Т.П. Караванова

ІНФОРМАТИКА

Основи алгоритмізації та програмування

777 задач з рекомендаціями та прикладами

Т.П. Караванова

ІНФОРМАТИКА

Основи алгоритмізації та програмування

777

задач з рекомендаціями та прикладами

Навчальний посібник
для 8-9 класів
із поглибленим вивченням
інформатики

Рекомендовано
Міністерством освіти і науки
України

За загальною редакцією
академіка НАН України
М.З. Згуровського

КИЇВ
« ГЕНЕЗА»
2006

ББК 32.81я721
К21

Рекомендовано Міністерством освіти і науки України
(лист МОН України №1/11-6488 від 20.12.04 р.)

За загальною редакцією
академіка НАН України Згуровського М.З.

Рецензенти:
Тимофієва Є.М. — канд. фіз.-мат. наук, доцент Чернівецького національного університету імені Юрія Федьковича;
Спориніна Т.Т. — викладач інформатики гімназії № 2 м. Чернівців, учитель-методист

Караванова Т.П.
К21 Інформатика: основи алгоритмізації та програм
777 задач з рек. та прикл.: Навч. посіб. для 8–9 кл
поглибл. вивч. інф-ки / За заг. ред. М.З. Згуровськог
К.: Генеза, 2006. - 286 с: іл. - Бібліограф. с. 286.

ISBN 966-504-466-4

У навчальному посібнику розглядаються питання а
ритмізації. Складання алгоритмів базується на мові програму
ня Pascal. Посібник містить велику кількість прикладів, ш
тичних порад, запитань для самоконтролю, різноманітних
складністю вправ та задач, що охоплюють усі теми алго
мізації, а також розглянуто основні можливості інтегрованог
редовища Turbo Pascal 7.0, описано типові помилки користув
до кожної з тем посібника.

Посібник розрахований на вчителів та учнів, що вивчають
форматику за «Програмою для спеціалізованих шкіл, гімна
ліцеїв. Інформатика і програмування для шкіл і ліцеїв. 8–11 к
си», рекомендованою Міністерством освіти і науки України, а
кож буде корисним учням 10-11 класів загальноосвітніх
вчальних закладів, при підготовці до олімпіад, турнірів, конкур
студентам середніх спеціальних навчальних закладів та ВНЗ

ББК 32.81я721

ISBN 966-504-466-4

...Програміст зобов'язаний володіти здібністю першокласного математика до абстракції та логічного мислення у сукупності з едісонівським талантом споруджувати все що завгодно з нуля і одиниці; він повинен поєднувати в собі акуратність банківського клерка з проникливістю розвідника, фантазію автора детективних романів із тверезою практичністю бізнесмена і, крім того, мати смак до колективної праці...

Академік А.П. Єршов

Від автора

Цей посібник є результатом багатьох років роботи з учнями-ліцеїстами. Я навіть і не думала, що якогось дня сяду за комп'ютер і почну мандрівку у світ літературних страждань, яка розтягнеться на багато років. Але це сталося!

Під час цієї роботи я намагалася поділитися з вами всім своїм досвідом викладання інформатики у школі, застерегти мас від помилок, які роблять мої учні, звернути вашу увагу на найбільш тонкі моменти алгоритмізації і на ті питання, які викликають в учнів найбільші складнощі. Маючи за плечима багаторічний досвід, я вважала це за можливе.

Працюючи над посібником, я намагалася викласти матеріал легкій, зручній для сприйняття формі, особливо там, де це стосується досить непростих питань. Як мені це вдалося, судити вам. Але якщо ви будете сумлінними і послідовними читачами, >удєте уважно розбирати і виконувати на комп'ютері всі запропоновані програми, то вас не спіткають жодні труднощі, ви закохастеся в інформатику так само, як і багато ваших друзів.

На жаль, на практиці дуже часто ототожнюють поняття алгоритмізація» та «програмування». Насправді вони лише астково перетинаються. Досвід свідчить, що, перш ніж зайнятися програмуванням, варто ознайомитися з основами алгоритмізації, навчитися логічно мислити. Дуже хотілося, щоб ме з таких позицій ви опрацьовували зміст цього посібника. дже я, насамперед, ставила за мету ознайомити вас з основними підходами до складання алгоритмів, а мова програмування ascal обрана лише як інструментарій для запису алгоритмів. ме тому ви не знайдете в посібнику детального опису середовища програмування Turbo Pascal 7.0. Зазначені лише деякі жливості цього середовища, ознайомлення з якими необхідне для зручного та швидкого налагодження програм, що дуть змогу перевірити складені алгоритми.

До посібника ввійшли переважно авторські завдання та ідеї найцікавіших задач з інших відомих збірників [9], [12], [15], а також завдання невідомих авторів, що зібрані у власній авторській картотеці.

Деякі задачі та вправи посібника можна виконувати усно або в зошиті, але коректність більшості з них варто все ж таки перевірити на комп'ютері. Можливо, при цьому вам удасться деякі з них удосконалити, скоротивши текст програми або час їх виконання.

Ті частини посібника, що мають відношення до основних алгоритмічних структур, починаються із «Серйозних розважалок». На тематику цих задач надихнула чудова книжка Григорія Остера «Задачник» [14]. Маю надію, що не тільки програмістам-початківцям молодшого віку сподобаються задачі з такою жартівливою тематикою. Адже за їх яскравою обгорткою прихованії досить серйозна логіка й математика.

Слід також звернути увагу на велику кількість завдань, включених до посібника. Всі вони тематично систематизовані розміщені після відповідних тем. Кожний такий блок задач, свою чергу, впорядковано за зростанням складності. Ал складність задачі - це поняття досить суб'єктивне. Одна й та сам задача одному може здатися дуже простою, іншому - надто склад ною. Може статися, що під час розв'язування складної задачі ва осяє цікава ідея, і, навпаки, напрочуд проста задача завдасть ва: багато клопоту. Тому не дуже зважайте на розташування задач посібнику – розв'язуйте ті задачі, які вам будуть до вподоби.

Я впевнена, що ви перш за все зазирнули в кінець посібн ка, аби пересвідчитися в наявності відповідей. Вас спіткал розчарування? Не дивуйтеся, переважно у збірниках задач інформатики, на відміну від математичних, фізичних ч: хімічних, відповіді не наводяться, оскільки відповіддю є са алгоритм чи програма. Не засмучуйтеся, вихід є. Відповідь н запитання, чи правильно складено вами алгоритм, дас комп'ютер. Лише від вас залежатиме, чи грамотно підібра тестові початкові дані для перевірки складеного алгоритму

Бажаю вам приємно і з користю провести час за комп'ютеро

Я вдячна своїм учням за те, що я їх учу і вони мене вчать, насолоду, яку я отримую від спілкування з ними, та за їх успіх: які мене тішать

Висловлюю подяку всім рецензентам за щирість, за слуш та справедливі зауваження і поради, які значно покращи зміст посібника.

Тетяна Караванс

ОСНОВНІ ПОНЯТТЯ АЛГОРИТМІЗАЦІЇ

ВСТУП В АЛГОРИТМІЗАЦІЮ

Поняття алгоритму

лгоритмом називається наперед задана скінченна ідовність чітких дій, виконання яких приводить до до-ення поставленої мети.

некому IX ст. жив і працював відомий середньоазіат-удрець, учений, філософ, математик Мухаммед бен Муса езмі. У своїх наукових трактатах він детально пояснив виконання арифметичних дій. При перекладі цих нау-аць уперше з'явився термін «алгоритм» (аль-Хорезмі – mi) і використовувався він спочатку для визначення бчислень у десятковій позиційній системі числення.

Ієє поняття слова «алгоритм» більш широке. Воно для тотожне зі словами «метод», «система правил», «про-ложна сказати, що алгоритм - це точна інструкція, а ї зустрічаються практично в усіх сферах нашого життя. итми виникли разом з появою математики. І саме у :у курсі математики ви часто зустрічаєтеся з алго-У молодших класах ви вивчали алгоритми покроко-5ання, множення, ділення, пізніше на уроках мате-ас учили правилам добування квадратного кореня, 5ісектриси кута, ділення відрізка навпіл та на задать рівних частин за допомогою олівця, циркуля та Ie обійшлося без алгоритмів на уроках фізики та оиклад, вам відомі алгоритми проведення фізичних експериментів, дослідження різних явищ. На уро-нітарних предметів ви вчили правила правопису. ми зустрічаються і в повсякденному житті. Наприк-дні учня мало чим відрізняються один від одного: я вранці в один і той самий час, умитися, одягнути-и, піти до школи, відвідати уроки, повернутися до-ати, виконати домашні завдання, відпочити, пове-и спати. У техніці добре відомі алгоритми **обробки**

деталей, у побуті - складання меблів, користування електричними приладами тощо.

Алгоритм є фундаментальним поняттям інформатики. Науковці виділяють три основні класи алгоритмів: обчислювальні, інформаційні та управляючі.

Обчислювальні алгоритми - це такі алгоритми, які працюють з порівняно простими видами даних, наприклад числами, векторами, матрицями.

Інформаційні алгоритми - це набір простих процедур, що працюють з великими об'ємами інформації. Прикладом такої процедури може бути пошук необхідної числової або символьної інформації, що відповідає певним ознакам. Ефективність роботи цих алгоритмів залежить від організації даних, як, наприклад, у базах даних.

Управляючі алгоритми характерні тим, що дані до них надходять від зовнішніх процесів, якими вони керують. Результатами роботи цих алгоритмів є вироблення своєчасного необхідного управляючого сигналу як реакції на швидку зміну вхідних даних.

У 30-ті роки XX ст. виникла теорія алгоритмів. До того часу поняття алгоритму зводилося до набору елементарних кроків: арифметичних дій, перевірки рівностей, нерівностей та інших відношень такого типу. Але на початку XX ст. об'єкти, якими оперували алгоритми, почали ускладнюватися, з'явилася потреба описувати операції над векторами, таблицями, множинами тощо. Постали питання щодо трактовки поняття елементарності кроків, тлумачення однозначності алгоритму, виникла думка, що не кожна математична задача може бути розв'язана за скінчений інтервал часу. Теорія алгоритмів досліджує питання побудови конкретних алгоритмічних моделей, кожна з яких містить конкретний набір елементарних кроків, способів визначення наступного кроку. Завданням теорії алгоритмів є також дослідження питання про існування чи не існування ефективних алгоритмів розв'язування окремих задач. З теоретичного погляду найбільшу цікавість викликають моделі, які одночасно були б і універсальними, і простими.

Бурхливий розвиток обчислювальної техніки, використання її в різноманітних наукових дослідженнях привели до створення великої кількості алгоритмів у багатьох прикладних галузях. Зрозуміло, що кожному алгоритму відповідає задача, яку він призначений розв'язувати, але, разом з тим, може існувати й кілька алгоритмів розв'язування даної задачі. Такі алгоритми називаються *еквівалентними,* і, зрозуміло, постає питання вивчення ефективності цих алгоритмів. Дослідники цих питань створили новий розділ теорії алґоритмів – теорію складності алгоритмів.

Складність алгоритму - це його кількісна характеристика. Вона визначається часом, за який виконується алгоритм (часова складність), та об'ємом пам'яті комп'ютера, необхідного для його виконання (ємнісна складність). Тому про складність алгоритму можна говорити стосовно саме машинних алгоритмічних моделей. Оскільки подолання обмежень на пам'ять комп'ютера - технічна проблема, що вирішується його удосконаленням майже щопівроку, то часова складність алгоритму, яка більшою мірою залежить від обраної алгоритмічної моделі, методу реалізації задачі, - творча, евристична проблема.

На практиці користувачів більше цікавлять не самі алгоритми, а задачі, які можна розв'язувати за їх допомогою. А оскільки для розв'язування задачі існують різні алгоритми, тому природно серед них визначити той, який має найменшу складність.

У посібнику найбільше нас цікавитимуть такі питання:

– ознайомлення з інструментальними засобами створення алгоритмів;

- ознайомлення з класичними алгоритмами для розв'язування типових задач;

– порівняння складності різних алгоритмів, які розв'язують одні й ті самі задачі.

Способи запису алгоритмів

Ми вже знаємо, що таке алгоритм. Тепер треба домовитися, як ми будемо його описувати.

Існує чотири способи запису алгоритмів, вибір яких залежить від того, хто його записує або на кого він орієнтований.

1. Словесний спосіб запису алгоритмів. Мабуть, поки що важко уявити собі інший спосіб запису. Це найпростіша і найдоступніша форма представлення алгоритму. Словесна форма зазвичай використовується для алгоритмів, орієнтованих на виконавця - людину. Справді, цей спосіб найдоступніший, незалежно від спеціальної підготовки.

Повертаючись до історії трансформації поняття «алгоритм», слід зазначити, що до 1950 р. під цим словом найчастіше розуміли викладений у «Елементах» Евкліда *алгоритм Евкліда* - процес знаходження найбільшого спільного дільника (НСД) двох цілих додатних чисел. Саме тому буде корисно навести як приклад опис цього алгоритму.

1. Дано два цілі додатні числа. Якщо вони рівні, то перше з них є найбільшим спільним дільником, якщо ж ні, то перейдемо до пункту 2.

2. Порівняємо два числа і виберемо більше з них.

3. Більше з двох чисел замінимо різницею більшого і меншого.

4. Перейдемо до пункту 1.

Зверніть увагу, що в першому пункті конкретно сказано, яке саме число необхідно вибрати у випадку рівності двох чисел. Саме це і є характерною ознакою алгоритму, виконавцем якого може бути навіть не підготовлена в даній галузі людина.

Наведений алгоритм застосуємо до конкретної пари чисел.

Нехай задано два числа: 45 і 12.

Продемонструємо процес знаходження НСД за наведеним алгоритмом у вигляді таблички, де на кожному кроці більше число, від якого віднімається друге (менше) число, виділятимемо жирним шрифтом:

1.	45	12
2.	33	12
3.	21	12
4.	9	12
5.	9	3
6.	6	3
7.	3	3

Отже, за сім кроків ми отримали результат:

НСД (45,12) = 3.

Перевірка роботи алгоритму є суттєвим кроком на шляху до його розуміння. Надалі домовимося, що кожний алгоритм, наведений у посібнику, повинен бути розібраний за кроками. Це найпростіший і найефективніший спосіб розуміння будь-якого алгоритму.

2. **Запис алгоритмів за допомогою схем.** Схеми дають змогу зобразити алгоритм в наочній графічній формі. Цей спосіб уже потребує деяких спеціальних знань. Це, зокрема, знання певних стандартів графічних зображень - блоків, усередині яких розміщують команди алгоритму. Деякі з цих блоків наведено в таблиці 1.

Таблиця 1

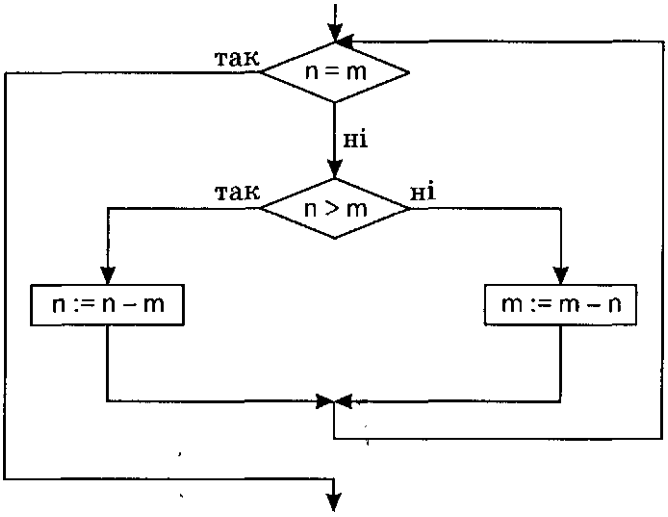
	Початок і кінець алгоритму
	Введення/виведення даних
	Вибір напрямку виконання алгоритму залежно від деяких змінних умов
	Виконання операцій, у результаті яких відбувається зміна значення даних

Під час створення схеми алгоритму блоки із записаними в них командами з'єднуються між собою стрілками, які визначають черговість виконання дій алгоритму.

Для запису команд усередині блоків використовується природна мова з елементами математичної символіки. В результаті перевірки умови під час вибору напрямку виконання алгоритму виникають два можливі шляхи для його продовження. Ці шляхи зображаються стрілками з позначеннями «так» і «ні». Перехід по стрілці з позначенням «так» відбувається в тому разі, коли умова виконується, а перехід по стрілці з позначенням «ні» - у протилежному випадку.

Блоки початку і кінця алгоритму використовуються при записі повного алгоритму задачі. Ми ж надалі вважатимемо, що алгоритми, які розглядаються в посібнику, можуть бути використані в інших алгоритмах як самостійні блоки. Тому блоки початку і кінця алгоритму не використовуватимуться.

Запишемо у вигляді блок-схеми алгоритм Евкліда (мал. 1).



Мал. 1

Наочність схематичного представлення алгоритму має свої переваги. Однак ця наочність швидко втрачається, якщо зображається великий алгоритм. У таких випадках у схемі алгоритму виділяються і відокремлюються її окремі частини - модулі, основною умовою яких є один вхід і один вихід. Згодом вони включаються у схему алгоритму як окремі блоки. Такий підхід до складання алгоритму відображає ідею структурного програмування.

3. Мова псевдокодів. Для запису алгоритмів за допомогою мови псевдокодів використовуються службові слова та спеціальні правила запису окремих дій. У мові псевдокодів

прийняті жорсткі синтаксичні правила для запису команд, що полегшує запис алгоритму на стадії його проектування і дає можливість використання широкого набору команд, розрахованих на абстрактного виконавця. Мова псевдокодів є проміжною між природною і формальною мовами.

Розглянемо як приклад алгоритм Евкліда.

```
АЛГОРИТМ найбільший_спільний_дільник
  ПОЧАТОК
    ВВЕДЕННЯ «Задайте два додатні цілі числа», n, m
    ПОКИ n≠m
    ПОЧАТОК
      ЯКЩО n > т
        ТО n: = n - т
        ІНАКШЕ т: = m - n
      ВСЕ
      КІНЕЦЬ
    ВИВЕДЕННЯ «Найбільший спільний дільник
    заданих чисел:», n
  КІНЕЦЬ
```

У наведеному алгоритмі неважко розібратися. Мова псевдокодів максимально наближена до звичайної розмовної мови. Усі службові слова, що використовуються для запису алгоритму, виділені жирним шрифтом. Їх треба записувати лише так, як це передбачено правилами мови. Окремі блоки алгоритму виділені вертикальними лініями і означають певні дії. Внутрішній блок визначає вибір тієї чи іншої дії залежно від відношення між значеннями величин *n* та *m*. Середній блок визначає дію повторення внутрішніх дій доти, поки значення величин *n* та *m* не стануть рівними. Зовнішній блок визначає повністю весь алгоритм, у якому передбачені також дії введення та виведення відповідно вхідної та результуючої інформації.

Для запису алгоритмів мовою псевдокодів треба знати всі службові слова цієї мови та правила запису деяких дій алгоритму.

4. **Мови програмування.** Найвищий професійний рівень для запису алгоритмів визначається знанням мов програмування. На практиці найчастіше виконавцями алгоритмів є комп'ютери. Тому алгоритми, призначені для виконання на обчислювальних машинах, повинні бути записані мовами, зрозумілими їм. Тут на перший план виходить необхідність точного запису команд, що унеможливлює довільне трактування їх виконавцем. Такі мови прийнято називати ***мовами програмування.***

Отже, мовою програмування називатимемо фіксовану систему позначень для опису структур даних і алгоритмів, призначених для виконання обчислювальними машинами.

Розроблено багато різних мов програмування, крім того ще відомі їхні різні версії. Чим це пояснюється? Чому не можна обійтися однією або принаймні кількома? Справа в тому, що на сьогоднішній день сфери використання комп'ютерів такі різноманітні і такі специфічні, що врахувати все в одній мові просто неможливо. Тому кожна мова програмування орієнтована на певний клас задач. Для програмування економічних задач використовуються мови програмування dBase, Paradox, Clipper; для розрахункових задач досить скористатися можливостями мов Basic, Pascal; для доступу до апаратних ресурсів комп'ютера – можливостями мов Assembler та C.

Усі наведені приклади мов програмування, крім Assembler, називаються мовами програмування високого рівня, бо їх конструкції максимально наближені до розмовних мов, зручні та зрозумілі користувачам. Мова Assembler називається машинно-орієнтованою, мовою низького рівня, тобто максимально наближеною до мови самого комп'ютера. На перший погляд ця мова здається складною, але, розібравшись у роботі комп'ютера, ви зрозумієте її простоту й доступність.

Надалі ми ознайомимося з можливостями мови програмування Pascal і розглянемо запис алгоритму пошуку найбільшого спільного дільника двох додатних цілих чисел саме цією мовою.

```
program NSD;
  var n, m: integer;
  begin
    writeln ('Задайте два додатні цілі числа:');
    readln (n, m);
    white n <> m do
      if n >m
        then n:= n - m
        else m:= m - n;
  writeln ('Найбільший спільний дільник заданих чисел:', n)
end.
```

Мабуть, ви звернули увагу на подібність запису алгоритму мовою псевдокодів і мовою програмування Pascal. Це справді так, але для запису алгоритму мовою Pascal бажано орієнтуватися в англійській мові та мати досить високий професійний рівень для вміння використовувати всі можливості цього потужного засобу програмування.

Для перетворення програм, написаних мовами програмування, у виконуваний вигляд існують спеціальні системні програми – інтерпретатори та компілятори.

Інтерпретатор – системна програма, яка здійснює синтаксичний контроль операторів початкового тексту програми та послідовне виконання її команд (операторів).

Компілятор — системна програма, яка здійснює переведення всього початкового тексту програми в машинний код.

Перші версії мови програмування Basic були реалізовані на комп'ютерах у вигляді інтерпретаторів. Тобто Basic-програми на кроці виконання виконувалися послідовно команда за командою. Тому навіть за наявності синтаксичних помилок у тексті програми та частина програми, яка передує помилкам, всеодно виконувалася.

До складу системного середовища Turbo Pascal 7.0, як і до всіх інших версій Pascal, входить компілятор, тому Pascal-програма виконуватиметься лише після виправлення всіх синтаксичних помилок у початковому тексті.

Типи алгоритмів

Після ознайомлення з різними формами запису алгоритмів ми вже зможемо розрізнити різні за типами команди, а відповідно і алгоритми, які їх використовують. З усіма цими командами ми будемо ознайомлюватися на прикладі алгоритму Евкліда.

Лінійні алгоритми. ***Лінійним алгоритмом називається такий алгоритм, в якому команди виконуються послідовно в часі одна за одною.***

Лінійні алгоритми складаються лише з лінійних команд. У таких алгоритмах після виконання однієї команди виконавець завжди переходить до виконання наступної за порядком запису в алгоритмі команди. В алгоритмі Евкліда лінійними командами є команди введення та виведення інформації, команди обчислення нових значень величин *n* та *m*. Звичайно, у складніших програмах вам буде важко знайти суто лінійні алгоритми. Тому краще вести мову про окремі лінійні фрагменти складних програм.

Розгалужені алгоритми. ***Алгоритм, що містить хоча б одну умову, в результаті перевірки якої здійснюється перехід до одного з можливих кроків, називається розгалуженим.***

Досить часто вибір тих чи інших дій для продовження алгоритму залежить від виконання або невиконання певних умов. Команди, які аналізують ці умови та наступний вибір виконуваних дій алгоритму, називаються ***командами розгалуження.***

Прикладом команди розгалуження в алгоритмі Евкліда є команда вибору зміни значення величини *n* або величини *m* залежно від співвідношення між ними (n > m - «так» або «ні»). У результаті вибору одна з команд (n := n - m, m := m - n) буде пропущена, хоча в алгоритмі вони записані одна за одною.

За аналогією з лінійними алгоритмами, мабуть, доцільніше говорити про розгалужені фрагменти алгоритмів, ніж про самі розгалужені алгоритми.

Циклічні алгоритми. ***Алгоритм, в якому певна послідовність команд повторюється кілька разів з новими вхідними даними, називається циклічним.***

Команди, що дають змогу кілька разів повторювати певний блок команд алгоритму, називаються ***циклічними.***

Необхідність у використанні циклічних команд виникає, коли треба кілька разів використовувати для обчислення одні й ті самі формули з різними значеннями змінних або якісь інші однотипні команди. В алгоритмі Евкліда ми спостерігаємо циклічність під час багатократного повторення перших трьох команд, доки нові значення чисел не стануть однаковими. Безумовно, циклічний алгоритм за розмірами набагато менший, ніж той, у якому команди були б повторені стільки разів, скільки їх треба виконати.

Допоміжні алгоритми. Набувши певного досвіду в алгоритмізації й переходячи до складання досить серйозних алгоритмів, ви обов'язково зіткнетеся з ситуацією, коли треба описувати дуже схожі фрагменти алгоритму. Звичайно, бажано якимось чином записати цей фрагмент лише один раз і звертатися до нього стільки разів, скільки в цьому виникне необхідність.

Допоміжним називається алгоритм, який створений наперед і викликається та повністю виконується в даному алгоритмі тоді, коли виникає в цьому необхідність.

Допоміжні алгоритми можна розглядати як:

— внутрішні, локальні, що створюються автором у межах даного алгоритму і доступні для використання тільки в цьому алгоритмі;

- зовнішні, глобальні, які можна використовувати в різних незалежних алгоритмах. Такі допоміжні алгоритми найчастіше об'єднуються у так звані бібліотеки. Для користування цими бібліотеками повинні існувати інструкції з описом усіх допоміжних алгоритмів, що входять до них, та правила користування ними.

Стосовно великих і складних алгоритмів доцільно, мабуть, говорити про тип не всього алгоритму, а лише визначаючи типи окремих його фрагментів.

Властивості алгоритмів

Засвоївши поняття алгоритму, можна зробити висновок, що будь-який алгоритм є інструкцією, послідовністю деяких вказівок. Але не кожну інструкцію чи послідовність дій можна назвати алгоритмом.

Отже, сформулюємо основні властивості алгоритму.

Дискретність. Будь-який алгоритм зображається у вигляді окремих дій. Виконання команд алгоритму повинно бути послідовним, з точною фіксацією моментів завершення виконання однієї команди і початку виконання наступної.

Скінченність. Виконання алгоритму припиняється після завершення скінченної кількості кроків. У математиці існують обчислювальні процедури, які мають алгоритмічний характер, але не володіють властивістю скінченності. Прикладом такої процедури може бути обчислення значення числа. Така процедура описує нескінченний процес, який ніколи не завершиться. Але якщо обмежитися певною кількістю знаків після коми, то ми отримаємо алгоритм обчислення числа із заданою точністю.

Визначеність. Кожний крок алгоритму має бути чітко й однозначно визначений, щоб не допустити довільного трактування виконавцем. Тобто алгоритм розрахований на механічне виконання. Якщо один і той самий алгоритм доручити для виконання різним виконавцям, то вони повинні отримати один і той самий результат.

Зрозумілість. Формулювання дій алгоритму повинно бути орієнтоване на конкретного виконавця. Якщо виконавець є некомпетентним у питаннях, що вирішуються даним алгоритмом, то необхідно вибрати найдоступнішу для його формулювання форму, якою є словесний спосіб. Якщо ж виконання алгоритму буде запропоновано комп'ютеру, то алгоритм потрібно зобразити відповідною машинною мовою.

Масовість. В алгоритмі повинна бути передбачена можливість виконання його для різних початкових значень, щоб використовувати його для розв'язування цілого класу однотипних задач.

Результативність. Алгоритм повинен забезпечувати можливість отримання результату після скінченної кількості кроків.

Ефективність. Кожний крок алгоритму повинен бути виконуваним, тобто кожна дія має бути достатньо простою, щоб її можна було виконати точно і за скінченний інтервал часу.

Як бачимо, для того щоб набір інструкцій, вказівок можна було назвати алгоритмом, він повинен задовольняти низку достатньо жорстких умов.

Виконавець алгоритму. Формальне виконання алгоритму

Точне визначення, що таке виконавець, дати дуже складно, та в цьому й немає потреби. Важливо зрозуміти основні характеристики виконавця: середовище, елементарні дії, систему команд, відмови. Середовище - це місце перебування виконав-

ця. Кожен виконавець може виконувати команди лише з деякого строго заданого списку - системи команд виконавця. Для кожної команди повинні бути задані умови застосування, тобто в яких станах середовища може бути виконана команда, й описані результати її виконання.

Виконавця можна представити у вигляді деякого пристрою, керування яким спричиняє виконання тієї чи іншої команди. Після виклику команди виконавець здійснює відповідну елементарну дію. Важливо зазначити, що в цьому процесі нас більше цікавить результат, а не механізм виконання команди. Відмови виконавця виникають за умови виклику команди в неприпустимому для даної команди стані середовища. Ситуації, при яких виникає відмова, різні для різних команд виконавця (наприклад, ділення на нуль, відсутність принтера при виведенні інформації на друк). При виконанні деяких команд відмови ніколи не виникають (наприклад, обчислення виразу 2 + 2).

У деяких випадках виконавцем алгоритмів є людина. Наприклад, якщо йдеться про правила, інструкції, функціональні та посадові обов'язки тощо. Але людина далеко не єдиний виконавець алгоритмів. Роботи-маніпулятори, верстати з програмним управлінням, жива клітина і навіть тварини в цирку виконують різноманітні алгоритми, в тому числі й ті алгоритми, які людина не може виконати.

Що ж таке виконавець? Спрощено виконавця можна уявити собі як деякий пристрій керування, з'єднаний з набором інструментів. Пристрій керування розуміє алгоритми і організовує їх виконання, користуючись при цьому відповідними інструментами. А інструменти, сприймаючи команди керуючого пристрою, виконують дії. Якщо людину розглядати як виконавця, то можна провести таку аналогію: мозок - пристрій керування, руки, ноги, очі, ніс тощо — інструменти. У роботів-маніпуляторів, верстатів з програмним управлінням, комп'ютерів пристроєм керування є процесор, а набір інструментів залежить від того, для розв'язування яких задач призначений той чи інший виконавець.

Під формальним виконанням алгоритму розуміється таке його виконання, коли сам виконавець не знає ні постановки задачі, ні змісту отриманих результатів, але, чітко виконуючи усі дії, записані в алгоритмі, досягає результату.

Найчастіше алгоритми виконуються саме формальним чином. Формальними виконавцями є користувачі різноманітних побутових електричних приладів, учасники спортивних ігор, які дотримуються правил цих ігор, учні, що застосовують у

своїх творчих роботах мовні правила, виконують завдання з математики чи фізики, використовуючи відомі формули, закони та методи розв'язування різних типів задач. Продовжуючи цю думку, комп'ютер також слід вважати формальним виконавцем алгоритмів при виконанні ним програм.

Аргументи, результати, проміжні величини

Для роботи багатьох алгоритмів треба задавати початкові значення. Ці значення передаються в алгоритм за допомогою аргументів.

Аргумент — це величина, значення якої необхідно задати для виконання алгоритму.

Проте є алгоритми, що не потребують жодних початкових значень для свого виконання. Пізніше ми ознайомимося з такими алгоритмами. Однак немає жодного алгоритму, що не дає ніякого результату. Справді, яка ж потреба в такому алгоритмі? Прикладом різноманітності результатів роботи програм є ігрові комп'ютерні програми. У цих програмах дані обробляються та певним чином перетворюються у графічні і звукові образи.

Результат - це величина, значення якої отримується внаслідок виконання алгоритму.

Під час складання багатьох алгоритмів виникає необхідність крім аргументів і результатів використовувати ще й додаткові величини. Введення в алгоритм таких величин задає автор алгоритму.

Проміжна величина — це величина, яка додатково вводиться в процесі розробки алгоритму.

Ми вже достатньо наочно уявили собі алгоритм. Але як він виглядає з погляду звичайного користувача цього алгоритму? Найчастіше користувач не знає, яким чином цей алгоритм записаний, за допомогою яких команд, які методи були застосовані для його реалізації, якою мовою програмування. У даному випадку йдеться про виконання алгоритму на комп'ютері у вигляді готової програми. Виконавець алгоритму бачить лише його зовнішню сторону: які початкові дані необхідно задати і в якому вигляді отримується результат. Кожний, мабуть, може пригадати номер фокусника, коли той у чорну скриньку закладає одні предмети, а витягує зовсім інші. Від глядачів прихований вміст цієї чорної скриньки, і залишається таємницею, яким чином у ній відбувається «перетворення» предметів. Саме такою «чор-

ною скринькою» для користувача і є програма, якою він користується на комп'ютері. Це можна зобразити у такому вигляді:

аргументи **АЛГОРИТМ** → **результати**

Запитання для самоконтролю

1. Що називається алгоритмом?
2. Чому словесний спосіб запису алгоритму вважається найпростішою формою його подання?
3. У чому полягають особливості схематичної форми зображення алгоритму?
4. У чому полягають особливості мови псевдокодів як одного із способів подання алгоритму?
5. Мови програмування - один із способів подання алгоритму.
6. Чим обумовлена така послідовність ознайомлення із способами запису алгоритмів: словесний, схема алгоритму, мова псевдокодів, мови програмування?
7. Які типи алгоритмів ви знаєте? У чому полягає основна характеристика кожного з них?
8. Яка різниця між інтерпретатором і компілятором?
9. Назвіть основні властивості алгоритмів та дайте їм коротку характеристику.
10. Що розуміється під формальним виконанням алгоритму?
11. Що називається аргументом алгоритму?
12. Що називається результатом алгоритму?
13. Коли виникає необхідність уведення проміжних даних?

Виконайте завдання

1. Записати у словесній формі алгоритм свого розпорядку дня на робочий тиждень. Визначити тип цього алгоритму.

2. Записати у словесній формі алгоритм поділу відрізка навпіл за допомогою олівця, лінійки та циркуля. Визначити тип цього алгоритму.

3. Записати у словесній формі алгоритм поділу відрізка на задану кількість рівних частин за допомогою олівця та лінійки. Визначити тип цього алгоритму.

4. Записати у словесній формі алгоритм прибирання квартири. Визначити тип цього алгоритму.

5. Записати у словесній формі алгоритм отримання числа 512 в результаті виконання арифметичних дій з числом 2, не використовуючи операцію піднесення до степеня. Визначити тип цього алгоритму.

6. Записати у словесній формі алгоритм смаження яєчні на сніданок. Визначити тип цього алгоритму.

7. Записати у словесній формі алгоритм вирішення проблеми «Чи брати парасольку?», коли ви збираєтесь вийти на вулицю. Визначити тип цього алгоритму.

8. Записати у вигляді схеми алгоритм переходу вулиці, де встановлено світлофор. Визначити тип цього алгоритму.

9. Записати у вигляді схеми алгоритм правопису префіксів *з-* та *с-* перед приголосними.

10. Записати у вигляді схеми алгоритм завантаження операційної системи після ввімкнення комп'ютера. Визначити тип цього алгоритму.

11. Записати у вигляді схеми алгоритму послідовність виконання дій для обчислення таких виразів:

1) $(x+y)^2 - \frac{x}{y} + 10$; 2) $x - \frac{z}{a+2}$; 3) $\frac{\sqrt{x-10}}{x}$;

4) $\frac{a+b}{ab}$; 5) $a - \frac{b}{a+1}$.

12. Як за допомогою двох пісочних годинників на 3 хв та на 8 хв відміряти 7 хв? Визначити тип цього алгоритму.

13. Як за допомогою скляних трилітрової і п'ятилітрової банок відміряти об'єм рідини, що дорівнює

1) 7 л; 2) 12 л; 3) 14 л; 4) 1 л?

14. Як поділити на дві рівні частини 12 відер хлібного квасу, налитого у дванадцятивідерну бочку, користуючись для цього порожніми бочками - восьмивідерною і п'ятивідерною?

15. (Старовинна задача). Мисливцю треба перевезти через річку вовка, козу і капусту. Човен настільки малий, що в ньому можуть поміститися мисливець і з ним або вовк, або коза, або капуста. Вовка не можна лишати з козою, а козу з капустою. Як бути мисливцю?

16. До двох хлопчиків, що каталися річкою на човні, звернулися два дорослі чоловіки з проханням перевезти їх на протилежний берег. Як це зробити хлопчикам, якщо за один раз у човен може сісти або лише два хлопчики, або один хлопчик і один чоловік?

17. Початкове розташування чорних і білих шашок таке:

Поміняти місцями білі й чорні шашки, враховуючи, що їх можна рухати лише на сусідню порожню клітинку або перестрибувати через одну зайняту. Скласти схему алгоритму, визначити тип алгоритму.

18. Записати у вигляді схеми алгоритм угадування задуманого числа в проміжку:

1) від 0 до 7; 2) від 0 до 15;
3) від 0 до 31; 4) від 0 до 100.

Під час вгадування можна задавати лише одне із запитань типу: «Ваше число менше за ...?» або ж «Ваше число більше за ...?». Відповіддю на запитання може бути «Так» або «Ні». За яку найменшу кількість кроків це можна зробити? Визначити тип цього алгоритму.

19. Два хлопчики грають у гру Ваше. За умовою кожний гравець за один хід має право брати від 1 до 3 предметів. Виграє той, хто візьме останній предмет. Як повинен грати перший хлопчик, щоб напевне виграти, якщо на столі є:

1) 15 предметів; 2) 17 предметів; 3) 8 предметів?

Сформулювати виграшну стратегію для першого гравця за наявності будь-якої кількості предметів. Визначити тип цього алгоритму.

20. Скількома способами хлопчик може піднятися східцями на 10 сходинку, якщо він може підніматися на наступний східець або переступати через один чи два східці? Сформулювати алгоритм визначення кількості способів сходження на сходинку.

21. Є 8 монет однакової вартості, серед яких одна фальшива. Відомо, що фальшива монета трохи легша за інші. Як визначити фальшиву монету двома зважуваннями на терезах з двома шальками без гирок? Скласти схему алгоритму, визначити його тип.

22. Є 12 монет, серед яких одна фальшива. За три зважування монет на терезах без гирок виявити фальшиву монету і дати відповідь на запитання: «Фальшива монета легша чи важча за нефальшиву?» Скласти схему алгоритму, визначити його тип.

23. Є 10 мішків з монетами. Відомо, що в одному з мішків усі монети фальшиві і кожна з них на 1 г легша від нефальшивої. За одне зважування на вагах з гирями визначити, в якому з мішків фальшиві монети. Скласти схему алгоритму, визначити його тип.

ЕТАПИ РОЗВ'ЯЗУВАННЯ ЗАДАЧ НА КОМП'ЮТЕРІ

Перш ніж отримати очікуваний результат роботи програми на комп'ютері, треба виконати досить багато копіткої підготовчої роботи.

Постановка задачі. Розв'язування будь-якої задачі починається з її постановки. На першому кроці треба добре уявити, в чому саме полягає дана задача, які початкові дані необхідні для її розв'язування, яку інформацію вважати результатами розв'язування.

Побудова математичної моделі. Побудова математичної моделі алгоритму - дуже відповідальний етап. Не завжди умова

сформульованої задачі містить у собі готову математичну формулу, яку можна застосувати для розробки алгоритму задачі, не завжди розв'язок задачі вдається отримати в явному математизованому вигляді, що зв'язує вхідні дані та результати. Для цього створюється інформаційна математична модель об'єкта, що вивчається.

Вибір виду моделі залежить від інформаційної сутності об'єкта, а не від його фізичної природи. Тобто не стільки важливе прикладне значення задач, скільки однотипність методів, якими вони розв'язуються. Наприклад, логічні моделі використовуються як для моделювання людських міркувань, так і для опису логічних схем автоматики.

Розв'язуючи задачу про рух тіла під дією прикладених до нього сил, ми перш за все записуємо рівняння його руху на основі законів механіки. Проте, крім сили тяжіння, на тіло діє і сила опору повітря. Постає питання достовірності математичної моделі та реальної картини досліджуваного процесу. Інколи буває неможливо врахувати всі реальні фактори, що впливають на нього. Тому дуже важливим є вміння виділяти серед усіх факторів головні і другорядні, для того щоб останніми можна було знехтувати. При цьому може скластися ситуація, коли наперед невідомо, якими саме факторами можна знехтувати, і тому може бути декілька математичних моделей, що описують одну і ту саму задачу чи явище з різним ступенем достовірності.

Ступінь відповідності моделі реальному об'єкту перевіряється практикою, комп'ютерним експериментом. Критерій застосування практики дає можливість оцінити побудовану модель і уточнити її в разі потреби. Чим достовірніше математична модель відображає реальні сторони процесу, тим точніші одержувані результати.

Побудова алгоритму. Наступним кроком є розробка алгоритму обробки інформації на основі побудованої математичної моделі. Тепер треба знайти спосіб розв'язування цієї задачі. Для цього можуть бути застосовані вже відомі методи, проведені їх оцінка, аналіз, відбір або розроблені нові методи. Наприклад, вибір методу розв'язування системи рівнянь, що описує дану математичну модель.

Під час створення складних алгоритмів застосовується метод покрокової розробки. Сутність цього методу полягає в тому, що алгоритм розробляється «зверху донизу».

На кожному етапі приймається невелика кількість рішень, що приводить до поступової деталізації, уточнення як виконуваної, так і інформаційної структури алгоритму. Такий підхід дає змогу поділити алгоритм на окремі частини — модулі, кожний з яких розв'язує свою самостійну підзадачу. Це дає мож-

ливість сконцентрувати зусилля на розв'язуванні кожної підзадачі, що реалізується у вигляді окремої процедури. Для кожного такого модуля визначаються свої методи реалізації алгоритму та структура даних, якими він оперує.

Останнім кроком у методі покрокової деталізації є об'єднання окремих незалежних модулів у єдине ціле. Для цього між модулями повинні бути встановлені зв'язки, тобто узгоджена передача інформації від одних модулів до інших: результати виконання одних модулів є вхідною інформацією для інших.

Якщо поставлена задача є складним завданням, то важливість поділу алгоритму на окремі модулі неможливо недооцінити.

Вибір мови програмування. Алгоритм, призначений для виконання на комп'ютері, має бути записаний мовою програмування. Різноманітність існуючих мов програмування вимагає від програміста реальної оцінки складності й характеру задачі. Тільки після цього він зможе здійснити вибір найоптимальнішої мови програмування для реалізації поставленої задачі.

Враховуючи можливість поділу всього алгоритму на окремі модулі, для кожного з них вибір мови програмування може бути здійснений окремо.

Процес розробки програми, як і алгоритму, може здійснюватися за принципом «зверху донизу». В результаті це дає змогу отримати добре структуровану програму, читання й розуміння якої значно полегшене.

Складання програми. Цей етап потребує лише знання обраної мови програмування. Суть його полягає в тому, щоб на основі розроблених алгоритмів і представлень інформації створити програму для комп'ютера.

Компіляція програми. Переведення програми на машинну мову здійснюється за допомогою спеціальних програм — компіляторів. Однією з їх функцій є перевірка наявності у програмі синтаксичних помилок після введення її у комп'ютер. Не тіште себе надією, що ваша, навіть найпростіша, програма написана бездоганно. Серед програмістів побутує таке прислів'я: «Якщо програма не має помилок, це означає, що у вас поганий компілятор».

Налагодження програми, контрольний прорахунок. Виправлення всіх синтаксичних помилок у програмі на попередньому етапі зовсім не позбавляє вас від помилок іншого типу - змістовних, логічних. Вони з'являються під час помилкового трактування умови поставленої задачі, через недосконаліть математичної моделі або недоліки у самому побудованому алгоритмі, що призводить до отримання помилкового результату. Такі помилки не можуть бути усунені на стадії компіляції,

тому що для їх виявлення необхідна інформація про сутність самої задачі. Їх може усунути лише сам розробник.

Смисл налагодження програми полягає в тому, що готується ***система тестів,*** за допомогою якої перевіряється робота програми в різних можливих режимах. Кожний тест містить набір вхідних даних, для яких відомий результат. Для більшості програм виникає необхідність добору не одного, а цілої серії тестів, щоб перевірити найбільше число можливих ситуацій, які можуть трапитися в процесі роботи програми. Якщо для всіх тестів результати роботи програми збіглися з розрахунками, то можна вважати, що логічних помилок немає.

Експлуатація програми. Якщо ваша програма успішно пройшла всі попередні етапи, то можна з певною мірою припущення вірити всім результатам, отриманим при будь-яких початкових даних. Тепер програму можна тиражувати і пропонувати іншим користувачам, доповнивши тексти програм описом обмежень на початкові дані, відповідною документацією для користувача, описом середовища даної програми тощо.

Більшість розглянутих етапів розв'язування задач на комп'ютері виконуються людиною і носять творчий, евристичний характер.

Запитання для самоконтролю

1. Назвіть основні етапи розв'язування задач на ЕОМ.
2. У чому полягає важливість побудови математичної моделі задачі?
3. У чому полягає суть побудови алгоритму як етапу розв'язування задачі на комп'ютері?
4. У чому полягає важливість тестування програми при її налагодженні?

Розділ II

БАЗОВІ АЛГОРИТМІЧНІ СТРУКТУРИ

МОВА ПРОГРАМУВАННЯ PASCAL. ПОЧАТКОВІ ПОНЯТТЯ

Історія створення мови програмування Pascal

У 1970 р. з'явилося повідомлення про створення ще однієї мови програмування, названої на честь відомого математика, розробника однієї з перших обчислювальних машин Блеза Паскаля. Автором цієї мови програмування був Ніклаус Вірт, професор, директор Інституту інформатики Швейцарської вищої політехнічної школи, лауреат Тьюрингівської премії, автор численних та широковідомих праць у галузі програмування. Н. Вірт є автором ще таких мов програмування, як Ейлер, Модула, Модула-2. Крім цього ним запропонована методика покрокової розробки програм від глобального до локального, від загального до часткового, тобто занурення в алгоритм зверху донизу. Ця методика була визнана найсильнішою ідеологією програмування в 70-ті роки.

Н. Вірт вирішив взятися за створення нової мови програмування з методичною метою: дати своїм студентам інструмент для вивчення програмування як систематичної, логічної дисципліни, що базується на фундаментальних поняттях.

У своїй праці «Систематичне програмування» Н. Вірт пише: «...мова, якою студента навчають висловлювати свої думки, здійснює глибокий вплив на його навички мислення та винахідницькі здібності...».

Мова програмування Pascal, на відміну від усіх попередніх популярних мов, дала в руки програмісту прості конструкції команд, що виглядають дуже природно. Тобто їх зміст збігається з тим, чого інтуїтивно чекає користувач. Набір команд мови Pascal зовсім невеликий і складає її базову частину. Всі додаткові можливості мови Pascal дають змогу програмістові користуватися типами змінних, властивих лише цій мові програмування, а також створювати свої власні типи.

Дуже швидко мова Pascal набула великої популярності серед програмістів і була реалізована для різних типів комп'ютерів.

Ми розглядатимемо інтегроване середовище програмування Turbo Pascal 7.0 і зазначатимемо, за потреби, його відмінності від інтегрованого середовища програмування Turbo Pascal 6.0, хоча всі команди і приклади, що будуть наведені далі, можна використати в усіх попередніх версіях.

Ви зможете належним чином оцінити всю красу і привабливість мови програмування Pascal, і після її вивчення у вас обов'язково з'явиться потреба в поглибленні своїх програмістських навичок.

Поняття величини. Типи величин

Алгоритми, так само як і програми, в першу чергу мають справу з ***величинами.***

Енциклопедичний словник тлумачить поняття величини як узагальнення конкретних понять: довжини, площі, ваги тощо. Вибравши для однієї з цих величин одиницю виміру, можна будь-яку іншу величину цього самого роду виразити числом відношення до одиниці виміру. Тобто це поняття носить кількісний, порівняльний характер.

З погляду алгоритмізації в якості величин та їх сукупності виступають дані, що обробляються цими алгоритмами. Якщо ж розглядати алгоритм на рівні його виконання на комп'ютері, то в цьому випадку величина трактується як інформація, якою оперує програма, а їй відповідно повинно бути відведено місце в пам'яті комп'ютера. Міркуючи таким чином, слід визнати, що сама програма для комп'ютера також є послідовністю даних, тобто програмних величин (команд, операторів), кожній з яких при введенні відводиться місце в пам'яті.

Отже, саме за допомогою величин передаються всі значення, вся інформація, виконуються різні обчислення.

Величини можна поділити на ***сталі*** та ***змінні.***

Сталі величини — це величини, які не змінюють свого значення протягом виконання всього алгоритму.

Наведемо кілька прикладів сталих величин:

10, −123.45, 'A', 'ліцей', 'моя програма'.

Що незвичного у цих прикладах? Перше число (10) є цілим, і його запис не викликає ніяких запитань. А от у другому числі, що є дійсним (−123.45), ви, мабуть, звернули увагу на те, що ціла частина числа відділяється від дробової знаком «.», а не «,», як прийнято в математиці.

Слід запам'ятати, що в усіх мовах програмування у записі дійсних чисел ціла частина числа в'ідділяється від дробової знаком «.».

Наступні три приклади сталих величин є дуже специфічними. Стала величина **'A'** - це знайомий нам символ **«A»,** який використовується як складова частина деякого тексту. Два останніх приклади є сталими величинами-текстами (рядками), які використовуються у програмах як коментар або текст.

Тепер можна зробити висновок, що такі сталі, як символи або послідовності символів, у Pascal-програмах необхідно брати в значок ***апострофа.***

На відміну від сталих величин, значення яких під час виконання алгоритму змінити неможливо, змінні величини можуть змінювати свої значення.

Змінними величинами називаються величини, які можуть змінювати своє значення протягом виконання алгоритму.

Як позначити ці величини, якщо ми наперед не знаємо їх значень? У математиці та фізиці такі величини мають якесь ім'я. Наприклад, *S* - площа, *V* - об'єм, *F* - сила, *m* - маса тіла, *U* - напруга електричного струму. У програмуванні є певні правила для найменування змінних величин.

Іменем, або ідентифікатором, змінної величини може бути будь-яка послідовність латинських літер (a — *z* та ***A — Z),*** цифр (0 — 9) та знака підкреслення, що починається з літери.

Характерно, що у Pascal-програмах не розрізняються великі та маленькі літери. Тобто ідентифікатори *MyResult* та ***myresu.lt*** вважаються однаковими. Але все ж таки слід рекомендувати застосовувати ідентифікатори змінних переважно за їх змістовим призначенням для того, щоб програма була **«читабельною».** І саме цього будемо дотримуватися у прикладах посібника.

На кількість символів, з яких складаються імена змінних, накладається обмеження: ідентифікатори не повинні містити більше як 63 символи. Ви можете використовувати й довші ідентифікатори, але всі символи після 63-го ігноруються. А оскільки кожне ім'я змінної повинно бути в даній програмі ***унікальним,*** то можна припуститися помилки, якщо ці ідентифікатори будуть відмінні у символах після 63-го.

Позначивши змінну ідентифікатором, ще треба вказати, яких саме значень може набувати ця величина.

З поняттям величини завжди пов'язують поняття ***типу:*** ціла, дійсна, символьна або рядкова. Ми вже знаємо, що на рівні виконання програми кожній величині відводиться відповідне місце в пам'яті комп'ютера, розмір якого залежить саме від типу величини. Яке місце в пам'яті займають різні типи змінних, з якими працює Pascal, ми розглянемо далі.

Наявність різноманітних типів величин у мові Pascal можна пояснити такими двома моментами:

- економія пам'яті комп'ютера під час виконання програм;
- перетворення типів величин під час виконання операцій, застосованих до них.

Перше твердження не повинно викликати ніяких непорозумінь. Особливо гостро це питання постане перед вами, якщо ви матимете справу зі створенням програм значних розмірів. А от стосовно другого, то на ньому варто зупинитися детальніше. Кожна операція в Pascal вимагає величин певного типу і видає результат так само певного типу. Саме тому при ознайомленні з операціями Pascal ми будемо звертати увагу на типи величин, якими вони оперують, та на тип результату, який одержується.

Якщо для сталих величин не вимагається вказувати їх тип, оскільки вони задаються явним чином, то для змінних величин обов'язково треба визначати тип.

І ще одна дуже суттєва деталь: у кожний момент часу при виконанні програми змінна величина повинна мати якесь значення!

Отже, про сталу величину за її записом зразу можна сказати все: яке її значення, якого вона типу. А змінна величина, на відміну від сталої, дуже таємнича. У програмах вона характеризується такими ознаками: ***іменем, типом*** і ***значенням.***

Стандартні ***ТИПИ ЗМІННИХ*** *у Pascal*

У самому Pascal вже закладені найпоширеніші типи змінних величин, які називаються ***стандартними,*** тобто такі, які зрозумілі без додаткових пояснень.

Типи змінних, що набувають цілих значень, позначаються службовим словом **Integer.** Існують певні межі для значень величин цього типу:

$$-32768 .. 32767.$$

У пам'яті комп'ютера такі значення займають *2* ***байти.***

У Pascal існує можливість використання ще декількох різновидів цілих типів, які мають інші діапазони змін своїх значень.

Подамо всі типи цілих чисел у вигляді таблиці 2.

Таблиця 2

Тип	Діапазон	Формат	Розмір у байтах
Integer	- 32768 .. 32767	Знаковий	2
Shortint	- 128 .. 127	Знаковий	1
Longint	- 2147483648 .. 2147483647	Знаковий	4
Byte	0 .. 255	Беззнаковий	1
Word	0 .. 65535	Беззнаковий	2

Якщо наперед відомо, що деяка змінна може набувати досить невеликих значень протягом виконання алгоритму, то її можна описати типом **Shortint,** тобто коротке ціле. Якщо потрібен дуже великий розбіг значень цілої змінної, то можна скористатися типом **Longint.** Бувають такі програми, що працюють тільки з цілими додатними числами. І для такого випадку у Pascal є відповідні типи - тип **Byte** та тип **Word,** які не враховують знак числа.

Дійсні змінні у Pascal позначаються типом **Real.** Їх значення займають у пам'яті комп'ютера ***6 байт,*** дають 11–12 значущих цифр числа. Їх значення визначаються інтервалом:

$$2.9*10^{-39} .. 1.7*10^{38},$$

тобто числа, менші за ліву межу, вважаються рівними 0, а більші за праву - залишаються рівними $1.7*10^{38}$.

Крім основного типу дійсних чисел, існують його різновиди, які відрізняються кількістю значущих цифр числа, кількістю байт, яке воно займає у пам'яті комп'ютера, та інтервалом точності значень. Для порівняння наведемо їх у вигляді таблиці 3.

Таблиця 3

Тип	Діапазон	Число значущих цифр	Розмір у байтах
Real	$2.9*10^{-39} .. 1.7*10^{38}$	11 - 12	6
Single	$1.5*10^{-45} .. 3.4*10^{38}$	7 - 8	4
Double	$5.0*10^{-324} .. 1.7*10^{308}$	15 – 16	8
Extended	$3.4*10^{-4932} .. 1.1*10^{4932}$	19 – 20	10
Comp	$-263 + 1 .. 263 - 1$	19 – 20	8

Ознайомлюючись з прикладами сталих величин, ви, напевно, звернули увагу на те, що величини, якими оперує Pascal, можуть бути не тільки числовими.

Тип **Char** (від англ. слова character – *символ*) описує символьні змінні, тобто такі, значенням яких може бути будь-який символ із таблиці ASCII-кодів. Коди цих символів займають у пам'яті ***1 байт,*** і їх можлива кількість складає 256 (2^8=256).

Послідовність символів визначається типом **String,** що перекладається як *рядок.* Кількість символів у таких рядках обмежена числом 255. Об'єм пам'яті, який займають дані цього типу, складає ***256 байт.*** Згодом у нас буде можливість детальніше ознайомитися і попрацювати зі змінними цього типу.

Завершуємо невеликий огляд стандартних типів змінних ще одним типом, що має назву **Boolean -** *логічний тип.* Змінні

цього типу можуть набувати лише двох значень - **False** і **True** ***(істина*** та ***хиба).*** Розмір пам'яті для значень цього типу складає всього ***1 байт.***

Форми подання дійсних чисел у Pascal

Виникає таке запитання: «Як числа зберігаються у пам'яті комп'ютера?»

Щодо цілих чисел, то вони зберігаються у двійковому вигляді. Ви можете самі переконатися у достовірності вибраних діапазонів зміни значень цілих типів. Для цього достатньо перевести їх мінімальне та максимальне двійкове значення, враховуючи довжину в байтах, у десяткову систему числення.

Для подання дійсних чисел у Pascal та в інших мовах програмування існує дві форми.

Природна форма дійсного числа. Це звичайний запис дійсного числа, до якого всі звикли в школі. Тільки необхідно пам'ятати, що ціла частина дійсного числа відділяється від дробової крапкою, а не комою.

Наведемо кілька прикладів:

10.123, 1.0123, 1012.3, 0.0010123.

Усі ці числа різні за значенням, але значуща частина, або мантиса, у них однакова.

Показникова форма дійсного числа. Не завжди числа, з якими ви матимете справу при програмуванні, виглядатимуть так зручно. Подекуди це будуть дуже великі або дуже маленькі числа, для яких задавати зайві нулі не зовсім зручно.

Розглянемо приклад малого

$$0.00000000010123 = 1.0123*10^{-10}$$

або дуже великого числа

$$10123000000000000000 = 1.0123*10^{20}.$$

Отже, у показниковій формі дійсного числа можна виділити такі основні характеристики: ***знак числа, мантису числа, знак порядку та порядок числа.*** Решта елементів запису числа в показниковій формі повторюється - знак множення, основа степеня (10). Це означає, що зберігати їх у пам'яті комп'ютера немає ніякого сенсу. І справді, саме зазначені чотири характеристики дійсного числа зберігаються у пам'яті.

Ви, мабуть, чекаєте відповіді на запитання: «Це ж не запис числа, а подання його у вигляді арифметичних дій множення та піднесення до степеня?» Щоб уникнути цих арифметичних дій, дійсне число в показниковій формі у Pascal подається таким чином:

1.0123E–10.

У цьому записі символ Е означає основу степеня 10 і компілятор розпізнає цей запис як форму представлення дійсного числа. Характерно, що при додатних значеннях числа і мантиси знак «+» можна не вказувати.

Визначимо схематично основні характеристики дійсного числа:

1.0123 E – 15

знак числа мантиса знак порядку порядок

Структура Pascal-програми

Будь-яку мову програмування можна собі уявити як деякий засіб спілкування з комп'ютером. Тому, як і у будь-якій іншій мові спілкування, тут є певні домовленості у поданні своїх висловів: своя абетка та синтаксичні правила.

Синтаксично Pascal-програма складається з окремих «речень». Такими «реченнями» можуть бути деякі описи для самої програми або команди, з яких складається даний алгоритм. ***Кожне «речення» Pascal-програми повинно закінчуватися символом «;».***

Послідовність представлення алгоритму у вигляді Pascal-програми має певну закономірність. Означимо її так:

program <ім'я програми>;
uses <розділ опису бібліотек (модулів), що підключаються>;
label <розділ опису міток>;
const <розділ опису констант>;
type <розділ опису типів>;
var <розділ опису змінних>;
procedure або **function** <розділ опису процедур і функцій>;
begin <розділ операторів> **end.**

Ви, мабуть, звернули увагу, що службові слова Pascal-програми виділені жирним шрифтом (для зручності). Надалі ***службовими словами*** вважатимемо ключові слова, з яких складаються оператори Pascal та назви його стандартних процедур і функцій, тобто ті слова, які зарезервовані в самій мові Pascal і які ви не можете використовувати як ідентифікатори. Крім того, вся програма розбивається на певні розділи. Бажано

зберігати таку послідовність використання розділів, хоча з досвідом ви побачите, що деякі з них можна міняти місцями. Однак треба враховувати, що розділ опису модулів повинен обов'язково йти першим, а розділом операторів завершується кожна програма. Розділ типів обов'язково повинен передувати розділу змінних, а розділ міток і розділ констант бажано розміщувати перед розділом типів. Першим рядком програми зі службовим словом **program** можна знехтувати.

Умовно цю структуру можна поділити на дві частини - описову і виконувану.

Описова частина **(program, uses, label, const, type, var)** містить інформацію про те, які можливості ви будете використовувати у своїй програмі, які у вас будуть константи, мітки, якими змінними будете користуватися. Саме за змістом цієї описової частини всім указаним змінним відводиться місце в пам'яті комп'ютера за їх типами і послідовністю, в якій вони записані в програмі.

Виконувана частина містить опис процедур, функцій та основний блок програми, який ще називається тілом програми і розташований між службовими словами **begin** та end, тобто ті завдання, які повинна виконати дана програма.

Завершується Pascal-програма завжди символом **«.»**.

Як приклад наведемо програму, за допомогою якої можна обрахувати площу круга, вказавши будь-який радіус.

```
program circle;
  const pi=3.1415926;
  var r, s: real;
begin
  writeln ('Задайте радіус круга:');
  read (r);
  s:=pi*sqr(r);
  writeln ('Площа круга з радіусом ', r, 'дорівнює ', s)
end.
```

Зверніть увагу на принцип **«вкладеності»,** за допомогою якого досягається максимальна «читабельність» програми. Вміст розділу операторів зміщений трохи вправо відносно службових слів **begin** та **end.** Аналогічно записана й описова частина Pascal-програми. Кожне «речення» програми записується з окремого рядка. Оператори, які містять інші оператори, зміщуються відносно них так само вправо. Це дає змогу розібратися у складних програмах, знайти в них помилки не тільки самому автору, а й сторонньому користувачу.

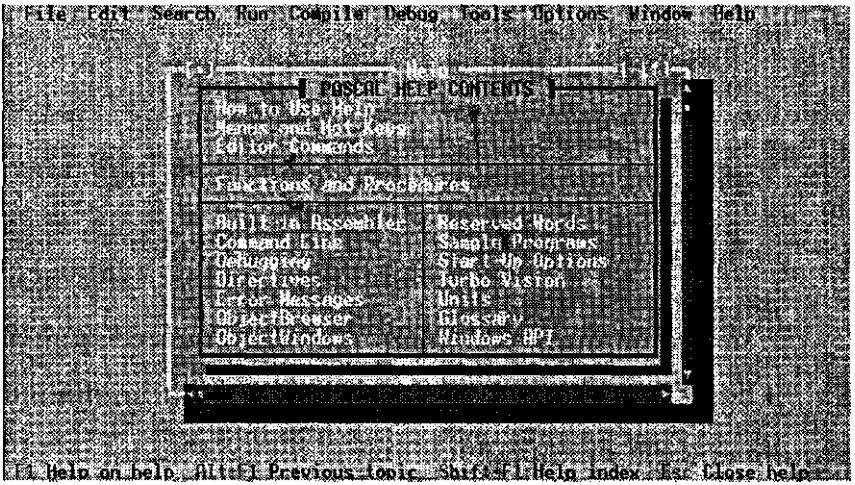
Здається, що ось-ось у вас виникне слушне запитання: «Як виконати написану програму за допомогою мови Pascal?» І справді, ви вже знаєте, що текст програми, написаний на аркуші паперу, сам не може бути виконаний. Для зручності роботи користувача з Pascal-програмами створено інтерактивне інтегроване середовище, яке об'єднало в собі можливості текстового редактора для набирання текстів програм, компілятора для визначення помилок у програмах та запуску програм на виконання в разі відсутності помилок, налагоджувача для покрокового виконання програм і виявлення складних помилок. Інтерактивним середовище називається тому, що воно працює в режимі постійного спілкування з користувачем, а інтегрованим - оскільки об'єднує в собі одночасно всі згадані вище можливості.

Робота в інтегрованому середовищі починається із запуску на виконання файла **turbo.exe.** Крім цього файла, на вашому диску бажано мати ще такі файли:

turbo.tpl - бібліотека стандартних процедур та функцій Turbo Pascal 7.0;

turbo.hlp — допомога користувачу середовища програмування Turbo Pascal 7.0.

Після запуску середовища Turbo Pascal 7.0 ви побачите у верхній частині екрана монітора головне меню, а в нижній - рядок повідомлень (мал. 2). Можливості середовища Turbo Pascal 7.0 дуже потужні, і розглядати їх усі в межах цього



Мал. 2. Екран інтегрованого середовища програмування Turbo Pascal 7.0

посібника немає ніякого сенсу. Тому зупинимося лише на основних і необхідних моментах.

Середовище Turbo Pascal 7.0 дає змогу користувачу працювати як за допомогою клавіатури, так і за допомогою «мишки». Для роботи з клавіатурою в назві кожного елемента головного меню іншим кольором виділені окремі літери, за допомогою яких можна активізувати те чи інше меню. Для цього достатньо натиснути комбінацію клавіш **ALT+«літера».** Наприклад, для активізації меню **File** треба скористатися комбінацією клавіш **ALT+F.** Якщо ж на вашому комп'ютері інстальована **«мишка»,** то достатньо підвести її курсор до необхідного елемента головного меню і натиснути на ній будь-яку кнопку.

Інтегроване середовище також надає можливість працювати з **«гарячими** клавішами» **(Hot key),** які активізують роботу найчастіше використовуваних команд. Наприклад, для запису тексту набраної програми на диск достатньо натиснути «гарячу клавішу» **F2** замість того, щоб використовувати команду меню **File.** Надалі можливе використання «гарячих клавіш» для виконання деяких команд середовища вказуватиметься в дужках.

Інтегроване середовище програмування Turbo Pascal **7.0** ще називають багатовіконним. Середовище надає користувачу змогу працювати одночасно із **100** вікнами, в яких може міститися різноманітна інформація. В кожний момент часу може бути ***активним*** лише те вікно, в якому ви працюєте. Слід зауважити, що практично кількість одночасно вікритих вікон залежить від об'єму оперативної пам'яті вашого комп'ютера, тому що інформація у вікнах під час сеансу роботи з середовищем зберігається саме в оперативній пам'яті.

Розглянемо пункти головного меню, не дотримуючись порядку, в якому ці пункти розташовані.

File — меню для роботи з файлами, які містять тексти програм. Після активізації цього елемента головного меню перед вами розгорнеться меню, яке міститиме такі команди (мал. 3).

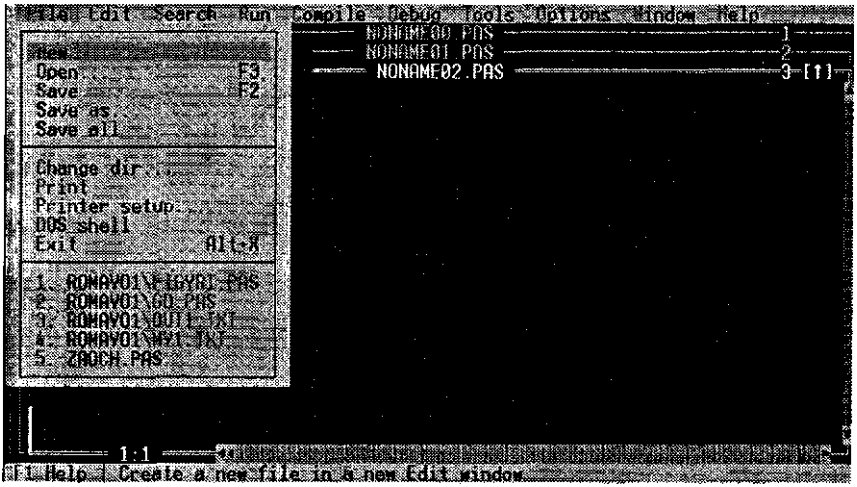
New - відкрити нове вікно для створення інформації (автоматично задається ім'я файла **NONAMExx.PAS,** де xx - порядковий номер відкритого вікна);

Open... (F3) - прочитати з диска для подальшої роботи текст збереженої раніше програми з поточного підкаталога, тобто з того, з якого було запущено середовище;

Save (F2) - зберегти вміст активного вікна в поточному підкаталозі;

Save as... — зберегти вміст активного вікна під новим вказаним ім'ям;

Save all - зберегти вміст усіх відкритих вікон;



Мал. 3. Меню File

Change dir... - зміна поточного підкаталогадля читання або запису файлів;

Print - виведення на пристрій друку вмісту активного вікна;

Get info... - інформація про розподіл інтегрованим середовищем пам'яті комп'ютера;

Dos shell - тимчасовий вихід в операційну систему для виконання дій на рівні ОС (повернення назад в інтегроване середовище здійснюється за допомогою системної команди **EXIT);**

Exit (ALT+X) - завершення сеансу роботи в інтегрованому середовищі.

Ви, мабуть, звернули увагу на те, що назви команд, які для свого виконання вимагатимуть від вас додаткової інформації, доповнені символами **«...»!**

Window - меню для роботи з вікнами (мал. 4). Під час роботи з цим елементом головного меню особливо відчутні переваги роботи з **«мишкою». Запропонований** нижче перелік можливостей даного елемента є неповним. Ми розглянемо лише найнеобхідніші з них на даний момент.

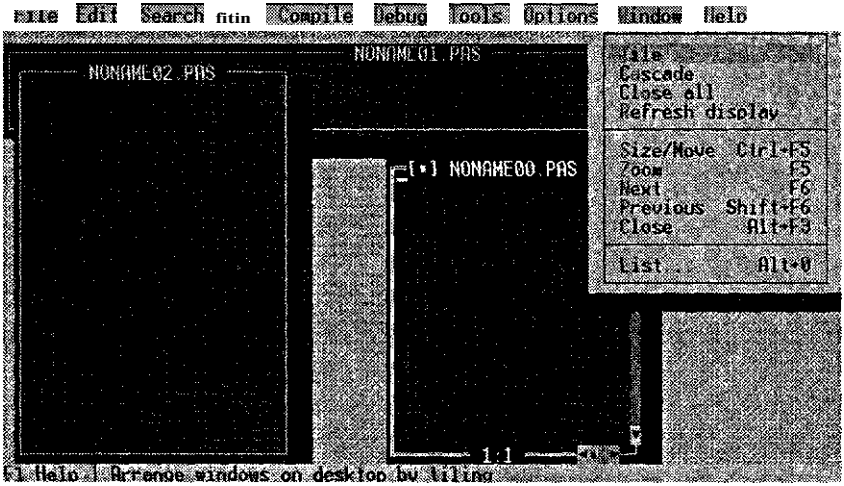
Зверніть увагу на те, що при роботі з багатьма вікнами активне вікно виділяється жирною подвійною рамкою.

Tile - рівномірний розподіл усіх відкритих вікон на екрані монітора;

Cascade - розподіл усіх відкритих вікон у вигляді каскаду, одне за одним;

Close All - закриття всіх активних вікон;

Size/Move (Ctrl+F5) - зміна розмірів та положення активного вікна. Для зміни розмірів вікна за допомогою «мишки»



Мал. 4. Вигляд середовища під час роботи з меню Window

необхідно нею «підчепити» вікно за правий нижній кут і рухати його в необхідному вам напрямі. Для зміни положення зменшеного вікна на екрані монітора необхідно «підчепити» його «мишкою» за верхню подвійну рамку і також рухати в потрібному напрямі;

Zoom (F5) - розгорнути вміст активного вікна на повний екран;

Next (F6) - активізація наступного вікна;

Previous (Shift+F6) - активізація попереднього вікна;

Close (Alt+F3) - закриття активного вікна;

List... (Alt+0) - список усіх відкритих вікон.

Debug - меню для роботи з налагодженням програми в активному вікні (мал. 5). Запропонований нижче перелік можливостей даного елемента також є неповним. Ми звернемо увагу лише на деякі з них, а до інших повернемося дещо пізніше.

Output - відкрити спеціальне службове вікно, в якому можна побачити результати роботи програми користувача;

User screen (Alt+F5) - відкрити «екран користувача», тобто перегляд результатів вашої роботи на комп'ютері поза середовищем, на повний екран.

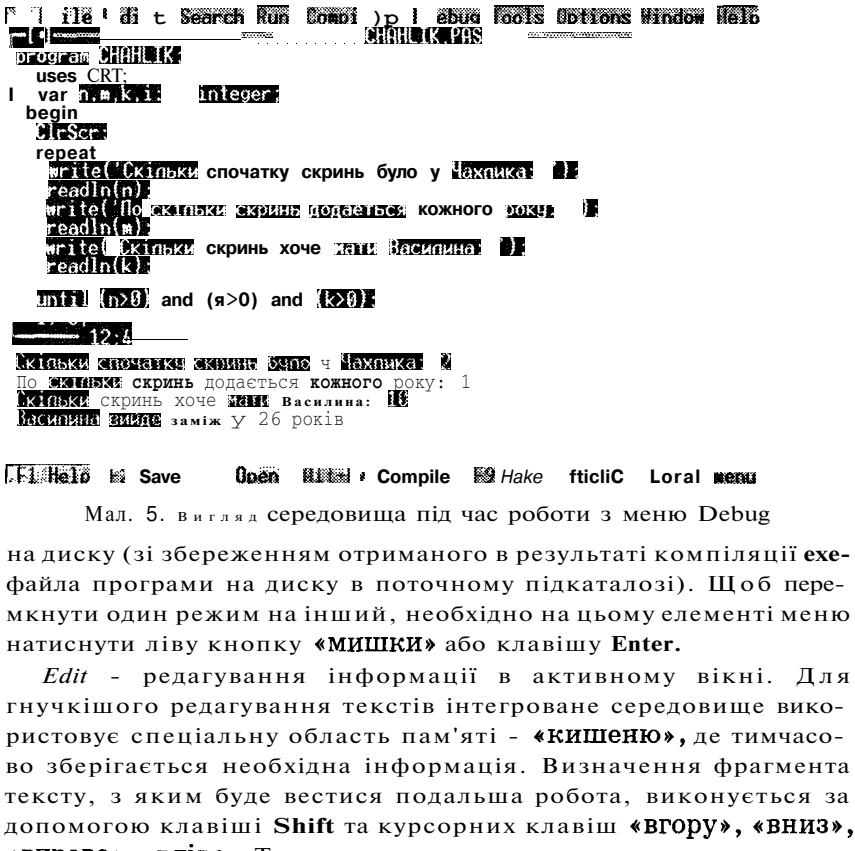
Зауваження. В інтегрованому середовищі Turbo Pascal 6.0 **Output** і **User screen** знаходяться в меню *Window*.

Run (Ctrl+F9) - запуск програми з активного вікна на виконання з одночасним пошуком синтаксичних помилок.

Compile - компіляція програми з активного вікна.

Compile (Alt+F9) - створення exe-файла програми, яка знаходиться в активному вікні;

Destination Memory (Disk) - компіляція програми в оперативній пам'яті комп'ютера без збереження результату компіляції



Мал. 5. Вигляд середовища під час роботи з меню Debug

на диску (зі збереженням отриманого в результаті компіляції **exe**-файла програми на диску в поточному підкаталозі). Щоб перемкнути один режим на інший, необхідно на цьому елементі меню натиснути ліву кнопку **«мишки»** або клавішу **Enter.**

Edit - редагування інформації в активному вікні. Для гнучкішого редагування текстів інтегроване середовище використовує спеціальну область пам'яті - **«кишеню»,** де тимчасово зберігається необхідна інформація. Визначення фрагмента тексту, з яким буде вестися подальша робота, виконується за допомогою клавіші **Shift** та курсорних клавіш **«вгору», «вниз», «вправо», «вліво».** Те саме можна виконати за допомогою «мишки», підвівши її на початок фрагмента тексту і натиснувши на її ліву кнопку та рухаючи її в необхідному напрямі. Зняти відмітку тексту можна за допомогою послідовного натиснення клавіш **Ctrl+K+H** або натисненням лівої кнопки «мишки» в будь-якому місці відміченого тексту.

Cut (Shift+Del) - вилучення відміченого фрагмента тексту з активного вікна і переміщення його в **«кишеню»;**

Copy (Ctrl+Ins) - копіювання відміченого фрагмента тексту з активного вікна в **«кишеню»;**

Paste (Shift+Ins) - копіювання інформації з «кишені» в активне вікно, починаючи з поточної позиції курсора;

Show clipboard - перегляд в окремому вікні вмісту «кишені» і можливість роботи з ним як зі звичайним текстом;

Clear (Ctrl+Del) - знищення відміченого фрагмента тексту.

На завершення пропонується невеликий сценарій створення й налагодження вашої першої програми в середовищі Turbo Pascal 7.0.

1. Запустити середовище Turbo Pascal 7.0 за допомогою файла **turbo.exe.**
2. Відкрити нове вікно за допомогою елемента меню *File* — **New.**
3. Набрати текст програми.
4. Зберегти текст набраної програми за допомогою елемента меню *File* — **Save as....**
5. Запустити на виконання програму з активного вікна за допомогою «гарячих клавіш» **Ctrl+F9.**
6. Виправити помилки, якщо вони будуть знайдені, і запустити програму на виконання ще раз.
7. Зберегти налагоджену програму за допомогою «гарячої клавіші» **F2.**
8. Переглянути результати роботи програми за допомогою «гарячих клавіш» **Alt+F5** і повернутися назад у середовище за допомогою клавіші **ESC.**
9. Створити exe-файл програми за допомогою «гарячих клавіш» **Alt+F9** (в режимі **Destination-Disk).**
10. Закрити вікно з виконаною програмою за допомогою «гарячих клавіш» **Alt+F3.**
11. Завершити сеанс роботи в інтегрованому середовищі програмування Turbo Pascal **7.0** за допомогою «гарячих клавіш» **Alt+X.**
12. Запустити на виконання одержаний exe-файл програми з операційної системи або з її оболонки.

Запитання для самоконтролю

1. Що таке величина з точки зору алгоритмізації? Які бувають величини?
2. Які стандартні типи величин ви можете назвати? Чому вони називаються стандартними?
3. Які дві форми подання дійсних чисел ви можете назвати?
4. У якому вигляді зберігаються в пам'яті комп'ютера дійсні числа?
5. Назвіть основні розділи Pascal-програми.
6. На які три основні частини можна поділити Pascal-програму? Яке їх призначення?
7. Що означає термін «інтерактивне інтегроване середовище»?
8. Які основні функції інтегрованого середовища Turbo Pascal 7.0?
9. Якими «гарячими клавішами» інтегрованого середовища зручно користуватися під час налагодження програми?

Виконайте завдання

24. Визначити тип результату обчислення таких виразів:

1) 1 + 0.0; **2)** 2 − 3; **3)** sin (0); **4)** 2*2.0;

5) $\frac{1}{2}$; **6)** 2*5; **7)** $\sqrt{4}$; **8)** [2.0].

25. Визначити тип результату обчислення таких виразів:

1) $x + y$, якщо x та y — цілі;
2) $x - y$, якщо x — ціле, y — дійсне;
3) $(x + y) * 10$, якщо x та y — цілі;
4) $\frac{x + y}{10}$, якщо x та y — цілі;
5) $\sin(x) + \cos(y)$, якщо x та y — цілі;
6) $\sin(x) * \cos(y)$, якщо x — ціле, y — дійсне;
7) $(x + 1) * (y - 2)$, якщо x та y — цілі;
8) $x^2 + 3 * x + 5$, якщо x — ціле;
9) $x^2 + 3\sqrt{x} + 5$, якщо x — ціле;
10) $\frac{x^2 + 3 * x + 5}{y}$, якщо x — ціле, y — дійсне.

ЛІНІЙНІ АЛГОРИТМИ

Арифметичні операції та арифметичні вирази

З математики вам відомо, що існує чотири основні арифметичні дії: додавання, віднімання, множення та ділення. Отже, сформулюємо цілком очевидне означення.

Арифметичними називатимемо такі вирази, які записуються за допомогою арифметичних операцій і в результаті обчислення яких одержуються числові значення.

Поки що це означення не дало ніякої нової інформації порівняно з математичними і фізичними. Це настільки очевидно, що взагалі незрозуміла його необхідність. Не кваптеся, трохи пізніше ви дізнаєтеся, що в Pascal використовують не лише арифметичні вирази.

У програмуванні, так само як і в математиці, існує пріоритет виконання арифметичних дій, тобто визначається, яким діям надається перевага перед іншими під час обчислення значення арифметичного виразу. Наведемо арифметичні операції мови Pascal саме в порядку зменшення їх пріоритетності:

***, /** - множення і ділення;
div - частка від ділення націло двох цілих чисел;
mod — остача від ділення націло двох цілих чисел;
+, — - додавання і віднімання.

Ви звернули увагу на те, що в Pascal відсутня операція піднесення до степеня? І, звісно, вам це здається дуже незручним. З часом до такого недоліку ви швидко звикнете і виявиться, що без цієї операції можна обійтися.

Перейдемо до ознайомлення з новими арифметичними операціями Pascal, з якими ви не зустрічалися в математиці.

Операції **div** та **mod** виконуються над величинами **тільки цілого типу.** Результатом виконання цих дій є також цілі числа. Принцип їх використання краще за все можна пояснити на прикладах.

10 div 3 = 3 - в результаті отримаємо відповідь на запитання, скільки разів число **3** входить у число **10.**

10 mod 3 = 1 - в результаті такої дії отримаємо відповідь на запитання, скільки залишиться від числа **10,** якщо вилучити з нього всі трійки.

Розглядаючи операції **div** та **mod,** ми вже торкнулися питання типів величин, над якими вони виконуються. Слід звернути увагу й на інші арифметичні операції.

Операції «*», «/», «+», «−» виконуються над величинами як цілого, так і дійсного типу. Питання лише в аналізі типу результату. Якщо операції «*», «+», «−» виконуються над величинами, які мають цілі значення, то й отриманий результат буде цілим числом. Якщо ж хоча б одна з величин, над якими виконується дія, матиме дійсний тип, то і результат буде дійсного типу. Виняток складає операція ділення «/». Якого б типу не були величини, над якими вона виконується, результат завжди буде дійсного типу.

Арифметичні вирази, як і всі інші конструкції мов програмування, вводяться з клавіатури підряд, в один рядок. Тому зрозуміла доречність використання дужок для задання необхідної послідовності виконання дій у виразі.

Розглянемо приклад арифметичного виразу в математичній формі та в Pascal:

$$\frac{a+b}{c+d} \Rightarrow (a + b)/(c + d).$$

А запису в Pascal $a + b/(c + d)$ відповідає така математична формула:

$$a + \frac{b}{c+d}.$$

Перевірте себе ще й на таких прикладах:

$$a + b/c + d, \quad (a + b)/c + d, \quad a + (b/c) + d.$$

В арифметичних виразах можуть використовуватися і стандартні функції. Деякі з них відомі з математики, наприклад: $\sin x$, $\cos x$, $\operatorname{tg} x$ і т. ін. Щоправда, у Pascal поняття функції ширше, але про це йтиметься пізніше. А поки що поговоримо про ***стандартні функції,*** тобто такі функції, які розуміє Pascal без додаткових пояснень та описів. Пріоритетність обчислення функцій найвища. Отже, якщо в арифметичному виразі вико-

ристовуються функції, то спочатку буде обчислене їх значення, а потім над цими результатами будуть виконані інші дії. На відміну від математики, серед стандартних функцій у Pascal є не лише тригонометричні. І ще одна домовленість - у мовах програмування аргументи функції вказуються в дужках. Тобто в математиці ви пишете sin x, а в Pascal треба писати sin *(x)*.

Наведемо список стандартних функцій мови Pascal із зазначенням типів їх аргументів і результатів.

sin (x) - синус[1];
cos (x) - косинус;
arctan (x) - арктангенс;
exp (x) - e^x (експонента);
ln (x) - $\ln x$ (логарифм натуральний);
sqr (x) - x^2;
sqrt (x) - $\sqrt{x}$.

Для вищенаведених функцій аргументами можуть бути цілі або дійсні значення, а результати завжди дійсні.

abs (x) – $|x|$ (модуль);

trunc (x) - ціла частина *x,* тобто відкидається дробова частина; *x* - дійсне, результат - цілий. Наприклад, **trunc** (-6.7) = -6.

frac (x) - дробова частина x; *x* - дійсне, результат - дійсний. Наприклад, **frac** (4.5) =0.5.

round (x) - заокруглення до цілого значення; *x* - дійсне, результат - цілий. Наприклад, **round** (4.2) =4, **round** (4.5) =5.

chr (I) - символ, порядковий номер якого в таблиці ASCII-кодів дорівнює I; *I* - ціле, результат - символ (тип **char**). Наприклад, **chr** (66) = 'B'.

ord (c) - порядковий номер вказаного символу; *c* - символ, результат – цілий. Наприклад, **ord** ('B') = 66, **chr** (**ord** ('C')) = 'C'.

pred (k) - попередній для *k* елемент у зчисленому типі (зчисленим називається такий тип, для елементів якого існує поняття «попередній» та «наступний», наприклад **integer, char**). В якості прикладів наведемо такі: **pred** ('B') = 'A', **pred** (10) = 9. Для першого елемента множини результат невизначений.

succ (к) - наступний для *k* елемент. Наприклад, **succ** ('B')='C', **succ** (**false**) = **true**. Для останнього елемента множини результат невизначений.

odd (x) – визначення непарності x; *x* - ціле, результат – **true** або **false**. Наприклад, **odd** (2) = **false, odd** (5) = **true**.

int (x) - повертає цілу частину аргументу. Наприклад, **int** (3.7) = 3, **int** (–3.7) = –3.

Pi – повертає значення числа л; аргументів немає.

[1] Аргументи тригонометричних функцій задаються в радіанах.

Тепер час навести математичну формулу, за допомогою якої можна піднести будь-яке число до будь-якого степеня:

$$x^y = e^{\ln x^{*y}}.$$

Це відповідає такому запису в Pascal:

$$\exp(\ln(x)^{*}y).$$

Виконайте завдання

26. Скільки арифметичних операцій містить вираз:

1) *(x + 1/2)*(y + 7/10) - 3/4;*
2) *(x + y)/(x - y)/x*y;*
3) *(x + y)* mod *(x - y)* div 10;
4) sqr *(x* mod *y* div *y))/x - y;*
5) trunc (sin *((x - y)/x*y)* div 10?

27. Перевести формулу у вигляд, доступний для програмування:

1) $a + bc$; 2) $x + \frac{2x - 10}{y + 2}$; 3) $10^5\alpha(\beta - 10^{-3})$;

4) $\frac{x - x_0}{x_1 - x_0} - \frac{y - y_0}{y_1 - y_0}$ 5) $\{a[b + (c + d)] - bc\}$; 6) $\cfrac{a}{x + \cfrac{b}{2! + \cfrac{c}{5! + d}}}$.

28. Записати арифметичний вираз у вигляді звичайної алгебраїчної формули:

1) *a*b*c - a/(n*m)/(a + 6/2);*
2) *a*b/(c + d) + (c - d)/b*(a + b);*
3) *(a/(1 + b/(2*x)) + c)/(1 - sqr (sqrt (a)));*
4) *x/(1 + sqr (x)/(2 + x*sqr (x)/3));*
5) *(x*sin (y + 2*(2 + ft)) + 1)/(0.7 - x/(y + sqr (z)))*x/(a + b).*

29. Вказати порядок виконання операцій під час обчислення значення простого арифметичного виразу:

1) sqr *(x)* + sqr (sqr *(y)*);
2) *x*sqr (y)/sqv* (2);
3) *a*b/c*d;*
4) sqrt *(a)* + 1 - sqr (6);
5) *- x*sqr (y)/sqr* (sqr *(z)).*

30. Нехай $A = 5$, $B = -4$, $C = 3$ та $P = 0.5$. Обчислити значення простого арифметичного виразу:

1) *(A + B)/C*P;* 2) *(A + B)/C/P;*
3) *(A + B)/(C*P);* 4) *-A*B* + sqr (C) + 0.5.

31. Нехай задано вектор *a*(1, 2, 3, 4, 5, 6), компоненти якого перенумеровані, починаючи з одиниці. Обчислити значення простого арифметичного виразу:

$(a[i + 2] + a[2])/\text{sqr}(a[2*i+1])*(-2)$

для $i = 2$ та $i = 4$.

32. Обчислити значення простого арифметичного виразу:

$$(a[1] + a[i*2])*a[5 - i]$$

для $i = 2$ та вектора $a(2, -5, 0, 4, 6.3, -4, 6)$, компоненти якого пронумеровані, починаючи з -2.

Оператор присвоювання

Зрозуміло, якщо у програмі буде обчислене значення арифметичного виразу, то його необхідно десь запам'ятати для подальшого використання. Для цього існує оператор присвоювання.

Загальний вигляд оператора:

<ім'я змінної> := <вираз>.

Операцію присвоювання можна ще назвати операцією заміщення. Дію n := m можна прокоментувати таким чином: *значення змінної n повинно бути замінене поточним значенням змінної m.*

Знак «=» треба відрізняти від знака «:=». Перший означає порівняння, умову, яку можна перевірити. Тобто логічно використовувати його зі знаком запитання: «*n* = Другий знак «:=» означає дію, яку можна виконати.

Під час виконання цього оператора спочатку обчислюється значення виразу в правій частині при поточних значеннях змінних, що входять до нього, а потім отриманим результатом замінюється попереднє значення змінної, що вказана зліва. Отримане значення записується в ту частину оперативної пам'яті комп'ютера, яка виділена для цієї змінної.

Під час виконання операції присвоювання важливим є збіг типів змінної величини в лівій частині оператора і виразу, що обчислюється в правій його частині. Це пояснюється відображенням виконання оператора присвоювання на пам'ять комп'ютера: типи повинні бути однаковими.

Наприклад:

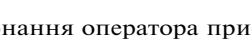
```
p := (a + b + c)/3;
a := sin (x) + sin (y);
d := sqrt (sqr (b) -4*a*c).
```

Вирази, що беруть участь в операторі присвоювання, можуть бути не лише арифметичними. Наприклад, можливі й такі присвоювання:

```
cn := 'a';
text := 'алгоритм';
flag := true.
```

Перший приклад присвоює змінній значення типу **char,** другий - типу **string,** третій - типу **boolean.** Пізніше розглядатимемо питання обчислення виразів типу **boolean.**

Операція збільшення а на 1 позначається таким чином: а := а + 1. Її слід читати так: «*а* замінити на *а* + 1». Оскільки це дуже важливо розуміти, щоб не робити зайвих помилок, продемонструємо роботу саме цього оператора схематично (мал. 6).



Мал. 6. Схема виконання оператора присвоювання а := а + 1

З малюнка бачимо, що спочатку в області пам'яті, яка відведена змінній *а,* знаходилося поточне значення 4. Виконання оператора а := а + 1 відбувалося поетапно таким чином:

- під час обчислення виразу справа взяли поточне значення змінної *а* (число 4) і до нього додалося число 1;
- отримане значення 5 було поміщене в ту область пам'яті, що відведена змінній *а*. При цьому попереднє значення змінної *а* втрачається.

Під час повторного виконання оператора а := а + 1 змінна *а* отримає вже значення 6.

Цікаво розглянути такі дві послідовності операторів присвоювання:

1) m := п, п := к;
2) п := к, m := п.

Ці послідовності операторів присвоювання різні щодо збереження значення змінної *п*. У першому випадку значення змінної *п* було збережене у змінній *т*. У другому випадку значення змінної *п* було втрачено перш, ніж його можна було використовувати для заміщення змінної *т*. Таким чином, остання послідовність операторів присвоювання фактично аналогічна такій: п := к, т := к.

Стандартні процедури введення/виведення інформації

Для створення найпростішої програми необхідно ознайомитися з можливістю введення аргументів та виведення результатів виконання програми.

У мові Pascal для цього існують уже готові, а отже, стандартні процедури.

Загальний вигляд процедури введення інформації:

read (<список елементів введення>)

або

readln (<список елементів введення>).

Елементами введення можуть бути один або кілька ідентифікаторів змінних, записаних через кому.

Наприклад:

read (a, b, c);

read (alfa);

readln (m, n).

Під час виконання процедур **read** та **readln** програма переходить у стан очікування введення даних. Причому процедури **read** та **readln** при введенні інформації з клавіатури мало чим відрізняються одна від одної. Поки що ви можете помітити лише те, що після введення інформації за допомогою процедури **readln** курсор переходить на початок нового рядка, а після роботи процедури **read** - ні. Їх суттєвіша відмінність буде визначена під час ознайомлення з файлами у мові Pascal.

Зауваження:

- якщо введення інформації передбачає зразу кілька значень, то їх можна вводити в одному рядку, відділяючи одне від одного символом «пробіл», або в окремих рядках, завершуючи введення кожного значення клавішею Enter;
- робота процедур не завершиться, доки не будуть введені значення для всіх змінних;
- змінні отримують свої значення послідовно в процесі введення даних (перше значення - першій змінній, друге - другій і т. д.);
- тип значень, що вводяться, повинен відповідати тому типові, яким описана дана змінна, інакше при введенні інформації буде видано повідомлення про помилку.

Як ви думаєте, яке значення отримає змінна *a* після виконання процедури **read** (a, a, a), якщо буде набраний такий рядок значень: 10 – 51? Ви не помилилися: змінна *a* отримає значення 1 (з третього разу)!

Загальний вигляд процедури виведення інформації:

write (<список елементів виведення>)

або

writeln (<список елементів виведення>).

Список елементів виведення інформації набагато ширший. У ньому можуть використовуватися:

- ідентифікатори змінних, значення яких будуть виведені на екран монітора;

- вирази, значення яких спочатку будуть обчислені, а потім виведені на екран монітора;
— сталі величини (числові, символьні, рядкові).

Різниця між процедурами **write** та **writeln** така сама, як і для процедур введення: після виведення інформації процедурою **write** курсор опиняється в наступній позиції після останнього виведеного символу, а після процедури **writeln** - на початку нового рядка.

Приклади:

```
write ('Задайте значення трьох цілих чисел');
write (a + b,' - така отримана сума двох чисел');
write ('a =',a,' b =',b);
writeln ('a =',10).
```

Зверніть увагу на те, що всі коментарі є рядковими сталими величинами і їх треба брати в апострофи. Причому, зрозуміло, що апострофів у списку елементів виведення має бути парна кількість (третій приклад). Якщо текст коментарю передбачає використання апострофів, то замість одного апострофа в тексті необхідно поставити два ('комп''ютер').

Оскільки елементами процедур введення інформації не можуть бути коментарі, треба обов'язково використовувати перед ними процедури виведення інформації з відповідними коментарями для того, щоб програма не була «німою», щоб вам було зрозуміло, скільки даних та які саме необхідно задати для її виконання.

У четвертому прикладі на екран монітора завжди буде виводитися один і той самий вираз, незалежно від того, яке значення змінна *a* отримуватиме у програмі: *a* = 10.

А як ви думаєте, що виведе ваша програма у третьому прикладі, якщо величинам *a* та *b* не будуть задані значення? Арифметичний вираз буде обчислено, але в якості значень *a* та *b* буде взяте з пам'яті комп'ютера так зване «сміття», тобто те що містилося там у той момент, коли ця частина пам'яті розподілялася для цих змінних. Це можуть бути залишки інформації попередньої програми, яка знаходилася на цьому місці, або випадкові нулі, якщо ви тільки увімкнули комп'ютер і в цю область оперативної пам'яті ще ніяка інформація не була внесена. Згадайте це зауваження, якщо буде ситуація, коли програма під час кожного запуску видає непередбачувані результати!

І ще одна можливість процедур виведення інформації, яку можна використовувати для отримання результатів у зручному, не показниковому вигляді. Це форматування значень змінних.

Вигляд форматування:

<ім'я змінної>: n : m,

де *n* - загальна кількість символів, що відводиться під значення змінної, *am* — кількість знаків дробової частини.

У загальну кількість символів входить і знак числа, і крапка, що відділяє дробову частину числа від цілої. Цей тип форматування, звісно, задається тільки для дійсних чисел. Для форматування цілих чисел параметр *m* зайвий.

Для більшої зрозумілості наведемо декілька прикладів з поясненнями.

```
write ('сума =', S:10:6);
write ('кількість елементів:', n:5);
writeln ('a =', a:0:3,' b =', b).
```

У першому прикладі при $S = -101.697$ буде виведено на екран монітора

сума = - 101.697000,

у другому при цілому $n = 57$ -

кількість елементів:_____57,

у третьому при $a = 10.2458$ та $b = -132.879$ -

a = 10.246 b = - 1.32879E + 02 .

Із цих прикладів можна зробити такий висновок. Під час форматування у дробовій частині за наявності зайвих зарезервованих позицій у кінці числа дописуються нулі, а за їх нестачі відбувається округлення останньої можливої цифри. У цілій частині надрукується стільки цифр, скільки необхідно для даного числа, навіть якщо є нестача позицій, або перед цілою частиною буде виведена необхідна кількість пробілів, якщо зазначено позицій більше, ніж потрібно. За відсутності форматування значення дійсних чисел будуть надруковані в показниковій формі.

Скориставшись цією інформацією, ви швидко оціните зручність форматування вихідних даних!

Використання процедур і функцій модуля CRT у лінійних програмах

Монітори персональних комп'ютерів можуть працювати у двох режимах - текстовому й графічному. Після запуску програми на виконання автоматично задається текстовий режим роботи, при якому на екран може виводитися 80 символів у 25 рядках. З графічним режимом роботи монітора ознайомимося пізніше. Але й для текстового режиму існує багато цікавих додаткових можливостей. Усі вони реалізовані у вигляді процедур і функцій, розміщених у модулі **CRT**.

У Pascal під модулями розуміють бібліотеки, в яких розміщені процедури та функції, що виконують роль допоміжних алгоритмів. Вони містяться в окремих файлах і підключаються до програми тільки тоді, коли їх імена вказані

у розділі **uses.** Це дає змогу не «перевантажувати» Pascal-програму зайвою інформацією!

Модулі (серед них і модуль **CRT),** що використовуються найчастіше, поміщені у файл **turbo.tpl (.tpl** - Turbo Pascal Library). Тому перед використанням можливостей цього модуля перевірте наявність файла **turbo.tpl** на диску.

Для підключення модуля **CRT** до програми необхідно до неї ввести рядок **uses CRT:**

```
program my_prog;
  uses CRT;
begin

end.
```

Наведемо список основних процедур модуля **CRT.**

ClsScr - очищення екрана монітора;

GoToXY (x, y) - виставлення курсора в позицію *x* рядка *y* ($1 \leqslant x \leqslant 80,\ 1 \leqslant y \leqslant 25$);

TextColor (color) - встановлення кольору тексту;

TextBackGround (color) – встановленя кольору фону символів;

Delay (Ms) — затримка виконання програми на вказаний у мілісекундах інтервал часу (спробуйте **Delay** (200));

Sound (Hz) - увімкнення звукового сигналу вказаної частоти у герцах;

NoSound - відключення звукового сигналу.

Залишилися невідомими значення параметрів процедур для роботи з кольорами. Наведемо перелік їх значень:

0	**Black**	чорний
1	**Blue**	синій
2	**Green**	зелений
3	**Cyan**	голубий
4	**Red**	червоний
5	**Magenta**	фіолетовий
6	**Brown**	коричневий
7	**LightGrey**	яскраво-сірий
8	**DarkGrey**	темно-сірий
9	**LightBlue**	яскраво-синій
10	**LightGreen**	яскраво-зелений
11	**LightCyan**	яскраво-голубий
12	**LightRed**	рожевий
13	**LightMagenta**	бузковий
14	**Yellow**	жовтий
15	**White**	білий
128	**Blink**	блимання символу

Для зазначення необхідного кольору ви можете користуватися або числовими значеннями, або їх мнемонічними альтер-

нативними позначеннями - вони передають одне й те саме числове значення. Pascal вас зрозуміє!

Те, що ми з вами розглянули, називається **процедурами.** Вони дуже схожі на оператори Pascal, бо записуються окремими «реченнями» та завершуються символом **«;».** Серед них є такі, яким для виконання необхідно передати параметри, а інші виконуються без будь-якої додаткової інформації.

Вам, напевно, доводилося писати заяви. Давайте складемо програму, яка це робитиме за вас.

```
program statement;
  uses CRT;
  var name_direct, my_name, data: string;
begin
  writeln ('Введіть ім''я директора організації');
  readln  (name_direct);
  writeln ('Введіть своє ім''я');
  readln (my_name);
  writeln ('Введіть дату подачі заяви');
  readln (data);
  ClrScr;
  TextColor (Red);
  Sound (500);
  GoToXY (50, 1);
  write ('Директору', name_direct);
  GoToXY (50,2);
  write (my_name);
  GoToXY (35, 6);
  write ('З А Я В А');
  GoToXY (5, 8);
  write ('Прошу прийняти мене,' ,my_name,' на роботу.');
  GoToXY (5, 10);
  write (data);
  Delay (500);
  NoSound
end.
```

Зверніть увагу на те, що значення змінних величин типу **string** задаються лише процедурами **readln,** а після використання процедур **GoToXY** ідуть процедури **write,** адже після встановлення курсора в потрібне місце екрана процедура **writeln** перемістить його на початок наступного рядка.

А тепер розглянемо дві цікаві функції модуля **CRT.**

KeyPressed - повертає значення **true,** якщо натиснена будь-яка клавіша, та **false** — у протилежному випадку;

ReadKey - повертає значення символу, клавіша з яким була натиснена.

Ви вже помітили, чим відрізняються функції від процедур? Процедури зразу виконують передбачені в них дії, а функції отримують певні значення, з якими щось необхідно зробити.

Отже, з'ясуємо, як відбувається введення інформації з клавіатури. Клавіатура, як і інші периферійні пристрої, має свій буфер, тобто виділену частину оперативної пам'яті. Саме в ньому послідовно «збираються» коди тих клавіш, які ви натискаєте. Робота стандартних процедур **read** і **readln** організована таким чином, що після натискання клавіші **Enter** уся накопичена в буфері інформація певним чином перетворюється і присвоюється черговій змінній. Після цього буфер клавіатури очищається і готовий для отримання нової інформації.

Повернімося до наших функцій. Функція **ReadKey** зчитує з буфера клавіатури одиницю інформації, що міститься першою в буфері, й отримує значення натиснутої клавіші. Якщо перед виконанням цієї функції буфер був порожнім, то програма переходить у режим очікування, поки у буфер щось буде занесено. У цьому випадку після виконання функції **ReadKey** буфер знову стає порожнім. Якщо ж під час виконання функції **ReadKey** буфер не був порожнім, то він спорожниться тільки тоді, коли функцію виконають стільки разів, скільки інформації є у буфері. Саме цією особливістю можна скористатися, щоб затримати виконання програми до натискання будь-якої клавіші. Цією особливістю зручно користуватися наприкінці програми, щоб мати можливість проаналізувати отримані результати, перш ніж середовище Turbo Pascal перейде в режим роботи з текстом вашої програми.

```
program myprog;
  uses CRT;
  var k: char;
begin
  { тіло програми }
  k := ReadKey;
end.
```

Виконання програми можна затримати, використавши процедуру **readln** без параметрів:

```
program myprog;
begin
  { тіло програми }
  readln;
end.
```

Особливість використання такої «затримки» полягає в тому, що процедура **readln** реагує лише на натискання клавіші **Enter**!

І ще один, третій варіант організації «затримки» роботи вашої програми. Це використання функції **KeyPressed**. Для пояс-

нення доведеться забігти вперед. Річ у тому, що функція **KeyPressed** нічого з буфера не вичитує, а лише перевіряє, чи там є щось. Скориставшись цією особливістю, запишемо оператор:

repeat until KeyPressed.

Прочитати цей запис можна таким чином: «повторювати, доки не буде натиснена якась клавіша». Доки буфер порожній, функція **KeyPressed** повертатиме значення **false,** і ми будемо «висіти» на цьому фрагменті програми. Але як тільки буде натиснено будь-яку клавішу, функція **KeyPressed** поверне значення true і зазначену дію буде завершено.

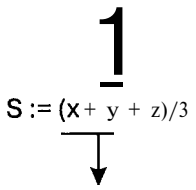
Отже, варіант використання функції **KeyPressed:**

```
program my_prog;
  uses CRT;
begin
  { тіло програми }
  repeat until KeyPressed;
end.
```

До речі, інформація в буфері після використання функції KeyPressed залишиться, тобто ця функція лише перевіряє наявність інформації в буфері клавіатури. Це означає, що всередині програми такий спосіб «затримки» треба використовувати дуже обережно і лише тоді, коли ви переконані, що перед цією операцією буфер клавіатури був порожнім. Можливо, вам колись стане в нагоді порада, яким чином спорожнити буфер клавіатури. Для цього можна використати такий фрагмент програми:

```
program empty;
  uses CRT;
  var k: char;
begin
  { тіло програми }
  while KeyPressed do
    k := ReadKey;
  { продовження тіла програми }
end.
```

Наведемо приклад програми, що обчислює середнє арифметичне значення будь-яких трьох чисел, та схему алгоритму (мал. 7), що їй відповідає:



Мал. 7

```
program my_prog;
  uses CRT;
  var x, y, z, s:  real;
      k:           char;
begin
  ClrScr;
  TextColor (Green);
  GoToXY (30,10);
  writeln ('Введіть три будь-які числа:');
  read (x, y, z);
  s := (x+ y + z)/3;
  GoToXY (30,15);
  write ('Середнє арифметичне чисел' , x:7:3, y:7:3, z:7:3);
  GoToXY (45,16);
  write (s:10:5);
  k := ReadKey;
end.
```

Запитання для самоконтролю

1. Що називають арифметичними виразами у Pascal?
2. Назвіть арифметичні операції Pascal в порядку спадання їх пріоритетності.
3. Над якими величинами можна виконувати операції **div** та **mod?**
4. Які правила запису арифметичних виразів у Pascal?
5. Як на машинному рівні виконується оператор присвоювання?
6. Яким чином організоване введення та виведення інформації у Pascal?
7. Для чого у Pascal використовується форматування змінних?
8. Що розуміється під модулями у Pascal?
9. Як використовують функції та процедури модуля?
10. У яких двох режимах можуть працювати Pascal-програми?
11. Назвіть основні функції та процедури модуля **CRT.**

Не припускайтеся помилок!

У цих розділах спробуємо застерегти вас від типових помилок, яких припускаються майже всі початківці[1].

І ще одне зауваження. Частина помилок виникатиме під час компіляції програми - це переважно помилки синтаксичного характеру. В результаті компіляції програми буде вказано на розташування таких помилок. Коли ж усі вони будуть виправлені, можуть з'явитися помилки на етапі виконання програми - це вже арифметичні або ж логічні помилки. Їх аналізувати важче, але можливо.

Якщо під час компіляції програми було видано повідомлення про помилку:

Error 4: Duplicate identifier,

це може означати, що ім'я програми та деякої змінної однакові.

[1] Тут і далі наведено перелік помилок за версією Turbo Pascal 7.0.

Якщо компілятор видасть повідомлення про помилку:

Error 3: Unknown identifier,

це означає, що вказане ім'я змінної не описане в розділі змінних.

Якщо тип вказаної компілятором змінної і тип значення, яке ви намагаєтеся їй присвоїти, не збігаються, буде видане повідомлення про помилку:

Error 26: Type mismatch.

Якщо ви були неуважні і в арифметичному виразі не вистачає дужок, то компілятор Pascal може видати повідомлення про таку помилку:

Error 89: Unexpected «(»

або ж

Error 89: Unexpected «)».

Подібне повідомлення про помилку також видається, якщо десь не вистачає символів «;» або «:=».

У разі непарної кількості апострофів у процедурі write компілятор видає повідомлення про помилку з таким коментарем:

Error 8: String constant exceeds line,

що означає «рядкова константа перевищує розміри рядка».

Будьте уважні під час введення дуже довгих рядків. На такі ситуації компілятор реагує повідомленням про помилку:

Error 11: Line too long.

Якщо в кінці програми немає крапки, то буде видане повідомлення про помилку:

Error 10: Unexpected end of file,

що означає «несподіване закінчення файла».

Якщо під час компіляції програми не розпізнається ім'я будь-якої процедури або функції з модуля **CRT,** то в першу чергу перевірте, чи підключений цей модуль.

Якщо на етапі виконання програми, тобто за повної відсутності в ній синтаксичних помилок, буде видане повідомлення про помилку:

Runtime error 106,

це означатиме, що введені дані не збігаються з описаним типом. Наприклад, треба було ввести числове значення, а ви випадково ввели символ і натиснули клавішу **Enter.**

Подібне повідомлення про помилку, але з іншим кодом, буде видано в разі, коли ви захочете виконати ділення на вираз, що набуває значення 0, або добути корінь з від'ємного числа. У таких ситуаціях треба ретельно проаналізувати початкові дані.

Вправи

33. Нехай задані такі значення дійсних змінних x $= 0.5$ та $h = 0.1$. Які значення матимуть ці змінні після виконання таких операторів присвоювання:

1) $x := 2.5$; 2) $x := x + 2*h$; $h :=$ Л/2;
3) $x := x +$ Л; 4) A := -A; x := x +A; A :=Л*2?

34. Є три цілочислові змінні з поточними значеннями $A = 3$, $B = 5$, C $= 7$. Які значення матимуть ці змінні в результаті виконання таких послідовностей операторів присвоювання:

1) $C := A*B$ 2; $B := B$ 1; $A :=$ C - (B);
2) P := C; C := B; B := A; $A := P$;
3) $B := B + A$; $C :=$ C B;
4) $A :=$ $A+$ 1; $B :=$ A + B; C :=A + B?

35. Задані дійсні змінні x та y з поточними значеннями x = 0.8, $y =$ - 2.1 та цілочислові змінні A та l із поточними значеннями A $= 5$, $l = -3$. Знайти значення, якого набуде відповідна змінна в результаті виконання кожного з наведених нижче операторів присвоювання із зазначенням його типу:

1) A :=x; 2) A := $(k+ 1)/10$;
3) $l :=$ 5 $+y$; 4) $x := (k+ 1)/10$;
5) A :=A/3; 6) $l := x+ y$;
7) x :=A; 8) x :=x $+y$;
9) y :=A +x; 10) $l := x+$ 0.9

36. Чому дорівнюватимуть значення змінних x та y після виконання таких дій:

$x := 2$, $y := 5$, $x := y$; $y :=$ x?

37. Задати за допомогою оператора присвоювання такі дії:

1) змінній z присвоїти значення, що дорівнює півсумі значень змінних x та y;
2) подвоїти значення змінної a;
3) значення змінної x збільшити на 0.1;
4) за нове значення змінної $a1$ прийняти її поточне значення, піднесене до квадрата;
5) змінити знак у значенні змінної q.

38. Змінній a присвоїти значення, що дорівнює сумі значень змінних x та y, а змінній b - подвоєному їх добутку.

39. Визначити, за допомогою якої послідовності дій можна поміняти місцями вміст змінних x та y:

1) $x :=$ xy; x :=y;
2) $x := y$; x :=y - x; $y := x - y$;
3) $y :=$ xy; $x' :=$ y - x; y := y -x;
4) x := $x + y$; y := x -y; x :=x - y.

40. Записати у вигляді оператора присвоювання обчислення виразу $y = 2x^3 + 3x^2 + x + 5$, не використовуючи при цьому операції піднесення до степеня.

Серйозні розважалки

41. Якщо на одну шальку терезів посадити Даринку, яка важить n кілограмів, і Наталку, яка важить на **5** кілограмів менше, а на іншу - насипати m кілограмів цукерок, то скільки кілограмів цукерок доведеться з'їсти нещасним дівчаткам, щоб шальки терезів зрівноважилися?

42. Невдаха-учень Сашко сів виконувати домашнє завдання і просидів за столом **2** год. З них x хв він чухав потилицю і дивився у вікно, y хв шукав у письмовому столі гумку, щоб стерти у підручнику з англійської мови карикатуру на свого товариша, на малювання якої він витратив перед цим z хв. Решту часу Сашко перекладав англійські слова. Скільки слів він устиг перекласти, якщо на переклад одного слова у нього йшло **5** хв?

43. Петрусь задумав число і нікому його не назвав. Друзі упіймали його і примусили подвоїти задумане число, а потім додати до нього **5.** І тільки після того, як вони пообіцяли Петрусеві благодійну допомогу на контрольній з математики, він зізнався, що вийшло число n. Визначте, яке число задумав і приховав від своїх друзів Петрусь.

44. Курочка Ряба знесла яєчко, а мишка розбила його. Після цього Ряба знесла на K яєчок більше, але мишка знову їх розбила. Ряба піднатужилася і знесла знову на K яєчок більше, ніж попереднього разу, але безсовісна мишка розтрощила й ці. Так продовжувалося п'ять разів, поки Ряба не здалася. Зі скількох яєць Дід і Баба змогли б врешті-решт зробити собі яєчню?

45. Із тераріуму втекли x гадюк, y кобр та z гюрз. Довжина кожної гадюки - **1** м, кожної кобри - **1** м ЗО см, а гюрзи - **1** м **15** см. Скільки повних метрів отруйних змій утекло з тераріуму? Яку довжину вони складають у сантиметрах?

46. У цариці Несміяни кругле обличчя, радіус якого дорівнює R. Визначте довжину сторони такого квадратного дзеркала, щоб, коли Несміяна милується собою, її відображення поміщалося в дзеркалі.

47. Чепуруха Катруся, взявши ножиці в руки, змоделювала собі з маминого капелюшка радіуса R капелюшок нового фасону - з квадратними полями. Якою має бути сторона квадратної коробки для нового капелюшка?

48. Надіслати вітальну телеграму собі на день народження від президента країни.

49. Створити програму, яка виводила б на екран монітора лист вашому найзапеклішому ворогові, задаючи його ім'я з клавіатури.

Задачі

50. Дано два дійсні числа a і b. Знайти їх суму, різницю і добуток.

51. Дано два дійсні числа x та y. Обчислити значення виразу:

$$\frac{|x|-|y|}{1+|xy|}$$

52. Дано довжину ребра куба. Знайти його об'єм та площу всієї поверхні.

53. Дано два дійсні числа. Знайти їх середнє арифметичне і середнє геометричне значення.

54. Дано катети прямокутного трикутника. Знайти його гіпотенузу і площу.

55. Визначити час падіння каменя на поверхню Землі з висоти h.

56. Квадрат заданий довжиною сторони. Визначити:

1) довжину вписаного в нього кола;
2) довжину описаного навколо нього кола;
3) площу вписаного в нього круга;
4) площу описаного навколо нього круга.

57. Дано довжину кола. Визначити площу круга, обмеженого цим колом.

58. Дано гіпотенузу і один із катетів прямокутного трикутника. Знайти другий його катет і площу вписаного круга.

59. Знайти площу кільця, внутрішній радіус якого дорівнює 10, а зовнішній - даному числу $r (r > 10)$.

60. Зростаючу арифметичну прогресію задано першим членом a, кроком h та загальною кількістю членів n. Знайти:

1) суму арифметичної прогресії;
2) k-й елемент арифметичної прогресії.

61. Зростаючу геометричну прогресію задано першим членом a, коефіцієнтом q та загальною кількістю членів n. Знайти:

1) суму геометричної прогресії;
2) k-й елемент геометричної прогресії.

62. Обчислити відстань між двома точками з координатами $(x_1; y_1)$ та $(x_2; y_2)$.

63. Трикутник задано координатами вершин. Знайти:

1) периметр трикутника;
2) площу трикутника.

64. Визначити координати проекцій кола з центром у точці $(x; y)$ та радіусом r на осі координат.

65. Визначити швидкість автомобіля через час t, який рушив із місця з прискоренням a.

66. Визначити, яку роботу необхідно виконати, щоб підняти тіло масою m на висоту h від поверхні Землі.

67. Визначити значення рівнодійних сил, що діють на тіло масою m, яке рухається з прискоренням a ($= ma$).

68. Визначити значення опору R в колі електричного струму за заданими значеннями сили струму I і напруги U ($R = U/I$).

69. Визначити, на якій відстані від місця кидання впаде тіло, кинуте під кутом a до горизонту з початковою швидкістю v_0.

70. Визначити, в який момент часу опиниться на висоті h тіло, кинуте вертикально вгору з початковою швидкістю v_0.

71. Визначити період коливання математичного маятника, довжина якого дорівнює L.

72. Визначити опір електричного кола, якщо в ньому резистори R_1, R_2, R_3, R_4 з'єднані:

1) послідовно;

2) паралельно.

73. Визначити, яку платню одержить на фірмі сумісник за виконану роботу, якщо йому нараховано S гривень, а податок становить 20 %.

74. Підприємство поклало в банк на депозитний рахунок суму в S тисяч гривень під 40 % річних. Яку суму зніме підприємство в кінці року?

75. У класі - N учнів, з них M - хлопці. Знайти співвідношення у відсотках кількості хлопців та кількості дівчат у цьому класі.

76. Скласти програму, що подає звуковий сигнал з частотою H Гц та тривалістю T мс.

77. Дано значення змінних x, y, z. Обчислити значення змінної t:

1) $t = \dfrac{x+y}{x - 1/2\,xy} - \dfrac{x-z}{}$; 2) $t = x^{y^{z+2}} + x^{y^2}$;

3) $t = (1 + z)^{\frac{y}{1}}$; 4) t $1 + \cfrac{}{3 + \cfrac{(2x)^2}{5 + (3x)^2}}$

78. Дано значення змінних x, y, z. Обчислити значення змінних a та b:

$$1 + x^2 \,|y + \operatorname{tg} z|$$

3) $a = (1 + y)^{\dfrac{x + y/(x^2+4)}{e^{-x-2} + 1/(x^2+4)}}$ $\quad b = \dfrac{1 + \cos(y-2)}{x^4/2 + \sin^2 z}$

4) $a = y + \dfrac{x}{y^2 + \dfrac{x^2}{y + x^3/3}}$ $\quad b = (1 + \operatorname{tg}^2 z);$

5) $a = \dfrac{2\cos(x - \pi/6)}{1/2 + \sin^2 y}$ $\quad 6 = 1 + \dfrac{}{3 + z^4/5}$

$$\frac{1 + \sin^2(x+y)}{2 + |x - 2x/(1+}$$ $\quad b = \cos^2(\operatorname{arctg}(1/z));$

7) $a = 1\text{п}\left(y - \sqrt{|x|}\right)$ x- $_{2 + x^2/2}$ $\quad b = x\dfrac{x}{3!} + \dfrac{x}{5!}$

79. Скласти програму, результатом виконання якої є виведення на екран монітора заданого тексту. Значення, вказані в «<...>», повинні задаватися користувачем:

1)

Директору <школа>
<прізвище директора>
<прізвище заявника>

ЗАЯВА

Прошу зарахувати мого сина (дочку) <прізвище учня> до <клас> Вашої школи.
<дата>

<прізвище заявника>

2) Шановні телеглядачі!

З технічних причин вихід в ефір передачі під назвою <назва телепередачі> переноситься з <дата 1> на <дата 2>. Просимо вибачення за створені незручності.

Дирекція телеканалу

3) Шановний(а) <прізвище батьків>!

<Дата> в приміщенні актового залу школи відбудуться батьківські збори на тему <тема зборів>.

Дирекція школи

РОЗГАЛУЖЕНІ АЛГОРИТМИ

Логічні вирази. Обчислення значень логічних виразів

Крім арифметичних виразів, у Pascal існує ще один тип виразів - **логічний.**

Логічним виразом називається такий вираз, внаслідок обчислення якого одержується логічне значення типу true або false («істина» або **«хиба»).**

Із стандартним типом змінних Boolean, які можуть набувати лише двох значень True або False, ви вже ознайомилися. Отже, саме такий тип і мають результати обчислення логічних **виразів.**

Логічні вирази поділяються на ***прості*** та ***складені.***

Простим логічним виразом називається вираз, який записується за допомогою знаків співвідношень **<, >, <=, >=, =** та **<>.**

Приклади простих логічних виразів можуть здатися вам простими:

$$a + b > c + d,\ n <> m,\ x - y.$$

Порівняйте тепер призначення символів **«:=»** та **«=»!**

Зверніть увагу на те, що спочатку виконуються арифметичні дії, а вже потім порівняння отриманих результатів.

Складеним логічним виразом називається вираз, в якому використовуються логічні операції and, or, not **(«так», «або», «ні»).**

Наведемо приклади. З математики вам відомі такі записи:

$$x \in [a, b] \text{ та } x \notin [a, b].$$

Спробуємо **записати їх** у вигляді логічних виразів

(x >= a) and (x <= b),

(x < a) **or** (x > b).

Під час запису складених логічних виразів прості логічні вирази обов'язково слід брати у круглі дужки!

Чи можна записати простий логічний вираз n <> m у вигляді складеного? Виявляється, можна:

not (n = m).

Визначимо правила, за якими обчислюються значення складних логічних виразів. Для цього існують таблиці істинності

(табл. 4-6), в яких цифра 0 означає **false,** а цифра 1 **- true.** Наведені таблиці можна перефразувати таким чином.

Таблиця 4

A	*B*	*A* **and** *B*
0	**0**	0
0	1	0
1	0	0
1	1	1

Таблиця 5

A	*в*	*A* **or** *B*
0	0	0
0	1	1
1	0	1
1	1	1

Таблиця 6

A	**not** *A*
0	1
1	0

***Логічна операція* and *дає результат* true *тоді і тільки тоді, коли обидва операнди мають значення* true.**

***Логічна операція* or *дає результат* true *тоді, коли хоча б один операнд має значення* true.**

Логічна операція* not *завжди дає результат, протилежний значенню Ті операнда.

Ми вели розмову про обчислення значень логічних виразів. Зрозумілим є запитання: «А де їх можна використовувати?»

По-перше, використання логічних виразів (як і арифметичних) можливе в операторі присвоювання.

Наприклад:

```
logical_1 := a > b;
logical_2 :=(N <=x) and (x <=M);
flag := false.
```

Умовою безпомилкового виконання таких операторів є збігання типів, тобто змінні в лівій частині цих операторів повинні бути описані типом **boolean.**

По-друге, результат обчислення логічних виразів **true** та **false** можна ще трактувати як «так» та «ні». Це наводить на думку про використання логічних виразів для визначення оцінки деякої ситуації, що склалася, і прийняття рішення про те, що робити далі.

Оператор умовного переходу. Повна та скорочена форми

Уявіть собі, що ви за кермом автомобіля і перед вами стоїть вибір дальшого руху: або їхати поганою, але коротшою дорогою, або ж гарною, але довшою. Звичайно, що ваш вибір буде залежати від певних умов: по-перше, чи є у вас зайвий час, по-друге, хто господар автомобіля?

Подібну проблему завжди вирішують оператори розгалуження.

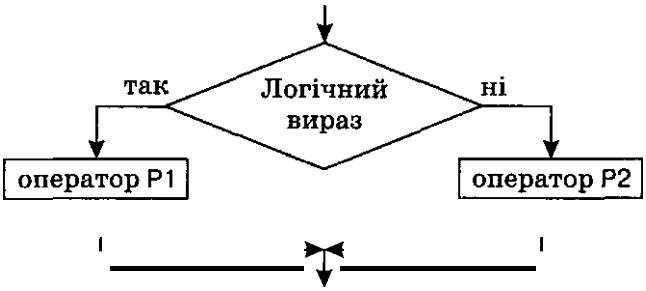
Загальний вигляд повної форми оператора умовного переходу:

if <логічний вираз> **then** P1 **else** P2,

де логічний вираз - це вираз, який може набувати одного з двох значень **true** або **false,** P1 та P2 - це оператори або процедури.

Робота оператора умовного переходу не викликає ніяких труднощів. Цей оператор використовує результат обчислення логічного виразу для вибору того чи іншого шляху наступного виконання алгоритму - виконання оператора Р1 або оператора Р2. Після цього робота алгоритму продовжується далі за вказаними операторами.

Схема алгоритму повного оператора умовного переходу показана на малюнку 8.



Мал. 8

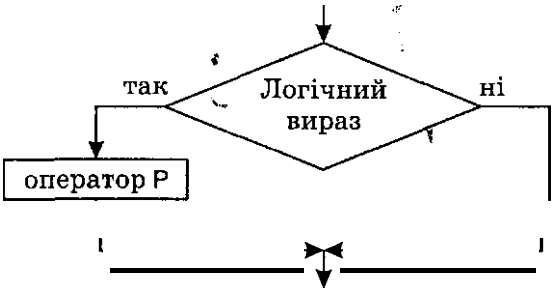
Схема алгоритму наочно демонструє, що після аналізу значення логічного виразу буде вибраний лише один з наступних напрямків виконання алгоритму (Р1 або Р2), після чого цей алгоритм буде виконуватися далі.

Загальний вигляд скороченої форми оператора умовного переходу:

if <логічний вираз> **then** Р,

де значення вказаних параметрів такі самі, як і в повній формі.

Схема алгоритму скороченої форми оператора умовного переходу дуже схожа на попередню (мал. 9).



Мал. 9

На схемі алгоритму добре видно відмінність між двома формами умовного оператора: в першій - повній - незалежно від значення логічного виразу якісь дії обов'язково будуть виконані, а вже потім продовжено виконання алгоритму далі, у

другій - скороченій - у випадку, коли логічний вираз набуде значення **true,** будуть виконані якісь дії, а потім продовжено виконання алгоритму, а у випадку, коли логічний вираз набуде значення **false,** алгоритм відразу буде продовжено далі.

Складений оператор

Розширимо поняття оператора в Pascal. Справа в тому, що ми з вами поки що мали справу лише з ***простими*** операторами - присвоювання і умовного переходу. А що робити, коли після службових слів **then** або **else** нам потрібно вказати не один такий оператор, а кілька? Для такого випадку в Pascal введене поняття ***складеного*** оператора.

Складеним оператором називають послідовність кількох операторів або викликів процедур, розділених символом «;»та взятих в операторні дужки **begin ... end.**

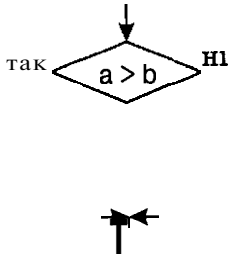
Тепер уже час переходити до прикладів. Розглянемо алгоритм пошуку найбільшого з двох заданих чисел *A* та *B*.

```
program max_A_B;
  var a, b, max: real;
begin
  write ('Задайте два будь-які числа: ');
  readln (a, b);
  if a > b
    then
      begin
        writeln ('Перше число більше за друге.');
        max := a
      end
    else
      begin
        writeln ('Друге число більше або дорівнює першому.');
        max := b
      end
  writeln ('Це число –', max:10:5);
  readln
end.
```

Розглянемо детальніше наведену програму. *По-перше,* у ній чітко спостерігається принцип **«вкладеності»** операторів, тобто сходинкова структура. Наочність такої програми явно виграє! Для цього в ній навіть з'єднані вертикальними лініями оператори різного рівня. *По-друге,* перед закриваючою операторною дужкою **(end)** не стоїть символ **«;».** І справді, ви ж не ставите кому в тексті перед закриваючою дужкою, коли перелічуєте в ньому в дужках декілька слів. Хоча в Pascal це не є

помилкою. Саме через це не поставлено і символ «;» після останньої процедури **readln.**

Схема алгоритму цієї програми виглядатиме так, як зображено на малюнку 10.



Мал. 10

Спробуйте відповісти на запитання: при виконанні якої умови буде виконано оператор max := b? Дійсно, за умовою, протилежною a > b, тобто при виконанні умови a <= b!

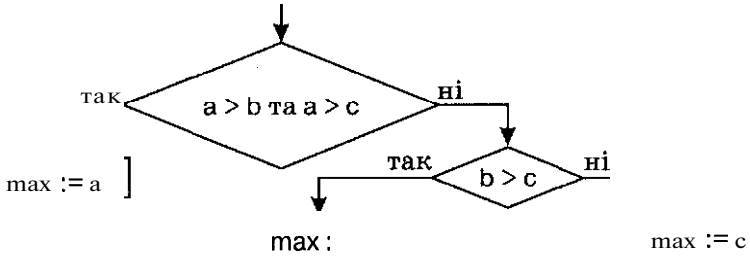
Настав час прикладу, який продемонструє використання складених логічних виразів. Розглянемо пошук найбільшої з трьох попарно різних величин *a, b, c.*

```
program max_A_B_C;
  var a, b, c, max: real;
begin
  write ('Задайте три будь-які числа: ');
  readln (a, b, c);
  if (a > b) and (a > c)
    then
      begin
        writeln ('Перше число більше за інші два.');
        max := a
      end
    else
      if (b > c)
        then
          begin
            writeln ('Друге число більше за інші два.');
            max := b
          end
        else
          begin
            writeln (Третє число більше за інші два.');
            max := c
          end;
  writeln ('Це число - ', max:10:5);
  readln
end.
```

У цьому прикладі ми бачимо використання двох вкладених умовних операторів.

Внутрішній оператор умовного переходу буде виконано тоді і тільки тоді, коли значення логічного виразу зовнішнього умовного оператора матиме значення **false**. Це означатиме, що значення змінної a не є найбільшим, тому у вкладеному умовному операторі перевіряється лише значення змінної *b*. Якщо ж і її значення не є найбільшим, то лишається визнати, що найбільшим є значення змінної *c*.

За традицією розглянемо схему цього алгоритму (мал. 11).



Мал. 11

Спробуємо записати інший варіант алгоритму пошуку найбільшої з трьох величин.

```
program max_A_B_C;
  var a, b, c, max: real;
begin
  write ('Задайте три будь-які числа: ');
  readln (a, b, c);
  if (a > b) and (a > c)
    then
      begin
        writeln ('Перше число більше за інші два.');
        max := a
      end;
  if (b > a) and (b > c)
    then
      begin
        writeln ('Друге число більше за інші два.');
        max := b
      end;
  if (c > a) and (c > b)
    then
```

```
        begin
          writeln (Третє число більше за інші два.');
          max := c
        end;
    writeln ('Це число - ', max:10:5);
    readln
end.
```

Запишіть схему цього алгоритму самі та порівняйте обидва варіанти алгоритмів задачі.

Отже, у першому алгоритмі внутрішній оператор розгалуження виконається тільки тоді, коли нас «пропустить» до нього зовнішній оператор розгалуження. Цей принцип можна порівняти із ситом.

У другому варіанті алгоритму всі оператори розгалуження є послідовними. Тобто всі вони виконуватимуться, перевірятимуть значення своїх логічних виразів і вирішуватимуть, виконувати чи ні вказані в них дії. Крім того, треба зауважити, що використані тут оператори розгалуження мають скорочену форму.

Звичайно, що з погляду ефективності виконання алгоритму перевагу має перший варіант - у ньому може бути виконано менше перевірок для досягнення необхідного результату. Але, зрештою, все залежить від конкретної умови задачі і вашого бачення її реалізації. Десь ви виграєте на етапі виконання програми, але програєте в її написанні, а десь програма буде зрозумілішою, але кількість виконуваних дій у ній буде більшою.

Оператор вибору

Для зручнішого запису вкладених розгалужень використовують оператор вибору, або, як ще кажуть, оператор варіанта.

Загальний вигляд повної форми оператора вибору:

```
case <вираз> of
  <значення_виразу_1>: P1;
  <значення_ виразу_2>: P2;

  <значення_ виразу_N>: PN;
  else PN+1
end;
```

де <значення_виразу_1> ... <значення_виразу_N> - перелік значень указаного в операторі виразу, при яких повинні виконуватися відповідні оператори P_1, ... P_N.

В операторі вибору на використання типів змінних величин накладається певне обмеження. Змінні величини, що використовуються для запису виразів в операторі case, можуть бути лише зчисленого типу!

З поняттям зчисленого типу ми вже зустрічалися. До таких типів належать типи, для елементів яких існує поняття «наступного» та «попереднього» елемента. Серед відомих стандартних типів до зчислених належать усі типи, що характеризують цілі числа (**integer, shortint, longint, byte, word**), символьний (**char**) та логічний (**boolean**) типи. До зчислених типів не можна віднести змінні дійсного типу (**real**) та рядкового (**string**). Адже, справді, неможливо сказати, яке наступне число іде за 1.012 або яке слово передує слову «алгоритм». Нам ще доведеться ширше ознайомитися із зчисленими типами у Pascal - це дуже потужна особливість цієї мови програмування.

Специфіка виконання оператора вибору полягає в тому, що перевірка значення виразу з указаними значеннями змінних в операторі відбувається до першого збігання. Решта значень уже не перевіряється. Якщо ж не відбулося жодного збігання, то виконається оператор, указаний після службового слова **else**.

Загальний вигляд скороченої форми оператора вибору відрізняється лише відсутністю службового слова **else**:

```
case <вираз> of
   <значення_виразу_1>: P1;
   <значення_виразу_2>: P2;

   <значення_виразу_N>: PN
end;
```

Щоб краще зрозуміти використання оператора вибору та пояснити деякі його особливості, розглянемо такий приклад. Нехай за заданими числовими значеннями днів тижня треба вивести інформацію про те, робочий це день чи вихідний.

```
program day;
  var n: integer;
begin
  write ('Задайте порядковий номер дня тижня: ');
  readln (n);
  case n of
     1..5 : writeln ('Це робочий день ');
     6, 7 : writeln ('Це вихідний день ')
     else writeln ('Це не день ')
  end;
  write('тижня')
end.
```

Виявляється, що в операторі **case** можна через кому вказувати зразу кілька значень або інтервал значень, при яких виконується даний оператор. І ще зверніть увагу, як можна комбінувати тексти, що виводяться на екран монітора - оператор **write ('тижня')** знаходиться після оператора **case** і тому виконається у будь-якому випадку.

Оператор безумовного переходу. Мітки

Для визначення поняття оператора безумовного переходу ознайомимося з поняттям мітки.

Міткою називається описаний в розділі міток **Label** ***ідентифікатор або беззнакове число від 0 до 9999.***

Мітками помічаються виконувані у програмі дії або службові слова **end.** Після кожної мітки повинен стояти символ **«:».** Кожна мітка у програмі повина бути унікальною. До речі, мітки 0010 та 10 вважаються однаковими!

Наведемо приклади міток:

Label StopLabel, 10, 909, Label_1, Label_2.

Тепер розглянемо **загальний вигляд оператора безумовного переходу:**

goto <мітка>.

Використання оператора безумовного переходу можна продемонструвати на прикладі захисту програми від уведення недопустимих даних.

Нехай за умовою задачі треба задавати лише додатні значення змінних *x, y, z*. Фрагмент програми виглядатиме приблизно так:

```
label my_label;
var x, y, z: real;

begin
myjabel: write (' Задайте три додатні числа: ');
readln (x, y, z);
   if (x <=0) or (y <=0) or (z <=0)
      then
         begin
            writeln (' Ви помилилися, повторіть: ')
            goto myjabel;
         end;
{ продовження програми }
```

У цьому прикладі використано можливість запису в програмі своїх власних коментарів. Для цього використовуються фігурні дужки **«{...}»,** тобто все те, що взято у фігурні дужки, у Pascal-програмі не обробляється, ігнорується. Цією можливістю буде вручно користуватися згодом під час налагодження програм для тимчасового відключення виконання деяких її фрагментів.

Л тепер підіб'ємо сумний підсумок усього сказаного про оператор безумовного переходу: ***користування оператором*** goto ***вважається у програмістів проявом поганого тону!***

У Pascal є багато можливостей обійтися без оператора безумовного переходу і в наступному розділі ми цьому навчимося.

Загальний вигляд порожнього оператора:

Тобто якщо ви в програмі випадково поставите підряд **«;;»,** то це компілятором Pascal не буде вважатися помилкою, оскільки між цими двома символами він розпізнає порожній оператор! Але хіба лише для цього потрібен порожній оператор? Інколи виникає необхідність передати керування на невиконувану частину програми, якою є, наприклад, службове слово **end.** А оскільки мітку можна ставити лише на виконуваній частині, то для цього існує порожній оператор:

```
label my_label;

begin
  { початок програми }
goto myjabel;
  { продовження програми }
myjabel: ;
end.
```

Запитання для самоконтролю

1. Що називають логічними виразами? На які два типи вони поділяються?
2. Назвіть логічні операції.
3. Поясніть використання логічних операцій за таблицями істинності.
4. Яким чином організоване розгалуження у Pascal?
5. Чим відрізняються повна і скорочена форми оператора умовного переходу?
6. Намалюйте схеми алгоритмів обох варіантів розгалуження.
7. Що називають складеним оператором? Які правила його запису?
8. У чому полягає відмінність виконання вкладених та послідовно розташованих розгалужень?
9. Запишіть загальний вигляд і схему алгоритму повної форми оператора вибору.
10. Запишіть загальний вигляд і схему алгоритму скороченої форми оператора вибору.
11. Коли доцільне використання оператора вибору?
12. Що називають мітками, які правила їх запису, для чого вони використовуються?
13. Запишіть загальний вигляд оператора безумовного переходу і поясність принцип його роботи.
14. Що таке порожній оператор і в яких ситуаціях його використовують?

Не припускайтеся помилок!

Якщо ви використали в операторах розгалуження непарну кількість операторних дужок, то може бути видано повідомлення про одну з двох помилок:

Error 85: «;» expected

або

Error 113: Error in statement.

Якщо перед службовим словом else в операторі умовного переходу стоїть «;», то курсор встановиться на цьому службовому слові і буде видано повідомлення про помилку:

Error 113: Error in statement.

Якщо в операторах умовного переходу або вибору в окремих блоках відсутні деякі операторні дужки, то вам про це повідомить та сама помилка Error 113.

Якщо в операторі case змінна не є змінною зчисленого типу, то вам буде видане повідомлення про таку помилку:

Error 29: Ordinal type expected.

Якщо в операторі case вказано змінну символьного типу, то її значення треба взяти в апострофи. Наприклад, 'A', 'z', '0'. Якщо ви цього не зробили, то буде видано повідомлення про помилку щодо незбігання типів.

Якщо в кінці оператора case відсутнє службове слово end, то вам буде видано повідомлення про помилку:

Error 42: Error in expression.

Виконайте завдання

Вправи

80. Поставити у відповідність розташованим ліворуч виразам вирази, що розташовані праворуч:

1) **not** (x= y), a) $x \in [0; 1]$,
2) (x < y) **or** (x= y), b) $x \neq y$,
3) (x < 0) **or** (x > 1), c) $x \leqslant y$,
4) (x >= 0) **and** (x <= 1), d) $x \notin [0, 1]$.

81. Обчислити значення логічних виразів:

1) x < y при $x = 2.5$, $y = 0.1$;
2) a **and not** (b = c) при a = **false**, b = **false**, c = **true**;
3) **not** (a **and** b) **or** b = a при a = **true**, b = **false**;
4) (**not** a **and** (x < y)) **or** (x < 0) при $x = -0.1$, $y = 0.7$, a = **true**.

82. Записати у вигляді логічних виразів висловлювання, наведені нижче:

1) значення x не належить інтервалу (0; 1);
2) значення x належить відрізку $[-1; 0]$ або відрізку $[2; 5]$;

3) точка $M(x;y)$ лежить у другій чверті координатної площини;

4) точка $M(x;y)$ лежить усередині або на межі одиничного круга з центром у початку координат;

5) точка $M(x;y)$ не лежить на одиничному колі з центром у початку координат;

6) компоненти дійсного вектора $x(x_1, x_2, x_3)$ утворюють неспадну послідовність і всі вони невід'ємні;

7) $0 \leqslant A \leqslant 1{,}5;$

8) $3 > B \geqslant C > 0{,}1;$

9) $5 \leqslant A < 6 \leqslant B < 7{,}6.$

83. Математична логіка - це розділ математики, який вивчає методи встановлення істинності або хибності висловлювань. У логіці визначені такі операції: «і» (л) - кон'юнкція, або логічне множення; «або» (v) - диз'юнкція, або логічне додавання; «не» (¬) - інверсія, або заперечення. Ці логічні операції повністю відповідають логічним операціям «and», «or», «not», що використовуються в логічних виразах. Представити у вигляді логічних виразів такі записи:

1) $x \vee \neg y \wedge (x \vee y) \wedge y = z;$

2) $\neg x \wedge (a * b - c)d \leqslant \frac{\quad}{c} \vee x;$

3) $5 - 2a \neq - 2 \wedge x \wedge y \vee \neg((x \vee y) \wedge z);$

4) $3{,}47 \frac{\quad}{a} + b \geqslant (a - 8{,}96d)^2 \vee a^3 - c < 3b;$

5) $\neg(x \vee y \vee z) \wedge a > \sin c;$

6) $a > \frac{a}{b-c} + \frac{b(a+d)}{c} \vee (0{,}6a\ 0{,}3b)^2 \leqslant \frac{b}{a+b+c};$

7) $\frac{\quad}{a + \frac{\quad}{b+c}} < 0{,}1x \wedge y \wedge z;$

8) $\sin^2 a + \cos^2 b \neq \ln x - \operatorname{tg} y;$

9) $\frac{e \quad -e}{e} > y \wedge \operatorname{arctg} 80° \geqslant z^3.$

84. Записати за допомогою умовного оператора виконання таких дій:

1) дійсне значення x замінити його абсолютною величиною;

2) менше з двох дійсних значень x та y (або будь-яке з них, якщо вони рівні) замінити нулем;

3) найбільше з трьох попарно різних дійсних значень x, y, z зменшити на 1/2;

4) присвоїти змінній x значення 0, якщо її початкове значення належало відрізку [0; 2];
5) якщо значення змінної x від'ємне, то залишити його без змін, у протилежному випадку зробити його таким, як і значення y;
6) якщо поточне значення змінної j не більше за 10, то значення у збільшити на одиницю, в протилежному випадку продовжити виконання дій алгоритму.

85. Використовуючи логічні величини, записати за допомогою умовного оператора виконання таких дій:
1) якщо $A < C$ і $B < C$, то виконується умова $A = B$;
2) якщо $1 < B < 3$ або $-3 < B < -1$, то виконується умова $B^2 = 4$;
3) $A \leqslant$ B тільки за умови, що B $\leqslant 5$;
4) $A =$ B $= 5$ тільки за умови, що $C < 3 < E$;
5) $A \neq 3$ тільки за умови, що $B \neq 5$ та $6 \geqslant C > 3$.

Серйозні розважалки

86. Чебурашка вирішив купити килими, щоб застелити кімнату, в якій він мешкав разом з Геною. Розмірами їхньої прямокутної кімнати виявилися цілі числа a і b. Коли Чебурашка запитав у магазині, яких розмірів є квадратні килими в продажу, то продавець назвав йому ціле число c. Яку кількість килимів треба придбати Чебурашці, щоб накрити максимальну площу кімнати? Килими не можна накладати і підгинати. Визначити, яка площа кімнати буде ненакритою килимами. Передбачити ситуацію, коли розміри килима перевищують розміри кімнати.

87. На одному маленькому квадратному безлюдному острові зі стороною a м мешкали k потерпілих корабельну аварію Робінзонів. Чи не порушені їхні права на житло, якщо на кожного Робінзона повинно припадати S м^2? Скільком Робінзонам ще вистачить місця на острові, якщо поблизу трапиться нова аварія?

88. Іван Петрович одягнув нові штани і сів на щойно пофарбовану табуретку. На його штанах з'явилася квадратна пляма зеленого кольору, довжина сторони якої становила a см. Виявилося, що у хімчистку беруть одяг, плями на якому не більші за S см^2. Визначити, чи вдалося Іванові Петровичу врятувати свої штани.

89. Від річкового вокзалу відійшли одночасно у протилежних напрямах теплохід і турист. Теплохід рухався зі швидкістю v_1 км/год, а турист стежкою уздовж річки - зі швидкістю v_2 км/год. Якщо через N год турист передумає і вирішить поплисти річкою назад за теплоходом зі швидкістю v_3 км/год, то

чи встигне він підсісти на теплохід, який має за графіком зупинку через $У$ годин після початку руху і стоїть на цій зупинці Z годин? Зважати на те, що всі події відбувалися протягом однієї доби.

90. Жили собі дід і баба, і був у них город прямокутної форми. Довжина городу становила A м, а ширина - В м. Якось дід посварився з бабою і вирішив поділити город порівну. Тепер у діда квадратний город зі стороною С м, відрізаний скраю, а решта дісталася бабі. Визначити, чи не залишилася баба ошуканою та якої форми дістався їй город - прямокутної чи квадратної?

91. Трьом Товстунам подали на десерт кремові тістечка. Маса одного тістечка становила x кг, а маса Товстунів відповідно Перший Товстун з'їв n тістечок. Кожний наступний Товстун з'їдав у два рази більше від попереднього, але при цьому всі вони не могли з'їсти більше, ніж половина їхньої власної маси. Скільки тістечок з'їли Товстуни?

92. Чи впізнає себе дільник m у залишку після ділення на нього націлого числа n?

93. Щоб бути завжди чистою, людині необхідно x ($24 \leqslant x \leqslant 50$) шматків мила на рік. Якщо мити лише п'яти, то мила знадобиться у 12 разів менше, а якщо мити лише вуха - ще на один шматок менше. Скласти програму, яка за вибором користувача давала б відповідь, яку кількість шматків мила необхідно закупити на n років уперед, щоб:

1) митися повністю;
2) мити лише п'яти;
3) мити лише вуха;
4) мити п'яти і вуха.

94. Три міліціонери гналися прямою стежкою за одним злочинцем. Вусатий міліціонер біг зі швидкістю x км/год, пузатий міліціонер - на h_1 км/год швидше, а лисий міліціонер - ще на h_2 км/год швидше від пузатого. Злочинець тікав зі швидкістю y км/год. Пробігши n годин, злочинець заліз на березу і причаївся. А міліціонери, пробігши по m годин кожний без сніданку, обіду та вечері, зупинилися і всі троє подивилися вгору. Той, у полі зору якого (до 5 м) виявився злочинець, був дуже щасливим і заарештував його. Визначити, хто з міліціонерів був щасливим, а хто залишився сумним. Скільки годин до арешту просидів на березі злочинець? Яка відстань була між міліціонерами в момент арешту злочинця?

95. Велосипедист Микола, стартувавши з точки $(x_0; y_0)$ та рухаючись уздовж прямої $A(x - x_0) + B(y - y_0) = 0$, мріє про те, як він покатає на своєму велосипеді сусідку Катрусю. Чи здійсняться мрії Миколи, якщо недалеко, у точці $(p; q)$, росте дерево?

Задачі

Прості та вкладені розгалуження

96. Дано значення цілих величин x та *y*. Знайти:

1) max (x, *y*) +min (x, *y*);
2) $\max^2(x, y) - \min^2(x, y)$;
3) max (x^2, y) +max (x, y^2);
4) min *(x +y, x - y)*.

97. Дано значення дійсних величин *a*, *b*, *c*. Знайти:

1) max *(a + b + c, a*b*c)*;
2) min $((a + b + c)/2, \sqrt{1/(a+1)+1/(b^2+1)} + 1/(c^2+1))$;
3) max *(a +b, b +c)* +min *(a +c, b)*.

98. Дано дійсне число *x*. Вивести за зростанням числа *tg x*, $1 + |x|$ та $(1 + x^2)^4$.

99. Дано два дійсні числа *x* та *y*. Визначити *x/y*, передбачивши можливе ділення на нуль виведенням повідомлення «Ділення на нуль».

100. Дано значення дійсної величини x. Визначити:

$$x + 3$$

$$x^2 - 2x - 10$$

3) $\frac{x - 5}{x^3 + x - 2}$;

4) $\frac{x^4}{x^5 + x^3 + x + 1}$.

101. При даному значенні *x* обчислити:

1) $\sqrt{x + 101}$;
2) $\sqrt{x^2 - 2.5}$;
3) $\sqrt{x^3 + 5}$;
4) $\sqrt{|x - 10| - x^2}$;
5) $\sqrt{x^3 - \sqrt{x - 1}}$.

102. Дано дійсні значення *x* та *y*. Обчислити:

$$\frac{x^2 + y^3}{\sqrt{x + y}},$$

2) $\frac{\sqrt{x^3 - y + 0.5}}{x^2 - y^2}$

3) $\frac{\sqrt{x + 2}}{\sqrt{y - 1}}$;

4) $\sqrt{x^2 - y^2 + 5}^{x+y}$.

103. На площині дано дві точки $(x_1; y_1)$ та $(x_2; y_2)$. Визначити, яка з них лежить далі від початку координат.

104. На осі *Ox* визначено три точки з попарно різними дійсними координатами x_1, x_2, x_3. Вивести значення тих двох точок, між якими лежить третя.

105. Нехай деяка точка на площині задана дійсними координатами $(x; y)$. Відомо, що $x \neq 0$, $y \neq 0$. Визначити, в якій чверті координатної площини вона розташована.

106. Автомобіль подолав відстань S км через населений пункт за T хв. Визначити, чи не порушив водій правил дорожнього руху, якщо швидкість автомобіля при цьому не повинна перевищувати 60 км/год.

107. За рейтинговою системою оцінка визначається таким чином: якщо сумарний бал учня становить не менш як 92 % від максимального, то виставляється оцінка 12, якщо не нижче за 70 %, то виставляється оцінка 8, якщо ж не нижче за 50 %, то — оцінка 5, в інших випадках - оцінка 2. Визначити оцінку учня, якщо він набрав N балів, а максимальне значення сумарного балу становить S.

108. Відома така таблиця співвідношення ваги і зросту людини за віком: від значення зросту людини беруться останні дві цифри; якщо вік людини до 25 років, то її вага повинна становити на 5 кг менше від отриманого двоцифрового числа, для людей від 25 до 45 років - дорівнювати цьому двоцифровому числу, для старших за 45 років - двоцифрове число необхідно збільшити на 5. Задані цілі додатні числа P - зріст людини і N — її вік ($30 < P < 200$, $10 \leqslant N \leqslant 100$). Визначити рекомендовану вагу людини в цьому віці.

109. Визначити, чи є неспадною послідовність дійсних чисел $X, y, 2$.

110. Визначити, чи попадають значення цілих величин a, b, c у відрізок $[x; y]$, де x і y - цілі числа.

111. Визначити, що більше: середнє арифметичне чи середнє геометричне значень трьох цілих величин $x, y, 2$.

112. Дано значення дійсних величин a, b, c. Подвоїти ці значення, якщо $a \geqslant b \geqslant c$, і замінити їх абсолютними значеннями, якщо це не так.

113. Дано ціле число k і три дійсні числа $x, y, 2$. При $k < 0$, $k = 0$ або $k > 0$ замінити модулем відповідно значення x, y або z, а два інших значення зменшити на 0.7.

114. Перетворити задані значення двох дійсних змінних x та y за таким правилом: якщо значення x та y від'ємні, то кожне з них замінити його модулем, якщо від'ємне тільки одне з них, то обидва значення збільшити на 0.5, якщо ж обидва значення невід'ємні і жодне з них не належить відрізку [0.5; 2.0], то обидва значення зменшити в 10 разів. У інших випадках значення x та y залишаються без змін.

115. Дано значення трьох попарно різних дійсних змінних $x, y, 2$. Змінити їх значення, надавши їм найбільшого із заданих значень.

116. Дано дійсні числа *a, b, c, d.* Якщо вони утворюють спадну послідовність, то замінити їх значення модулями, якщо зростаючу, то все залишити без змін, в протилежному випадку — збільшити всі значення в 10 разів.

117. Дано три цілі додатні числа *x, y, 2.* Визначити, чи можна з відрізків із цими довжинами утворити трикутник.

118. Дано трикутник зі сторонами *a, Ь, c.* Визначити, який це трикутник: гострокутний, тупокутний чи прямокутний.

119. Залежно від розміру суми, розмір податку з неї розраховується за такою схемою:

1) якщо сума не перевищує деяку величину *a,* то податок не вираховується;
2) якщо сума більша за *a,* але не перевищує *Ь,* то податок становить 10 %;
3) якщо сума більша за *Ь,* але не перевищує *c,* то податок становить 25 %;
4) якщо сума більша за с, то податок становить 50 %.

Визначити, який податок буде вираховано із суми розміром *S.*

120. Дано натуральне число n ($n \leqslant 1000$). Визначити:

1) найстаршу цифру цього числа;
2) суму першої і останньої цифр;
3) порядок числа.

121. Дано дійсне число *x.* Не користуючись відповідними стандартними функціями Pascal, визначити:

1) цілу частину числа;
2) його округлення до найближчого цілого значення.

122. Дано ціле число *n.* Не користуючись відповідною функцією Pascal, визначити чи є це число парним.

123. Дано цілі числа *a, b, s, q* ($a \neq 0$). Визначити, чи буде при діленні націло a на ft остача s або *q.*

124. Дано натуральне число *n* ($n \leqslant 99$). Визначити, чи правильно, що n^2 рівне кубу суми цифр цього числа.

125. Дано натуральне число n ($n \leqslant 9999$). Визначити, чи є це число паліндромом, тобто таким числом, яке зліва направо і справа наліво читається однаково (усі чотири цифри враховуються; наприклад, 0110).

126. Дано натуральне число n ($n \leqslant 9999$). Враховуючи всі чотири цифри числа, визначити, чи правильно, що воно містить:

1) рівно три однакові цифри;
2) усі різні цифри;
3) дві пари цифр, що повторюються;
4) цифри, що утворюють неспадну послідовність;
5) цифри, що утворюють арифметичну прогресію в порядку їх запису в числі;
6) цифри, що утворюють геометричну прогресію в порядку їх запису в числі.

127. Дано цілі числа a і с, де $a \neq 0$. Знайти розв'язки рівняння:

1) $ax^2 + c - 10 = 0$;

2) $ax^2 + c/2 = 0$;

3) $2ax^2 - 5c = 0$.

128. Квадратний багаточлен заданий своїми коефіцієнтами a, b, c, де а $\neq 0$. Визначити:

1) чи корені відповідного рівняння є парними числами;

2) області додатних значень багаточлена;

3) області від'ємних значень багаточлена.

129. Два квадратні рівняння задано своїми коефіцієнтами a_1, b_1, c_1 та a_2, b_2, c_2, де $\neq 0$ і $a_2 \neq 0$. Визначити:

1) чи мають ці рівняння однакові пари коренів;

2) чи збігаються відношення між меншими і більшими коренями обох рівнянь;

3) чи утворюють у сукупності корені обох рівнянь арифметичну прогресію;

4) чи можуть бути корені обох рівнянь сторонами деякого прямокутника;

5) чи можуть бути корені обох рівнянь сторонами деякого чотирикутника.

130. Дано дійсні додатні числа a, b, c, x, y. Визначити, чи пройде цеглина з ребрами a, b, c у прямокутний отвір зі сторонами x та y. Проштовхувати цеглину можна лише таким чином, щоб кожне з її ребер було паралельним чи перпендикулярним кожній зі сторін отвору.

131. Трикутник на площині заданий координатами вершин. Визначити:

1) чи належить центр координат цьому трикутнику;

2) чи належить точка $(x; y)$ цьому трикутнику.

132. На площині задано три прямі коефіцієнтами рівнянь $(a_1; b_1; c_1), (a_2; b_2; c_2), (a_3; b_3; c_3)$ (пряма на площині задається рівнянням $ax + by + c = 0$). Визначити, чи можуть ці прямі утворити трикутник, і якщо так, то знайти координати його вершин.

133. Дано чотири числа, які визначають довжини відрізків a, b, c, d. Визначити, чи можна з цих відрізків утворити прямокутник.

134. Розробити діалогову програму, яка запитує ім'я користувача та його вік і визначає, до якої вікової категорії він належить:

1) від 1 до 10 років - дитина;

2) від 11 до 15 років - підліток;

3) від 16 до 20 років - юнак (юнка);

4) від 21 до 30 років - молода людина;

5) після 31 року - доросла людина.

135. Два відрізки прямих задані координатами своїх кінців $(x_1;y_1)-(x_2;y_2)$ та $(x_3;y_3)-(x_4;y_4)$. Визначити:

1) чи належать ці відрізки одній прямій;
2) чи збігаються ці відрізки хоча б одним зі своїх кінців та чи лежать вони на одній прямій, утворюючи при цьому новий відрізок; вказати координати кінців утвореного відрізка;
3) чи є ці відрізки частинами деякого третього відрізка і мають спільну частину.

136. Чотири відрізки прямих задані координатами кінців $(x_1;y_1)-(x_2;y_2)$, $(x_3;y_3)-(x_4;y_4)$, $(x_5;y_5)-(x_6;y_6)$ та $(x_7;y_7)-(x_8;y_8)$. Визначити, чи утворюють вони прямокутник і, якщо так, то яка його площа.

137. Три відрізки прямих задані координатами кінців $(x_1;y_1)-(x_2;y_2)$, $(x_3;y_3)-(x_4;y_4)$ та $(x_5;y_5)-(x_6;y_6)$ і не мають спільних точок. Визначити, чи лежать вони на одній прямій і, якщо так, то який із них є «внутрішнім» відносно двох інших.

138. Розмір мішені для стрільби задається координатами центра концентричних кіл (x; *y)* та відповідними радіусами $R_i=5i$, де *i* - 1, 2, 10. Спортсмен виконав 10 контрольних пострілів. Координати влучень у мішень задаються відповідними парами чисел $(x_j;y_j)$, де j - 1, 2, 10. Визначити, скільки очок набрав спортсмен за дану серію пострілів (вважається, що попадання на межу кілець зараховується як кращий результат).

139. На площину за допомогою трафарета наносяться послідовно чотири круги різних радіусів та кольорів. Їх місцеположення й розміри задаються відповідно координатами центрів $(x_1;y_1)$, $(x_2;y_2)$, $(x_3;y_3)$ та $(x_4;y_4)$ і радіусами R_1, R_2, R_3, R_4. Визначити:

1) чи можна їх повністю накрити кругом із центром у точці (x; *y)* та радіусом R;
2) які пари кругів мають спільні точки.

140. Дано дійсне число *a*. Обчислити $f(a)$, якщо

1) $f(x)=\begin{cases} x^2 & \text{при } -2 \leqslant x<2; \\ 10 & \text{в протилежному випадку;} \end{cases}$

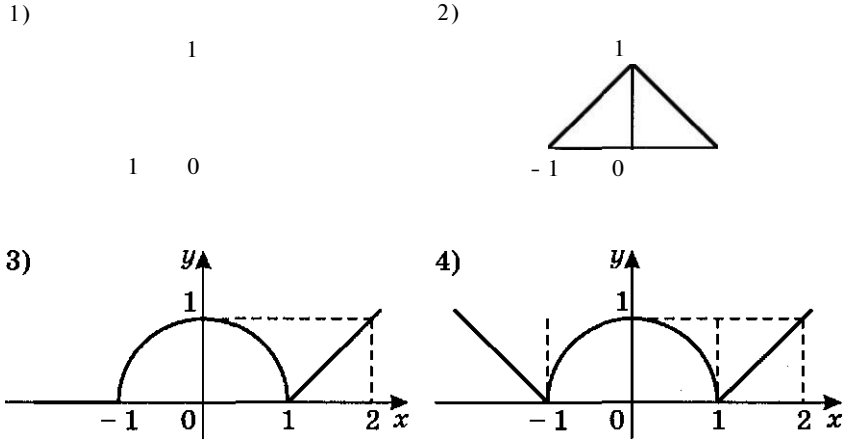
2) $f(x)=\begin{cases} \sin x+|x| & \text{при } x\leqslant 2; \\ (x^2+5)/2 & \text{в протилежному випадку;} \end{cases}$

3) $f(x)=\begin{cases} x^2+5x-6 & \text{при } x>0; \\ x+\cos x & \text{в протилежному випадку;} \end{cases}$

4) $f(x)=\begin{cases} x^2+4x+5 & \text{при } x<0; \\ 1 & \\ x^2+4x+5 & \text{в протилежному випадку;} \end{cases}$

5) $f(x)=\begin{cases} 3x^3-x^2 & \text{при } x<-10; \\ \sqrt{10-x} & \text{при } -10\leqslant x\leqslant 10; \\ 2x+1 & \text{при } x>10. \end{cases}$

141. Дано дійсне число a. Для функцій $f(x)$, графіки яких зображені на малюнку 12, обчислити $f(a)$.



Мал. 12

142. Дано дійсні числа a, b, c, d, s, t, u (s та t одночасно не дорівнюють нулю). Відомо, що точки (a; b) і (c; d) не лежать на одній прямій l, яка задана рівнянням $sx+ty+u=0$. Пряма l розбиває координатну площину на дві півплощини. З'ясувати, чи належать точки (a; b) і (c; d) різним півплощинам.

Зауваження. У цій задачі необхідно скористатися тією властивістю, що:

по-перше, якщо точки не лежать на прямій $sx+ty+u=0$, то це означає, що $sa+tb+u\neq 0$ та $+td+u\neq 0$;

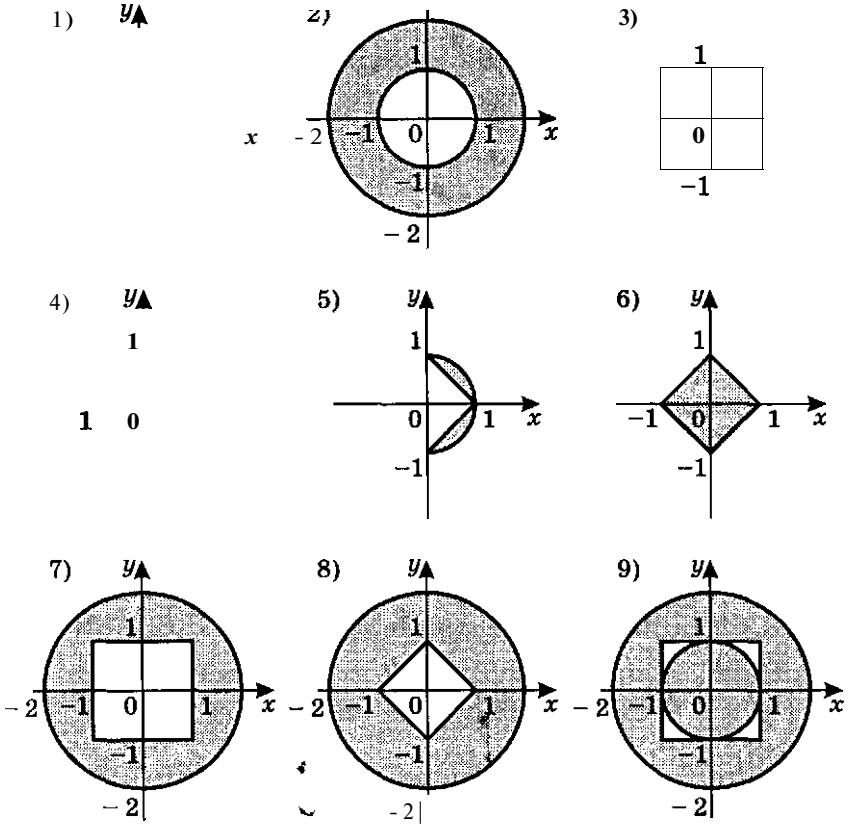
по-друге, якщо задані точки належать одній півплощині, то $sa+$ $+u$ та $sc+td+u$ - числа одного знака.

143. Дано дійсні числа a, b, c, d, e, f, h. Відомо, що точки (a; b) і (c; d) не лежать на прямій l, яка проходить через точки $(e; f)$ та $(g;$ $к)$. Пряма l розбиває координатну площину на дві півплощини. З'ясувати, чи належать точки (a; b) і (c; d) одній і тій самій півплощині.

Зауваження. Для розв'язування цієї задачі необхідно враховувати, що рівняння прямої, яка проходить через дві різні точки $(e; f)$ та $(g; h)$, має вигляд:

$$(x - e)(h - f) - (y - f)(g - e) = 0.$$

144. Дано дійсні числа x, y. Визначити, чи належить точка з координатами $(x; y)$ зафарбованій частині площини (мал.13).



Мал. 13

145. Три точки на площині задано координатами $(x_1; y_1)$, $(x_2; y_2)$, $(x_3; y_3)$. Визначити, чи лежать вони на одній прямій.

146. Координати клітинок шахової дошки визначаються парою натуральних чисел, кожне з яких не перевищує 8. Перше число - номер стовпчика (перелік зліва направо), друге - номер рядка (перелік знизу вгору). Дано натуральні числа k, l, m, n, кожне з яких не перевищує восьми. Визначити:

1) чи є поля $(k; l)$ та $(m; n)$ полями однакового кольору;
2) чи загрожує полю $(m; n)$ слон, який розташований на полі $(k; l)$;

3) чи загрожує полю $(m; n)$ тура, яка розташована на полі $(k; 0)$;

4) чи загрожує полю $(m; n)$ ферзь, який розташований на полі $(k; l)$;

5) чи загрожує полю $(m;$ л) кінь, який розташований на полі $(k; l)$.

Вибір

147. Дано ціле число л ($1 \leqslant n \leqslant 12$), яке визначає порядковий номер місяця в році. За введеним значенням л надрукувати назву відповідного місяця.

148. Дано ціле число л ($1 \leqslant$ л $\leqslant 4$), яке визначає порядковий номер кварталу року (січень, лютий, березень - I квартал і т. д.). За вказаним значенням л надрукувати перелік місяців, які відносяться до цього кварталу.

149. Розробити програму видачі номера кварталу, до якого відноситься місяць, заданий числом від 1 до 12.

150. Розробити програму виведення назви дня тижня (понеділок, вівторок тощо), якщо він заданий цілим числом від 1 до 7.

151. Розробити програму виведення інформації про день тижня - вихідний він чи робочий, якщо задано його номер від 1 до 7.

152. Розробити програму виведення кількості днів у місяці, якщо останній задається цілим числом від 1 до 12.

153. Дано ціле число n ($1 \leqslant$га$\leqslant 4$), яке визначає пору року. За вказаним значенням л надрукувати перелік місяців, які відносяться до цієї пори року.

154. Розробити програму видачі текстового варіанта шкільних оцінок:

1) 1, 2, 3 - початковий рівень;

2) 4, 5, 6 - середній рівень;

3) 7, 8, 9 – достатній рівень;

4) 10, 11, 12 - високий рівень.

155. За даним цілим значенням змінної k ($1 \leqslant k \leqslant 6$), яка визначає день тижня, надрукувати свій розклад уроків.

156. За даним порядковим номером л ($1 \leqslant$ л $\leqslant 10$) надрукувати прізвище учня вашого класу з класного журналу.

157. Введемо такі позначення для відмінків в українській мові:

«називний» - «н» або «Н»;

«родовий» – «р» або «Р»;

«давальний» - «д» або «Д»;

«знахідний» - «з» або «З»;

«орудний» - «о» або «О»;

«місцевий» - «м» або «М»;

«кличний» - «к» або «К».

Розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на які відповідає іменник у вказаному відмінку, наприклад:

«називний» - «хто?, що?».

158. Використовуючи позначення відмінків із попередньої задачі, розробити програму, яка за введеним позначенням відмінка видаватиме запитання, на яке відповідає прикметник у вказаному відмінку, наприклад:

«називний» – «який?».

159. Введемо позначення для визначення родів в українській мові:

чоловічий рід- «ч»;
жіночий рід – «ж»;
середній рід - «с».

Розробити програму, яка за введеним позначенням роду видаватиме закінчення відповідних йому слів у називному відмінку, наприклад:

«жіночий рід» - «-а, -я».

160. Дано ціле число n $(1 \leqslant n \leqslant 3)$ і дійсне число x. За заданим значенням змінної n, яке визначає порядковий номер функції, визначити:

1) sin x; 2) cos x; 3) tg x.

161. Дано дійсне значення x і ціле значення n $(1 \leqslant n \leqslant 4)$. За введеним значенням n знайти значення відповідної функції:

1) $x^2 + 2x - 3$; 2) $3x - 10$; 3) $\frac{}{|x|+10}$.

162. Розробити програму-довідник, яка за введеним значенням радіуса R пропонуватиме користувачу послуги в обчисленні:

1) 1 – довжини кола;
2) 2 - площі круга;
3) 3 - об'єму кулі;
4) 4 - площі поверхні кулі.

163. Розробити «меню»-орієнтований алгоритм, який за введеними початковими даними і порядковим номером виводить таку інформацію:

1) прізвище учня;
2) ім'я;
3) номер школи;
4) клас;
5) номер телефону.

164. Розробити алгоритм-«лотерею», який, використовуючи генератор випадкових чисел, визначатиме призи:

1) комп'ютер;
2) принтер;
3) сканер;
4) компакт-диск;
5) набір дискет.

165. Дано натуральне число n ($n \leqslant 100$), яке визначає вік людини. Додати до цього числа відповідно «рік», «роки», «років». Наприклад, 1 рік, 12 років, 94 роки.

166. Розробити програму, яка за введеною датою народження людини визначає, до якого знаку зодіаку вона належить:

20.01 - 18.02 Водолій	23.07 – 22.08 Лев
19.02 - 20.03 Риби	23.08 – 22.09 Діва
21.03 - 19.04 Овен	23.09 – 22.10 Терези
20.04 - 20.05 Телець	23.10 – 22.11 Скорпіон
21.05 – 21.06 Близнюки	23.11 - 21.12 Стрілець
22.06 – 22.07 Рак	22.12 – 19.01 Козеріг

167. Виявляється, що будь-яку цілочислову грошову суму, більшу за 7 грошових одиниць, можна видати лише купюрами в 3 та 5 грошових одиниць. Дано натуральне n ($n > 7$). Визначити, якою кількістю купюр у 3 та 5 грошових одиниць відповідно можна видати суму в n грошових одиниць, щоб їх загальна кількість була найменшою.

ЦИКЛІЧНІ АЛГОРИТМИ

Оператори повторення в Pascal

Як же ж цикли полегшують життя програмістам! Уявіть собі на хвилинку, що вам довелося б писати повторення одних і тих самих фрагментів програм багато разів! У Pascal передбачено три різновиди операторів циклу. Всі вони різні за своїм записом і застосуванням.

Загальний вигляд оператора циклу з передумовою:

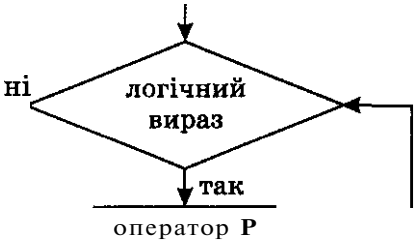
while <логічний вираз> **do** P,

де логічний вираз приймає одне з двох значень **true** або **false**, P - простий чи складений оператор. Цикл з передумовою працює за таким принципом: ***повторення оператора P буде виконуватися доти, поки логічний вираз в операторі циклу отримує значення*** **true**. ***Якщо тільки на деякому кроці циклу логічний вираз набуде значення*** **false**, ***цикл припинить свою роботу.***

Одне важливе зауваження: оскільки виконувані дії знаходяться за всіма службовими словами оператора циклу з перед-

умовою, то не можна забувати про операторні дужки у випадку, коли тіло циклу складається з кількох таких дій.

Оператор дуже простий і зрозумілий. Схема алгоритму оператора повторення з передумовою не викличе у вас ніяких непорозумінь (мал. 14).



Мал. 14

Ви пам'ятаєте захист програми від введення неприпустимих даних, який ми робили за допомогою оператора розгалуження та з використанням мітки? Наведемо приклад такого самого захисту, але більш раціонального:

```
program my_defence;
  var x, y, z: real;
begin
  write ('Задайте три додатні числа: ');
  readln (x, y, z);
  while (x <= 0) or (y <= 0) or (z <= 0) do
    begin
      writeln ('Задайте три додатні числа: ');
      read (x, y, z);
    end;
{продовження програми}
```

Справді, без оператора **goto** можна спокійно обійтися. Один недолік у цієї програми: нам довелося двічі повторити групу операторів опитування значень змінних **(write ... readln).** Спробуємо обійти і цю незручність.

Загальний вигляд оператора циклу з післяумовою:

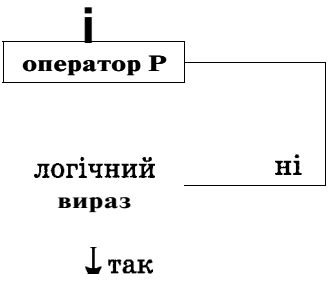
repeat P **until** <логічний вираз>;

де значення всіх параметрів такі самі, як і в попередньому операторі.

Цикл з післяумовою працює таким чином: ***повторення оператора*** **P** ***відбувається доти, поки логічний вираз отримує значення*** **false.** ***Якщо тільки на деякому кроці циклу***

логічний вираз набуде значення **true**, ***цикл завершить свою роботу.***

Чи відчули ви відмінність між операторами **while ... do** та **repeat ... until?** Перший оператор спочатку перевіряє значення логічного виразу, а потім вирішує, виконувати йому оператор, чи ні, а другий, навпаки, спочатку виконує вказаний оператор, а потім перевіряє, чи треба його виконувати ще раз. Ось як цю різницю наочно демонструє схема алгоритму (мал. 15):



Мал. 15

А тепер віддайте належне самому елегантному захисту програми від введення неприпустимих даних за допомогою оператора циклу з післяумовою.

```
program my_defence;
  var x, y, z:  real;
begin
  repeat
    write ('Задайте три додатні числа: ');
    readln (x, y, z);
  until (x > 0) and (y > 0) and (z > 0);
{продовження програми}
```

Згадайте один з варіантів затримки виконання програми, що був наведений у попередніх розділах

repeat until KeyPressed.

Тепер його можна зрозуміти без додаткових пояснень.

Аби ви не подумали, що за допомогою циклів можна записувати лише повторення «стратегічного» типу і використовувати їх для зручності роботи з вашою програмою, наведемо приклад використання циклу для обчислення суми чисел, значення яких вводяться користувачем. Кількість чисел, що сумується, наперед невідома. Домовимося лише про пароль, який буде означати завершення введення даних. Нехай це буде число 0.

```
program suma;
  var a, sum: real;
begin
  sum := 0;
  write ('Задайте число (ознака завершення введення - число 0) ');
  readln (a);
  while (a<>0) do
    begin
      sum := sum + a;
      write ('Задайте число (ознака завершення введення - ');
      write ('число 0) ');
      readln (a);
    end;
  writeln ('Сума заданих чисел: ', sum:10:5)
end.
```

Проаналізуємо цю програму.

По-перше, ми обійшлися лише двома змінними, хоча даних під час виконання програми може бути введено багато. Цей підхід називається ***покроковим введенням*** початкових даних.

По-друге, перш ніж починати виконувати цикл, змінній *sum* було присвоєне значення 0, потім на кожному кроці циклу додавалося до її попереднього значення нове значення, введене з клавіатури. Це дуже схоже на скриньку, з якої ми спочатку все викинули, перш ніж починати туди щось складати. Весь цей процес називається «накопиченням суми».

Виконайте самостійно таке завдання. Скориставшись тим, що додавання до суми числа 0 не змінить її значення, перепишіть останню програму за допомогою оператора циклу з післяумовою.

Загальний вигляд оператора циклу з параметром:

for <параметр циклу> := $X_{\text{поч}}$ **to** (**downto**) $X_{\text{кінц}}$ **do** *P*,

де параметр циклу - змінна зчисленого типу, $X_{\text{поч}}$ - початкове значення параметра циклу, $X_{\text{кінц}}$ - кінцеве значення параметра циклу, *P* - простий чи складений оператор.

Службове слово **to** означає, що зміна значення параметра циклу йде від $X_{\text{поч}}$ до $X_{\text{кінц}}$ в порядку збільшення, а **down to** - в порядку зменшення.

Зверніть увагу на те, що нам невідомий крок, з яким іде зміна значення параметра циклу. А це тому, що в циклі **for ... to (downto) ... do** на кожному наступному кроці в якості значення параметра циклу береться наступне (або попереднє) значення. В цьому сенс того, що параметр циклу повинен бути зчисленого типу! Цикл припинить своє виконання тоді, коли значення параметра циклу сягне кінця вказаного інтервалу $X_{\text{поч}} \ldots X_{\text{кінц}}$.

Схема алгоритму оператора циклу з параметром абсолютно не відрізняється від такої самої схеми алгоритму для оператора циклу з передумовою. Тому наводити її ще раз немає сенсу.

Розглянемо такий приклад. У математиці існує поняття факторіала. Факторіал обчислюється за такою формулою:

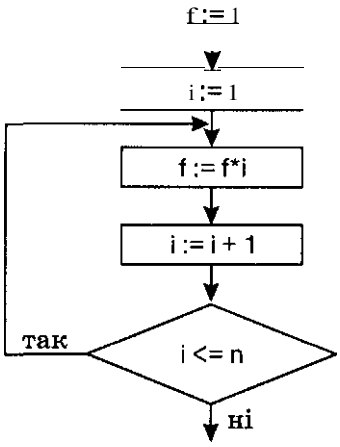
$$n! = 1*2*3*4*5*\ldots *n.$$

Тобто $n!$ - добуток усіх натуральних чисел від 1 до n.

```
program factorial;
  var i, f: integer;
begin
  f := 1;
  write ('Задайте значення числа n: ');
  readln (n);
  for i := 1 to n do
    f := f*i;
  writeln ('Значення факторіала числа ', n:5, 'дорівнює: ', f: 10)
end.
```

Накопичення добутку відрізняється від накопичення суми тим, що початкове значення обов'язково повинне дорівнювати 1!

Схема алгоритму цієї програми зображена на малюнку 16.



Мал. 16

Якщо ми лише незначним чином змінимо нашу програму, ввівши виведення результату в тіло циклу, то отримаємо зовсім інший ефект. Результатом її виконання буде виведення значень факторіала на кожному кроці. Такий принцип називається ***покроковим виведенням*** результатів.

```
for i := 1 to n do
  begin
        f := f*i;
        write ('Значення факторіала числа ', i:5);
        writeln ('дорівнює: ', f: 10)
end;
```

Інший приклад. Для того щоб вивести на екран монітора всі літери латинської абетки, введіть таку програму:

```
program alphabet;
  var i : char;
begin
  for i :='a' to 'z' do
    write (i, ' ');
  repeat until KeyPressed
end.
```

Як розібратися, коли який оператор циклу використовувати? Саме в цьому й полягає майстерність програміста — оцінити постановку задачі, скласти алгоритм, залучити всі свої аналітичні здібності й остаточно вибрати оптимальний варіант циклічного оператора.

Порада. Якщо вам наперед відома кількість повторень оператора циклу, то доцільніше використовувати цикл із параметром. Якщо кількість повторень залежить від виконання деякої умови і постановка задачі передбачає обов'язкове виконання циклу хоча б один раз, то в цьому випадку рекомендується лише цикл з післяумовою. Якщо ж наперед невідома кількість повторень циклу, і, за умовою задачі, можливе невиконання циклу жодного разу, то прогноз єдиний - вам підходить лише цикл з передумовою! У циклах з перед/післяумовою створені вами умови повинні містити хоча б одну змінну величину, значення якої змінюється в тілі цього циклу. Інакше ви маєте нагоду створити **«вічний двигун».**

Інколи при складанні циклічних алгоритмів зручно використовувати так звані **«перемикачі».** Розглянемо два типові «перемикачі» і приклади їх використання.

1. Розглянемо, як змінюватиметься значення змінної k під час виконання в циклі такого оператора:

$$k := -k.$$

Зрозуміло, що на кожному наступному кроці абсолютна величина значення змінної k змінюватися не буде, а лише буде змінюватися на протилежний знак цієї величини. Наприклад, при початковому значенні $k = 1$ буде утворюватися послідовність:

$$-1, 1, -1, 1, \ldots .$$

При початковому значенні $k = -1$ ця послідовність матиме вигляд:

$$1, -1, 1, -1, \ldots .$$

Застосування такого «перемикача» можна продемонструвати на прикладі. Нехай при заданій кількості доданків n треба знайти суму: 1 - 2 + 3 - 4 + 5 - 6 + ...

Скориставшись вищезазначеною ідеєю представимо шукану суму у вигляді:

$$(+1)*1 + (-1)*2 + (+1)*3 + (-1)*4 + (+1)*5 + (-1)*6 + \ldots .$$

Тепер уже зрозуміло, який вигляд матиме програма обчислення значення шуканої суми.

```
program sum1;
  var i, s, n, k: integer;
begin
  s := 0;
  k := 1;
  write ('Задайте значення числа n: ');
  readln (n);
  for i := 1 to n do
    begin
      s := s + k*i;
      k := - k;
    end;
  writeln ('Значення суми дорівнює: ', s:10)
end.
```

2. Виконання оператора k := 1 - k приводить до одержання «перемикача» типу

$$0, 1, 0, 1, \ldots$$

при початковому значенні $k = 1$ і типу

$$1, 0, 1, 0, \ldots$$

при початковому значенні $k = 0$. Де ж можна застосувати такий «перемикач»? Для прикладу перефразуємо задачу з попереднього пункту таким чином.

Нехай при заданій кількості доданків n треба знайти суму 1 + 3 + 5 + 7 + ... Перепишемо її у зручнішому для нашого випадку вигляді:

$$1*1 + 0*2 + 1*3 + 0*4 + 1*5 + 0*6 + 1*7 + \ldots .$$

Програма, що буде обчислювати цю суму, матиме такий вигляд:

```
program sum2;
  var i, s, n, k: integer;
begin
  s := 0;
  k:= 1;
  write ('Задайте значення числа n: ');
  readln (n);
  for i := 1 to n do
```

```
    begin
      s := s + k*i;
      k := 1 - k
    end;
  writeln ('Значення суми дорівнює: ', s:10)
end.
```

Рекурентні програми

У математиці серед «золотих» чисел почесне місце посіли числа Фібоначчі (Леонардо Пізанський, XIII ст.). Кажуть, що свого часу відомий учений у послідовності цих чисел відобразив закон розмноження кроликів. Ці числа мають такий вигляд:

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89 . . . .

Можна помітити, що починаючи з третього числа, кожне наступне число дорівнює сумі двох попередніх. Тобто, починаючи з третього члена цієї послідовності існує така залежність:

$$F_{new} = F_{old1} + F_{old2}.$$

Залежність нового значення члена послідовності від певної кількості попередніх називається ***рекурентною,*** а програми, які використовують рекурентні формули, називаються відповідно ***рекурентними.***

Розглянемо таку програму.

```
program fibonachi;
  var i, f_old1, f_old2, fnew: integer;
begin
  write ('Задайте значення числа n: ');
  readln (n);
  f_old1 := 1;
  f_old2 := 1;
    for i := 3 to n do
      begin
        f_new := f_old1 + f_old2;
        f_old1 := f_old2; {пересилання нових обчислених}
        f_old2 := f_new{значень на місце попередніх}
      end;
    write ('Значення ', n:5, 'числа ряду Фібоначчі ');
    writeln ('дорівнює:',f_new:10);
  repeat until KeyPressed
end.
```

Цій програмі варто приділити більше уваги.

По-перше, для обчислення великої кількості значень можна обійтися послугами лише трьох змінних.

По-друге, цикл починається з розрахунку третього члена, оскільки перші два члени вже відомі.

По-третє, після обчислення чергового члена необхідно підготуватися до наступного кроку, тобто зробити такий зсув значень змінних (мал. 17):

1-й крок 2-й крок

f old1 f old2 f new

Мал. 17

Якщо перший і другий кроки зсуву поміняти місцями, то втратяться попередні значення змінних!

Вкладені цикли

Мабуть ви знаєте такий пристрій, як спідометр, і пам'ятаєте, як він працює. У цьому пристрої найчастіше змінюється остання його цифра. Коли вона набуде значення **9**, то цифра перед нею збільшиться на одиницю, а сама вона знову почне змінюватися від **0** до **9**. Так само змінюватимуться і решта цифр спідометра.

Отже, ми маємо справу зі зміною відразу кількох значень у межах від **0** до **9**. Причому зміна на одиницю значень старших цифр залежить від того, чи пробігла весь діапазон значень молодша цифра. Виявляється, що програма, яка імітує роботу спідометра, досить проста:

```
program spidometer;
  uses CRT;
  var i, j, k: integer;
begin
  GoToXY(45,10);
  write ('Спідометр: ')
  for i := 0 to 9 do
    begin
      GoToXY (45,12);
      write (i);
      for j := 0 to 9 do
        begin
          GoToXY (48,12);
          write (j)
          for k := 0 to 9 do
            begin
              GoToXY (51,12);
              write (k)
            end;
        end;
    end;
  repeat until KeyPressed
end.
```

Вкладені цикли організовуються за принципом «матрьошки»: спочатку повністю відпрацьовує внутрішній цикл, після

цього значення параметра зовнішнього циклу міняється на наступне, а внутрішній цикл починає свою роботу спочатку.

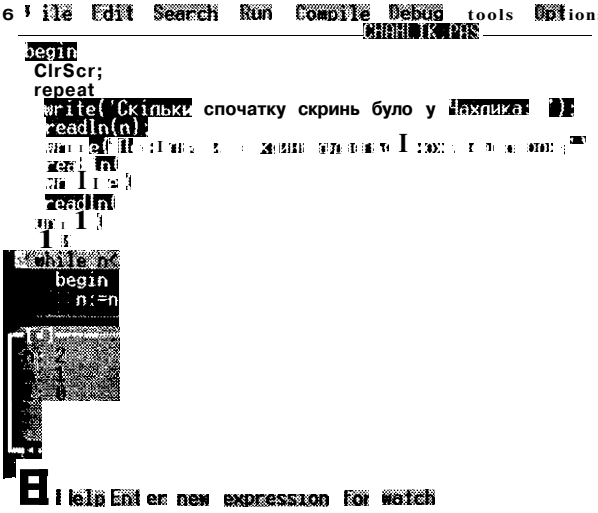
На відміну від вкладених циклів решту циклів називають *простими*.

можливості інтегрованого середовища програмування Turbo Pascal 7.0 для роботи з циклами

Розглянемо ще деякі можливості середовища програмування Turbo Pascal 7.0, які ви оціните належним чином під час налагодження циклічних та складніших програм.

Процес пошуку помилок називають *налагоджуванням*. Логічні помилки інколи бувають так приховані, що на їх аналіз та пошук витрачається чимало часу. Виконувати складні покрокові дії без допомоги комп'ютера дуже складно. І ось тут на допомогу програмісту приходить саме інтегроване середовище, де передбачене покрокове виконання всієї програми або її фрагмента за допомогою опції головного меню Debug. У прямому перекладі з англійської bug означає «комаха», «жук» (мал. 18).

Елемент головного меню Debug містить меню Watch, яке дає змогу відкрити нове вікно для перегляду поточних значень указаних змінних. Для задання імені змінної необхідно у вікні Watch натиснути на клавішу Enter. Результатом цієї дії буде поява на екрані ще одного вікна, де ви і введете ім'я необхідної змінної (якщо ця змінна є іменем масиву, то у вікні Watch ви спостерігатимете за зміною значень усього масиву одночасно).



Мал. 18. Вигляд інтегрованого середовища Turbo Pascal 7.0 під час роботи з елементом меню Watch

В інтегрованому середовищі Turbo Pascal 6.0 меню **Watch** розташоване в меню **Window.**

Звичайно, переглядати поточну зміну значень величин доречно лише при покроковому виконанні програми. В інтегрованому середовищі передбачена і така можливість. Розглянемо декілька її варіантів.

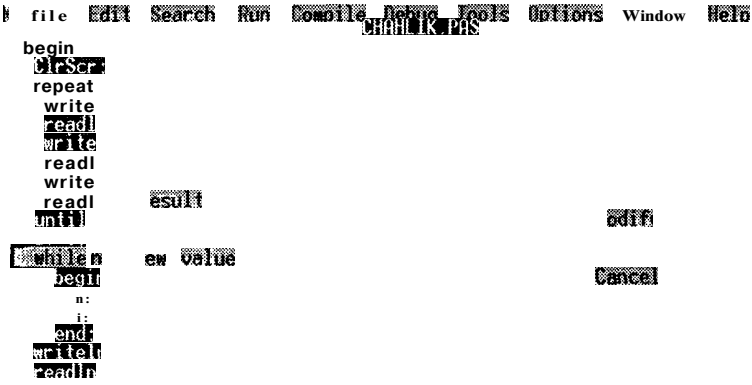
Якщо натиснути на функціональну клавішу **F8,** програма буде виконуватися в покроковому режимі. Тобто для виконання кожного наступного рядка в тексті програми необхідно натискати на клавішу **F8.** При цьому рядок, який буде виконуватися, помічається блакитною смужкою. До речі, якщо ви в одному рядку розмістите кілька команд програми, то за один крок буде виконано всю цю групу команд. Якщо ви хочете простежити, як виконуються команди в тілі циклу, то розбийте їх на окремі рядки.

Якщо початок програми не викликає у вас ніяких сумнівів, можете встановити курсор у той рядок програми, з якого необхідно виконувати покроковий перегляд проміжних результатів. Запуск програми на виконання здійснюється за допомогою функціональної клавіші **F4.** Програму буде виконано до вказаного рядка, а далі ви, користуючись уже знайомою клавішею **F8,** продовжите покрокове виконання дій і відслідковування значень змінних у вікні **Watch.**

Якщо ще до завершення виконання програми ви вже знайшли помилку, припинити її подальше виконання можна за допомогою клавіш **Ctrl+F2.**

Якщо вас цікавить якийсь конкретний фрагмент у програмі і ви хочете «прокручувати» цикл, зупиняючись весь час на цьому місці, то його можна позначити «червоним рядком» за допомогою клавіш **Ctrl+F8.** Таких контрольних точок можна встановити в програмі кілька. Виконання програми між ними здійснюється за допомогою клавіш **Ctrl+F9** або покроково клавішею **F8.** Зняття контрольних точок здійснюється тією самою комбінацією клавіш **Ctrl+F8,** коли курсор знаходиться у відміченому рядку.

Існує ще одна можливість для перегляду поточного значення будь-якої змінної у програмі. Не відкриваючи вікна **Watch,** під час зупинки виконання програми ви можете скористатися комбінацією клавіш **Ctrl+F4.** При цьому відкриється нове вікно, де ви зможете набрати ім'я змінної величини, значення якої вас цікавить на даний момент, і переглянути його в наступному рядку цього самого вікна (мал. 19).



Мал. 19. Перегляд поточного значення змінної в інтегрованому середовищі Turbo Pascal 7.0

Запитання для самоконтролю

1. Напишіть загальний вигляд оператора циклу з предумовою.
2. Який загальний вигляд має оператор циклу з післяумовою?
3. Напишіть загальний вигляд оператора циклу з параметром та поясніть призначення його елементів.
4. Яка характерна особливість усіх трьох операторів циклу?
5. Чим відрізняються схеми алгоритмів операторів циклу з передумовою та післяумовою?
6. Що таке рекурентність та рекурентні програми?
7. Поясніть роботу вкладених циклів.
8. Що зміниться, якщо два вкладені цикли замінити двома послідовними циклами?

Не припускайтеся помилок!

Використання в циклах непарної кількості операторних дужок призводить до однієї з двох помилок:

Error 85: «;» expected

або

Error 113: Error in statement.

Якщо тіло циклу типу **for ... to (downto) ... do** або **while ... do** не взяте в операторні дужки, то ви це помітите на етапі налагодження програми: вміло підібрані тестові початкові дані не дадуть очікуваних результатів.

Якщо в циклі **for ... to (dowto) ... do** параметр циклу не зчисленого типу, то буде видано повідомлення про помилку:

Error 29: Ordinal type expected

Якщо програма не завершує роботу, виводить одні й ті самі значення, то це може означати, що в операторах типу **while ... do** або **repeat ... until** ви не передбачили зміну параметра циклу.

Якщо не описано параметр циклу в розділі змінних, то він буде невідомий програмі і ви отримаєте повідомлення про помилку:

Error 3: Unknown identifier.

Виконайте завдання

Вправи

168. Нехай $M = 128$. Чому дорівнюватиме значення змінної M в результаті виконання послідовності операторів:

```
i: = – 2;
while i <= 5 do
   begin
      M := M/2;
      i :=i + 2
   end?
```

169. Якого значення набуде змінна m в результаті виконання послідовності операторів:

```
for i := 1 to 5 do
   begin
      read (x);
      if x >= Othen m := 2*i
   end;
```

де змінна x послідовно набуватиме значення:

1) 1, 4, – 2, 5, 0; 2) 1, 3, 5, 2, 4?

170. Якого значення набуде змінна x в результаті виконання такої послідовності операторів:

```
x := 0.5;
for i := 1 to 5 do
   begin
      x := x*2;
      x := x + 1
   end?
```

171. Якого значення набуде змінна f в результаті виконання послідовності дій:

```
1) f :=false;
   for i := 1 to 5 do
      begin
         if a < 0 then f :=true;
         a := a - 1
      end;
```

```
2) f := false;
   for i := 1 to 5 do
      begin
         if a < 0 then f := true;
         a := - a
      end;
```

для таких початкових значень змінної а:

а) 10; б) 5; в) 1; г) - 10; д) 4.

172. Якого значення набуде змінна *t* в результаті виконання таких дій:

```
1) t := 10;
   while t > 1 do
      t := t/2;
```

```
2) t := 1;
   while t > 1 do
      t := t/2;
```

```
3) t := 1;
   repeat
      t := t/2;
   until t <= 1.
```

173. Якого значення набуде змінна s після виконання такої послідовності дій:

```
1) s := 0;
   i := 0;
   while i < 5 do
      i := i + 1;
      s := s + 1/i;
```

```
3) s := 0;
   i := 0;
   while i < 5 do
      begin
         i := i + 1;
         s := s + 1/i
      end;
```

```
2) s := 0;
   i := 1;
   repeat
      s := s + 1/i;
      i := i - 1
   until i <= 1;
```

```
4) s := 1;
   n := 1;
   for i := 2 to n do
      s := s + 1/i.
```

174. Проаналізувати фрагменти алгоритмів і визначити, які значення будуть надруковані в результаті виконання таких дій:

```
1) s := 1;
   for i := 1 to 10 do
      s := s + 1;
writeln (s);
```

```
2) s := 1;
for i := 1 to 10 do
   begin
      s := s + 1;
      writeln (s)
   end.
```

175. Скільки разів будуть надруковані числа **2** і **3** в результаті виконання таких дій:

```
1) for i := 1 to 3 do
      writeln (2);
   for j := 1 to 2 do
      writeln (3);
```

```
2) for i := 1 to 3 do
      begin
         writeln (2);
         for j := 1 to 2 do
            writeln (3);
      end
```

Серйозні розважалки

176. Вивести на екран монітора своє прізвище задану кількість разів.

177. Ненажера Стецько пробрався перед обідом у шкільну їдальню, де вже були накриті столи, і почав швиденько з'їдати ще тепленькі булочки, що стояли на столах. З першого столу він з'їв x_1 булочок, з другого - x_2 булочок, і, відповідно, з останнього - x_n булочок. Але за ним стежив черговий по їдальні Андрійко і ретельно все фіксував на своєму калькуляторі: до

булочок, з'їдених з першого столу, додав кількість булочок, що зникла з другого столу і т. д. Допоможіть покроково відтворити інформацію, яку отримував Андрійко на своєму калькуляторі.

178. Нещасний Петрик їсть несмачну макаронину завдовжки n км. Першого дня він з'їв половину всієї довжини, другого дня - третину від того, що залишилося, третього дня - четверту частину від того, що залишилося другого дня, і т. д. Скільки макаронини ще залишиться йому «домучувати» на m-й день?

179. На дверях ліфта висіло загрозливе попередження про те, що двері зачиняються самі в той момент, коли зайвий за вагою пасажир переступить поріг ліфта. Котрий за рахунком пасажир постраждає, якщо ліфт витримує вагу не більш як S кг, а вага пасажирів, що стоять у черзі до ліфта, дорівнює відповідно $a_1, a_2, \ldots a_n$?

180. Коли Василині Премудрій виповнилося 18 років, Чахлик Невмирущий вирішив взяти її заміж. Василина запитала Чахлика, скільки у нього скринь із золотом. Чахлик сказав, що в нього зараз n скринь і кожний рік додається ще по m скринь. Василина пообіцяла, що вийде заміж тоді, коли у Чахлика буде k повних скринь із золотом. Скільки років буде тоді нареченій?

181. Капосний папуга навчився висмикувати в дідуся Василя волосся, яке ще залишилося в того на голові. Почавши з однієї волосини, він щодня збільшував порцію вдвічі. Через скільки днів дідусеві не знадобиться гребінець, якщо на початку в нього на голові було аж N волосин?

182. У понеділок Толя позичив у Миколки 2 цукерки і з задоволенням їх з'їв. У вівторок він позичив удвічі більше цукерок, після чого віддав половину боргу, а решту цукерок знову із задоволенням з'їв. Кожного наступного дня він позичав удвічі більше цукерок, ніж попереднього дня, віддавши з них цілу частину від половини боргу, а решту цукерок із задоволенням з'їдав. Скільки цукерок з'їсть Толя через N тижнів? Який у нього при цьому буде борг? Скільки цукерок устигне повернути за цей час Толя Миколці?

183. Компанія бабусь поїхала на мотоциклах на курси з комп'ютерної грамотності. Попереду на мотоциклі без глушника їхала одна бабуся, за нею - дві, потім - три і т. д. Скільки бабусь їхало на заняття, якщо приголомшені пішоходи всього нарахували n рядів? Чи змогли бабусі зайняти всі місця у класі, якщо там стояло k рядів по l комп'ютерів у кожному? Скільки вільних місць залишилося?

184. Два хлопчики одночасно стартували з однієї точки і побігли - один по колу, а другий по сторонах квадрата. Якщо вважати, що радіус кола може бути лише цілим числом, а $\pi = 3{,}14$, то при якому найменшому радіусі і при якій стороні квадрата вони знову одночасно зустрінуться в початковій точці?

185. Маленька Моська хоче помірятися зростом зі Слоном і біжить за ним зі швидкістю у, м/хв, а Слон тікає від неї зі швидкістю v_2 м/хв $(v_1 > v_2)$. У змореної Моськи, яка почала бігти на t хв пізніше від Слона, швидкість через кожні 10 хв падає на h м/хв. Чи здійсниться Мосьчина мрія і, якщо так, то через скільки хвилин це станеться?

186. Василина Премудра грала у шашки зі Змієм Гориничем. Спочатку Василина з'їла у Горинича 3 шашки, а Горинич у Василини - 5 шашок, потім Василина у Горинича з'їла 9 шашок, а Горинич у Василини - 10 шашок, на третьому ході Василина проковтнула 15 шашок, а Горинич - 20. Ця серйозна гра тривала ще довго, аж поки Горинич не втомився і на N-му ході не з'їв саму Василину Премудру. Скільки всього шашок проковтнув Змій Горинич?

187. Коли в кімнаті розважалося вже N мух, Петро Петрович відкрив кватирку і, розмахуючи рушником, почав виганяти їх на вулицю. На виганяння однієї мухи він затрачав 1 хв, але через кожні 5 хв у кімнату залітала нова муха. Коли в кімнаті ставало менше як 10 % від початкової кількості мух, то процес виганяння мух уповільнювався вдвічі. Скільки мух залишилося розважатися в кімнаті через K хв? Через скільки хвилин Петро Петрович залишиться у кімнаті на самоті?

188. Капітан Флінт із своїми піратами на безлюдному острові викопав величезний скарб із старовинних золотих монет. Спочатку Флінт узяв собі найбільшу кількість монет, яка не перевищувала половини скарбу, а решту віддав своїм розбійникам. Але на цю частину скарбу наклав лапу його заступник, який за прикладом свого начальника зробив те саме, а решту віддав підлеглим. Таким чином у кожній компанії, що залишалася, знаходився старший, який забирав свою частину скарбу, тобто найбільшу кількість монет, яка не перевищувала половини того, що ділили, залишаючи решту всім іншим. Скільки монет дісталося останньому розбійникові, якщо всього було K розбійників і M монет? Чи залишилися обділені розбійники?

Задачі

Прості ЦИКЛИ

189. Знайти значення:

1) $1*3*5*\ldots*101$;

2) $(1+0.1)(2+0.2)\ldots(9+0.9)$;

3) $(1+2)+(1+2+3)+\ldots+(1+2+\ldots+50)$;

4) $(100+10\cos 0.1)(100+10\cos 0.2)\ldots(100+10\cos 10)$.

190. Дано ціле n. Визначити:

1) $n!$; 2) 2^n; 3) $2+4+6+\ldots+2n$;

4) $1*3*5*7*\ldots*(2n+1)$;

5) $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$

6) $\sin(1)*\sin(1+2)*\ldots*\sin(1+2+\ldots+n)$;

$\frac{1}{\sin n}+\frac{1}{\sin n+\sin(n-1)}+\ldots+\frac{1}{\sin n+\ldots+\sin 1}$

191. За даним натуральним значенням змінної n обчислити

1) $\frac{1}{\cos 1}*\frac{1+2}{\cos(1+2)}*\ldots*\frac{1+2+\ldots+n}{\cos(1+2+\ldots+n)}$,

2) $\frac{10\sin 1}{\sqrt{1}}*\frac{10\sin 2}{\sqrt{1}+\sqrt{2}}*\ldots*\frac{10\sin n}{\sqrt{1}+\sqrt{2}+\ldots+\sqrt{n}}$,

3) $\left(1+\frac{\cos 1}{1^2}\right)\left(1+\frac{\cos 2}{1^2+2^2}\right)\ldots\left(1+\frac{\cos n}{1^2+2^2+\ldots+n^2}\right)$,

4) $\frac{\cos n}{\sin 1}+\frac{\cos n*\cos(n-1)}{\sin 1*\sin 2}+\ldots+\frac{\cos n*\ldots*\cos 1}{\sin 1*\ldots*\sin n}$

5) $1^2-(1+2)^2+(1+2+3)^2-\ldots(-1)^{n+1}(1+2+\ldots+n)^2$;

$1^2-(1+2)^2+(1-$

6) $\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}$;

7) $\sqrt{3+\sqrt{6+\ldots+\sqrt{3n}}}$;

n разів

8) $|\cos 1|+|\cos 3|+\ldots+|\cos(2n+1)|$

9) $\sqrt{|\sin 1|+\sqrt{|\sin 2|+\ldots+\sqrt{\sin n}}}$

10) $(\cos 1+1)*\left(\cos 3+\frac{1}{2}\right)*\ldots*\cos(2n+1)+\frac{1}{n}$

192. Дано натуральне число n і дійсне число x. Обчислити:

1) $\cos x+\cos^2 x+\ldots+\cos^n x$;

2) $\cos x+\cos x^2+\ldots+\cos x^n$;

3) $\underbrace{\cos x+\cos\cos x+\ldots+\cos\cos\ldots\cos x}_{n\text{ разів}}$.

193. Дано натуральне число n і дійсне число x. Обчислити:

1) $\cos x+\cos^2 x+\ldots+\cos^n x-\sin x+\sin^2 x+\ldots+\sin^n x$;

2) $\sin x + \sin\sin x + \ldots + \sin\sin\ldots\sin x + \cos x + \cos x^2 + \cos x^n$;

3) $(\sin x + \sin^2 x + \ldots + \sin^n x) * (\cos X + \underbrace{\cos\cos\ldots\cos}_{n \text{ разів}} X + \ldots + \underbrace{\cos\cos\ldots\cos}_{n \text{ разів}} x)$.

194. Дано дійсне число a і натуральне число n. Обчислити:

1) a^n;

2) $a(a+1)\ldots(a+n-1)$;

3) $\frac{1}{a} + \frac{1}{a(a+1)} + \ldots + \frac{1}{a(a+1)\ldots(a+n)}$;

4) $a(a-n)(a-2n)\ldots(a-n^2)$;

5) $(a+1)(a+1+2)\ldots(a+1+2+\ldots+n)$;

6) $\ln|a^n| + \ln|a^{n-1}| + \ldots + \ln|a|$;

7) $\frac{1}{1+\cos a} + \frac{1}{1+\cos a^2} + \ldots + \frac{1}{1+\cos a^n}$,

8) $\frac{a}{\cos a + 1} \quad \frac{a^2}{\cos^2 a + 1} \quad \frac{a^n}{\cos^n a + 1}$

195. Обчислити значення:

1) $\sum_{i=1}^{50} i^{2*}$; 2) $\sum_{i=10}^{50} \frac{1}{i}$; 3) $\sum^{10}$

196. Обчислити значення:

1) $\prod_{i=10}^{100} \frac{i+1}{i-2}$ ** 2) $\prod^{10}\left(2 - \frac{1}{i!+2}\right.$ 3) $\prod^{16}\left(1\right)^2$

197. Дано натуральне число n. Визначити:

1) кількість цифр у цьому числі;

2) суму цифр у числі;

3) суму першої й останньої цифри числа;

4) чи утворюють цифри числа неспадну послідовність.

198. Дано два натуральні числа n і m. Визначити:

1) в якому з чисел частіше трапляється цифра 5;

2) порядок якого з чисел більший;

3) переставити місцями відповідно перші й останні цифри обох чисел.

199. Дано ціле число $n \leqslant 10\,000$. Представити його у вигляді

$$a_p * 10^p + a_{p-1} * 10^{p-1} + \ldots + a_1 * 10 + a_0,$$

$\sum$ - означає суму елементів у вказаному діапазоні.

$\prod$ - означає добуток елементів у вказаному діапазоні.

де p - порядок числа n; a_p, a_{p-1}, a_1, a_0 - цифри числа n. Наприклад,

$$123 \Rightarrow 1*10\hat{}2 + 2*10 + 3.$$

200. Дано дійсне число a. Знайти:

1) серед чисел $1{,}1 + \frac{1}{2}, 1 + \frac{1}{2} + \frac{1}{3}, \ldots$ перше, яке більше за a;

2) таке наименше n, що $1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} > a$.

201. Дано натуральне число n. Обчислити:

1) $\sum \sqrt{k}$; 2) $\sum (-1)^k (2k+1)$; $\frac{2+}{}$.

202. Дано натуральне число n. Обчислити:

$\prod_{i=1}^{n} \frac{i}{i!}$ 2) $\prod \frac{1+\cos i}{2^i}$; 3) $\prod_{i=1}^{n-2} \frac{(-1)^i}{((i+1)!+1)^2}$

203. Знайти найбільше додатне ціле число n, для якого виконується умова:

1) $3n^2 - 730n < 5$;

2) $-4n + 84\sqrt{n} + 3 \geqslant 0$;

3) $7n^3 + 81n^2 + 10^6 < 0$;

4) $e^n - 1000 \ln n \leqslant 5$;

5) $-n^n + 30n^2 - 1 > 0$.

204. Дано ціле число $m > 1$. Одержати найбільше число при якому $4^k < m$.

205. Дано натуральне число n. Одержати найменше число вигляду 2^k, що перевищує n.

206. Обчислити:

$$1 + \frac{}{3 + \frac{}{5 + \frac{}{\ddots .101 + \frac{1}{103}}}}$$

207. Дано дійсне число $x \neq 0$. Обчислити значення величини

$$x^2 + \frac{x}{x^2 + \frac{x^2}{x^2 + \frac{8}{2} 512}}$$

208. Під час обчислення результатів деяких експериментів виникає потреба отримання результату із заданою похибкою. Нехай результатом є нескінченна сума, що задається певною формулою, і відома похибка ε ($\varepsilon > 0$) для знаходження наближеного значення результату. Вважатимемо, що необхідна точність досягнена, якщо додавання наступного доданка змінює суму на величину, меншу за ε. Обчислити:

$$1 + \sum_{i=1}^{\infty} \frac{(-1)^i}{i!} \qquad \sum_{i=1}^{\infty} \frac{(-2)^i}{i(i+1)(i+2)}$$

209. На скільки років необхідно покласти в банк суму в X грошових одиниць, щоб отримати суму в N грошових одиниць ($N > X$), якщо банк нараховує 14 % щорічних?

210. Дано ціле число n, яке набуває значень шкільних балів. Визначити відповідною кількістю звукових сигналів, який саме бал був заданий («1» - один звуковий сигнал, «2» — два звукові сигнали і т. д.). Якщо ж задане число не відповідає значенню шкільного бала — подати довгий звуковий сигнал.

211. Заповнити екран заданим символом у кількості n, починаючи з указаної позиції екрана (x; y).

212. Обчислити значення числа π, використовуючи формулу $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$. Визначити, яка кількість елементів послідовності дає значення числа π з точністю до 3, 5, 7 знаків після коми.

213. Англійський математик Дж. Мечином (1680–1751) винайшов формулу для обчислення значення числа π:

$$\frac{\pi}{4} = 4 * \operatorname{arctg}(1/5) * \operatorname{arctg}(1/239).$$

Для обчислення значення $\operatorname{arctg}(1/x)$ можна скористатися формулою:

$$\operatorname{arctg}(1/x) = \frac{1}{x} - \frac{1}{3x^3} + \frac{1}{5x^5} - \dots .$$

Для спрощення необхідно обмежитися лише вісьмома елементами цієї послідовності. Скласти програму для обчислення наближеного значення числа π.

Покрокове введення/виведення даних

214. Дано натуральне число n і дійсні числа $a_1, a_2, \dots a_n$. Обчислити:

1) $a_1 + a_2 + \dots + a_n$;
2) $|a_1 + a_2 + \dots + a_n|$;

3) $|a_1| + |a_2| + \ldots + |a_n|$;

4) $\sqrt{|a_1|} + \sqrt{|a_2|} + \ldots + \sqrt{|a_n|}$;

5) $a_1 * a_2 * \ldots * a_n$;

6 $\frac{a_1 + a_2 + \ldots + a_n}{}$

215. Дано натуральне число я і дійсні числа a_1, a_2, Обчислити:

1) $(a_1 + 1) + (a_2 + 2) + \ldots + (a_n + л)$;

2) $a_1 - a_2 + a_3 \ldots (-1)^{n+1} a_n$;

3) $\frac{a_1}{1!} + \frac{a_2}{2!} + \ldots + \frac{a_n}{л!}$;

4) $\frac{a_1}{1 + 2} + \ldots + \frac{a_n}{1 + 2 + \ldots + л}$

5) $\sqrt{|a_1 a_2 \ldots a_n|}$.

216. Дано натуральне число л і дійсні числа a_1, a_2, Використовуючи задачі № 214, 215, обчислити:

1) $\sin(a_1 + a_2 + \ldots + a_n) \cos(a_1 a_2 \ldots a_n)$;

2) $(\sqrt{|a_1|} - a_1)^2 + (\sqrt{|a_2|} - a_2)^2 + \ldots + (\sqrt{|a_n|} - a_n)^2$;

3) $\sqrt{a_1^2 + 1} + \sqrt{a_2^2 + 2} + \ldots + \sqrt{a_n^2 + n} + n^2$

217. Дано натуральне число л і дійсні числа a_1, a_2, a_n! Обчислити:

1) $a_1, + a_2, - (-1)^n a_n$;

2) $a_1, a, + a_2,$ $a, + a_2 + \ldots + a_n$;

3) $a_1, a_1 a_2,$ $a_1 a_2 \ldots a_n$;

4) $\sin a_1, \sin(a_1 + a_2) \ldots \sin(a_1 + a_2 + \ldots + a_n)$;

5) $a_1 + 1!, a_2 + 2!,$ $a_n + n!$.

218. Дано натуральне число л. Отримати послідовність значень b_i при i = 1, 2, л, якщо відомо, що:

1) $b_i = i$;

2) $b_i = 1 + 2 + \ldots + i$;

3) $b_i = \frac{1}{i}$;

4) $b_i = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{i}$;

5) $b_i = 1! + 2! + \ldots + i!$.

219. Дано натуральне число n. Отримати послідовність значень b_i при $i = 1, 2, \ldots, n$, якщо відомо, що:

1) $b_i = 1 + \frac{1}{2!} + \frac{1}{3!} + \ldots + \frac{1}{i!}$;

2) $b_i = 2 + \frac{2}{2!} + \frac{2}{3!} + \ldots + \frac{2}{i!}$;

3) $b_i = 5(1 + 3 + \ldots + (2i - 1))$;

4) $b_i = i(1 - 2 + 3 - \ldots + (-1)^{i+1} i)$;

5) $b_i = \frac{3^i}{(i + 1)!}$.

220. Дано натуральне число n і дійсні числа a, b $(a \neq b)$. Отримати $x_0, x_1, \ldots, x_n$, де $x_i = a + ih$, $h = (b - a)/n$.

221. Вивести таблицю температур за Цельсієм від - 50 до 50 градусів та їх еквівалентів за шкалою Фаренгейта, використовуючи співвідношення $t_f = \frac{9}{5} t_c + 32$.

222. Вивести таблицю значень функції $y = x^4 - 5x^3 - 2x^2 + 0.1$ для значень x, які змінюються в інтервалі від 5 до 3 з кроком 0.2.

223. Обчислити значення виразу $\frac{\sin a + 1}{5a - 3a + 9}$ для $a = 0, 0.1, \ldots, 10$.

224. Обчислити значення багаточлена

$$12x^5 + 0.5x^3 - 1.7x^2 + x + 6$$

при $x = 0, 1, \ldots, 24$.

225. Дано натуральне число n. Обчислити значення функції

$$y = \frac{x^2 - \sin x + 1}{\sqrt{2x^3 - 1}}$$

для $x = 1, 1.1, 1.2, \ldots, 1 + 0.1n$.

226. Дано дійсні числа $a_1, \ldots, a_{50}$. Визначити b_{50}, якщо

$$b_i = \frac{\sin a_i + 1}{\cos a_i} + \sqrt{|a_i - 1|}.$$

227. Обчислити значення функцій $f_1(x) = \frac{x^3 + 1}{5}$, $f_2(x) = x^2 + x - 5$, $f_3(x) = \frac{x^3 - x^2 - 0.01}{x + 1}$ для значень аргумента $x = 1, 2, \ldots, 10$.

228. Визначити, при яких значеннях радіуса основи R об'єм циліндра висотою k набуватиме значення від 10 до 100 з кроком 10.

229. Вивести таблицю довжин відрізків, початок яких знаходиться в точці (a; b), а координати кінця (c; d) змінюються за такою залежністю:

1) $c_i = i$, $d_i = i/2$ $(i = 1, \ldots, 10)$;

2) $c_i = 0.1i$, $d_i = 2c_i + a$ $(i = -5, \ldots, 5)$;

3) $c_i = \sin(i/10*\pi)$, гі, $= \cos(i/10*\pi)$ $(i = 1,$ 20);

4) $c_i = \sqrt{i}$, $d_i = ac_i^2 + 2bc_i - 1$, $(i = 1,$ 10).

230. Дано натуральне число я і дійсні числа a_1, a_2, а„. Визначити:

1) середнє арифметичне значення a_1, a_2, a_n;
2) a_1, a_1a_2, $a_1a_2a_3$, ..., $a_1a_2 \ldots a_n$;
3) a_1a_2, a_3a_4 ... $a_{n-1}a_n$ *(n – парне)*;
4) а, $+a_2$, $a_2 + a_3$, $a_3 + a_4$ ... $a_{n-1} + a_n$;
5) $2a_1 + 3a_2 + 2a_3 + 3a_4 + \ldots + 2a_n$ *(я - непарне)*.

Поєднання повторення і розгалуження

231. Скласти програму, яка допомогла б працівникам ДАІ визначати кількість порушників перевищення швидкості на трасі, якщо відомо, що на даному проміжку траси встановлено обмеження на швидкість *MAXV*, а прилад фіксує швидкість автомобілів v_1, v_2, v_n.

232. Розробити алгоритм зарахування результатів учасників спортивних змагань за певними віковими категоріями, якщо відомо, що:

1) для учасників до 15 років результат повинен знаходитись у проміжку [10; 15];
2) для учасників від 16 до 20 років - у проміжку [16; 20];
3) для учасників від 21 до 35 років – у проміжку [21; 25];
4) для учасників від 36 до 45 років - у проміжку [26; 30];
5) старші 45 років - у проміжку [20; 30].

Дані задаються парами цілих додатних чисел $(a_i; b_i)$, де a_i - вік i-го учасника, b_i — результат i-го учасника, $i = 1, 2,$ n.

233. Дано цілі додатні числа n, R і пари дійсних чисел (а; b), $(x_1; y_1)$, $(x_2; y_2)$, $(x_n; y_n)$. Визначити, скільки точок з координатами $(x_1; y_1)$, $(x_2; y_2)$, $(x_n; y_n)$ попадають усередину кола з центром у точці $(a; b)$ і радіусом R.

234. Дано натуральні числа n і a_1, a_2, ..., a_n. Визначити кількість членів a_k послідовності a_1, a_2, a_n, які

1) є непарними числами;
2) кратні числу 3 і не кратні числу 7;
3) є квадратами парних чисел;
4) є подвоєними непарними числами;
5) задовольняють умову $2^k < a_k < k!$;
6) задовольняють умову $a_k < \frac{R-1}{2}$;
7) мають парні порядкові номери та є непарними числами;
8) при діленні на 7 дають остачу 1, 2 або 5.

235. Дано цілі числа $q_1, q_2, \ldots q_n$. Знайти суму тих членів послідовності, які:

1) є додатними та парними;

2) не кратні 7;

3) задовольняють умову $|q_k| < \text{ft}^2$;

4) мають порядкові номери, кратні 3.

236. Дано дійсні значення a і c. Відомо, що значення змінної b змінюються в проміжку [5; 5] з кроком 0.5. Вивести таблицю дійсних коренів квадратного рівняння $ax^2 + bx + c = 0$ і відповідних їм значень коефіцієнта b.

237. Дано натуральне число n і дійсні числа $x, a_1, a_2, \ldots, a_n$. Визначити, скільки разів число x трапляється серед таких чисел:

1) fl|, $a_2, \ldots, a_n$;

2) $a_1, a_1 + a_2, a_1 + a_2 + a_3$, a, $+a_2 + \ldots + a_n$;

3) $a_1^2, a_2^2, \ldots, a_n^2$;

4) $\sin a_1, \sin a_2, \ldots \sin a_n$.

238. Дано натуральне число n і дійсні числа $x, a_1, a_2, \ldots a_n$. Визначити, чи трапляється число x серед таких чисел:

1) $\cos a_1, \cos$ (a, $+a_2$), $\cos (a_1 + a_2 + \ldots + a_n)$;

2) $a_1, a_3, a_5 \ldots a_{2i-1}$, де $i = 1, 2, \ldots [n/2]$;

3) $\ldots, \frac{a_{2i-1}}{a_{2i}}$, де $a_{2i} \neq 0$, $i = 1, 2, \ldots [n/2]$.
$a_2 \quad a_4$

239. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots a_n$. Визначити:

1) max $(a_1, a_2, \ldots a_n)$;

2) min $(a_1, a_2, \ldots a_n)$;

3) max $(a_1^2, a_2^2, \ldots, a_n^2)$;

4) max $(a_1^3, a_2^3, \ldots, a_n^3)$;

5) min $(a_1, 2a_2, \ldots na_n)$.

240. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots a_n$. Обчислити:

1) max $(a_1, (a, +a_2), (a, +a_2 + a_3), \ldots, (a_1 + a_2 + \ldots + a_n))$;

2) max $(a_1, a_2, \ldots a_n) + \min (a_1, \ldots a_n)$;

3) min (sin (a, $+a_2 + \ldots + a_n$), $\cos (a_1 + a_2 + \ldots + a_n)$);

4) min (sin a, $+\sin a_2 + \ldots + \sin a_n$, cos a, $+\cos a_2 + \ldots + \cos a_n$);

5) min (sin (a, $+a_3 + \ldots + a_{2i-1}$), cos $(a_2 + a_4 + \ldots + a_{2i})$), де $i = 1, 2, \ldots, [n/2]$.

241. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. Обчислити кількість додатних, від'ємних і нульових членів заданої послідовності, а також визначити, яких із них більше.

242. Дано натуральне число n і цілі числа $a_1, a_2, \ldots a_n$. Визначити кількість і суму тих членів послідовності, які діляться націло на 5 і не діляться націло на 7.

243. Дано натуральні числа n,p і цілі числа $c_1, c_2, \ldots, c_n$. Знайти добуток тих членів послідовності $c_1, c_2, \ldots, c_n$, які кратні p.

244. Дано цілі числа p, д, $a_1, a_2, \ldots, a_{15}$ ($p > q \geqslant 0$). У послідовності $a_1, a_2, \ldots, a_{15}$ замінити нулями члени, модуль яких при діленні на p дає в остачі q.

245. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. Визначити:

1) подвоєну суму додатних членів послідовності $a_1, a_2, \ldots, a_n$;
2) обернену величину добутку тих членів послідовності $a_1, a_2, \ldots, a_n$, для яких виконується нерівність $i + 1 < a_i < i!$;
3) кількість членів послідовності $a_1, a_2, \ldots, a_n$, значення яких належать відрізку [0; 1], та суму всіх інших членів даної послідовності;
4) суму додатних та кількість від'ємних членів послідовності $a_1, a_2, \ldots, a_n$.

246. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. У послідовності $a_1, a_2, \ldots, a_n$ усі невід'ємні числа, що не належать відрізку [1; 2], замінити на одиницю та отримати кількість від'ємних елементів і кількість членів, що належать відрізку [1; 2].

247. Дано натуральне число n і дійсні числа $x_1, x_2, \ldots, x_n$. У послідовності $x_1, x_2, \ldots, x_n$ усі члени, менші за 2, замінити числом 2, а також отримати суму та кількість членів, що належать відрізку [5; 10].

248. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. У послідовності $a_1, a_2, \ldots, a_n$ усі від'ємні члени замінити числом -1, додатні - числом 1, а нульові члени залишити без змін.

249. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. У послідовності $a_1, a_2, \ldots, a_n$ усі члени, більші за 5, замінити числом 5 та знайти кількість таких членів.

250. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$. У послідовності $a_1, a_2, \ldots, a_n$ усі додатні члени збільшити на 5, а недодатні замінити на -10.

251. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots, a_n$ ($a_n \neq 0$). Відомо, що в заданій послідовності є хоча б одне нульове значення. Розглядаючи члени послідовності, що розташовані до першого нульового елемента, визначити:

1) кількість додатних членів;
2) суму від'ємних членів;
3) суму квадратів членів;
4) яких членів більше - додатних чи від'ємних;
5) середнє арифметичне членів;
6) найбільший член;
7) порядковий номер найменшого члена;
8) суму найбільшого і найменшого членів;

9) який член розташований раніше - найбільший чи найменший;

10) добуток членів з непарними порядковими номерами.

252. Дано натуральне число N. Знайти всі цілі додатні числа, що не перевищують N, дільниками яких є тільки числа 2, 3 і 5.

253. Дано два цілі додатні числа n і m. Визначити всі спільні дільники цих чисел.

254. Дано натуральне число n. Визначити всі нескорочувані дроби із знаменником n в інтервалі (0; 1).

255. Дано два цілі додатні числа n ($n \leqslant 9999$) і m ($m \leqslant 9$). Розробити власний навчальний алгоритм додавання в стовпчик чисел n та m, демонструючи по кроках процес виконання дій: запис одиниць результату в поточний розряд і перенесення десятків у наступний розряд. Перехід до наступного кроку здійснюється натисненням клавіші «пробіл».

256. Дано два цілих додатних числа n ($n \leqslant 9999$) і m ($m \leqslant 9$). Розробити власний навчальний алгоритм віднімання в стовпчик чисел n і m, демонструючи по кроках процес виконання дій: запис одиниць результату в поточний розряд і перенесення десятків у наступний розряд. Перехід до наступного кроку здійснюється натисненням клавіші «пробіл».

257. Дано два цілі додатні числа n ($n \leqslant 9999$) і m ($m \leqslant 9$). Розробити власний навчальний алгоритм множення в стовпчик чисел n і m, демонструючи по кроках процес виконання дій: запис одиниць результату в поточний розряд і перенесення десятків у наступний розряд. Перехід до наступного кроку здійснюється натисненням клавіші «пробіл».

258. Дано натуральні числа N і M. Знайти трійки послідовних натуральних чисел на проміжку $[1; M]$, добуток яких дорівнює N. Передбачити відповідь «Послідовність не знайдена».

259. Дано цілі числа п (п >0) і Р (Р >1). Скласти алгоритм переведення десяткового числа п у систему числення з основою P за таких умов:

1) число n описується стандартним типом мови програмування, $P > 10$;
2) число n описується стандартним типом мови програмування, $P > 32$.

260. Дано ціле число n (п >0) у системі числення з основою P ($1 < P < 10$). Скласти алгоритм переведення числа n у десяткову систему числення за умови, що число п описується стандартним типом мови програмування.

Рекурентні послідовності

261. Нехай $x_0 = 1$, $x_i = x_{i-1} + 2i$, де $i = 1, 2, \ldots$. Визначити значення x_{10}.

262. Послідовність чисел Фібоначчі u_0, u_1, ... утворюється за законом $u_0 = 0$; $u_1 = 1$; $u_i = u_{i-1} + u_{i-2}$, (i = 2, 3, ...). Дано натуральне число $n > 1$. Визначити члени послідовності u_1, u_2, u_n.

263. Нехай $x_0 = x_1 = 1$, $x_i =$ $+ 2x_{i-2}$, де i = 2, 3, ... Дано натуральне число n. Визначити значення x_n.

264. За даними співвідношеннями визначити n-й елемент числової послідовності:

1) $x_n = x_{n-1} + x_{n-2}$; $x_0 = x_1 = 1$; $(n > 1)$
2) $x_n = 2x_{n-1} + 3x_{n-2}$; $x_0 = 0$, $x_1 = 9$; $(n > 1)$
3) $x_n = x_{n-1} + x_{n-2} + x_{n-3}$; $x_0 = x_1 = 1$, x, 6; $(n > 2)$
4) $x_n = x_{n-1} + 4x_{n-3}$; $x_0 = x_1 = x_2 = 2$; $(n > 2)$
5) $x_n = x_{n-1}(x_{n-2} + 1)$; $x_0 = 0$, $x_1 = 1$. $(n > 1)$.

265. Нехай $b_0 = b_1 = 0$; $b_2 = 1$; $b_i = \frac{i+1}{i+2} b_{i-1} + b_{i-2} b_{i-3}$, де $i = 4, 5,$ За даним значенням л $(n \geqslant 4)$ знайти b_n.

266. Нехай $a_0 = a_1 = 1$; $a_i = a_{i-2} \frac{a_{i-1}}{2^{i-1}}$, де $i = 2, 3,$ Знайти добуток $a_0 a_1 \ldots a_{20}$.

267. Нехай $x_1 = x_2 = x_3 = 1$; x, $= x_{i-1} - x_{i-3}$, $i = 4, 5,$ Обчислити

268. Нехай $a_1 = b_1 = 1$; $a_k = 3b_{k-1} + 2a_{k-1}$; $b_k = 2a_{k-1} b_{k-1}$, де k 2, 3, ... Дано натуральне число n. Знайти

$$\overleftarrow{\prod_{k=1}^{n}} \frac{2^k}{(1 + a_k^2 + b_k^2) k!}.$$

269. Нехай $a_1 = u$; $b_1 = V$, $a_k = 2b_{k-1} + a_{k-1}$; $b_k = 2a_{k-1}^2 + b_{k-1}$, k - 2, 3, ... Дано дійсні числа u, V і натуральне n. Знайти

$$\overleftarrow{\prod_{k=1}} \frac{a_k b_k}{(k + 1)!}$$

270. Визначити найменший додатний член числової послідовності, заданої рекурентними співвідношеннями:

1) $x_n = x_{n-1} + x_{n-2} + 100$; $x_1 = x_2 = -99$;
2) $x_n = x_{n-1} + x_{n-2} + x_{n-3} + 200$; $x_1 = x_2 = x_3 = -99$;
3) $x_n = x_{n-1} + x_{n-3} + 100$; $x_1 = x_2 = x_3 = -99$.

271. Нехай $x_0 = 0$; $x_k = \frac{k + 1}{2x_{k-1} + 1}$, $k = 1, 2, \ldots$. Дано дійсне число $\varepsilon > 0$. Знайти перший член послідовності та його порядковий номер n, для якого виконується нерівність $|x_n - x_{n-1}| < \varepsilon$.

Вкладені цикли

272. Обчислити:

1) $\sum_{i=100}^{100}\sum_{j=50}^{50}(i+j)$;

2) $\sum_{i=1}^{30}\sum_{j=1}^{i}(1+i^2+j^2)$;

3)

4) $\sum_{i=10}^{100}\sum_{j=5}^{50}\cos(i-j)$.

273. Обчислити

1) $\sum_{i=1}^{10} i\sum_{j=1}^{20}(i+j)^2$;

2) $\sum_{i=1}^{\text{£}}\left(\frac{}{i}+\right.$ $_{j=1}$ +

3) $\prod^{20}{}_{2}$)

4) $\sum_{i=1}^{50}\prod_{j=1}^{20} ij$

274. Дано натуральне число n і дійсне число x. Обчислити:

1) $\sum_{k=1}\sum_{m=k}\frac{x+n}{m}$;

2) $\sum x^{2k}$;

3) $\sum(2k)!$;

4) $\sum k^k$

275. Дано натуральні числа n і m. Визначити $b_1, b_2, \ldots, b_n$, якщо відомо, що $b_i = \sum k^i$.

276. Дано натуральне число N. Знайти всі прості числа, менші за N.

277. Дано натуральні числа n і m. Визначити всі прості числа, що належать проміжку $[n; m]$.

278. Дано натуральні числа n і m. Визначити всі числа, що належать проміжку $[n; m]$ і сума цифр яких є простим числом.

279. Визначити 100 перших простих чисел.

280. Дано натуральне число N. Знайти всі числа в проміжку $[1; N]$, цифри яких утворюють неспадну послідовність.

281. Дано натуральне число N. Знайти всі цілі додатні числа, що не перевищують N, сума кубів цифр яких дорівнює самим цим числам.

282. Дано натуральне число N. Знайти всі цілі додатні числа, що не перевищують N і діляться на кожну зі своїх цифр.

283. Серед натуральних чисел, що не перевищують заданого N, знайти такі, які в двійковому представленні мають більше як половину 1.

284. Дано натуральне число n. Визначити, чи можна представити це число у вигляді $n = x^2 + y^2 + z^2$, де x, y, z — натуральні числа.

285. Дано два натуральні числа n і m. Усі цілі числа, що належать проміжку $[n; m]$, розбити на вісім груп за умовою ділення націло на 2, 3, 4, 5, 6, 7, 8 і 9.

286. Дано два цілі додатні числа n і m. Відомо, що n і m є відповідно чисельником та знаменником деякого дробу. Розробити програму розкладання чисельника і знаменника на прості множники та продемонструвати процес скорочення цього дробу.

287. Скласти програму, яка б у текстовому режимі заповнила екран монітора різнокольоровими вертикальними смугами шириною в n позицій.

288. Скласти програму, яка б у текстовому режимі виводила на екран монітора зображення шахової дошки.

289. Скласти програму, яка б у текстовому режимі моделювала каркас панелей NORTON COMMANDER.

290. Скласти програму, яка б імітувала роботу електронного годинника.

291. Скласти програму-модель лічильника використання електроенергії.

МАСИВИ

Одновимірні та багатовимірні масиви

Досі ми навіть при обчисленні цілої послідовності значень обходилися декількома змінними. Але це не завжди можливо.

Поставимо перед собою таке завдання. Нехай ми працюємо на метеостанції і в наші функції входить підрахунок щогодинного відхилення від середньодобової температури. Зрозуміло, що спочатку треба щогодини записувати значення температури, потім знайти середнє арифметичне цих значень і тільки після цього можна обчислити відхилення кожного записаного значення від середнього арифметичного.

Ми дійшли висновку, що доведеться мати справу з 24-ма змінними величинами, в значеннях яких немає ніякої закономірності. Якими ідентифікаторами їх позначити, яким чином одержати суму їх значень? Використовувати для цього набутий нами скромний досвід недоречно. Пропонується такий вихід. У математиці елементи послідовності позначають таким чином:

$$a_1, a_2, \quad a_n.$$

Скористаємося схожим позначенням і ми:

$$T[1], T[2], \quad T[24].$$

Називатимемо цю послідовність масивом.

Масивом називається скінченна послідовність змінних одного типу, які мають однакове ім'я та різняться порядковим номером.

Індексом називається порядковий номер елемента масиву.

Отже, введено новий тип - **масив.** Усі типи, які досі були вам відомі, називаються ***простими.*** Масив є прикладом ***структурованого*** типу, тобто він, у свою чергу, складається з елементів іншого типу.

Можемо зробити висновок: у поставленій задачі ми маємо справу з масивом значень температур *T[i],* де $i = 1, 2, 3, \ldots 24$.

Як звернутися до елементів цього масиву? Для цього необхідно вказати індекс. Наприклад,

$$T[2],\ T[5],\ T[i],\ T[i + j].$$

Але в третьому і четвертому прикладах для визначення необхідного елемента масиву треба знати значення величин *i* та у. Така загальність визначення індексу масиву є дуже потужним засобом програмування, але разом з цим і провокує можливі помилки: отриманий результат обчислення індексу масиву може виходити за межі інтервалу, виділеного для індексів даного масиву.

І ще один важливий момент, яким у жодному разі не можна нехтувати. Масиви відносяться до структур з так званим ***прямим*** або ***довільним доступом:*** щоб визначити окремий елемент масиву, достатньо вказати його індекс.

Тепер зрозуміло, як у циклі перебирати різні значення елементів масиву: для цього достатньо змінювати їх індекси. А закон зміни індексів дуже простий - кожне наступне значення більше попереднього на одиницю. Дуже зручна закономірність!

Масив називається одновимірним, якщо для задання місцеположення елемента в масиві необхідно вказати значення лише одного індексу.

Це означає, що наш масив є ще й одновимірним.

А що станеться, якщо сформульовану на початку задачу ми поширимо з однієї доби на весь місяць? Логічною буде така заміна позначення елементів:

$$T[1, 1], T[1, 2], T[1, 3], \ldots T[1, 24],$$
$$T[2, 1], T[2, 2], T[2, 3], \ldots T[2, 24],$$
$$\ldots$$
$$T[30, 1], T[30, 2], T[30, 3], \ldots T[30, 24].$$

Перший індекс кожного елемента такого масиву означатиме добу, а другий - годину в цій добі. Тобто в загальному вигляді всі змінні можна записати так:

T[*i*, *j*де *i* = *1,2,* 3, ..., 30, *j* = 1,2, 3, ..., 24.

Масив називається двовимірним, якщо для **задання** місцеположення елемента в масиві необхідно вказати значення двох індексів.

Запам'ятайте, що у двовимірних масивах перший індекс завжди вказує на номер рядка, а другий - на номер стовпчика в цьому рядку!

Розмірність масивів у Pascal необмежена, вона визначається лише об'ємом пам'яті вашого комп'ютера.

Резонним буде **запитання:** а як же розташовуються масиви в пам'яті комп'ютера? Пояснення для одновимірних масивів дуже просте - всі вони розташовані в пам'яті підряд. Двовимірні масиви розташовуються дещо інакше - спочатку елементи першого рядка, потім другого і т. д. Розташування масивів більшої розмірності пояснюється аналогічно.

Залишилося з'ясувати, як пояснити програмі, що ви працюватимете з елементами, які утворюють масив значень.

Загальний вигляд опису масивів:

<ім'я змінної>: **array** [<межі зміни індексів>] **of** <тип>.

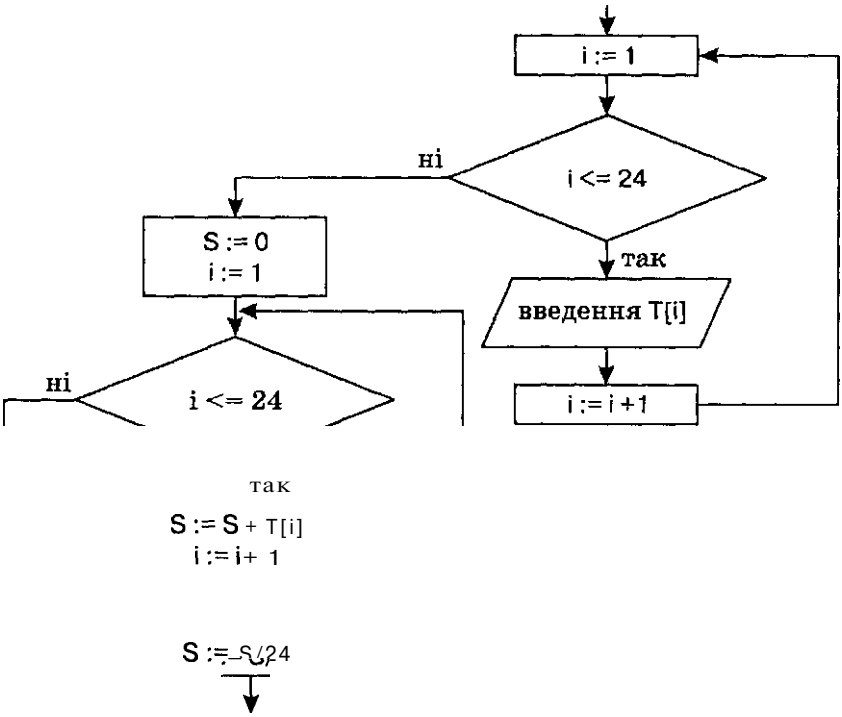
Наприклад,

```
var
  A: array [1 ..10] of real;
  B: array [1..100,1..100] of byte;
  C: array [1..100] of array [1 ..100] of byte.
```

Цікаво, що другий і третій приклади описують однакові масиви. Справді, адже будь-яку таблицю можна розглядати як послідовність рядків, де кожний рядок у свою чергу є також послідовністю. Звернення до елементів останнього масиву буде мати такий вигляд: *C*[*i*][*j*].

Зауваження. *По-перше,* межі індексів завжди вказуються через два символи **«.»**. *По-друге,* при розподілі пам'яті в описовій частині програми під масив буде зарезервовано стільки місця, скільки передбачає вказана кількість елементів масиву. Тому при виконанні програми ви можете використовувати кількість елементів не більшу, ніж описана в розділі змінних. *По-третє,* межі зміни індексів повинні бути сталими величинами, а не змінними, інакше невідомо буде, скільки місця необхідно відвести в пам'яті під такий масив.

Час переходити до завдання, яке було сформульоване на початку розділу. Розглянемо схему **алгоритму,** за допомогою якої можна визначити середнє арифметичне значення добової температури (мал. 20).



Мал. 20

Розглянемо ще таку задачу. Нехай дано таблицю здійснення рейсів *N* автобусами протягом *M* днів. Рейси, що відбулися, позначаються в таблиці цифрою 1, а ті, що з деяких причин не відбулися, - цифрою 0. Треба скласти програму, яка підраховує кількість рейсів, що не відбулися.

```
program voyage;
  var
    R                 :array[1.. 30, 1..100] of byte;
    i, j, n, m, count : integer;
begin
  write ('Задайте кількість автобусів: ');
    repeat
      readln (n)
    until (n > 0) and (n <= 30);
  write ('Задайте кількість днів: ');
    repeat
    j  readln (m)
    until (m > 0) and (m <= 100);
    writeln ('Задайте інформацію про здійснення рейсів: ');
      {Введення початкових даних }
```

```
    for i := 1 to n do
      for j := 1 to m do
      begin
        write (i, ' автобус, ', j, ' день: ');
          repeat
            readln (R[i, j])
          until (R[i, j] = 0) or (R[i, j] = 1)
      end;
    count := 0; {Підготовка змінної для накопичення суми}
    for i := 1 to n do  {Підрахунок кількості рейсів,}
    for j := 1 to m do {що не відбулися}
      if R[i, j] = 0 then count := Inc (count);
    writeln ('Кількість нездійснених рейсів: ', count:5);
    repeat until KeyPressed
end.
```

У цій програмі ми використали нову стандартну процедуру мови Pascal **Inc** (count). Її призначення - заміна значення параметра на **1.** Можливий ще такий варіант використання цієї процедури: **Inc** (n, step). У цьому разі збільшення параметра *n* відбуватиметься з кроком *step*. У Pascal існує альтернативна процедура **Dec,** яка зменшує значення вказаного параметра.

Якщо ви спробували виконати цю програму на комп'ютері, то, напевно, під час виправлення можливих помилок вам щоразу доводилося знову і знову задавати початкові дані. Можливості мови Pascal дають змогу уникнути такої незручності.

Повернемося до початку знайомства з Pascal, де вводилося поняття про розділ констант **const.** У цьому розділі ідентифікаторами позначаються сталі величини. Надалі в програмі замість цих сталих величин вказуватимемо лише відповідні їм ідентифікатори. Зручність полягає в тому, що якщо знадобиться поміняти значення якоїсь константи, то достатньо це зробити тільки в розділі const, не чіпаючи всієї програми.

Наприклад, можна розміри масиву задати таким чином:

```
const SizeLine = 100;
      SizeColumn = 100;
var A: array [1..SizeLine, 1..SizeColumn] of real;

repeat
  read (n, m)
until (n > 0) and (n <= SizeLine) and (m > 0) and (m <= SizeColumn);
```

Зверніть увагу на те, що в розділі констант стоїть символ «=», а не символ «:=».

Використовуючи в програмі константи, необхідно врахувати таку особливість: їх значення не можна змінювати під час виконання програми. Тобто ідентифікаторам, що визначають

константи, не можна присвоювати ніяких нових значень. Їх можна використовувати лише для обчислення інших значень.

Ми говорили про те, як незручно під час редагування програми щоразу заново задавати значення елементів масиву. Спробуємо і це зробити за допомогою розділу констант:

```
const
  A: array[1..3, 1..4] of real = ((1.5, 1.2, 2.1, -4.42),
                                  (2.4, 5.7, -1, 45.4),
                                  (-1.1,7,  45, -10));
```

З останнього прикладу видно, що в розділі констант можна задавати ще й тип. Такі константи називаються ***типізованими.*** Значення масивам задаються таким чином: в одновимірних масивах вони записуються через кому, у двовимірних - значення елементів рядків беруться у круглі дужки, для тривимірних — аналогічним чином. Відмінність типізованих констант від простих, для яких тип не вказується, полягає в тому, що їх значення можна змінювати під час виконання програми.

Якщо ж вас не цікавлять конкретні дані, які будуть надані елементам масиву під час виконання програми, то скористайтеся можливостями стандартної процедури Pascal **Randomize** та функції **Random** *(n),* що генерують випадкові числа. Параметр *n* (типу **Word)** у процедурі **Random** визначає праву межу інтервалу, в якому будуть визначатися випадкові числа (ліва межа завжди 0). Функція **Random** може задаватися і без параметра. У цьому разі вона генеруватиме те дійсне число в діапазоні [0; 1). А для того щоб випадкові числа в програмі з кожним її наступним запуском не повторювалися (хоча вони і випадкові, але послідовність цих чисел постійна), то скористайтеся процедурою **Randomize,** яка встановить початок відрахунку випадкових чисел залежно від поточного стану системного годинника вашого комп'ютера.

Фрагмент програми, що використовує випадкові числа, може виглядати так:

```
randomize;
for i:=1 to n do
  a[i] :=random (100);
```

Тепер робота з масивами не викликатиме незручностей.

Символьні масиви. Тип STRING

З рядковими величинами ми вже мали справу. Тепер подивимося на них з іншого боку. Адже це масиви символів. Справді, замість запису str: **string** можна записати str: **array** [1..255] **of char.** Найочевидніша відмінність у цих записах полягає в тому, що в першому випадку ви зможете ввести текст одним рядком, а в другому - лише по одній літері.

Давайте заглибимося у світ рядкових величин і дізнаємося багато корисного.

Наприклад, якщо вам наперед відома приблизна довжина символьного рядка, який буде введений користувачем під час виконання програми, то можна тип **string** дещо обмежити:

string [n],

де *n* - будь-яке число в межах 1..255.

У цьому разі під таку змінну буде відведено стільки байтів, якою є довжина описаної змінної.

Зверніть увагу, що тип **string** є простим типом, а тип **string** [n] - структурованим.

Насправді рядкові величини мають на один байт більше, ніж для зберігання символів рядка, тобто максимально 256 символів. Увесь секрет криється у байті з номером 0 (str [0]). Там міститься інформація про справжню кількість уведених символів рядкової величини. Виконайте таку операцію

write (ord (str[0]))

і ви отримаєте інформацію про те, скільки символів у рядку було насправді введено.

До окремих символів рядкової величини можна добутися так само, як і до елементів одновимірного масиву, вказавши порядковий номер необхідного символу.

Наприклад, якщо була виконана операція

str := 'алгоритм', то str [4] = 'о'.

Визначимо ***операції над рядковими величинами.***

Зчеплення. Ця операція дає змогу поєднувати кілька рядкових величин і позначається символом «+». Наведемо приклади:

str1 := 'алгоритм'; str2 := 'найкращий';
str1 + ' ' + str2 ⇒ 'алгоритм найкращий';
str2 +' ' + str1 ⇒ 'найкращий алгоритм'.

Порівняння. Оскільки всі символи кодуються своїми порядковими номерами, які ще називають ASCII-кодами (див. таблицю ASCII-кодів), то рядки можна порівнювати. При цьому будуть спочатку порівнюватися перші символи рядків, потім другі і т. д.

Наприклад:

'Алгоритм' < 'алгоритм', оскільки в таблиці спочатку йдуть великі літери, а потім малі;

'алгоритм' > 'алго', оскільки довжина першої величини більша за другу;

'алгоритм' <>'alhoritm';

'алг' = 'алг'.

Розглянемо ***функції та процедури***, які існують у Pascal для роботи з рядковими величинами.

Про довжину рядкової величини вам допоможе дізнатися стандартна функція Pascal, що має такий вигляд:

Length (<ім'я змінної>).

Результатом виконання цієї функції буде ціле число, а точніше типу **byte.**

Для визначення місцеположення шуканого фрагмента рядкової величини можна скористатися функцією

pos(S_find,S),

де S_find - шуканий підрядок, S - рядок.

Результат виконання цієї функції - тип **byte.** Якщо шуканого підрядка в рядку не існує, то функція **pos** повертає значення 0.

Наступна функція дасть той самий ефект, що й операція зчеплення:

concat (<список змінних>),

де список змінних - це змінні, значення яких будуть зчеплені. Результат виконання - тип **string.**

Якщо необхідно виділити деякий фрагмент рядкової величини, скористайтеся функцією

copy (S, Start, Len),

де S — рядкова величина, з якої виділяється фрагмент, **Start** - ціле значення, що визначає позицію початку фрагмента, який виділяється, Len — величина типу **integer,** що визначає кількість символів фрагмента. Результатом виконання цієї процедури є величина типу **string.** Будьте уважні: якщо виділяєте фрагмент лише в один символ, то результатом буде значення типу **string,** а не **char!**

Функція

UpCase (a)

дає змогу перетворити символ будь-якої літери а зі строкового вигляду в прописний. При цьому змінна а і результат роботи функції описуються типом **char.**

Тепер ознайомимося з процедурами. Першою буде процедура, що дає змогу вилучити деякий фрагмент з рядкової величини. Вона має вигляд:

delete (S, Start, Len),

де S - рядкова величина, з якої вилучається фрагмент, Start — порядковий номер першого символу фрагмента, що вилучається, Len — кількість символів фрагмента, що вилучається.

Так само можна вставити фрагмент у рядкову величину. Для цього існує процедура

insert (S, S_new, Start),

де S - рядкова величина, фрагмент, що вставляється, S_new - рядкова величина, в яку вставляється фрагмент і де отри-

мується результат вставлення, Start - порядковий номер символу в рядку S_new, перед яким вставляється фрагмент S.

Досить часто трапляється ситуація, коли рядкові величини мають числовий вигляд і виникає необхідність перетворення їх у числовий тип. У цьому вам допоможе процедура

val (S, V, ErrCode),

де S _рядкова величина, V - величина числового типу, який ви чекаєте отримати, **ErrCode** - код результату виконання процедури: 0, якщо перетворення відбулося успішно, або номер позиції в рядку S, який не піддається перетворенню.

Зворотна процедура до **val** виглядає таким чином:

str (V, S),

де V - величина числового типу, значення якої перетворюється у рядковий вигляд, S - результат перетворення рядкового типу.

Розглянемо приклад, який дає змогу записати задане слово навпаки ('алгоритм' ⇒ 'мтирогла').

```
program on_the_contrary;
  var
    S, S_new : string;
    i        : integer;
begin
  write ('Введіть слово: ');
  readln (S);
  S_new := '';
  for i := 1 to length (S) do
    S_new := S[i] + S_new; {дописування символу ліворуч}
  writeln ('Слово навпаки: ', S_new);
  repeat until KeyPressed
end.
```

До речі, якщо оператор S_new := S[i] + S_new замінити на оператор S_new := S_new + S[i], то в результаті виконання програми заданий текст не зміниться.

Варто розглянути ще один приклад роботи з рядковими величинами. Це масив рядкових величин. Слід нагадати, що значення типу **String** вводяться за допомогою процедури **Readln**, а не **Read**. Більше того, за один раз може бути введений лише один рядок.

Отже, розглянемо програму, яка вводить список рядків, а потім виводить їх в упорядкованому за алфавітом вигляді.

```
program Alphabetize;
  var
    S      : array [1.. 100] of string;
    i, j, n : integer;
begin
  write ('Введіть кількість слів: ');
  readln (n);
  writeln ('Введіть слова:');
```

```
    for i := 1 to n do
      readln (S[i]);
    writeln ('Список слів за алфавітом:');
    for i := 1 to n do
      begin
        k := 1;
        for j := 2 to n do
          if (S[j] < S[k]) and (S[j] <> ' ') then k := j;
        writeln (S[k]);
        S[k] := ''
      end
  end.
```

Класичні задачі для роботи з масивами

На практиці найчастіше зустрічаються пошукові задачі, тобто пошук заданого елемента в деякій послідовності значень. Але зрозуміло, що зручніше шукати необхідну інформацію там, де є певний порядок. Наприклад, пошук у бібліотечній картотеці, де формуляри упорядковані за алфавітом. Опираючись на це, можна сформулювати такі три класичні задачі.

Пошук заданого елемента в масиві. Нехай дано деякий масив значень $a_1, a_2, \quad a_n$ і значення величини x. Треба визначити, чи містить указаний масив величину x, і якщо так, то на якому місці.

Ця задача розв'язується послідовною, лінійною перевіркою всіх елементів масиву $a_1, a_2, \quad a_n$ на збігання з величиною x.

```
  program search;
    uses CRT;
    var a    : array [1..100] of real;
        x    : real;
        i, n : integer;
  begin
    write ('Задайте кількість елементів масиву: ');
    readln (n);
    writeln ('Задайте елементи масиву:');
    for i := 1 to n do
      readln (a[i]);
    write ('Задайте значення шуканої величини: ');
    readln (x);
    i := 1;
    while (a[i] <> x) and (i <= n) do
      i := i + 1;
    if i <= n
      then
        writeln ('Елемент ', x:5:2, ' знаходиться в масиві на ', i, ' місці');
      else
        writeln ('Елемент ', x:5:2,' в масиві відсутній');
    repeat until KeyPressed
  end.
```

Це не єдиний алгоритм пошуку заданого елемента в масиві.

Наприклад, якщо відомо, що масив є упорядкованим, то пошук у ньому можна здійснити набагато швидше. Такий алгоритм називається *алгоритмом бінарного пошуку*. Ідея методу така. Порівняємо значення шуканого елемента з елементом масиву, який розташований у його центрі. Може бути три варіанти такого порівняння:

1) шуканий елемент менший за даний;

2) шуканий елемент більший за даний;

3) шуканий елемент рівний даному.

Наступні дії в кожному з цих можливих варіантів такі:

1) звузити межі масиву, в якому відбувається пошук, відкинувши його другу половину, і продовжити пошук у першій;

2) звузити межі масиву, в якому відбувається пошук, відкинувши його першу половину, і продовжити пошук у другій;

3) завершити пошук шуканого елемента в масиві.

Для спрощення вважатимемо, що шуканий елемент обов'язково знаходиться в заданому масиві. В протилежному випадку наш алгоритм треба дещо модифікувати.

```
program binary_search;
  uses crt;
  var          a : array [1..100] of integer;
               x : integer;
      i, n, L, R, k : integer;
begin
  write ('Задайте кількість елементів масиву: ');
  readln (n);
  writeln ('Задайте елементи масиву:');
  for i := 1 to n do
    readln (a[i]);
  write ('Задайте значення шуканої величини: ');
  readln (x);
  L := 1; R := n;
  while (a[k] <> x) do
    begin
      k := (L+ R) div 2;
      if x < a[k] then R := k - 1;
      if x > a[k] then L := k + 1;
    end;
  writeln ('Елемент ', x, ' знаходиться в масиві на ', k, 'місці');
  repeat until KeyPressed
end.
```

Пошук мінімального (максимального) елемента в масиві. Нехай задано деякий масив $a_1, a_2, \ldots a_n$. Треба знайти мінімальний елемент цього масиву.

Пошук найменшого елемента здійснюватимемо за таким алгоритмом. Від початку до кінця масиву відкриватимемс

послідовно по одному елементу і на кожному кроці визначатимемо поточний найменший елемент. На першому кроці найменшим буде саме цей перший елемент a_1. На другому кроці порівняємо новий відкритий елемент a_2 з тим мінімумом, який ми вже маємо. Якщо новий елемент менший від того мінімального, що визначений на попередньому кроці, то запам'ятаємо його, інакше залишимо старий результат. Таким чином, дійшовши до кінця масиву, визначимо мінімальний елемент усього масиву.

```
program minimum;
  var    a : array [1..100] of real;
       min : real;
     i, n, k : integer;
  begin
    write ('Задайте кількість елементів масиву: ');
    readln (n);
    writeln ('Задайте елементи масиву:');
    for i := 1 to n do
      readln (a[i]);
    min := a[1]; k := 1;
    for i := 2 to n do
      if a[i] < min then
        begin
          min := a[i]; k := i
        end;
    write ('Мінімальний елемент ', min);
    writeln (' знаходиться в масиві на ', k, ' місці');
    repeat until KeyPressed
  end.
```

Запропонована програма не тільки знаходить найменший елемент, а й визначає, на якому місці в масиві він знаходиться.

Проаналізуємо декілька цікавих моментів стосовно даного алгоритму.

Наведений приклад алгоритму визначає перший мінімальний елемент у масиві, хоча в ньому може бути і кілька таких елементів. Наприклад, розглянемо масив 1, 2, 2, 1, 3, 1. Весь секрет ховається в умові a[i] < min. Адже новий елемент масиву буде запам'ятовуватися як найменший лише тоді, коли він строго менший за попередній.

Якщо в алгоритмі поміняти умову a[i] < min на умову a[i] <= min, то визначатиметься останній найменший елемент, оскільки новий елемент масиву буде запам'ятовуватися ще й тоді, коли він дорівнює попередньому.

Щоб розглянутий алгоритм виконував пошук найбільшого елемента, достатньо умову a[i] < min поміняти на a[i] > max.

Упорядкування масиву за зростанням (спаданням) Нехай дано деякий масив значень a_1, a_2, a_n. Треба перетво-

рити його таким чином, щоб виконувалося співвідношення $a'_1 \leqslant a'_2 \leqslant \ldots \leqslant a'_n$. Якщо ви спробували б навести порядок у картках, на яких написані деякі числа, то ви дійшли б такого висновку. Спочатку треба знайти мінімальний елемент у всьому заданому масиві і поміняти його з елементом, який поки що стоїть на першому місці, бо саме на цьому місці повинен стояти знайдений мінімальний елемент. Тепер уже перший елемент знаходиться на своєму місці, і ми можемо його не розглядати далі, тобто масив, у якому вестимемо подальший пошук зменшився на один елемент (перший). З новим масивом виконаємо ті самі дії: знайдемо в ньому мінімальний елемент і поміняємо його місцями з першим відкритим елементом (насправді він є другим елементом у початковому масиві). Так продовжуватимемо доти, поки на останньому кроці не залишаться два останні елементи. Якщо необхідно, поміняємо і їх місцями.

```
program order;
  var     a : array [1..100] of real;
      min, c : real;
    i, j, n, k : integer;
  begin
    write ('Задайте кількість елементів масиву: ');
    readln (n);
    writeln ('Задайте елементи масиву:');
    for i := 1 to n do
      readln (a[i]);
    for i := 1 to n - 1 do
      begin
        min := a[i]; k := i;
        for j := i + 1 to n do
          if a[j] < min then
            begin
              | min := a[j]; k := j
            end
        c := a[i]; a[i] := a[k]; a[k] := c;
      end;
    writeln('Упорядкованиймасив:');
    for i := 1 to n do
      write (a[i],' ');
    repeat until KeyPressed
  end.
```

Зрозуміло, щоб описаний алгоритм виконував упорядкування масиву за спаданням, необхідно логічний вираз a[j] < min замінити на a[j] > min, замінивши також при бажанні ідентифікатор *min* на ідентифікатор *max*.

Наведений алгоритм упорядкування послідовності є далеко не єдиним. Існують алгоритми впорядкування вже частково

впорядкованих послідовностей, алгоритми, ефективніші за часом виконання, але такі, що використовують більше пам'яті комп'ютера, алгоритми, що впорядковують дуже великі за об'ємом послідовності, які не можна повністю завантажити в пам'ять комп'ютера тощо.

Неважко здогадатися, що всі ці програми можна використовувати і для рядкових величин. Необхідно лише створити масив із елементів типу **string.**

Запитання для самоконтролю

1. Що називається масивом?
2. Що називається індексом масиву?
3. Які типи змінних називаються простими, а які структурованими?
4. Дайте означення одновимірних і двовимірних масивів. Як елементи цих масивів розташовуються в пам'яті комп'ютера?
5. Яку важливу роль можуть відіграти константи під час роботи з масивами?
6. У чому полягає ідея пошуку мінімального елемента в масиві?
7. Яким чином використовується пошук мінімального (або максимального) елемента масиву для його впорядкування?
8. Чим пояснити те, що рядкові величини розглядаються в розділі масивів?
9. Яка максимальна довжина символьних масивів?
10. Де знаходиться інформація про кількість заданих елементів символьного рядка?
11. Назвіть функції для роботи з рядковими величинами та призначення їх параметрів.
12. Назвіть процедури для роботи з рядковими величинами та призначення їх параметрів.

Не припускайтеся помилок!

Якщо кількість елементів описаного масиву завелика і не може бути розміщена в пам'яті комп'ютера, то ви отримаєте повідомлення про помилку:

Error 29: Ordinal type expected.

Якщо при визначенні розмірів масиву в розділі **var,** були використані ідентифікатори, які не є константами, то ви отримаєте повідомлення про помилку:

Error 3: Unknown identifier.

Якщо індекси масиву не описані в розділі змінних, то вам буде видано повідомлення про помилку:

Error 3: Unknown identifier.

Якщо індекс масиву виходить за межі заданих розмірів, то ви отримаєте повідомлення про помилку:

Error 76: Constant out of range.

Якщо розмірність указаного масиву не збігається з його описом у розділі змінних, то ви отримаєте попереднє повідомлення про помилку.

Якщо результат функції **copy** присвоюється змінній типу **char,** то з'явиться повідомлення про помилку щодо незбігання типів **(Error 26).**

Якщо в результаті використання вкладених циклів ви отримуєте результат, якого не чекали, розберіться з вкладеними циклами за допомогою покрокового виконання програми **(F8, Ctrl+F4)!**

Виконайте завдання

Вправи

292. Які групи наведених нижче змінних з індексами визначають один і той самий елемент одного і того самого масиву за умови, що змінні *i* та у набувають цілих додатних значень:

1) *x[i, j* +0, −5];
2) *a*[6];
3) *x[i,* 2, −5];
4) *x[i, j,* - 5];
5) *a*[0, 6];
6) *a*[6 + 0];
7) *x[i* +0, *j,* −5];
8) *a*[10 − 4];
9) *x*[0, y, - 5];
10) *x[i* +0, *j* +0];
11) *a*[6*2 − 6];
12) *b*[0, - 2**j,* 1];
13) *ax*[0, 6];
14) *x[i, j,* - 5];
15) *b*[0, *j*(− 2), 1];
16) *b*[0, 2, 1];
17) *ax*[0, − 8+ 2];
18) *x*[0, *j* +0, −5];
19) *x*[*i*+ 0, 1, − 5];
20) *ax*[0.01, 6.01]?

293. Розглядаючи наведені нижче вектори як одновимірні масиви, записати їх у вигляді послідовності змінних з індексами:

1) (m_i), де $i = 1, 2$ ... 10;
2) (x_k)де k = 0, 1, 5;
3) (b_j), де $j = -8, -7, \ldots, -1$;
4) (c_j), де у $= -3, -2,$ 3;
5) (d_i), де i = 4, 5, 10;
6) (M_n), де n = - 3;
7) (A_n)де n = 0, 1, 7;
8) (Q_l), де $l = n + 1, n + 2,$ $n + 5$;
9) (P_k), де $k = i - 2, i - 1, \ldots, i + 1$;
10) (B_i), де і $= k, k + 1,$ $k + 6$.

294. Розглядаючи наведені нижче матриці як двовимірні масиви, записати їх у вигляді послідовності змінних з індексами:

1) (a_{ij}), де i= 1, 2, 3; j= 1, 2, 3;
2) де i = 1,2; j = 1,2, 3, 4;
3) (M_{kl}), де κ = - 8, - 7; l = 8, 9, 10;
4) (X_{km}), де k = 3, 4, 5; те = - 1, 0, 1;
5) (b_{ij}), де і = га + 1, n + 2; у = - 7, - 6, - 5;
6) (Q_{im}), де i = 1; m = 1, 2, 3, 4, 5;

7) (P_{nm}), де $n = 1, 2, 3, 4, 5$; $m = 3$;
8) (B_{km}), де $k = 0$; $m = 0$;
9) (D_{ij}), де $i =$ л - 7, л - 6; $j =$ - л + 1, - л + 2, - л + 3.

295. Нехай нижня і верхня межі індексів одновимірного масиву B відповідно дорівнюють - 5 і 10. Обчислити порядкові номери наступних елементів масиву:

1) $B[\sim 5]$;	5) $B[8]$;	9) $B[6]$;
2) $B[10]$;	6) $B[-4]$;	10) $B[5]$;
3) $B[0]$;	7) $B[7]$;	11) $B[-0]$;
4) $B[3]$;	8) $B[-1]$;	12) $B[+0]$.

296. Нехай нижня і верхня межі індексів двовимірного масиву Z відповідно дорівнюють 1 та 8 - по першому виміру (номер рядка), 1 і 4 - по другому виміру (номер стовпчика). Обчислити порядкові номери елементів масиву Z:

1) $Z[2, 1]$;	5) $Z[1, 4]$	9) $Z[7, 2]$;
2) $Z[7, 4]$;	6) $Z[1, 1]$	10) $Z[1, 2]$;
3) $Z[4, 2]$;	7) $Z[8, 3]$	11) $Z[7, 3]$;
4) $Z[5, 2]$;	8) $Z[3, 2]$	12) $Z[5, 4]$.

297. Нехай нижня і верхня межі індексів одновимірного масиву S відповідно дорівнюють - 10 і 32. Визначити значення індексів елементів масиву S, порядковими номерами яких є:

1) 1;	6) 17;	11) 8;	16) 27
2) 3;	7) 9;	12) 25;	17) 21
3) 5;	8) 10;	13) 39;	18) 19
4) 12;	9) 42;	14) 16;	19) 40
5) 32;	10) 37;	15) 14;	20) 11.

298. Нехай нижня і верхня межі індексів двовимірного масиву M відповідно дорівнюють 1 і 7 — по першому виміру та - 1 і 12 - по другому виміру. Визначити значення індексів елементів масиву S, порядковими номерами яких є:

1) 2;	6) 57;	11) 19;	16) 41;
2) 54;	7) 84;	12) 70;	17) 5;
3) 11;	8) 8;	13) 48;	18) 69;
4) 37;	9) 61;	14) 39;	19) 30;
5) 13;	10) 27;	15) 21;	20) 40.

299. Нехай елементи одновимірного масиву $A[1..10]$ набувають відповідно значень $-5, -3, -1, 1, 3, 5, 7, 9, 11, 13$. Які значення буде надруковано в результаті виконання таких операторів:

1)
```
for i := 1 to 10 do
    writeln (A[i]);
```

2)
```
for i := 1 to 5 do
    writeln (A[i + 5]);
```

3)
```
for i := 10 downto 1 do
    writeln (A[i]);
```

4)
```
for i := 1 to 5 do
    writeln (A[2*i]);
```

```
5) i := 1;
   while A[i] < 0 do
      begin
         i:=i + 1;
         writeln (A[i])
      end;
```

```
6) i := 1;
   repeat
      i := i + 1;
      writeln (A[i])
   until A[i] >= 0;
```

```
7) i  = 10;
   while A[i] > 0 do
      i := i – 1;
   writeln (A[i]);
```

```
8) i := 1;
   repeat
      writeln (A[i]);
      i :=i + 1
   until A[i] < 0;
```

```
9) i := 10;
   while A[i – 9] < 0 do
      begin
         writeln (A[i]);
         i :=i + 1
      end;
```

```
10) i := 1;
    repeat
       writeln (A[2*i - 1]);
       i := i+1
    until i >= 10?
```

Серйозні розважалки

300. Барон Мюнхаузен, вийшовши на екологічно чисте полювання, зарядив свою рушницю кісточками вишень. Після того як він вдало влучив поміж роги N оленям, в яких попало відповідно k_1, k_2, k_N кісточок, у них на головах виросли чудові молоді вишеньки. Скільки нових саджанців зміг подарувати барон Мюнхаузен садівникам-дослідникам?

301. Мама розвела цілу оранжерею кактусів, деякі з яких були гольчасті, а решта - голі. Маленька донечка Яринка вирішила, що голки на кактусах - це надто зухвало, і тому старанно поголила їх бритвою. Добре, що в мами залишився записник, в якому всі кактуси були позначені кількістю голочок $a_1, a_2, \ldots a_n$ (голі кактуси були позначені **0**). Скількох кактусів не торкнулася рука юної перукарки?

302. Середню групу дитячого садочка вивели на прогулянку. Скільки дівчаток і скільки хлопчиків видно з-за паркана, якщо зріст хлопчиків задається у сантиметрах від'ємними числами, а дівчаток - додатними у вигляді цілих значень $a_1, a_2, \ldots a_n$? Крім того, в усіх дівчаток на голівках зав'язані бантики заввишки **10** см, а висота паркана — H см.

303. Маленький онучок вирішив допомогти бабусі підстригти квіти на її дорогоцінному квітнику, зрізавши лише бутони та квіточки на них. На щастя, кмітливий хлопчик зрізав лише ті квіти, які були заввишки від h_1 до h_2 см від землі. Скільком квіточкам повезло бути підстриженими, якщо висота їх у сантиметрах становить $a_1, a_2, \ldots a_n$?

304. Коли барон Мюнхаузен захотів пообідати, він прив'язав до довгої мотузки шматок сала і закинув його високо в повітря. Зграя диких гусей, що пролітала тим часом над помешканням барона, зацікавилася незвичним предметом, і найстарший гусак, що очолював зграю, проковтнув його. Не встиг він насолодитися відчуттям ситості, як шматок сала проскочив через нього і зник у дзьобі другого гусака і т. д. Тепер доля обіду барона Мюнхаузена залежала лише від довжини мотузки! Скільки кілограмів підсмаженої гусятини було подано на обід баронові Мюнхаузену, якщо довжина мотузка становила L см, N гусей летіли на відстані h см один від одного, довжина кожного з них дорівнює k см, а вага цих гусей у кілограмах становила $m_1, m_2, \quad m_N$?

305. Маленький Дмитрик кожний місяць виростає на 2 см, а в бабусі в комірчині облаштовано полички з різними ласощами - варенням, джемом, повидлом. Акуратистка бабуся записувала висоту і ставила наступний порядковий номер у свій записник кожної нової полички в тій послідовності, як вона з'являлася в комірчині стараннями дідуся. Висота цих поличок була a_1, a_2, ... , a_n (у сантиметрах). Нові полички дідусь весь час прибивав, де йому заманеться, - вище, нижче і між тими, що вже були. З'ясувати, через скільки місяців до яких поличок, враховуючи їх порядок запису в бабусиній книжці, добереться Дмитрик (наприклад, спочатку до п'ятої, занотованої у записнику, потім до другої і т. д.), якщо він відкрив для себе бабусину комірчину, коли його зріст був H_1 см, а дорослий Дмитрик до H_2 см.

306. Велосипедист-початківець Павлуша виїхав на широку дорогу. Але їхати інакше, ніж за законом синусоїди, йому ніяк не вдавалося. Юний спортсмен стартував у точці x_0 на осі Ox, а центри основ стовпів знаходяться у точках $x_1, x_2, \quad x_n$ на цій же осі, яку перетинає синусоїда руху велосипедиста. Скільки стовпів трапляться на шляху Павлуші, якщо шириною стовпа можна знехтувати?

307. Лихач-водій Василь Іванович вирішив поставити рекорд перегонів між містом Мляшем та містом Пляшем. Спочатку він запасся картою розташування заправок пальним на шляху між Мляшем та Пляшем, де відстань між заправками була позначена числами $a_1, a_2, \quad a_n$, а потім побився об заклад зі своїми друзями, що встановить рекорд за T годин. Одна проблема мучила Василя Івановича: об'єм бака для пального складав K літрів, а запасної каністри у нього не було. Витрати пального в дорозі становили 1 л на 10 км шляху. Для економії часу Василь Іванович на кожній заправці визначав, чи варто витрачати 10 дорогоцінних хвилин на дозаправку, чи можна доїхати до наступної заправки на залишках пального у баку. Чи зможе Василь Іванович встановити рекорд, якщо перша заправка знаходиться у місті Мляші, остання - у місті Пляші, а сам Василь

Іванович їде зі сталою швидкістю v км/год? Чи може таке статися, що Василь Іванович взагалі не доїде до міста Пляша?

308. Лікар-психіатр призначив Сергійкові лікування від лайливих слів. Виконуючи поради психіатра, хворий повинен був записувати у таблицю по днях упродовж місяця кількість використаних ввічливих слів «дякую», «пробачте», «прошу». У який день місяця друзям Сергійка повезло на ввічливі слова найбільше? Якого дня місяця у хлопчика був найгірший настрій? Які ввічливі слова Сергійкові найбільше до вподоби?

Задачі

Одновимірні масиви

309. Дано натуральне число A. Складіть програму, що представляє його у вигляді многочлена. Наприклад,

$$123 \Rightarrow 1*10^2+2*10^1+3*10^0.$$

310. Випадковим чином[1] заповнити цілочисловий одновимірний масив A, що складається з десяти елементів, та вивести його значення на екран монітора:

1) в рядок; 2) у стовпчик.

311. Дано одновимірний масив цілих чисел $A[i]$, де $i = 1, 2, \ldots, n$. Вивести значення елементів масиву:

1) у зворотному порядку;
2) з парними індексами;
3) з непарними індексами;
4) що є недодатними числами;
5) що є невід'ємними числами;
6) що є парними числами;
7) що є непарними числами.

312. Дано дійсні числа $a_{1951}, a_{1952}, \ldots, a_{2000}$ — кількість опадів (у мм), що випали в місті за останні 50 років минулого століття. Обчислити середню кількість опадів за цей період і щорічне відхилення від середнього значення.

313. Дано одновимірний масив цілих чисел $A[i]$, де $i = 1, 2, \ldots, n$. Надрукувати окремо додатні та окремо від'ємні елементи заданого масиву і вказати їх індекси.

314. Задані натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots, a_n$. У заданій послідовності визначити кількість сусідств:

1) двох додатних чисел;
2) двох чисел різного знака;
3) двох нульових членів;
4) трьох нульових членів.

[1] У задачах, які передбачають введення великої кількості будь-яких значень, зручно користуватися генератором випадкових чисел.

315. Дано натуральне число n і цілі числа $a_1, a_2, \ldots a_n$ та $b_1, b_2, \ldots b_n$. Визначити значення $c_1, c_2, \ldots c_n$, якщо:

1) $c_i = a_i + b_i$;
2) $c_i = \frac{a_i}{b_i}, (b_i \neq 0)$;
3) $c_i = \sin a_i + \cos b_i$;
4) $c_i = i!(a_i + b_i)$.

316. Задані натуральне число n і цілі числа $a_1, \ldots, a_n$ та $b_1, \ldots, b_n$. Обчислити значення елементів $c_1, \ldots c_n$ і визначити кількість від'ємних серед них, якщо відомо, що:

1) $c_i = 2^i a_i + i! b_i$;
2) $c_i = \frac{a_1 + a_2 + \ldots + a_i}{n - i + 1} + b_{i+1} + b_{i+2} + \ldots + b_n$.
3) $c_i = \sum (a_j + b_j)$;
5) $c_i = \frac{a_1 + a_2 + \ldots + a}{b_{i+1} + b_{i+2} + \ldots + b_n}$.

317. Дано дійсні числа $a_1, a_2, \ldots a_{20}$. Отримати числа $b_1, b_2, \ldots b_{20}$, де ft, - середнє арифметичне чисел $a_1, a_2, \ldots a_n$ (i = 1, 2 ... 20).

318. Дано дійсні числа $a_1, a_2 \ldots a_{30}, b_1, b_2, \ldots, b_{30}$. Обчислити:

1) $(a, + b_{30})(a_2 + b_{29}) \ldots (a_{30} + b_1)$;
2) $a_1 a_{30} + a_2 a_{29} + \ldots + a_{30} a_1$;
3) $\frac{a_1 b_1 + a_2 b_2 + \ldots + a_{30} b_{30}}{a_1 + a_2 + \ldots + a_{30}}$
4) $\frac{a_1 b_1 + a_3 b_3 + \ldots + a_{29} b_{29}}{a_2 b_2 + a_4 b_4 + \ldots + a_{30} b_{30}}$;
5) $\frac{a_1 b_{30} + a_2 b_{29} + \ldots + a_{30} b_1}{b_1 b_2 \ldots b_{30}}$.
6) $\frac{a_1^2 + a_2^2 + \ldots + a_{30}^2}{\sqrt{|b_1| + |b_3| + \ldots + |b_{29}|}}$.

319. Дано натуральне число n і дійсні числа $a_1, a_2, \ldots a_{2n}$. Обчислити:

1) $a_1, a_2, \ldots, a_n, a_{2n}, a_{2n-1}, \ldots, a_{n+1}$;
2) $a_n, a_{n-1}, \ldots a_1, a_{n+1}, a_{n+2}, \ldots, a_{2n}$;

3) $a_{n+1}, a_{n+2}, \ldots, a_{2n}, a_1, a_2, \ldots, a_n$;

4) $a_1, a_{n+1}, a_2, a_{n+2}, a_n, a_{2n}$;

5) $a_1, a_{2n}, a_2, a_{2n-1} a_3, \ldots, a_n, \ldots, a_{n+1}$;

6) $a_{2n}, a_1, a_{2n-1}, a_2, \ldots, a_{n+1>} a_n$;

7) $a_{2n}, a_n, a_{2n-1}, a_{n-1}, \ldots, a_{n+1} a_1$.

320. Біля прилавка в магазині вишикувалася черга покупців. Час обслуговування продавцем і-го покупця дорівнює t_i (i = 1, *n)*. Нехай задані натуральне *n* і дійсні числа $t_1, t_2, \ldots, t_n$. Обчислити $c_1, c_2, \ldots c_n$, де c_i - час перебування і-го покупця в черзі *(i* = 1, *n)*. Вказати номер покупця, для обслуговування якого продавцю знадобився найменший час.

321. Дано одновимірний масив цілих чисел *A[i],* де *i* = 1, 2, *n*. Визначити:

1) скільки разів максимальний елемент зустрічається в даному масиві;
2) порядковий номер першого найбільшого елемента;
3) порядковий номер останнього найменшого елемента;
4) яких елементів більше - максимальних чи мінімальних.

322. Нехай дано натуральне число *n* і послідовність попарно різних дійсних чисел $a_1, a_2, \ldots a_n$. У даній послідовності поміняти місцями:

1) найбільший член з першим по порядку (якщо їх індекси збігаються - повідомити про це);
2) найменший член з останнім по порядку (якщо їх індекси збігаються - повідомити про це);
3) найбільший і найменший члени;
4) найменший член із членом, що стоїть на *k*-му місці.

323. Дано одновимірний масив завдовжки *n*. «Стиснути» його, вилучивши з нього всі від'ємні елементи.

324. Дано одновимірний масив завдовжки *n*. Розділити його на два нові масиви так, щоб у першому опинилися додатні елементи, а в другому - від'ємні.

325. Дано два одновимірні масиви. «Злити» їх у третій новий масив, у який спочатку ввійдуть елементи першого масиву, а потім - другого, зберігаючи свою первісну послідовність.

326. Дано два однакові за довжиною одновимірні масиви. «Злити» їх у третій новий масив, чергуючи елементи першого та другого масивів.

327. Дано два одновимірні масиви. Утворити новий масив, в якому спочатку розташовані всі додатні елементи першого масиву, потім додатні елементи другого масиву, далі - всі від'ємні елементи першого масиву, потім від'ємні елементи другого.

328. Дано два одновимірні масиви. Утворити новий масив, в якому спочатку розташовані всі елементи першого масиву в їх

первісній послідовності, а потім елементи другого масиву в зворотній послідовності.

329. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Отримати відповідні члени нової послідовності $b_1, b_2, \ldots b_n$ за такими правилами:

1) збільшити всі відмінні від максимального члени послідовності на $\max(a_1, a_2, \ldots a_n)$;
2) усі від'ємні члени замінити мінімальним членом, а всі додатні - максимальним;
3) усі члени, менші за середнє арифметичне, замінити мінімальним членом, а всі більші - максимальним.

330. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Перетворити дану послідовність, вилучивши з неї всі члени, що дорівнюють найменшому значенню.

331. Дано два одновимірні масиви *x[i]* та *y[i]*, де $i = 1, 2, \ldots n$. У масиві *x[i]* задано стаж працівників деякого підприємства, а у масиві *y[i]* — відповідно розмір їх заробітної платні. Визначити:

1) яка заробітна платня працівника з найбільшим стажем роботи (якщо їх кілька - вивести інформацію про всіх таких працівників);
2) який стаж працівника з найбільшою заробітною платнею (якщо їх кілька - вивести інформацію про всіх таких працівників);
3) кількість працівників з найбільшою заробітною платнею та їхні порядкові номери в масиві;
4) кількість працівників з найбільшим стажем роботи та їхні порядкові номери в масиві;
5) упорядкувати за зростанням інформацію про заробітну платню і відповідно до цього зробити зміни в масиві, який містить інформацію про стаж роботи працівників;
6) упорядкувати за спаданням інформацію про стаж роботи працівників підприємства і відповідно до цього зробити зміни в масиві, який містить інформацію про заробітну платню.

332. У деяких видах спортивних змагань виступ кожного спортсмена незалежно оцінюється кількома суддями, потім з усієї сукупності оцінок вилучаються найвища і найнижча, а для тих оцінок, що залишилися, визначається середнє арифметичне, яке й заліковується спортсмену. Якщо найвищу оцінку виставили кілька суддів, то із сукупності оцінок вилучається лише одна така оцінка (аналогічно поводяться з найнижчими оцінками).

Нехай n ($n \geqslant 3$) суддів виставили одному зі спортсменів відповідні оцінки: $a_1, a_2, \ldots a_n$. Визначити, яка остаточна оцінка піде в залік цьому спортсмену.

333. Дано одновимірний масив цілих чисел *A[i],* де $i = 1, 2, \ldots n$ та $0 < A[i] \leqslant n$. Надрукувати елементи масиву за таким правилом:

1) першим елементом завжди є $A[1]$;
2) значення поточного виведеного елемента масиву є порядковим номером наступного елемента цього масиву.

Процес завершується тоді, коли буде надруковано останній елемент масиву або ж порядковий номер виведеного елемента дорівнюватиме його значенню.

334. Визначити довжину інтервалу, на який припадають значення членів цілочислової послідовності $x_1, x_2, \ldots x_n$ ($n > 1$, ціле).

335. Дано дійсні числа K, L ($K \leqslant L$), натуральне число n та послідовність цілих чисел $a_1, a_2, \ldots a_n$. Визначити, які цілі числа з інтервалу $[K; L]$ є членами послідовності $a_1, a_2, \ldots a_n$.

336. Дано натуральне число n і послідовність цілих чисел $a_1, a_2, \ldots a_n$. Визначити в порядку зростання всі цілі числа з відрізка $[K; L]$, що не входять у послідовність $a_1, a_2, \ldots a_n$, якщо K - мінімальне значення цієї послідовності, а L - відповідно максимальне значення.

337. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots a_n$. З'ясувати, чи буде ця послідовність неспадною, якщо в ній замінити всі від'ємні члени їх модулями.

338. Дано непарне ціле n ($n > 1$) та дві послідовності чисел $a_1, a_2, \ldots a_n, b_1, b_2, \ldots b_n$. Визначити, в якій із цих послідовностей після впорядкування посередині стоятиме більший член.

339. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Після впорядкування цієї послідовності за спаданням визначити:

1) скільки членів заданої послідовності залишилося на своїх місцях;
2) порядкові номери членів заданої послідовності, які перемістилися на інше місце;
3) на якому місці в заданій послідовності знаходився член, що стоїть тепер на k-му місці.

340. Дано натуральне число n і дійсні числа x, $a_1, a_2, \ldots a_n$ ($a_1 \leqslant a_2 \leqslant \ldots \leqslant a_n$).

1) Визначити індекси членів послідовності $a_1, a_2, \ldots a_n$, між якими треба поставити число x таким чином, щоб не порушити її неспадання.
2) Утворити послідовність $b_1, b_2, \ldots b_{n+1}$, членами якої є числа x, $a_1, a_2, \ldots a_n$ і для якої виконується умова $b_1 \leqslant b_2 \leqslant \ldots \leqslant b_{n+1}$.

341. Дано натуральні числа n, m ($n < m$) та послідовності дійсних чисел $a_1, a_2, \ldots a_n$, $b_2, \ldots b_m$. Визначити:

1) члени, які належать обом послідовностям;
2) сукупність значень членів, з яких складаються обидві послідовності;

3) чи всі члени послідовності $a_1, a_2, \ldots a_n$ входять у послідовність $b_1, b_2, \ldots b_m$;
4) які члени послідовності $a_1, a_2, \ldots, a_n$ не входять у послідовність $b_2, \ldots b_m$;
5) які члени послідовності $b_1, b_2, \ldots, b_m$ не входять у послідовність $a_1, a_2, \ldots a_n$.

342. Дано натуральне число n і послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Визначити:
1) усі числа, що входять у послідовність по одному разу;
2) числа, взяті по одному з кожної групи однакових чисел;
3) кількість груп однакових членів послідовності;
4) кількість різних членів послідовності;
5) кількість членів, що входять до послідовності більш як один раз.

Двовимірні масиви, таблиці, матриці[1]

343. Дано натуральні числа n, m та випадкові[2] дійсні числа, що утворюють прямокутну таблицю $A[i, j]$, де $i = 1, 2, \ldots n$; $j = 1, 2, \ldots m$.

Роздрукувати[3]:
1) елементи таблиці, розташувавши їх по рядках і стовпчиках відповідно один під одним;
2) у рядок елементи, розташовані в першому стовпчику;
3) у рядок елементи, розташовані в останньому стовпчику;
4) у рядок елементи, розташовані на головній діагоналі (обидва індекси яких збігаються);
5) у рядок елементи, розташовані на бічній діагоналі;
6) елементи головної діагоналі, розташувавши їх на своїх місцях у таблиці і залишивши місця для інших елементів порожніми;
7) елементи бічної діагоналі, розташувавши їх на своїх місцях у таблиці і залишивши місця для інших елементів порожніми;
8) елементи, розташовані по зовнішньому контуру таблиці, залишивши місця для інших елементів порожніми.

---

[1] У різних розділах математики та інших наук дані, що мають вигляд інформації, заданої як послідовність рядків і стовпчиків, називають по-різному: матриці - у вищій алгебрі, таблиці - у розрахункових задачах, масиви - у програмуванні.

[2] У задачах, які передбачають введення великої кількості довільних початкових даних, для задання інформації зручно використовувати генератор випадкових чисел.

[3] У задачах, які передбачають роботу з таблицями значень, результати для кращої читабельності зручно виводити у вигляді справжньої таблиці, розташовуючи рядок під рядком, а числа у стовпчиках одне під одним.

344. Дано натуральні числа л, *m* і матриці цілих чисел *A[i, j], B[i,* у], де $i = 1, 2,$ л; $j = 1, 2,$ *m*. Обчислити значення елементів матриці $C[i,j]$, якщо відомо, що:

1) $C_{ij} = A_{ij} + B_{ij}$;
2) $C_{ij} = A_{ij}(i + j) + B_{ij}(i^2 + j^2)$;
3) $C_{ij} = \sin A_{ij} + \cos B_{ij}$.

345. Дано натуральні числа n і m. Обчислити значення елементів матриці $C[i, j]$ $(i = 1, 2,$ л; у $= 1, 2, \ldots, m)$, якщо:

1) $C_{ij} = \begin{cases} i + j, & \text{якщо і} < \text{у}; \\ i^2 - j^2, & \text{в інших випадках}; \end{cases}$

2) $C_{ij} = \begin{cases} \text{якщо } i < j; \\ 2, & \text{якщо } i = \text{у}; \\ i + j, & \text{в інших випадках}; \end{cases}$

3) $C_{ii} = \begin{cases} i^2 + j^2, & i < j; \\ (i - j)^3 & i = j; \\ \sin \mathrm{I} + \cos \text{у}, & \text{в інших випадках}. \end{cases}$

346. Дано натуральні числа *n, mi* дійсні елементи матриць *A[i, j], B[i,* у], де $i = 1, 2,$ л; у $= 1, 2,$ *m*. Визначити значення елементів матриці $C[i,$ у], якщо відомо, що:

A_{ij}, якщо $i \leqslant j$;

B_{ij}, якщо і $> j$;

2) $C_{ij} = \begin{cases} i^2 + j^2 + 2, & \text{якщо } i < \text{у}; \\ \frac{1}{(i + j)^3} & \text{якщо і} = \text{у}; \\ \sin^2(A_{ij}) + \sin^2(\text{Ву}), & \text{в інших випадках}; \end{cases}$

$\sin(A_{ij} 4 - B_{ij})$, якщо $A_{ij} * B_{ij} \leqslant 0$ та $A_{ij} \neq B_{ij}$;

3) $C_{ii} = \begin{cases} 1, & \text{якщо } A_{ij} = B_{ij}; \\ \frac{A_{ii} + B_{ii}}{2i + 3j} \end{cases}$

347. Дано цілі числа a_1, a_2, a_3. Побудувати цілочислову матрицю $b[i,j]$, де i, у $= \overline{1, 2, 3}$, для якої $b_{ij} = a_i - 3a_j$.

348. Дано дійсні числа $a_1, a_2, \quad a_{10}, b_1, b_2, \quad b_{20}$. Обчислити елементи дійсної таблиці $c[i, j]$, де $i = 1, 2, \quad 10$, у $= 1, 2, \quad 20$ за формулою:

$$a_i + 1$$

349. Дано натуральне число n. Обчислити елементи квадратної таблиці a_{ij}, де $i, j = 1, 2,$ за такою формулою:

$$a_{ij} = i^3 + \sin y - 5.$$

350. Дано натуральне число n. Визначити кількість додатних і кількість від'ємних елементів таблиці a_{ij}, де $i, j = 1, 2, \quad n$, якщо:

1) $a_{ij} = \sin (i + j)$;

2) $a_{ij} = \cos (i^2 + n)$;

3) $a_{ij} = \sin \frac{i+j}{2} + \cos \sqrt{i * j}$.

351. Дано таблицю $B[i, j]$, де $i = 1, 2, \quad n$, у $= 1, 2, \quad m$. Надрукувати:

1) суму елементів кожного стовпчика;

2) середнє арифметичне кожного стовпчика.

352. Дано квадратну дійсну таблицю порядку n. Усі максимальні елементи заданої таблиці замінити нулями.

353. Елементи цілочислової прямокутної матриці розміром $n \times m$ задано випадковим чином. Надрукувати пари індексів:

1) першого максимального елемента;

2) останнього мінімального елемента;

3) усіх максимальних елементів.

354. Дано прямокутну цілочислову таблицю порядку $10 \times n$. Визначити середнє арифметичне максимального і мінімального її значень.

355. Дано квадратну дійсну таблицю розмірності n. Обчислити кількість:

1) входжень заданого елемента x;

2) максимальних елементів;

3) мінімальних елементів.

356. Дано цілочислову прямокутну таблицю порядку $n \times m$. Усі елементи таблиці, менші за середнє арифметичне її значень, замінити на - 1, більші - на 1.

357. Дано таблицю $B[i, j]$, де $i, j = 1, 2, \quad n$. За допомогою заданої таблиці знайти елементи масиву C_j (у $= 1, 2, \ldots, n$), значення яких дорівнюють:

1) сумі елементів відповідних рядків таблиці;

2) сумі елементів відповідних стовпчиків таблиці;

3) мінімальним елементам відповідних стовпчиків таблиці;

4) сумі максимального і мінімального значень відповідних рядків таблиці;

5) сумі першого й останнього елементів відповідних стовпчиків таблиці.

358. У даній дійсній матриці розмірністю 6×9 знайти суму елементів рядка, що містить найбільший елемент. Вважається, що такий елемент в матриці єдиний.

359. Дано таблицю $A[i, j]$, де $i = 1, 2, \ldots n$; $j = 1, 2, \ldots m$. У цій таблиці поміняти місцями елементи:

1) першого й останнього рядків;

2) першого й останнього стовпчиків;

3) k-го та l-го рядків;

4) k-го рядка і l-го стовпчика.

360. Дано квадратну матрицю розмірності n. Надрукувати:

1) індекси елементів, розташованих на бічній діагоналі;

2) суму елементів бічної діагоналі;

3) елементи бічної діагоналі в порядку зростання.

361. Знайти різницю між найменшим і найбільшим значеннями елементів головної діагоналі квадратної матриці розмірністю n.

362. Розклад руху N маршрутів автобусів по M зупинках міста протягом однієї години записаний в таблицю розміром $N\times M$. Вважатимемо, що кожний автобус може побувати на деякій зупинці лише один раз, а значення 0 в таблиці означає, що даний автобус на цій зупинці протягом цього часу не зупиняється. Визначити:

1) чи є маршрути, на яких не зустрінеться жодна пара автобусів;

2) автобуси яких маршрутів зустрінуться на одній зупинці (якщо такі є);

3) на яких зупинках не зупиниться жоден автобус;

4) автобуси яких маршрутів не ходять протягом цього часу;

5) на якій зупинці зупиняється найбільше автобусів (якщо їх кілька – указати всі);

6) на якій зупинці зупиняється найменше автобусів (якщо їх кілька - указати всі).

363. Відомо, що в школі N класів ($15 \leqslant N \leqslant 20$) і в кожному класі навчається по 30 учнів. Підсумковий результат з математики (сума всіх оцінок за семестр) учнів усієї школи виписаний у вигляді таблиці розміром $N\times30$, де кожний рядок - це підсумкові оцінки учнів даного класу з математики відповідно до списку в класному журналі. Визначити порядкові номери учнів у кожному класі, які мають найкращі та найгірші підсумкові бали з математики. Визначити також найкращих і найгірших учнів цієї школи.

364. Таблицю футбольного чемпіонату задано квадратною матрицею порядку я, в якій усі елементи, що належать головній діагоналі, дорівнюють нулю, а кожний елемент, що не належить головній діагоналі, дорівнює 2, 1 або 0 (кількості очок, що набрані в грі: 2 - виграш, 1 - нічия, 0 - програш). Визначити:

1) кількість команд, які мають більше перемог, ніж поразок;
2) номери команд, що не мають поразок;
3) чи є хоча б одна команда, що виграла більше половини ігор.

365. Дано натуральне число $n \geqslant 2$ та елементи дійсного двовимірного масиву a_{ij}, де $i, j = 1, 2, \ldots n$. Побудувати послідовність $b_1, b_2, \ldots b_n$, що складається з нулів і одиниць, у якій $b_i = 1$ тоді і тільки тоді, коли:

1) елементи i-го рядка утворюють зростаючу послідовність;
2) елементи і-го рядка утворюють зростаючу або спадну послідовність.

366. Дано натуральне число n і дійсні елементи квадратної матриці порядку n. Вважатимемо рядок матриці позначеним, якщо елемент цього рядка, що належить головній діагоналі, додатний. Для кожного позначеного рядка обчислити:

1) максимальний елемент;
2) різницю між першим та останнім елементами;
3) середнє арифметичне елементів.

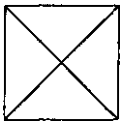
367. Дано натуральні числа n, m та дійсні числа, що утворюють прямокутну таблицю порядку $n \times m$. У таблиці впорядкувати за зростанням елементи:

1) у кожному рядку; 2) у кожному стовпчику.

368. Вважатимемо рядок квадратної матриці порядку n позначеним, якщо перший елемент цього рядка - нульовий. Вивести елементи заданої матриці в такому вигляді: в кожному позначеному рядку - упорядковані за зростанням, а в інших - за спаданням.

369. Дано двовимірний квадратний масив чисел розмірністю n. Повернути його за годинниковою стрілкою на 90°, 180°, 270°.

370. Дано квадратну матрицю розмірністю n. Вважатимемо, що ця таблиця умовно розбита на такі частини:



Знайти:

1) середнє арифметичне кожної з частин;
2) максимальне значення кожної з частин;
3) мінімальне значення кожної з частин.

Примітка. У пункті в) діагональні елементи не враховуються.

371. Дано ціле непарне число я (я > 0). Розташувати в квадратній таблиці розміром я послідовність натуральних чисел 1, 2, n^2 таким чином, щоб найменше число знаходилося в центрі, а всі наступні заповнювали таблицю послідовно по контурах, починаючи з верхнього лівого кута кожного наступного контура. Наприклад, для я = 5:

10	11	12	13	14
25	2	3	4	15
24	9	1	5	16
23	8	7	6	17
22	21	20	19	18

372. Дано ціле я число (я > 0). Розташувати в квадратній таблиці розміром я послідовність натуральних чисел 1, 2, n^2 таким чином, щоб найменше число було в лівому нижньому куті, найбільше - у правому верхньому, а інші заповнювали квадрат по діагоналях з лівого верхнього кута до правого нижнього. Наприклад, для я = 3:

4	7	9
2	5	8
1	3	6

373. Дано таблицю *A[i, j]*, де *i* = 1, 2, я; *J* = 1, 2, *m*. Серед усіх найменших елементів кожного рядка цієї таблиці визначити найбільший і вказати його індекси.

374. Визначити суму елементів, розташованих по зовнішньому контуру таблиці *A[i, j]*, де *i* = 1, 2, я; *J* = 1, 2, *m*.

375. Дано дві таблиці *A[i, j]*, де *i* = 1, 2, я; у = 1, 2, *m* та *B*[*k*, *l*], де *k* = 1, 2, *m*; *l* = 1, 2, я. Знайти елементи таблиці *C*[*i*, у], *i* = 1, 2, *n*; у = 1, 2, *m*, які дорівнюють сумам поелементних добутків рядків таблиці *A* на відповідні стовпчики таблиці *B*.

376. Дано дві таблиці *A[i, j]*, де і = 1, 2, я; у = 1, 2, *m* та *B*[*i*, *j*], де *i* = 1, 2, яг; *J* = 1, 2, я. Скласти таблицю, елементи якої є сумою елементів таблиць *A* і *B*, індекси яких збігаються.

Робота з символьними і рядковими величинами[1]

377. Нехай дано деякий текст. Обчислити кількість входжень у цей текст символу, значення якого задається.

[1] У різних мовах програмування робота з рядковими величинами або послідовністю символів організована по-різному. В деяких значення тексту необхідно вводити окремими символами, використовуючи одновимірні масиви, а в деяких передбачене введення змінних рядкового типу.

378. У даній формулі порахувати кількість входжень символів «+» та кількість символів «−».

379. Обчислити загальну кількість символів «+», «−» та «*» у заданому тексті.

380. Дано текст. Визначити, які символи зустрічаються у тексті частіше: «а» чи «о». Якщо якийсь із символів відсутній - повідомити про це.

381. У заданому тексті замінити всі символи «:» на символи «−» і навпаки.

382. У заданому тексті замінити всі символи «.» на послідовність символів «...». Якщо в тексті зустрічаються підряд три крапки, то залишити їх без змін.

383. У даному тексті всі послідовності крапок замінити на одну крапку.

384. Дано деякий текст, в якому є хоча б одна кома. Визначити порядковий номер:

1) першої коми в тексті;

2) останньої коми в тексті.

385. Роздрукувати даний текст у зворотному порядку.

386. Дано деякий текст. Створити новий текст, який утворено з даного читанням з кінця в початок.

387. Перевірити, чи однаково читається дане слово зліва направо і навпаки.

388. Перевірити, чи є дані два слова оберненими одне до одного, тобто перше читається зліва направо так само, як друге справа наліво.

389. Визначити, скільки разів у даному тексті зустрічається послідовність символів:

1) аб; 2) абв; 3) аба; 4) абаб.

390. Нехай дано текст-формулу, яка містить лише арифметичні операції +, −, *, / і не містить дужок та функцій. Визначити загальну кількість арифметичних дій, передбачених у цій формулі, та кількість чисел, над якими вони здійснюються.

391. Дано деякий текст. Відредагувати текст таким чином, щоб після кожного розділового знака стояв хоча б один пробіл, а всі перші слова в реченнях починалися з великої літери.

392. Дано деякий текст. Відредагувати його таким чином, щоб усі символи «.» були замінені на «...», символи «:» на «−», символи «−» на «:».

393. Скласти програму, яка після кожної цифри в тексті вставляє в дужках її текстовий еквівалент. Наприклад, 0 (нуль), 1 (один), ...

394. Нехай дано текст-формулу. Визначити коректність формули щодо кількості відкритих і закритих дужок. Ва-

жається, що закриті дужки не стоять перед відкритими. Якщо дужки у формулі відсутні - повідомити про це.

395. Скласти програму, яка вилучає із заданого слова всі літери «а» (наприклад, «застава» - «зств»).

396. Скласти програму, яка кожну літеру «а», що зустрічається в тексті, замінює на групу символів «ку» (наприклад, «ади» - «куди»).

397. Нехай дано деякий текст S і значення символьних змінних x та y. Із тексту вилучити всі символи, що збігаються з x, і повторити двічі всі символи, що збігаються з y.

398. Дано текст S, в якому є хоча б одна крапка. Роздрукувати ту частину тексту, що розташована:

1) до першої крапки;
2) після другої крапки;
3) між першою і другою крапками (якщо друга крапка відсутня, то до кінця тексту).

399. Нехай текст дано у вигляді одного слова, тобто в ньому відсутні пробіли. Скласти програму, яка перевіряє, чи є частиною заданого слова слово «рак». Відповіддю має бути «так» чи «ні» (наприклад, для слова «ракета» - «так», а для слова «карета» - «ні»).

400. У даному двійковому числі замінити всі цифри «0» на «1» і навпаки. Якщо старшими цифрами отриманого двійкового числа стануть цифри «0», то ними знехтувати.

401. Розробити програму-шифрувальник тексту, що замінює кожну його літеру наступною по порядку в абетці. Останню літеру абетки необхідно замінити першою.

402. Скласти алгоритм, що замінює перше й останнє слова в даному реченні заданим словом.

403. Скласти алгоритм-шифрувальник, який заміняє кожний символ тексту його ASCII-кодом.

404. Скласти алгоритм, що міняє місцями парні й непарні за порядком слова у тексті.

405. Дано два слова A і B. Перевірити, чи можна з літер, що входять у слово А, скласти слово B.

406. Дано деякий текст. Групи символів, які розділені пробілами (одним або кількома) та не містять усередині себе пробілів, називатимемо словами. Вважатимемо, що текст завжди починається зі слова. Визначити:

1) кількість слів у тексті;
2) кількість слів, які починаються з літер «а» або «А»;
3) кількість слів, у яких перша й остання літери однакові;
4) кількість слів, довжина яких дорівнює k.

407. У даному тексті обчислити найбільшу кількість символів пробілів, розташованих підряд.

408. Розробити програму, яка міняє місцями перше й останнє слова даного речення.

409. Нехай дано текст, що складається з окремих речень і в якому використовуються розділові знаки «.», «?», «!». Обчислити кількість речень у заданому тексті.

410. Дано символьну квадратну матрицю порядку 20 та значення символьної змінної x. Замінити значенням x усі елементи, розташовані нижче від головної діагоналі.

411. Дано натуральне число n і символьну квадратну матрицю порядку n. Отримати послідовність рядкових величин b_2, b_n, що складається зі значень «так» і «ні», в якій b_i - «так» тоді і тільки тоді, коли в i-му рядку кількість пробілів перевищує кількість інших символів.

412. Дано символьну прямокутну матрицю розмірністю гсхЗО. Знайти:

1) номер першого по порядку рядка, що містить найбільшу кількість цифр;
2) номер першого по порядку стовпчика, який містить найменшу кількість пробілів;
3) номер останнього по порядку рядка, який містить найбільшу кількість символів «ш» та «щ»;
4) номер останнього за порядком стовпчика, який містить найбільшу кількість пар однакових сусідніх символів.

ДОПОМІЖНІ АЛГОРИТМИ

Локальні та глобальні змінні. Формальні та фактичні параметри

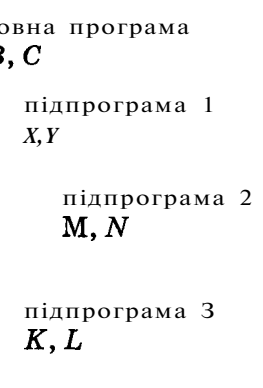
Із процедурами і функціями ми вже мали справу. В цьому розділі ми навчимося їх писати.

Кожна процедура або функція, тобто допоміжний алгоритм, це - «держава в державі», міні-програма в програмі. У ній діють усі закони, притаманні будь-якій програмі. Структура процедур і функцій така сама, як і всієї програми, тобто в них можуть бути свої розділи констант, міток, типів, змінних, процедур та функцій. Проте все, що описане в процедурах і функціях, доступно лише цим процедурам та функціям і не поширюється на всю програму. Логічно ці можливості назвати ***локальними.*** Л всі можливості, що передбачені в основній програмі, доступні як основній програмі, так і всім процедурам та функціям, що в ній описані. Вони називаються ***глобальними.***

Усе це поширюється, в першу чергу, на змінні. Цікавий сам процес виконання процедур і функцій. Усім змінним, що описані в них, місце в пам'яті відводиться тільки на час виконан-

ня цих процедур або функцій. Після того як процедура або функція відпрацює, пам'ять від них вивільняється. Це наводить на думку, що в різних процедурах і функціях програми, та й у самій цій програмі, можуть використовуватися однакові ідентифікатори, і змінні, що їм відповідають, не будуть заважати одні одним.

Схематично принцип дії глобальних і локальних змінних можна зобразити так:



A, B, C - глобальні змінні;
X, Y - локальні змінні для підпрограми 1 та глобальні змінні для підпрограми 2;
M, N - локальні змінні для підпрограми 2;
K, L - локальні змінні для підпрограми 3.

Для того щоб викликати на виконання більшість процедур чи функцій, необхідно задати певні параметри. Зрозуміло, що кожна з цих процедур і функцій повинна якимось чином «прийняти» ці значення, тобто необхідно передбачити їх одержання.

Параметри, які передаються у процедуру чи функцію під час виклику, називаються фактичними.

Параметри, що вказуються в заголовку процедури або функції і замість яких під час виконання підставляються фактичні значення, називаються формальними.

Зрозуміло, що кількість фактичних і формальних параметрів при роботі з однією і тією самою процедурою чи функцією повинна бути однаковою. Крім того, повинні збігатися типи цих параметрів.

Опис функцій

Функції характеризуються тим, що в результаті їх виконання отримуємо лише один результат.

Загальний вигляд функції:

```
function <ім'я функції> (<список формальних параметрів>):
   <тип результату>;
   <описова частина>;
begin
   <виконувана частина, або тіло функції>
end;
```

Оскільки функція є самостійною програмною одиницею в межах основної програми, то всі змінні, які з'являються в ній, повинні бути описані. І починається все з опису формальних параметрів - для кожного з них повинен бути вказаний тип. Описова частина може містити розділ змінних, необхідність в яких з'явилася при створенні даної функції (наприклад, лічильник циклу) і наявність яких у цій функції не впливає на основну програму. Те ж саме стосується і всіх інших розділів.

Одне, але дуже значне ***обмеження на типи параметрів,*** які можна передавати у функцію (а надалі і в процедуру), - вони можуть бути лише змінними простого типу. Тобто поки що масиви і змінні типу **string**[n], які є структурованими типами, ми не використовуватимемо.

Яким же чином результат обчислення функції потрапить в основну програму? Для цього функція повинна містити оператор присвоювання, який і виконає пересилання результату в основну програму. Він має такий загальний вигляд:

<ім'я функції> := <вираз>.

Зауваження. Ім'я функції може використовуватися лише для присвоювання йому деякого значення і не може брати участь в обчисленнях. Тобто запис

my_fun := my_fun + 1

буде сприйнятий як помилковий. Адже використання імені функції в деякому виразі розцінюється як спроба повторного звертання до цієї функції. Такий запис при вказанні параметрів процедури чи функції має право на існування, але це особлива ситуація, і ми про неї поговоримо трохи пізніше. Як бути в нашому випадку? Треба ввести ще одну локальну змінну, обчислити в ній необхідний результат, а потім переслати його імені функції.

А тепер приклад функції. Нехай нам треба обчислити таке значення:

$$z = x! + y!.$$

Оскільки факторіал - це одне число, то нас влаштує функція, яка буде обчислювати його значення.

```
function factorial (n: integer): integer;
  var i, f: integer;
begin
  f := 1;
  for i := 1 to n do
    f := f*i;
  factorial := f;
end;
```

Використати цю функцію в основній програмі для обчислення значення z можна таким чином:

z := factorial(x) + factorial(y).

Зверніть увагу на те, що оскільки після виконання функції її ім'я «містить у собі» обчислене значення, то в основній програмі його треба кудись обов'язково «покласти». Це означає, що звертання до функції повинно використовуватися лише в операторах присвоювання або в процедурі виведення інформації **write** (**writeln**).

Наведемо приклад функції, яка визначає, чи є в даному тексті вказаний символ. Результат виконання цієї функції - логічне значення.

```
function symbol_in_text(S: string; ch: char): boolean;
  var i : integer;
begin
  symbol_in_text := false;

  while (i <= Length(s)) and not (symbolJn_text) do
    begin
      if S[i] = ch then symbol_in_text:= true;
      i :=i + 1
    end;
end;
```

Присутність формальних параметрів у функції необов'язкова. Розглянемо приклад функції, яка дає змогу організовувати очікування натиснення будь-якої клавіші на клавіатурі перед продовженням роботи програми. У тілі функції покроково виконується спочатку очищення буфера клавіатури на випадок, якщо попередньо в ньому залишилася деяка невідома кількість інформації, потім відбувається очікування натиснення будь-якої клавіші, а наприкінець - вичитування коду останньої натиснутої клавіші, що виконає остаточне очищення буфера клавіатури із застосуванням функції **ReadKey**. Назвемо цю функцію «гортанням сторінок».

```
function list: char;
begin
  {очищення буфера клавіатури}
  while KeyPressed do list := ReadKey;
    {очікування натиснення будь-якої клавіші}
  repeat until KeyPressed;
    {очищення буфера клавіатури}
  list := ReadKey
end;
```

Скористатися цією функцією в основній програмі можна буде таким чином:

```
var k: char;

  k := list;
```

Загальний вигляд процедури:

```
procedure <ім'я процедури> (<список формальних параметрів>);
   <описова частина >;
begin
   <виконувана частина, або тіло процедури>
end;
```

Необхідно зразу зазначити, що в заголовку процедури не вказується її тип. Це тому, що за допомогою процедур можна відразу знаходити і передавати в основну програму кілька обчислених значень. Це і є основною відмінністю процедур від функцій.

Щоб зрозуміти, яким чином можна повернути результати з процедури в основну програму, введемо два нових поняття.

Параметри-значення. Почнемо з прикладу.

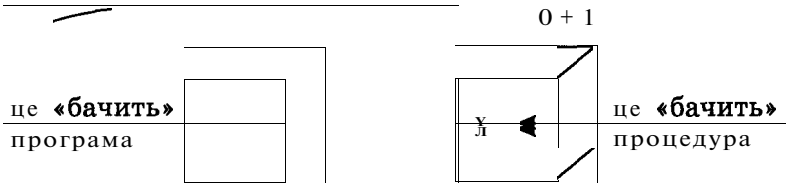
```
procedure proba (x, y: integer);
begin
   x :=x + 1; y :=y + 1
end;

begin

   a := 0; b := 0;
   proba (a, b);
   writeln (a, b);
```

Ця програма надрукує два числа: 0 0. Розберемося детальніше в цій ситуації.

Під час виконання процедури параметрам-значенням відводиться нове місце в пам'яті, тобто створюється їх копія, куди передаються відповідйі значення фактичних параметрів, над ними виконуються дії, вказані в процедурі, і після завершення роботи процедури всі ці змінні знищуються, а їх значення відповідно втрачаються. Тобто які значення для a та *b* були задані перед зверненням до процедури, такими вони і залишилися після її виконання. Схематично це показано на малюнку 21.

Зрозуміло, що параметрами-значеннями варто користуватися тоді, коли у процедуру передаються величини, які братимуть участь у ній у деяких обчисленнях і які після повернення в основну програму повинні зберегти свої початкові значення.

І ще одна важлива деталь: ***при звертанні до процедури параметрам-значенням можна передавати сталі величини*** (proba (2, 5)).



Мал. 21

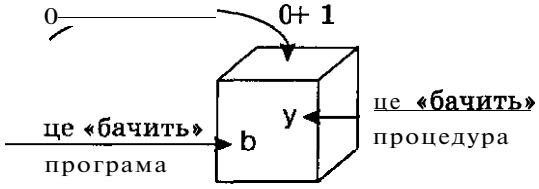
Пара метри-з мінні. Параметри-змінні на відміну від параметрів-значень після виконання процедури повертають ті значення, які вони отримали в процедурі. Справа в тому, що при звертанні до процедури для них не створюються нові місця в пам'яті, і вони використовують ті самі області пам'яті, які відведені відповідним фактичним параметрам. Тому всі зміни, які відбуваються над формальними параметрами, насправді відбуваються і над їхніми фактичними образами. Щоб їх відрізняти від параметрів-значень, перед такими ідентифікаторами вказується службове слово **var.**

```
procedure proba (x: integer; var y: integer);
begin
  x :=x + 1; y :=y + 1
end;

begin

  a := 0; b := 0;
  proba (a, b);
  writeln (a,  b);
```

Ця програма надрукує такі результати:

0 1.

Зобразимо схематично роботу процедури з параметрами-змінними (мал. 22).



Мал. 22

Зробимо ще один висновок: ***при зверненні до процедури параметрам-змінним не можна передавати значення сталих величин,*** оскільки їх значення не можна змінювати.

Можна запропонувати такий вихід: присвоїти значення сталої деякій змінній, а потім використати цю змінну при звертанні до процедури. Наприклад,

```
• b :=0;
 proba (0, b);
```

Запишемо невелику корисну програму, використавши в ній процедуру, яка перетворює всі малі літери тексту на великі і підраховує, скільки таких перетворень зроблено.

```
program invers;
  var txt: string;
   count: integer;
  procedure letters (var S: string; var n: integer);
    var i: integer;
  begin
    n :=0;
    for i := 1 to Length (S) do
      if S[i] <> UpCase (S[i]) then
        begin
          S[i] := UpCase (S[i]);
          Inc (n)
        end
  end;
begin
  writeln ('Введіть текст:');
  readln (txt);
  letters (txt, count);
  writeln ('Перетворений текст: ', txt,);
  writeln ('В заданому тексті було ', count, ' маленьких літер');
  repeat until KeyPressed
end.
```

У цій програмі використана функція Pascal **UpCase.** Вона перетворює малі літери на відповідні їм великі.

Розглянемо ще один приклад процедури, яка ущільнює текст, вилучаючи з нього всі пробіли.

```
procedure compress (var S: string);
  var i: integer;                          t
begin
  i := Pos (' ', S);
  while i <> 0 do
    begin
      Delete (S, i, 1);
      i := Pos (' ', S)
    end
end;
```

Зверніть увагу на те, що виклик процедури є лінійним оператором, оскільки після його виконання ми завжди переходимо до виконання наступного записаного в програмі оператора.

Спробуйте переписати функцію, що робить «гортання сторінок» вашої програми, у вигляді процедури. Це буде раціональніше.

Корисна порада. Інколи перша процедура або функція використовує в собі звертання до другої, а друга - звертання до першої. Оскільки в Pascal розподіл пам'яті та виконання програми йде в тій послідовності, в якій вони записані, то виникає запитання: «Яку процедуру чи функцію описати першою?» Для цього існує можливість випереджаючого опису. Необхідно заголовок другої за чергою процедури або функції винести перед першою зі службовим словом **forward,** а саму процедуру описати після першої. Наведемо приклад:

```
procedure a (<список формальних параметрів>); forward;
procedure b (<список формальних параметрів>);

begin

   a (<список фактичних параметрів>);
end;

procedure a;

begin
b (<список фактичних параметрів>);
end;
```

Рекурсивні функції та процедури

Процедури і функції можуть не тільки викликатися «звідкись», а й викликати самі себе. Така самоактивізація називається ***рекурсією.*** Мова програмування Pascal дає змогу використання рекурсивності, тобто звертання в тілі процедури або функції до самої себе. Застосування рекурсії дає можливість отримувати витончені алгоритми, а вміння записувати рекурсію є досить високим класом програмування.

Рекурсивне звернення до процедури чи функції має сенс тоді, коли сам алгоритм визначений рекурсивно. Звернемося до теоретичних питань теорії алгоритмів. Один із напрямів пошуку теоретичних моделей алгоритмів заснований на арифметизації алгоритмів. Тобто будь-який алгоритм у таких моделях можна представити як послідовність елементарних кроків, кожен з яких є арифметичною операцією. Існують два способи визначення такої послідовності. Перший спосіб — це підставлення функцій у функції таким чином, що порядок дій визначається розстановкою дужок. Другий спосіб зустрічається в елемен-

тарній математиці. Це рекурсія, тобто визначення чергового значення функції через раніше обчислені її значення.

Рекурсивні функції історично є першим уточненням поняття алгоритму, тобто першою універсальною алгоритмічною моделлю.

Звернемося до класичного прикладу факторіалу:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot n.$$

Але ж наведений вираз можна записати і таким чином:

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n = (n-1)! \cdot n.$$

І справді, щоб знайти 3!, треба знати значення 2! і домножити його на 3 і т. д. За домовленістю математиків 0! = 1 і 1! = 1.

У загальному випадку наші міркування можуть виглядати таким чином. Для того щоб обчислити значення $n!$, треба знати значення $(n-1)!$ і домножити його на n. А для того щоб обчислити значення $(n-1)!$, нам треба знати значення $(n-2)!$ і домножити його на $(n-1)$. Постає запитання: «А коли ж цей процес завершиться?» Зрозуміло, тоді, коли ми дійдемо до такого значення факторіалу, яке нам наперед відоме. Ним може бути 0! або 1!.

Розглянемо тепер послідовність наших міркувань з точки зору алгоритмізації. Що таке $n!$? Це алгоритм обчислення значення факторіалу від параметра n, який ми намагаємося створити. А що таке $(n-1)!$? Це той самий алгоритм обчислення значення факторіалу, але від параметра $n-1$. Напрошується такий запис математичної моделі алгоритму обчислення значення $n!$ в термінах відомих нам алгоритмічних структур:

$$n! = \begin{cases} 1, & \text{якщо } n = 1; \\ (n-1)! * n, & \text{якщо } n > 1. \end{cases}$$

Структура такого визначення функції складається з двох частин: визначення значення функції для початкового значення аргументу (в нашому випадку для 1) та визначення $f(n)$ через $f(n-1)$ і операцію множення. Для обчислення $f(n)$ треба обчислити значення функції в попередніх точках: $1, 2, 3, \ldots (n-1)$.

Скориставшись математичною моделлю, можемо записати рекурсивну функцію:

```
function factorial (n: integer): longint;
  begin
    if n = 1
      then
        factorial := 1
      else
        factorial := factorial (n - 1 )*n;
  end;
```

Проаналізуємо, як працюватиме ця програма. Спочатку ми звернемося до неї із заданим значенням *n*. Якщо воно не 0, то перейдемо до виконання формули

factorial := factorial (n - 1)*n

і в ній знову зустрінемо звертання до функції **factorial** уже з меншим параметром (*n* - 1). Але ж ми попередній крок ще не завершили, тому логічно його запам'ятати, записавши значення всіх змінних. Цей процес продовжуватимемо доти, доки не отримаємо значення *n = 1*. Тепер почнемо «розкручувати» ланцюжок збережених станів у зворотному порядку, одночасно виконуючи обчислення.

Так само працює і комп'ютер під час виконання рекурсій. Для цього потрібна додаткова пам'ять і чимала. Частина оперативної пам'яті комп'ютера, де зберігається інформація про всі незавершені стани рекурсивних процедур і функцій, називається *стековою*.Її максимальний розмір складає 64 Кб.

Спробуємо схематично зобразити використання стекової пам'яті під час виконання рекурсивного алгоритму обчислення значення 3!.

1. При зверненні в тілі основної програми до функції factorial(3) замість формального параметра *n* передається значення 3. При цьому для формального параметра *n* відводиться місце в пам'яті комп'ютера, що відповідає вказаному типу. Виконувана частина описаної функції доходить до оператора factorial := factorial(2)*3 і тимчасово припиняє її виконання, оскільки відбувається повторний виклик цієї самої функції. Зазначимо, що end; поточного виклику функції ще не досягнуто! У стековій пам'яті зберігається вся інформація про стан виконання функції factorial з параметром 3. На схемі визначено лише ту дію, на якій припинено виконання функції:

factorial(2)*3

Зрозуміло, що для завершення обчислення значення factorial треба лише знати значення factorial(2).

2. Викликається функція factorial з параметром 2. У пам'яті комп'ютера виділяється місце для значення формального параметра *n,* куди передається значення 2. Виконання функції factorial доходить лише до оператора factorial := factorial(1)*2. Але для обчислення необхідного значення результат factorial(1) невідомий. Виконання функції припиняється, в стековій пам'яті після інформації про виконання функції factorial(3) розміщується інформація про виконання функції factorial(2). Схема стекової пам'яті умовно тепер виглядатиме так:

factorial(2)*3 factorial(1)*2

3. Викликається функція factorial з параметром 1. У пам'яті комп'ютера знову виділяється місце для значення формального параметра *n,* куди тепер передається значення 1. Згідно із записаним нами алгоритмом, виконується оператор factorial := 1 і цей виклик функції добігає свого завершення - досягається службове слово **end.**

4. Тепер, згідно з алгоритмом виконання рекурсивних процедур і функцій, повертаємося до виконання останнього незавершеного варіанта нашої рекурсивної функції, що збережена в стековій пам'яті. Оскільки значення factorial(1) уже відоме і дорівнює 1, то буде завершено і обчислення значення factorial(2): factorial := factorial(1)*2 (1*2).

Стекова пам'ять вивільняється від інформації про недовиконання функції factorial(2) і виглядатиме так:

factorial(2)*3

5. Система повертається до стекової пам'яті, і відбувається спроба завершити її виконання. На даний момент уже відомо значення factorial(2) = 2, яке підставляється в арифметичний вираз factorial(2)*3. Тобто реально виконується такий оператор присвоювання factorial := 2*3. Тепер уже і цей варіант виклику функції завершується, і стекова пам'ять вивільняється від інформації про неї.

6. Саме тоді, коли в стековій пам'яті не залишається інформації про жоден варіант незавершеної функції або процедури, рекурсія вважається завершеною.

Зробимо *перший* дуже важливий висновок. При створенні рекурсивного алгоритму необхідно обов'язково передбачити умову його завершення, тобто за виконання певних умов результату надавати наперед відомого значення. А тому напрошується *другий* логічний висновок: тіло рекурсивного алгоритму повинне містити розгалуження, що й дає змогу передбачити його завершення.

І *третє.* Організація роботи рекурсивного алгоритму сама собою нагадує циклічність: поки ми не отримаємо наперед відомого результату (у нашому випадку — 1 при $n = 1$) «розкручування» рекурсії продовжуватиметься.

Тепер зрозуміло, що завжди варто дуже ретельно оцінювати кожну конкретну ситуацію з приводу використання рекурсії. Досить часто її можна замінити рекурентним співвідношенням, для якого додаткової пам'яті не потрібно. Наприклад, у даному випадку це відомий алгоритм:

$\text{factorial}_i = \text{factorial}_{i-1} * i$ при $i = 1, 2, \quad n$ та $\text{factorial}_0 = 1.$

Цей варіант алгоритму обчислює результат ефективніше, ніж рекурсивна версія. Причина в тому, що на виділення до-

даткової пам'яті також витрачається час. Команди, що забезпечують повторення, витрачають на це менше часу. І хоча ця відмінність подекуди не дуже суттєва, рекомендується все ж таки використовувати циклічні конструкції замість рекурсивних завжди, коли це можливо.

Однак бувають ситуації, коли циклічна версія дуже «заплутує» алгоритм. У цьому разі мінусів виявляється набагато більше, ніж плюсів.

Існує багато гарних прикладів виправданого застосування рекурсії. Наприклад, треба знайти всі можливі переставлення *n* різних елементів.

Назвемо цю операцію change (n) і опишемо її алгоритм у вигляді процедури.

Спочатку a_n залишимо на своєму місці і згенеруємо всі переставлення елементів a_1, a_2, a_{n-1}, викликавши для цього процедуру change(n–1), потім повторимо процес, помінявши місцями a_n з a_i при $i = 1$. Повторимо це для всіх значень $i = 2, n - 1$. Цей алгоритм виглядатиме так:

```
procedure change (k: integer);
  var i: integer;
      t: char;
begin
  if k = 1 then
    begin
      for i := 1 to n do write (a[i], ' ');
      writeln
    end
  else
    begin
      change (k - 1);
      for i := 1 to k - 1 do
        begin
          t := a[i]; a[i] := a[k]; a[k] :=t;
          change (k - 1);
          t := a[i]; a[i] := a[k]; a[k] := t;
        end
    end
end; {завершенняпроцедури}
```

Виклик цієї процедури здійснюється командою change (n).

Результати, отримані для випадку використання трьох літер, будуть такими:

ABC, BAC, CBA, BCA, ACB, CAB.

Нагадаємо ще раз, що кожний ланцюжок рекурсивних викликів повинен на якомусь кроці завершитися і тому *будь-яка рекурсивна процедура повинна містити рекурсивний виклик усередині умовного оператора, який припиняв би повторні виклики процедур.*

У наведеному прикладі рекурсія завершується, коли кількість елементів, які необхідно переставити, стає рівною 1.

Запишемо, використовуючи рекурсивну процедуру, алгоритм упорядкування елементів масиву, розглянутий раніше. Щоб упорядкувати останній елемент масиву, треба перед цим для двох останніх елементів поставити спочатку менший з них, а потім більший. Але перш ніж це робити, треба зробити ще один крок назад, розглянувши три останні елементи, і найменший з них поставити першим. Цей процес крокування назад припиниться лише тоді, коли ми дійдемо до першого елемента масиву. Отже, умовним оператором, що містить рекурсивний виклик, у даному разі є оператор, який перевіряє умову, чи існують ще з початку масиву не розглянуті елементи. Сам же ж рекурсивний процес складається з двох дій: розглянути ситуацію на попередньому кроці і виконати для нього переміщення найменшого елемента на перше місце.

```
program sort;
  var a: array [1..100] of integer;
      i, n: integer;
procedure min (i: integer);
  var min, k, j: integer;
  begin
    min :=a[i]; k :=i;
    for j :=i + 1 to n do
      if a[j] < min then
        begin
          min :=a[j];
          k:=j
        end;
      a[k] :=a[i]; a[i] := min;
  end;
procedure go_sort (i: integer);
  var j: integer;
    begin
      if i >= 1 then
        begin
          go_sort (i - 1);
          min (i);
        end
    end;
begin
  write ('Задайте n: ');
  readln (n);
  for i :=1 to n do
    read (a[i]);
  go_sort (n);
  for i :=1 to n do
      write (a[i], ' ');
end.
```

Наступний приклад демонструє рекурсію в застосуванні до класу задач пошуку розв'язку *методом спроб і помилок*. У цьому методі використовується послідовна побудова часткових розв'язків та перевірка їх допустимості. Кожний новий частковий розв'язок отримується доповненням деякого іншого. Тобто якщо отриманий частковий розв'язок незадовільний, то відбувається повернення до часткового розв'язку, з якого його було отримано, і робиться спроба доповнити його іншим способом. Тому такий підхід ще називається *пошук з поверненням*. Вибір розв'язку задачі здійснюється серед множини часткових розв'язків. Для запису таких алгоритмів дуже зручно використовувати рекурсію.

Запишемо приклад визначення всіх можливих шляхів переміщення у прямокутній таблиці розміром $n \times m$, якщо початкове положення - верхній лівий кут з координатами (1; 1), кінцеве положення - нижній правий кут з координатами (я; яг), а рух можна здійснювати лише на сусідню праву або нижню позиції в таблиці.

```
program seach;
  var a: array [1..100] of integer;
  n, m: integer;
procedure go_seach(i, j, k: integer);
  var L: integer;
begin
  a[k] := i; a[k + 1] := j;
  if j<m then
    go_seach(i, j + 1, k + 2);
  if i<n then
    go_seach(i + 1, j, k + 2);
  if (i = n) and (j = m) then
    begin
      L := 1;
      while L <= 2*(n + m - 1) do
        begin
          write ('(',a[L],',',a[L + 1],');');
          L := L + 2
        end;
      writeln;
    end;
end;
begin
  write ('Задайте n, m: ');
  readln (n, m);
  go_seach (1, 1, 1);
end.
```

У цій рекурсивній процедурі параметр *k* і є тією ланкою між частковими розв'язками, яка запам'ятовує кількість кроків,

зроблених при визначенні даного варіанта повного шляху, і дає змогу відновити це значення при поверненні назад до часткового розв'язку, з якого його було отримано.

У розділі процедур і функцій ми навчилися елементам структурного програмування, вмінню виділяти основні моменти програми в окремі блоки, які називаються процедурами та функціями. Такий процес, коли ми спочатку аналізуємо всю задачу, виділяючи в ній глобальні проблеми (основний алгоритм), і лише потім деталізуємо їх (допоміжні алгоритми), за необхідності виділяючи в них нові проблеми (допоміжні алгоритми нижчого рівня), називається ***програмуванням «зверху донизу»***.

Запитання для самоконтролю

1. Яка відмінність між глобальними та локальними змінними?
2. Які параметри називаються фактичними, а які - формальними?
3. Які типи змінних можуть використовуватися в якості параметрів процедур і функцій?
4. Запишіть загальний вигляд функції.
5. Чи можуть існувати функції, в заголовку яких не вказано формальні параметри?
6. Чим відрізняються процедури від функцій? Запишіть загальний вигляд процедури.
7. Яка відмінність між параметрами-значеннями і параметрами-змінними?
8. Поясніть, як відбувається розподіл пам'яті комп'ютера для параметрів-значень і параметрів-змінних.
9. Чи можна формальному параметру-змінній ставити у відповідність сталу величину? Поясніть відповідь.
10. Коли застосовують випереджаючий опис процедури чи функції? Яке службове слово для цього використовується?
11. Що таке рекурсія? У чому полягає техніка виконання рекурсивних алгоритмів?
12. Що ми розуміємо під програмуванням «зверху донизу»?

Не припускайтеся помилок!

Якщо процедури і функції, на які є пЬсилання в інших процедурах або функціях, описані пізніше, то видається повідомлення про помилку:

Error 3: Unknown identifier.

Якщо ви отримали повідомлення про помилку:

Error 87: «,» expected

при звертанні до процедури або функції, то це, найімовірніше, означає, що кількість фактичних параметрів не збігається з кількістю формальних.

Якщо при звертанні до процедури формальному параметру-змінній поставлено у відповідність фактичний параметр-константу, то буде видано повідомлення про помилку:

Error 20: Variable identefier expected.

Щоб позбутися цієї помилки, треба попередньо значення константи присвоїти якійсь змінній.

Якщо в основній програмі ви не отримуєте результатів, обчислених у процедурі, це означає, що формальні параметри, значення яких повертаються з процедури, не описані як параметри-змінні.

Значення функції в основній програмі повинне присвоюватися деякій змінній того самого типу. Якщо ж ви цього не зробили, про це надійде повідомлення про помилку:

Error 122: Invalid variable reference.

Якщо в якості формальних параметрів випадково використано структуровані типи (масиви, обмежені рядкові величини), то компілятор видасть повідомлення про помилку:

Error 12: Type identifier expected

або

Error 89: «)» expected.

Виконайте завдання

Вправи

413. Чи можна будь-яку функцію оформити як процедуру будь-яку процедуру представити як функцію?

414. Як можна спростити запис:

1) max (a, 1, 15);

2) min (− *1, n* - *m*, 10);

3) min (1, max (sin (a + *b*), π, 5), a - *b*)?

415. Нехай у програмі описано такі процедури:

```
procedure P(x, y: integer);
  begin
    y := x + 1
  end;
procedure Q(x: integer, var y: integer);
  begin
    y := x+ 1
  end;
procedure R(var x, y: integer);
  begin
    y := x+ 1
  end;
```

1) визначити, що буде надруковано в результаті виконання таких дій:

```
c := 2; d := 0; P(sqrt(c) + c, d); writeln (d);
c := 2; d := 0; Q(sqrt(c) + c, d); writeln (d);
c := 2; d := 0; R(sqrt(c) + c, d); writeln (d);
```

2) чи припустиме таке звернення до процедур:

```
P(sqrt(c), d);    P(1,d);    P(c, 2);
Q(sqrt(c), d);    Q(1,d);    Q(c, 2);
R(sqrt(c), d);    R(1,d);    R(c, 2)?
```

416. Знайдіть і поясніть помилки в записі функцій:

1)
```
function max (n: integer): real;
  var a, max: real;
begin
  read (max);
  for i := 1 to n - 1 do
    begin
      read (a);
      if a > max then max := a;
    end;
end;
```

2)
```
function max (n: integer): real;
  var a: real;
begin
  read (a);
  max := a;
  for i := 1 to n - 1 do
    begin
      read (a);
      if a > max then max := a;
    end;
end.
```

Серйозні розважалки

417. Баба-Яга записалася на курси водіїв літальних апаратів. Але справи в неї були кепські, бо вона ніяк не могла запам'ятати, як визначається тривалість польоту при відомій швидкості і відстані. Довелося їй звернутися по допомогу до маленького Хлопчика-Мізинчика, який швиденько написав їй шпаргалку, куди Бабі-Язі треба було лише підставити свої значення. Як виглядала послідовність дій у цій шпаргалці і як нею користувалася Баба-Яга?

418. Щоб відібрати найсильнішу команду з п'яти чоловік на міжнародні змагання по з'їданню апельсинів, десять приятелів розбилися по порядку на пари і визначили найобжерливішого з двох. Скільки апельсинів зможуть з'їсти на змаганнях члени збірної команди? Чи найсильніших гравців вибрали хлопці для своєї команди? Як відібрати гравців?

419. Юрко не переймався математичними формулами і тому весь час на уроки математики носив із собою довідник, у якому під номером 1 була записана формула обчислення периметра квадрата, під номером 2 - площі квадрата, під номером 3 - об'єму куба, під номером 4 - площі бічної поверхні куба. Допоможіть Юркові скласти для комп'ютера підпрограму, яка за заданим номером формули та довжиною сторони квадрата замінила б йому довідник для розв'язування домашніх завдань.

420. Браві солдати Мряка, Бряка, Брусик, Шмарик та Слюник служили у війську під командуванням сержанта Хрямзика. Перед сержантом Хрямзиком було поставлене завдання розробити стратегію визначення чотирьох вояків для штурму фортеці супротивника. Досвідчений сержант вирішив визначати найкращу четвірку за таким правилом: відношення суми ваги вояків до добутку їх зросту повинно бути максимальним. Хто з п'яти служак був удостоєний честі захищати своє військо?

421. Мряка, Бряка, Слюник і Хрямзик ішли, ішли і натрапили на камінь, на якому було викарбувано напис: «Хто праворуч піде, той назбирає n довгиків і m шириків, хто ліворуч піде, той може назбирати в два рази більше довгиків і цілу третину від шириків, порівняно з попереднім випадком, а хто прямо піде, тому дістанеться ціла половина від довгиків і в три рази більше шириків, ніж у першому випадку». Мряка, Бряка, Слюник і Хрямзик швиденько дістали свій лаптоп і мовою програмування Брюндаль склали підпрограми для визначення, куди їм вигідніше йти, щоб зібрати найкращий урожай довгиків і шириків. Їм було відомо, що, за інструкцією, один довгик важить k фряків і l лигів, а один ширик важить p фряків і q лигів, а 20 лигів дорівнюють 1 фряку. Спробуйте зробити це і ви знайомою вам мовою програмування.

422. Три Товстуни загрожують з'їсти своїх куховарів за те, що ті повільніше підшукують і готують нові страви, ніж Товстуни їх з'їдають. Треба врятувати нещасних, склавши для них підпрограму, яка б за кількістю і переліком назв продуктів миттєво видавала можливі варіанти страв - вінегрет, борщ, картопляний салат, котлети тощо. Для зручності при описі рецептури страв їх компоненти впорядковуються за алфавітом.

423. Стадо баранів закохалося в одну і ту саму вівцю. Два сусідніх стада побачили це і собі закохалися в цю ж саму вівцю. Барани першого стада, в якому було n баранів, зізнавалися вівці в коханні відповідно по $a_1, a_2, \ldots a_n$ хвилин, барани другого стада, де значилося m баранів, зізнавалися по $b_1, b_2, \ldots b_m$ хвилин, а барани третього стада, в якому змучена вівця нараховувала k баранів, розливалися в коханні $c_1, c_2, \ldots c_k$ хвилин. Яке стадо закоханих баранів ощасливлювало вівцю своїми зізнаннями найдовший час? У якому стаді знаходився найзакоханіший баран?

424. Сірий Вовк на п'ятому десятку вирішив спокутувати свої гріхи і почав рахувати, в який період свого життя він з'їв менше козенят. Виявилося, що за перші десять років він скуштував відповідно $a_1, a_2, \ldots a_{10}$ козенят, за другі - $b_1, b_2, \ldots b_{10}$, за треті - $c_1, c_2, \ldots c_{10}$, а за четверті кількість нещасних становила $d_1, d_2, \ldots d_{10}$ за рік. Який період порочного вовчого життя втішив Сірого найбільше?

Задачі

Використання функцій і процедур

425. Організувати «гортання» текстової інформації на екрані монітора за допомогою власної підпрограми PAGE, яка забезпечує очищення екрана монітора від попередньої інформації та встановлює курсор у дану екранну позицію.

426. Організувати подання звукового сигналу, тональність якого залежить від кількості символів у слові, скориставшись власною підпрограмою VOICE. Слова - це складові тексту, розділені хоча б одним пробілом.

427. Організувати зміну кольору екрана для покрокового введення текстової інформації - прізвище, ім'я, школа, клас - за допомогою власної підпрограми COLOR_PAGE. Для вибору номера кольору скористатися генератором випадкових чисел.

428. Організувати підпрограму, яка під час натиснення клавіші «пробіл» видає інформацію про розробника програми.

429. Скориставшись стандартною процедурою визначення поточного часу, написати підпрограму, яка щохвилини подає звуковий сигнал.

430. Написати підпрограму, яка за запитом користувача (наприклад, натиснення певної клавіші) видає поточну дату, скориставшись для цього відповідною процедурою мови програмування.

431. Записати функцію знаходження максимального з двох заданих значень і організувати її виклик для трьох пар довільних чисел.

432. Використовуючи функцію max2(a, b), яка визначає максимальне з двох даних значень, записати функцію max3(a, b, c), що визначатиме максимальне з трьох даних значень, і організувати виклик цієї функції для обчислення суми найбільших значень двох трійок довільних дійсних чисел.

433. Написати підпрограму обчислення значення функції $y = \text{sign}\ (x)$, що визначається за правилом:

$$\text{sign}\,(x) = \begin{cases} -1, & \text{якщо } x < 0; \\ 0, & \text{якщо } x = 0; \\ 1, & \text{якщо } x > 0. \end{cases}$$

Використати створену функцію для обчислення

$$\sum_{i=1}^{n} \text{sign}\,(a_i),$$

де n - ціле $(n > 1)$, a_i - дійсні числа.

434. Записати власну функцію обчислення модуля числа n modul(n) і використати її для обчислення середнього арифметичного модулів трьох довільних чисел.

435. За даними дійсними числами *a*, *b* обчислити:

$$u = f(0.5, a) + f(a + b, a - b),$$

де $f(x,y) = \dfrac{x^2 + y^2}{1 + x + y} + \dfrac{x - y}{x + y}$.

436. Дано дійсні числа *a*, *b*, *c*, *d*. Обчислити:

$$y = (p(a) + p(b) + p(c) + p(d))/4,$$

де $p(x) = 4x^4 + 3x^3 + 2x^2 + x + 0.5$.

437. Дано дійсні числа *u* та *v*. Визначити значення:

$$z = f(u, v) + f(u + v, uv) + f(u^2, v^2) + f(0.1, 0.1),$$

де $f(x,y) = \dfrac{x + y}{x + xy + y^2} + \dfrac{x}{1 + y^2} + \dfrac{y}{1 + x^2}$.

438. Дано дійсні значення *a* та ft. Обчислити:

$$u = f(1.7, a) + f(b, a) + f(a + b, b - a),$$

де $f(x,y) = \dfrac{x^2 + xy - y^2}{1 + x^2 + y^2}$

439. Записати підпрограму обчислення факторіалу цілого аргументу, коли відомо, що

$$n! = 1*2*\ldots*n,$$

і організувати її виклик для заданого значення *n*.

440. Використовуючи підпрограму попередньої задачі, розробити програму знаходження суми факторіалів усіх цілих чисел від 1 до 10.

441. Дано чотири цілі додатні числа *a*, *b*, *c*, *d*. Визначити ту пару, для якої функція $F(n, m) = n!*m!*(n + m)!$ набуває найбільшого значення.

442. Дано дійсні числа x, *y*. Обчислити:

$$f(y, -2x, x + y - 1.17) + f(2.2, y, x^2 - y^2),$$

де $f(a,b,c) = \dfrac{2a - b - \sin c}{5 + |c|}$.

443. Дано дійсні числа x, *y*. Обчислити:

$$g(1.2, x) + g(y, x) - g(2x - 1, xy),$$

де $g(a,b) = \dfrac{}{a^2 + 2ab + 3b + 4}$

444. Дано ціле число *a*. Визначити:

$$12.5f(a/2) + 3.1f(a^2 + a + 1), \text{ де } f(x) = \sum^{10}$$

445. Дано натуральне число *n*. Обчислити:

$$f(n + 5) \quad \overleftarrow{i=1}\ i!$$

446. Дано натуральне число n і дійсне число y. Обчислити:

$$\frac{1.7t(0.25,5)+2t(1+y,2)}{6-t(y^2-1,n)}, \text{ де } t(x,m)\frac{\sum_{k=0}^{m}(2k+1)!}{\sum_{k=0}^{m}\frac{x^{2k}}{(2k)!}}.$$

447. Дано дійсні числа x, y. Обчислити:

$$f(f(1, 2), x) + f(y, f(x + y, f(2x, 3y))),$$

де $f(n,m)=\frac{n^2-m^2}{n^2+2nm+3m^2+n+m}$

448. Дано дійсні числа x, y та z. Обчислити:

$$\frac{\max(x,y)+\max(x+y,xz)}{\max^2(0.5,x+z)}$$

449. Дано дійсні числа a, b, c. Обчислити:

$$\max(a, a+b)+\max(a, b+c)$$
$$1+\max(a+bc, b-c, 2.5)$$

450. Дано дійсні числа s, t. Обчислити:

$$h(s, t)+\max(h^2(s-t, st), h^4(s-t, s+t))+h(1,2),$$

де $h(a,b)=\frac{a}{1+b^2}+\frac{b}{1+a^2}-(a-b)^3$.

451. Дано дійсні числа $x_1, y_1, x_2, y_2, x_3, y_3$, які визначають координати вершин трикутника. Визначити периметр і площу трикутника, створивши функцію, що обчислює довжину відрізка, та функцію для визначення площі трикутника.

452. Дано координати вершин двох трикутників. Визначити, який з них має більшу площу.

453. Дано дійсні координати чотирьох точок на площині. Визначити, які трійки з них утворюють трикутники, і обчислити площу більшого трикутника, створивши для цього функції обчислення довжини відрізка та площі трикутника.

454. Дано три цілі додатні числа n, m, k. Визначити їх найбільший спільний дільник.

455. Дано дійсні числа $x_1, y_1, \ldots, x_i, y_i$, пари яких визначають координати вершин багатокутника (координати багатокутника задаються в порядку обходу за годинниковою стрілкою). Визначити периметр:

1) десятикутника;

2) n-кутника (n - ціле, $n>2$).

456. Відомі сторони п'ятикутника a, b, c, d, e та довжини двох діагоналей x, y, що з'єднують одну з вершин із двома іншими. Визначити площу п'ятикутника, написавши підпрограму обчислення площі трикутника за його сторонами.

457. Дано натуральне число n і дійсні числа $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$. Знайти площу n-кутника, вершини якого при послідовному обході мають координати $(x_1;y_1), (x_2;y_2), \ldots (x_n;y_n)$, написавши при цьому підпрограму обчислення площі трикутника за координатами його вершин.

458. Дано деякий текст. Визначити позицію останнього входження даного символу в текст, створивши для цього відповідну підпрограму. Якщо текст не містить символу, то результатом роботи підпрограми повинно бути -1.

459. У даному тексті замінити всі символи 1 на символи 0 і навпаки, використавши для цього відповідну підпрограму. Заміна виконується починаючи з даної позиції в тексті.

460. Скласти і використати для заданого тексту підпрограму «стискання» інформації, в якій кожне повторення одного і того самого символу замінюється на $x(k)$, де x - сам символ, k - кількість його повторень.

461. Скласти і використати для заданого тексту підпрограму, яка вилучає з нього повторення символів, залишаючи лише одне їх входження в даному місці тексту.

462. Дано дві послідовності дійсних чисел з n елементів кожна. Знайти значення елементів третьої послідовності як поелементної суми двох даних, створивши підпрограми введення та виведення членів цих послідовностей.

463. Створити підпрограму, яка б із тексту S вилучала вказаний символ x і рахувала кількість вилучень. У програмі організувати виклик цієї підпрограми.

464. Створити підпрограму, яка б у тексті S визначала перший і останній номери входження даного символу x. У програмі організувати виклик цієї підпрограми.

465. Створити підпрограму, яка б у тексті S заміняла даний фрагмент тексту на текст S_2 і визначала кількість таких замін. У програмі організувати виклик цієї підпрограми.

466. Створити підпрограму, яка б у тексті S визначала кількість розповідних, окличних та питальних речень. У програмі організувати виклик цієї підпрограми.

467. Створити підпрограму, яка б за даними двома дійсними числами a і b визначала їх суму та добуток. У програмі організувати виклик цієї підпрограми.

468. Дано натуральне число л і послідовність пар дійсних чисел $(x_1;y_1), (x_2;y_2), \ldots (x_n;y_n)$. Скориставшись підпрограмою попередньої задачі, визначити ту пару чисел, для якої модуль різниці між сумою та добутком є найменшим.

469. Створити підпрограму, яка б за радіусом R визначала довжину кола і площу круга. У програмі організувати виклик цієї підпрограми.

470. Дано натуральні числа n, R і послідовність пар цілих чисел $(l_1; s_1)$, $(l_2; s_2)$, $(l_n; s_n)$, де і? - радіус круглої коробки, а l_i, s_i - відповідно довжина стрічки, якою можна зовні по колу обв'язати коробку, та мінімальна площа в один шар, яку займають у коробці цукерки (вважатимемо, що цукерки в коробку можна насипати лише в один шар). Скориставшись підпрограмою попередньої задачі, визначити ті пари чисел $(l_i; s_i)$, для яких може підійти кругла упаковка з даним радіусом.

471. Створити підпрограму, яка б за заданою стороною a визначала периметр квадрата і його діагональ. У програмі організувати виклик цієї підпрограми.

472. Дано дві пари дійсних чисел $(x_1; y_1)$ та $(x_2; y_2)$, які є координатами відповідно лівого верхнього і правого нижнього кутів прямокутника, сторони якого паралельні осям координат. Створити підпрограму, що визначатиме довжини сторін цього прямокутника.

473. Дано натуральне число n та послідовність пар дійсних чисел $(x_1; y_1)$, $(x_2; y_2)$, $(x_n; y_n)$. Скориставшись підпрограмою попередньої задачі, визначити дві пари чисел $(x_i; y_i)$ та $(x_j; y_j)$, які можуть утворити прямокутник найбільшої площі.

474. Створити підпрограму, яка б за даними дійсними координатами двох точок $(x_1; y_1)$ та $(x_2; y_2)$ визначала довжину відповідного відрізка та довжину його проекції на вісь Ox. У програмі організувати виклик цієї підпрограми.

475. Дано натуральне число n і послідовність пар дійсних чисел $(x_1; y_1)$, $(x_2; y_2)$, $(x_n; y_n)$. Скориставшись підпрограмою попередньої задачі, визначити ту пару чисел $(x_i; y_i)$, для якої довжина відповідного відрізка найменша, а також обчислити довжину його проекції на вісь Ox.

476. Створити підпрограму, яка б за даною цілою сумою S двох сусідніх сторін прямокутника визначала два цілі числа a і b, для яких виконується умова $S = a + b$ і які визначають прямокутник найбільшої площі.

477. Дано натуральне число n та послідовність цілих чисел $S_1, S_2 \ldots S_n$, де S_i - сума двох сусідніх сторін i-го прямокутника. Скориставшись підпрограмою попередньої задачі, визначити такі пари цілих чисел $(a_i; b_i)$, для яких виконується умова $S_i = a_i + b_i$ і які визначають прямокутник найбільшої площі.

478. Записати підпрограму, що одночасно обчислює найменше і найбільше значення серед двох дійсних чисел, та використати її для визначення, у скільки разів найбільше значення з трьох заданих дійсних чисел перевищує найменше з них.

479. Дано ціле n і дійсні числа $a_1, b_1, c_1, \ldots a_n, b_n, c_n$, які визначають коефіцієнти n квадратних рівнянь. Для кожної трійки чисел визначити суму квадратів коренів відповідних квадратних рівнянь (передбачити можливість відсутності коренів).

480.[1] Дано дійсні числа $a_0, \quad a_{10}$. Обчислити для $x = 1$, 3, 5 значення $p(x + 1) + p(x - 1) + p(x)$, де

$$p(y) = a_{10}y^{10} + a_9y^9 + \ldots + a_0.$$

481. Дано ціле число n і дійсні числа s, $a_0, \quad a_n$. Обчислити:

$$p(1) - p(t) + p^2(s - t) + p^2(1) + p^3(5),$$

де $p(x)\ a_nx^n + a_{n-1}x^{n-1} + \ldots + a_0$.

482. Дано натуральні числа n, m дійсні числа $a_1, a_2, \quad a_n, b_1, b_2, \quad b_m$. У послідовностях $a_1, a_2, \quad a_n$ і $b_1, b_2, \quad b_m$ усі члени, що знаходяться за першим максимальним елементом відповідного масиву, замінити на 0.

483. Дано натуральні числа n, m цілі числа $a_1, a_2, \quad a_n, b_1, b_2, \quad b_m$, k. Скласти підпрограму, яка повертає значення true і замінює перше найбільше значення послідовності $a_1, a_2, \quad a_n$ на k, якщо в ній жодного разу не зустрічається член із значенням k, i повертає значення false, не змінюючи при цьому послідовності, якщо значення k трапилося. Ту ж саму операцію виконати з послідовністю $b_1, b_2, \quad b_m$.

484. За даними 20-елементними цілими масивами x і y обчислити:

$$S = \begin{cases} \sum_{i=1}^{20} x_i^2 & \text{при } \sum_{i=10}^{15} x_i y_i > 0; \\ \sum_{i=10}^{20} & \text{в інших випадках.} \end{cases}$$

485. За даними 50-елементними масивами дійсних чисел a, b, c обчислити:

$$t = \begin{cases} \max(b_i) \quad \max(c_i) \\ \max(a_i) \quad \max(b_i + c_i) & \text{при } \min(a_i) < \max(b_i); \\ \min(b_i + c_i) + \min(c_i) & \text{в інших випадках.} \end{cases}$$

У цій і наступних задачах $\max(x_i)$ - максимальний член послідовності $x_1, x_2, \quad x_n$.

486. Дано натуральне число k і дійсні числа $x_1, x_2, \quad x_k, y_1, y_2, \quad y_k, z_1, z_2, \quad z_{50}$. Обчислити:

$$\begin{cases} (\max(x_i) + \max(y_i))/2 & \text{при } \max(z_i) \geqslant 0; \\ 1 + \max^2(z_i) & \text{при } \max(z_i) < 0. \end{cases}$$

[1] Для виконання завдань № 480-493 мовою програмування Pascal необхідно спочатку ознайомитися з розділом «Зчислені типи. Типи користувача».

487. Дано натуральні числа k, m та цілочислові послідовності $a_1, a_2, \ldots a_k, b_1, b_2, \ldots b_m, c_1, c_2, \ldots c_m$. Обчислити:

$$\begin{cases} \min(b_i) + \min(c_i) & \text{при } |\min(a_i)| > 10; \\ \min(b_i + c_i) & \text{при } |\min(a_i)| \leqslant 10. \end{cases}$$

488. За даними 40-елементними дійсними векторами x, y, z обчислити:

$$u = \begin{cases} \prod_{i=1}^{40}(\sin(x_i)+2) & \text{при } \prod_{i=1}^{40}(1-y_i^2) > 0.5; \\ \prod_{i=1}^{40}(1-z_i^2) & \text{в інших випадках.} \end{cases}$$

489. Дано три квадратні таблиці A, B, C розмірністю 10. Створити підпрограму введення елементів цих таблиць та визначення максимальних елементів у кожній із них.

490. Дано дві дійсні прямокутні таблиці A і B розмірністю $n \times m$. Визначити, в якій із них сума всіх елементів найбільша.

491. Дано дві квадратні дійсні матриці порядку 10. Вивести значення обох матриць таким чином, щоб першою була та, в якій сума елементів на головній діагоналі найменша.

492. Дано три дійсні прямокутні таблиці A, B, C розмірністю $n \times m$. Визначити, в якій із них найбільший елемент зустрічається раніше (пошук вести по рядках зліва направо).

493. Дано дві дійсні прямокутні таблиці A і B розмірністю $n \times m$. Визначити, в якій із них кількість мінімальних елементів менша.

Рекурсія

494. Обчислити значення факторіала цілого числа n, коли відомо, що:

$$f(n) = \begin{cases} 1 & \text{при } n = 0; \\ n * f(n-1) & \text{при } n > 0. \end{cases}$$

495. Знайти найбільший спільний дільник двох додатних цілих чисел n і m за алгоритмом Евкліда, скориставшись такою залежністю:

$$f(n,m) = \begin{cases} n, & \text{якщо } n = m; \\ f(n-m, m), & \text{якщо } n > m; \\ f(n, m-n), & \text{якщо } n < m. \end{cases}$$

496. Обчислити значення n-го елемента послідовності чисел Фібоначчі, якщо існує така залежність:

$$f(n)=\begin{cases}0, & \text{якщо } n=0;\\ 1, & \text{якщо } n=0;\\ f(n-1)+f(n-2), & \text{якщо } n>0.\end{cases}$$

497. Обчислити площу прямокутника розміром $n\times m$, скориставшись залежністю:

$$S(n,m)=\begin{cases}1, & \text{якщо } n=m=1;\\ S(n-1,\ m)+1, & \text{якщо } n>1;\\ S(n,\ m-1)+1, & \text{якщо } n<1.\end{cases}$$

498. Обчислити значення функції Аккермана для двох невід'ємних цілих чисел n і m, де:

$$A(n,m)=\begin{cases}m+1, & \text{якщо } n=0;\\ A(n-1,1), & \text{якщо } n\neq 0,\ m=0;\\ A(n-1,A(n,m-1)), & \text{якщо } n>0,\ m>0.\end{cases}$$

499. Обчислити значення квадрата цілого додатного числа n, скориставшись залежністю $n^2=(n-1)^2+2(n-1)+1$ або

$$f(n)=\begin{cases}1, & \text{якщо } n=1;\\ f(n-1)+n+n-1, & \text{якщо } n>1.\end{cases}$$

Слід зауважити, що у запропонованому варіанті обчислення квадрата цілого числа використовуються лише операції додавання й віднімання.

500. Обчислити кількість комбінацій з n різних елементів по m, тобто кількість неупорядкованих підмножин з m елементів, що належать заданій множині з n елементів C_n^m $(n\geqslant m)$, скориставшись залежністю:

$$C_n^m=\begin{cases}1, & \text{якщо } m=0,\ n>0 \text{ або } m=n\geqslant 0;\\ 0, & \text{якщо } m>n\geqslant 0;\\ C_{n-1}^{m-1}+C_{n-1}^{m} & \text{в інших випадках.}\end{cases}$$

501. Скласти програму, яка реалізує синтаксичний аналізатор для поняття *ідентифікатор* де[1]

$$\textit{ідентифікатор} ::= \textit{ідентифікатор}\ \begin{Bmatrix}\textit{цифра}\\ \textit{літера}\end{Bmatrix}\ \textit{літера}.$$

[1] У цій і наступних задачах запис типу {...} розуміється як вибір можливих варіантів із зазначених у дужках. Тобто необхідно читати «ідентифікатор - це є літера або ідентифікатор, який, у свою чергу, є цифрою або літерою». Цей запис і визначає організацію рекурсії.

502. Побудувати синтаксичний аналізатор для поняття *сума*, де:

сума ::= *ціле* {*знак-операції**ціле*}*[1],

ціле ::= *цифра* {*цифра*}*,

знак-операції ::= ‹ › .

503. Скласти програму, що реалізує синтаксичний аналізатор для поняття *простий-вираз*, де:

простий-вираз ::= *простий-ідентифікатор* | ⌈*простий-знак*⌉ ‹ *знак-операції* › ⌊*простий-вираз*⌋

простий-ідентифікатор ::= *літера*,

знак-операції ::= ‹ + › .

[1] Символ « * » означає повторення.

РОЗШИРЕННЯ МОВИ ПРОГРАМУВАННЯ PASCAL

ДОДАТКОВІ МОЖЛИВОСТІ МОВИ PASCAL

Ми ознайомилися з основною частиною Pascal, яка ще називається ядром. Додаткові можливості, до яких ми переходимо, дадуть вам змогу записувати алгоритми компактніше й виконувати їх ефективніше. Ознайомлення з деякими новими типами змінних дасть змогу глибше зануритися у структуру самого комп'ютера, пригадати принципи його будови й функціонування.

Зчислені та інтервальні типи. Типи користувача[1]

Ми користувалися лише тими типами змінних, інформацію про які містить Pascal. Але можливості його настільки великі, що він дає змогу створювати свої власні типи. Правда, їх буде «розуміти» лише та програма, в якій вони описані.

Нехай складемо програму, зміст якої стосується правил дорожнього руху. Можна створити такий власний тип:

type color = (red, yellow, green).

Тепер у цій програмі можна використовувати змінні типу color так само, як і типу integer або boolean. Наприклад, можна організувати такий цикл:

```
var i: color;

for i := red to green do
   {тіло циклу}
```

Нехай ми працюємо з програмою нарахування заробітної платні. Логічно було б створити такий тип:

type month = (jan, feb, math, aprl, may, june, july, aug, sep, oct, nov, dec).

[1] Мова програмування Pascal має свою особливість - надає можливість користувачеві створювати власні типи змінних, що не належать до стандартних. Але це не означає, що задачі, запропоновані в цьому розділі, можуть бути запрограмовані лише в Pascal, адже всі мови програмування мають достатньо можливостей для реалізації такого типу задач.

Якщо ми маємо справу з розкладом занять, то наш власний тип виглядатиме так:

type day = (sunday, monday, **tuesday,** Wednesday, **thursday,** friday, **Saturday).**

Отже, з наведених прикладів випдиває, що елементами власного типу користувача можуть бути створені ним ідентифікатори на зразок імен змінних. Відмінність лише в тому, що в даному випадку ці ідентифікатори самі є значеннями, у той час як змінні набувають деяких значень. У переліку елементів власного типу користувача є чіткий порядок, що дає змогу використовувати їх в якості лічильника в циклі. Саме тому такі типи носять назву *зчислених.*

Із зчисленим типом ми вже ознайомлювалися, коли вивчали оператори **case** і **for ... to (downto) ... do.** Нагадаємо, що тоді зчисленим ми називали такий тип даних, для елементів якого існують поняття «попередній» і «наступний».

Зі зчисленних типів можна утворювати похідні типи, які визначають підмножину підряд розміщених елементів стандартного або описаного зчисленого типу.

Наприклад, у тому ж розділі типів можна описати ще такі типи:

work_day = monday..friday;
weekday = (sunday, **Saturday).**

Тип **workday** називається *інтервальним* або *обмеженим типом.* Подібні типи створюються з уже відомих стандартних або описаних власних типів. У інтервальних типах достатньо вказати лише межі зміни значень відомих типів, розділяючи їх символами «..». У інтервальному типі значення лівої межі завжди менше за значення правої. Інтервальним типом ми вже користувалися, вказуючи зміни індексів у масивах: **array** [1..100].

Наприклад, тип

mark = 1..12

створено з шдмножини елементів цілого типу і він дає змогу вводити лише числові значення шкільних балів. На введення будь-яких інших цілих значень програма видаватиме повідомлення про помилку.

Ще раз наголосимо, що елементи зчисленного типу розміщуються в чіткому порядку. Найменшим за величиною є перший записаний елемент, найбільшим - останній, другий елемент більший, ніж перший, і т. д. Одна незручність - усі значення власних типів існують лише в самій програмі та ще й у закодованому вигляді, а ввести їх із клавіатури або вивести на екран монітора неможливо. У цьому полягає найбільший секрет власних типів користувача. Обійти цю незручність можна, лише створивши процедури «перекладачі». Розглянемо, як це робиться.

Запишемо програму, що підраховує рейтинг за перше півріччя з таких предметів: фізика, математика, інформатика й англійська мова.

```
program rating;
  type science = (physics, machematics, informatics, english);
       month = (jan, feb, math, aprl, may, june, july, aug, sep,
                oct, nov, dec);
       marks = 1..5;
var subject: science;
    semestr: sept .. dec;
    result: array[science, sept ..dec] of marks;
    sum: integer;
procedure OutPutScience;
  begin
    case subject of
      physics: write ('фізика');
      machematics: write ('математика');
      informatics: write ('інформатика');
      english: write ('англійська мова');
    end
  end;
procedure OutPutMonths;
  begin
    case semestr of
      sept: write ('вересень');
      oct: write ('жовтень');
      nov: write ('листопад');
      dec: write ('грудень');
    end
  end;
begin
  for subject := physics to english do
    for semestr:=sept to dec do
      begin
        write ('Введіть оцінку за ');
        OutPutScience;
        write (' за ');
        OutPutMonths;
        write ('місяць ');
        readln (result[subject, semestr]);
      end;
    writeln;
  for subject := physics to english do
    begin
      write ('Рейтинг за семестр із предмета ');
      OutPutScience;
      sum := 0;
      for semestr:=sept to dec do
      sum := sum + result[subject, semestr];
      writeln (': ', sum);
    end;
```

```
    for semestr:=sept to dec do
      begin
        write ('Рейтинг з усіх предметів за ');
        OutPutMonths;
        write (' місяць: ');
        sum := 0;
        for subject:=physics to english do
          sum := sum + result[subject, semestr];
        writeln (sum);
      end;
    readln
  end.
```

Можна передбачити одну помилку на початку роботи із власними типами. Не використовуйте ідентифікатор типу (наприклад, **science** або **month)** як ім'я змінної, не присвоюйте йому значень! Тобто тип - це лише образ. Типам не відводиться місце в пам'яті комп'ютера, на них можна лише робити посилання: «Зроби мені цю змінну такою, як записано в розділі типів під назвою **xxx».** І тільки після цього змінна буде розподілена в пам'яті комп'ютера за образом вказаного типу.

Використання масивів як формальних параметрів функцій і процедур

Уведення поняття власних типів дає змогу організувати роботу процедур і функцій, де в якості формальних параметрів використовуються масиви.

Створимо, наприклад, такий тип:

```
type A = array [1..100] of integer.
```

Тепер змінні, які будуть посилатися на цей тип, можна використовувати в якості формальних параметрів.

Наведемо приклад фрагменту програми, що знаходить значення:

$$R = \max(a_1, a_2 \ldots a_n) + \max(b_1, b_2 \ldots b_m).$$

```
type mas = array [1..100] of real;
var a, b: mas;

function max mas (k: integer; x: mas): real;
  var     i: integer;
      max: real;
  begin
    max := x[1];
    for i := 2 to k do
      if x[i] > max then max := x[i];
    max mas := max
  end;
```

```
begin

  read (n, m);
  {введення значень масивів a та b}

  R := max_mas(n, a) + max_mas(m, b);

end.
```

Запитання для самоконтролю

1. Які типи змінних можна назвати зчисленими?
2. Що таке інтервальні типи?
3. Чи можна значення власних типів користувача задавати як аргументи і виводити як результати? Поясніть свою відповідь.
4. Яким чином можна використовувати структуровані типи як формальні параметри в процедурах і функціях?

Не припускайтеся помилок!

Якщо значення власних типів користувача вводитимуться з клавіатури або виводитимуться процедурою **write (writeln),** то буде повідомлення про помилку **Error 64.** Щоб позбутися її, треба скористатися процедурами або функціями «перекладачами».

Якщо ви використаєте ім'я типу як ім'я змінної, то це призведе до виведення повідомлення про помилку

Error 88: «(» expected,

тобто компілятор «подумає», що ви хочете звернутися до якоїсь функції.

Виконайте завдання

Вправи

504. Нехай у програмі зроблено такі описи:

```
type vegetables = (tomato, potato, carrot, cabbage,
                   pepper, onion, radish);
type fruit = (apple, pear, cherry, lemon, orange, kiwi, grapefruit);
var a, b: vegetables; c, d: fruit;
    v: tomato..onion;
    f: apple..cherry;
```

Дайте відповіді на такі запитання та виконайте завдання.

1) Яких значень набуватимуть змінні *a, b, c, d*?

2) Чи можливе виконання таких операторів присвоювання:

a) a := tomato;
b) a := b;
c) b := apple;
d) b := c;
e) d := f;
f) v := pear?

3) Чи можливе використання оператора циклу з таким заголовком:

a) **for** a := tomato **to** radish **do** ...;

b) **for** b := tomato **downto** pepper **do** ...;

c) **for** v := potato **to** pepper **do** ...;

d) **for** f := pepper **downto** carrot **do** ...;

e) **for** f := lemon **to** grapefruit **do** ...?

4) Чи можливе використання таких стандартних процедур введення/виведення даних:

a) **read** (a, b);

b) **readln** (vegetables);

c) **writeln** ('vegetables');

d) **write** (tomato)?

5) Чим відрізняються такі описи:

a) **var** v: tomato..onion;
f: apple..cherry;

b) **type** v = tomato..onion;
f = apple..cherry?

6) Обчислити значення виразів:

a) apple < orange;

b) cabbage >= tomato;

c) radish = grapefruit;

d) kiwi > grapefruit;

e) lemon <> orange;

f) **pred** (tomato);

g) **succ** (apple);

h) **pred** (cabbage);

i) **pred** (cabbage) = **succ** (potato);

j) **pred** (pepper) < **succ** (pepper);

k) **pred** (carrot) = **pred** (cherry);

l) **ord** (tomato) = **ord** (apple);

m) **ord** (lemon) > **ord** (tomato);

n) **ord** (apple) + **ord** (grapefruit);

o) **ord** (onion) <> **ord** (orange);

p) **ord** (vegetables).

Задачі

505. Нехай у програмі дані описи:

```
type colors = (red, blue, yellow, green);
var c: colors;
```

Скласти програму, яка друкувала б переклад описаних кольорів українською мовою.

506. У програмі зроблено описи:

```
type month = (jan, feb, math, apr, may, june, july, aug,
              sept, <5kt, nov, dec);
     day = 1 .. 31;
var d1, d2: day; m1, m2: month; t: boolean;
```

За даною датою (днем і місяцем) визначити, чи передує дата d1, m1 даті d2, m2, присвоївши змінній *t* відповідно значення **true** або **false**.

507. У програмі визначений тип **month** за умовою попередньої задачі, а також задані натуральне число n $(1 \leqslant n \leqslant 12)$ та значення змінної m типу **month**. Вивести назву місяця:

1) наступного для m;

2) попереднього для m;

3) n-го.

508. Нехай, крім типу **month,** у програмі описаний тип season = (winter, spring, summer, autumn). Визначити:

1) до якої пори року відноситься вказаний місяць;

2) які місяці відносяться до вказаної пори року.

509. Вважатимемо, що країни Європи та їх столиці у програмі описані відповідними типами:

type country = (France, England, Spain, Italy, Greece, Poland);
capital = (Paris, London, Madrid, Rome, Afine, Warchava);

Використовуючи генератор випадкових чисел, визначити і вивести назву країни та її столиці.

510. Текстові значення шкільних балів задані типом

type mark = (one, two, three, four, five, six, seven, eight, nine, ten, eleven, twelve).

Використовуючи функцію **ord,** за введеним текстовим варіантом шкільних балів визначити її числовий еквівалент.

511. У музиці використовують такі музичні інтервали: секунда (інтервал між двома сусідніми нотами; наприклад, ре-мі, сі-до), терція (інтервал через ноту; наприклад, фа-ля, ля-до), кварта (за аналогією до попередніх двох), квінта, секста, септіма.

Нехай у програмі є описи:

type note = (doo, re, mi, fa, sol, la, ci);
interval = (second, tercia, kvarta, kvinta, sexta, septima);
var n1, n2: note; i: interval;

Визначити інтервал, який утворюється двома нотами n1 та n2 (n1 ≠ n2).

512. У програмі зроблено описи:

type country = (France, England, Spain, Italy, China, Laos, USA, Canada);
continent = (Europe, Asia, America);

За введеною назвою країни визначити, якому континенту вона належить.

513. Нехай у програмі визначено такий тип:

type weekdays = (sun, mon, tues, wednes, thurs, fri, satur);

За введеним порядковим номером дня тижня (початок тижня - понеділок) вивести його повну назву англійською мовою та переклад українською мовою.

Множини

Уявіть, що ви маєте скриньку, в яку складаєте коштовності і ведете облік її вмісту. Але ця картотека специфічна - коштовностей так багато, що ви вже збилися з рахунку і не записуєте кількість однотипних речей, а тільки їх різновиди. Саме так виглядають множини і в Pascal.

ЦІ Множина — це сукупність елементів, якій притаманна якісна, а не кількісна **характеристика.**

Отже, ознайомимося з новим структурованим типом - множиною. Загальний вигляд опису множин:

<ім'я змінної>: **set of** <тип>;

Приклади визначення множин:

```
type
   symbol = set of char;
   number = set of byte;
   LetterUkr = set of ('і', Т, 'є');
var
   digit: set of 0..9;
   txt: set of ('0'..'9', 'х', 'у', 'z', 'а'..'я');
begin
   digit := [1, 2, 3];
   txt := ['0'..'z'];

end.
```

Під час роботи з множинами *треба пам'ятати,* що кількість елементів множини не може перевищувати 256. Так само як і елементи власних типів користувача, елементи множин знаходяться в пам'яті комп'ютера в закодованому вигляді і займають лише по 1 біту, тобто максимальний розмір множини — 32 байта. Тому в деяких випадках використання множин у програмі дає значний виграш.

Цікаво, що в Pascal існує поняття порожньої множини (а := []), тобто множини, в якій відсутні елементи.

Для роботи з множинами в Pascal введені додаткові операції.

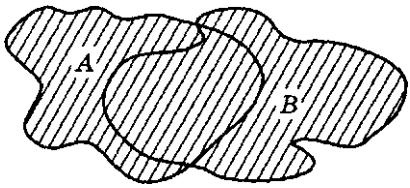
Операція входження має такий загальний вигляд:

<елемент> **in** <множина>;

Ця операція визначає, входить означений елемент у вказану множину чи ні. Зрозуміло, що результатом виконання цієї операції буде тип **boolean.**

Операція об'єднання множин позначається символом «+».

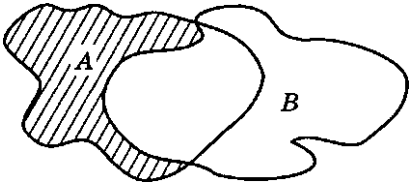
Якщо виконати операцію С := А + В, де *А, В, С* - множини, то множина С буде містити елементи, які належать множинам *А* і *В*. Можна уявити собі, ніби елементи *А* і *В* зсипали докупи. Схематично множину С будемо зображати заштрихованою, і на малюнку це виглядає так (мал. 23):



Мал. 23

Операція різниці множин позначається символом «–» і має такий вигляд (мал. 24):

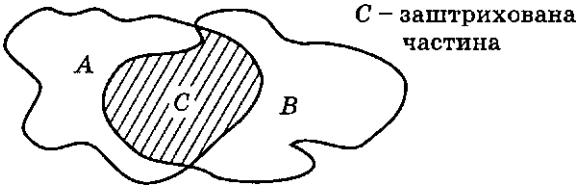
C := A - B.



Мал. 24

Операція перерізу множин позначається символом «*» і записується таким чином (мал. 25):

C := A*B.



Мал. 25

Операції порівняння множин:

A <= B - дає результат **true**, якщо всі елементи множини *A* належать множині *B*;

A >= B - дає результат **true**, якщо всі елементи множини *B* належать множині *A*;

A = B - дає результат true, якщо множини *A* і *B* складаються з однакових елементів;

A <> B - дає результат **true**, якщо множини *A* і *B* відрізняються хоча б одним елементом.

При створенні діалогових програм інколи важливо знати, яка встановлена розкладка клавіатури - англомовна чи україномовна. Запишемо програму, яка дає відповідь на це запитання.

```
program dialog;
  var M:  set of char;
      ch: char;
begin
  M := ['A'..'Я', 'І', Т, 'Є'];
  repeat
    repeat until KeyPressed;
      ch := UpCase(ReadKey);
      if not (ch in M) then
      writeln ('Переключіться з режиму латинських літер!')
  until ch in M;
end.
```

Щоб використати множини для роботи з масивами, треба спочатку «скинути» всі елементи масиву в множину, а потім проводити аналіз. Єдине зауваження: діапазон зміни значень масивів не повинен перевищувати 256 елементів. Наступний приклад демонструє, як можна визначити спільні елементи двох масивів.

```
program commons;
  var a, b:        array[1..100] of byte;
      S, S1, S2: set of 0..255;
      i:           integer;
begin
  {введення n елементів масиву a та m елементів масиву b}

    S1 := [];
    S2 := [];
  for i := 1 to n do
    S1 := S1 + a[i];
  for i := 1 to m do
    S2 := S2 + b[i];
    S := S1*S2;
  writeln ('Елементи, що є спільними для обох масивів: ');
  for i := 0 to 255 do
    if i in S then write (i, ' ');
end.
```

Цікаво, що значення змінних типу «множина» не можна виводити у програмі. Тобто використання процедури **write** (S) буде помилковим.

Як вивести значення елементів множини? Це залежить від того, в якій послідовності хочемо це робити. У попередньому прикладі елементи множини будуть виведені в упорядкованому варіанті: або за зростанням, або за спаданням. У такому разі елементи послідовності будуть зустрічатися лише по одному разу і для цього необхідно знати інтервал зміни їх значень. Нехай це буде L..R. Виведення елементів множини може виглядати так:

```
for i := L to R do
  if i in S then write (i, ' ');
```

Якщо ж є необхідність виведення елементів множини в порядку їх слідування у послідовності, тобто без повторень, то це можна зробити, наприклад, так:

```
for i := 1 to n do
  if a[i] in S then
    write (a[i], ' ')
  else
    S1 := S1 – a[i];
```

Запитання для самоконтролю

1. Що таке множина?
2. Яку кількість елементів множини можна використовувати у Pascal?
3. Які операції можна виконувати над множинами?
4. Чи можна виводити значення змінних типу «множина»?

Не припускайтеся помилок!

Якщо кількість елементів множини перевищує 256, то буде видано помилку:

Error 23: Set base type out of range.

Якщо ви намагаєтеся значення змінної типу «множина» виводити процедурою **write**, то компілятор видасть вам помилку:

Error 64: Cannot read or write variables of this type.

Виконайте *завдання*[1]

Вправи

514. Які з наведених конструкцій є множинами (в термінах мови програмування Pascal), а які ні і чому?

1) [9, 6, 3, 0];
2) [1..15, 4..18];
3) [0..0];
4) [2, **sqrt**(9)];
5) [[], [5]];
6) [0..500];
7) [0.. – 255];
8) [2..3, 5, 7];
9) ['*', '*'];
10) [**true**..**false**];
11) ['=', '>=', '>'];
12) [**odd**(7), 0 < 2];
13) [100..355];
14) [– 255..0].

515. Нехай програма містить такий опис:

var s: **set of char**; c, d: **char**;

Змінній s присвоїти:

1) порожню множину;
2) множину з малих голосних латинських літер (a, *e, i, o, u*);
3) множину із усіх цифр;
4) множину літер, яка починається з літери *c* і закінчується літерою *d*.

516. Обчислити значення виразів:

1) [2] <>[2, 2, 2];
2) ['4', '5', '6'] = [4, 5, 6];
3) [2, 3, 5, 7] <= [1..9];
4) ['a', 'b'] = ['b', 'a'];
5) ['c', 'b'] = ['c'..'b'];
6) [3, 6..8] <= [2..7, 9];

[1] Усі наведені в цьому розділі задачі можна розв'язувати як з використанням типу «множина», так і циклічних конструкцій та масивів, якщо в мові програмування не реалізований такий тип змінних, як «множина».

7) [] <= ['0'..'9'];
8) trunc(3.9) in [1,3, 5] = false;
9) [2] < [1..3];
10) 'q' in ['a'..'z'] = true;
11) odd(4) in [];
12) 66 = [66].

517. Обчислити значення виразів:

1) [1, 3, 5] + [2, 4];
2) [1..6] + [3..8];
3) [2, 4] + [1..5];
4) [] + [4];
5) [1, 3, 5]*[2, 4];
6) [1..6]*[3..8];
7) [2, 4]*[1..5];
8) []*[4];
9) [1, 3, 5] – [2, 4];
10) [1..6] – [3..8];
11) [2, 4] - [1 ..5];
12) [] – [4].

518. Обчислити значення виразів:

1) [2..13]*[3, 13..60] + [4..10] - [5..15] + [6];
2) [2..10] – [4, 6] – [2..12]*[8..15];
3) (['0'..'7'] + ['2'..'9'])*(['a'] + ['z']).

519. Чи еквівалентні такі записи:

1) x in [1,3, 5] та (x = 1) or (x = 3) or (x = 5);
2) x in [1 ..100] та (x => 1) and (x <= 100);
3) x in [1, 10..100] та (x= 1) or ((x => 10) and (x <= 100));
4) x in [1, 10..100] та (x= 1) or (x => 10) and (x <= 100)?

520. Спростити вирази (*A* і *B* - множини):

1) A*B – A;
2) (A + B) – (A – B) – (B – A);
3) A – (A – B);
4) (A – B) + (B – A) + A*B.

521. Нехай існує тип користувача, який задає всі дні тижня - від понеділка до неділі включно. Описати множинний тип, який містить множини з:

1) назв будь-яких днів тижня;
2) назв робочих днів тижня;
3) назв вихідних днів тижня.

Задачі

522. Дано текст, що складається з латинських літер і символу «пробіл». Визначити, які символи використані в тексті, виводячи їх:

1) в алфавітному порядку;
2) перші входження літер у текст, зберігаючи їх взаємний порядок.

523. Дано два слова (слово - це текст, що не містить пробілів). Визначити символи, які:

1) є спільними для обох слів;
2) є в першому слові і відсутні в другому;
3) використовуються і в першому, і в другому словах.

524. Дано текст, що складається з літер української абетки. Вивести в алфавітному порядку літери абетки, використані в тексті.

525. Дано текст. Вивести символи цього тексту, які входять у нього більше як один раз.

526. Дано текст-формулу. Визначити:

1) які цифри використані в цій формулі;

2) які цифри не використовуються в цій формулі;

3) які арифметичні операції присутні у формулі;

4) які арифметичні операції відсутні у формулі.

527. У даному тексті визначити кількість використаних розділових знаків («.»; «!»; «?»; «,»; «—»; «:»).

528. Визначити, які розділові знаки («.»; «!»; «?»; «, »; «»; «:»):

1) використані у даному тексті;

2) не входять у даний текст.

529. У даному тексті визначити кількість груп однакових символів.

530. Дано текст, що складається з цифр і малих латинських літер, який закінчується крапкою. Визначити, яких літер більше в цьому тексті: голосних (а, е, і, о, и) чи приголосних.

531. Дано текст, за яким іде крапка. В алфавітному порядку надрукувати по одному разу всі малі українські голосні літери (а, е, є, и, і, ї, о, у, ю, я), що входять у цей текст.

532. Дано текст, записаний малими літерами української абетки. Між сусідніми словами - кома, за останнім словом - крапка. Надрукувати в алфавітному порядку:

1) усі голосні літери, які входять у текст;

2) усі приголосні літери, які не входять у жодне слово;

3) усі дзвінкі приголосні літери, які входять хоча б в одне слово;

4) усі глухі приголосні літери, які не входять хоча б в одне слово;

5) усі приголосні літери, які входять тільки в одне слово;

6) усі глухі приголосні літери, які не входять лише в одне слово;

7) усі дзвінкі приголосні літери, які входять більше ніж в одне слово;

8) усі дзвінкі приголосні літери, які входять у кожне непарне по порядку слово і не входять хоча б в одне парне слово;

9) усі глухі приголосні літери, які входять у кожне непарне по порядку слово і не входять хоча б в одне парне слово.

Примітка. Голосні літери - а, е, є, и, і, ї, о, у, ю, я; приголосні - усі решта літер окрім й, ь; дзвінкі приголосні - б, в, г, д, ж, з, л, м, н, р; глухі приголосні — л, п, с, т, ф, х, ц, ч, ш, щ.

533. Дано натуральне число n і послідовність цілих чисел

$$a_1, a_2, \quad a_n, \text{ де } 0 \leqslant a_i \leqslant 255.$$

Визначити в порядку зростання:

1) усі числа, які входять у послідовність по одному разу;
2) числа, взяті по одному з кожної групи однакових чисел;
3) усі числа із заданого інтервалу, які не входять у послідовність.

534. Дано натуральне число n та послідовність цілих чисел

$$a_1, a_2, \quad a_n, \text{ де } 0 \leqslant a_i \leqslant 255.$$

Визначити:

1) кількість різних членів послідовності;
2) кількість чисел, що входять у послідовність по одному разу;
3) кількість чисел, що входить у послідовність більш як один раз.

535. Дано натуральне число n і цілі числа $a_1, a_2, \quad a_n, b_1, b_2, \quad b_{25}$, де $0 \leqslant a_i \leqslant 255$, $0 \leqslant b_j \leqslant 255$. У послідовності $a_1, a_2, \quad a_n$ немає елементів, що повторюються; у послідовності $b_1, b_2, \quad b_{25}$ таких елементів також немає.

1) Побудувати переріз послідовностей $a_1, a_2, \quad a_n$ та $b_1, b_2, \quad b_{25}$, тобто отримати в довільному порядку числа, які належать послідовності $a_1, a_2, \quad a_n$ та послідовності $b_1, b_2, \quad b_{25}$ одночасно.
2) Побудувати об'єднання цих послідовностей.
3) Отримати всі елементи послідовності $a_1, a_2, \quad a_n$, які не входять у послідовність $b_2, \quad b_{25}$.
4) Чи правильно, що всі елементи послідовності $b_1, b_2, \ldots, b_{25}$ входять у послідовність $a_1, a_2, \quad a_n$?
5) Чи правильно, що всі елементи послідовності $a_1, a_2, \quad a_n$ входять у послідовність $b_1, b_2, \quad b_{25}$?

536. Дано натуральне число p ($1 < p < 10$). Визначити, чи може введене ціле додатне число N належати системі числення з основою p.

537. Визначити, чи може задана послідовність символів бути числом, записаним у шістнадцятковій системі числення.

538. Дано послідовність символів, яка є цілим додатним числом, записаним у шістнадцятковій системі числення. Визначити:

1) в яких ще системах числення може існувати таке число;
2) найменшу основу системи числення, в якій може існувати таке число.

539. Нехай дано тип користувача, який складається з двадцяти різноманітних продуктів (хліб, молоко, цукор, ковбаса, сир тощо). З інформації, яку подають *n* магазинів, визначити:

1) множину продуктів, які є в усіх магазинах;

2) множину продуктів, які є хоча б в одному з магазинів;

3) множину продуктів, яких немає в жодному магазині.

Вивести отриману інформацію.

540. Дано тип користувача, який складається з деяких найпоширеніших жіночих і чоловічих імен. Усі імена учнів школи занесені по класах у таблицю *A[i, j]*, де $i = 1, 2, \ldots n$; $I = 1, 2, \ldots m$ (n - кількість класів в школі, m - найбільша кількість учнів у класі). Якщо кількість учнів у класі менша за m, то зайві позиції відповідного рядка заповнюються символом «пробіл». Визначити:

1) які імена найчастіше зустрічаються в класі з номером k;

2) які імена найчастіше зустрічаються у кожному з класів;

3) які імена найчастіше зустрічаються у школі;

4) які імена зустрічаються в кожному класі один раз;

5) які рідкісні імена мають учні школи.

541. Для перевірки зорової пам'яті N пацієнтів була використана картина відомого художника. Для дослідження дані задавалися у вигляді $N + 1$ послідовності, де в першій містилися кольори, використані художником у картині, у другій, третій і т. д. - кольори, які запам'ятали пацієнти. Нехай дано тип користувача, який складається із семи кольорів: червоний, оранжевий, жовтий, зелений, блакитний, синій, фіолетовий. Визначити:

1) які кольори, використані в картині, помітили всі пацієнти;

2) які кольори, використані в картині, не помітив жоден із пацієнтів;

3) які кольори, не використані в картині, назвав хоча б один пацієнт;

4) які кольори, використані в картині, називали пацієнти найчастіше.

542. У багатьох програмах передбачається введення інформації, що складається лише з певних символів. Розробити програму-захист, яка контролює введення:

1) числового коду;

2) латинських символів;

3) кирилиці;

4) символів української абетки.

Записи[1]

Розглянемо новий тип - **record,** компактний у записі, зручний у користуванні.

Загальний вигляд опису записів:

```
< ім'я змінної > : record
                     <список ідентифікаторів полів
                     із зазначенням їх типів>
                   end;
```

Приклади:

```
type
  person = record
             name, surname: string;
           end;
  man = record
          name: string;
          age: integer;
          birth_day: record
                       day, month, year: integer;
                     end;
        end;
var
  list: array [1..50] of man;
  student: record
             school, class, group: integer;
           end;
  a, b, c: person;
```

Як звернутися до змінних типу «запис»? Це зробити дуже просто:

```
list[i].name,  list[i].birth_day.year,  student.school  і т. д.
```

Пояснимо деталі.

По-перше, ідентифікатори типів не можна використовувати як імена змінних. Тобто ви не повинні писати таким чином:

```
person.surname.
```

У даному разі person — це деякий образ, на який ми можемо лише посилатися, описуючи змінні. Ви ж не пишете **integer** := ... .

По-друге, записи є структурованими типами, оскільки вони складаються з елементів певного типу. (Нагадаємо, що типи Pascal, які у свою чергу складаються з інших типів, вважаються структурованими.) Елементи, з яких складаються записи, називаються його *полями.* Вони визначаються ідентифікаторами за правилами найменування змінних величин. У наведе-

[1] Задачі цього розділу специфічні саме для Pascal, але це зовсім не означає, що вони не можуть бути реалізованими за допомогою можливостей інших мов програмування.

них прикладах поля можуть бути записами, тобто розбиватися на свої поля.

По-третє, щоб добутися до значення якогось поля запису, треба вказати спочатку ім'я змінної типу **record** і через крапку - ім'я необхідного поля. Якщо воно має наступну вкладеність, то через крапку - ім'я його внутрішнього поля і т. д.

Тип «запис» — це об'єднана разом під одним іменем наперед визначена сукупність змінних різного типу.

Коли зручно користуватися таким типом? Ця нагода трапляється тоді, коли маємо справу з декількома масивами різних типів змінних, над якими паралельно треба виконувати певні дії. Наприклад, у задачі з обробки інформації про види продукції деякого підприємства: назва виду продукції, його вартість, вироблена кількість.

Зрозуміло, що змінна типу **record** може брати участь лише в операторі присвоювання, якщо їй присвоювати значення іншої змінної такого самого типу. Ми не можемо змінні цього типу множити, додавати і тому подібне. Зовсім інша справа - поля змінної типу «запис». Якщо вони не є типом «запис», то над ними можна виконувати всі дії, які дозволяє їх тип.

Для того щоб позбутися незручностей при «глибоких» вкладеностях записів, у Pascal введений ще один оператор - оператор приєднання

with < ім'я змінної типу **record** > **do** P,

де P - оператор.

Використовуючи оператор приєднання для змінних масиву list з попереднього прикладу, можна записати:

```
with list[i] do
  begin
    readln (name);
    readin (age);
    with birthday do
      begin
        readln (day);
        readln (month);
        readln (year);
      end
  end;
```

Розглянемо приклад програми з використанням змінних типу **record,** яка дає змогу визначити телефон абонента за прізвищем і адресою.

```
program telefon;
  type
    S1 = string[20];
    S2 = string[40];
```

```
    abonent = record
                name:   S1;
                adress: S2;
                tel:    integer;
              end;
var
  list:   array[1..500] of abonent;
  person: S1;
  adr:    S2;
  i, n:   integer;
begin
  repeat
    write ('Задайте кількість абонентів телефонної мережі ');
    readln(n);
  until (n >= 1) and (n <= 500);
  writeln ('Задайте інформацію про абонентів:');
  for i := 1 to n do
    with list [i] do
      begin
        write ('Прізвище: ');
        readln (name);
        write ('Де мешкає: ');
        readln (adress);
        write ('Телефон: ');
        readln (tel);
      end;
  writeln ('Назвіть прізвище абонента, чий телефон вас цікавить:');
  readln (person);
  writeln ('Де він мешкає:');
  readln (adr);
  i := 1;
  while (i <= n) and (list[i].name <> person) and
        (list[i].adress <> adr) do inc(i);
  if i <= n then
        writeln ('Телефон абонента: ', list[i].tel)
      else
        writeln ('У цього абонента немає телефону');
end.
```

Запитання для самоконтролю

1. Запишіть загальний вигляд опису змінних типу «запис».
2. У яких ситуаціях зручніше використовувати тип record?
3. Яким чином можна звернутися до значень полів змінної типу «запис»?
4. Запишіть загальний вигляд оператора приєднання.
5. У чому полягає зручність використання оператора приєднання?

Не припускайтеся помилок!

Якщо в процедурі **read** або **write** використати ім'я типу «запис» без зазначення імені поля, то буде видано знайоме вам повідомлення про помилку:

Error 3: Unknown identifier.

Виконайте завдання

Вправи

543. Як виглядатиме опис змінної типу **record,** якщо вона повинна відображати таку інформацію:

1) координати точки на площині;
2) вартість у гривнях і копійках;
3) дату (рік, місяць, число);
4) час (година, хвилина, секунда);
5) прізвище, ім'я, по батькові співробітника;
6) картку на книжку в бібліотеці у такому вигляді: шифр, автор, рік видання, назва, дані про читача (прізвище, ім'я), адреса (назва вулиці, номер будинку, номер квартири)?

544. У програмі описані такі дані:

```
var x: array[1..100] of integer;
    y: array[1..100] of integer;
    a: array[1..100] of string[20].
```

Замінити цей опис описом змінної типу **record.**

545. Нехай у програмі описано змінну:

```
var a: record
         x, y: integer;
         z:    record
                 n: real;
                 m: char
               end
       end;
```

Яка група операторів присвоювання буде некоректною і чому:

1)
```
a.x := 10; a.y := 5;
with z do
  begin
    n := 0;
    m := 'A'
  end;
```

2)
```
with a do
  begin
    x := 10; y := 5;
    with z do
      begin
        n := 0.5; m := 'д'
      end;
  end;
```

3)
```
with a do
  begin
    x := 10; y := 5;
    n := 0.5; m := A
  end;
```

4)
```
a.x := 10;
a.y := 5;
a.z.n := 0.5;
a.z.m := 'A';
```

5)
```
a.x := 10.5;
a.y := 5;
z.n := 0.5;
z.m := 'A'?
```

546. Нехай у програмі, що обробляє координати і колір точок на екрані монітора, описано змінну:

```
var t: record
         x, y: integer;
         color: string[15]
       end;
```

Змінній *t* треба надати значення точки з координатами (10; 25), що зображена червоним (red) кольором. Які з наступних операторів приєднання виконують це завдання без помилок:

1)
```
with t do
  begin
    x := 10; y := 25
  end;
t.color := 'red';
```

2)
```
with t do
  begin
    t.color := 'red';
    x := 10; y := 25
  end;
```

3)
```
t.color := 'red';
with t do
  begin
    x := 10; y := 25
  end;
```

4)
```
color := 'red';
with t do
  begin
    t.x := 10; t.y := 25
  end?
```

Задачі

547. Вважатимемо, що в програмі зроблено описи:

```
type time = record
              hour: 0..23;
              minute, second: 0..59
            end;
```

Розробити програму, за допомогою якої можна було б визначити:

1) чи передує в межах доби час t1 часу t2, створивши для цього логічну функцію before(t1, t2);
2) час t1, що на одну секунду більший за заданий час t, врахувавши при цьому перехід на наступну добу і створивши для цього процедуру NextSec з параметрами t,t1;
3) який час d (у годинах, хвилинах, секундах) пройшов від часу t1 до часу t2 (t2 >t1), створивши для цього процедуру interval з параметрами d, t1, t2.

548. Нехай у програмі описані такі типи і змінні:

```
type name = (Anne, Valentina, Eugene, Petr, Alex, Max,
             Maria, Jura, Nataly);
     data = record
              sex: (female, male);
              height: 140..200
            end;
var group: array [name] of data;
```

Скласти програму, яка б за даним масиву group визначала:

1) середній зріст жінок;
2) ім'я найвищого чоловіка;
3) чи є хоча б дві людини однакового зросту;
4) дані про всіх жінок.

549. Програма містить описи типів і змінних:

```
type str = string[16];
     inhabitant = record
                    surname, city: str;
                    address: record
                               street: str;
                               house, flat: 1..999
                             end
                  end;
var list: array [1..20] of inhabitant;
```

Скласти програму, яка б визначала, чи є в даному списку люди, які мають однакову адресу, але мешкають у різних містах.

550. В описовій частині програми містяться описи:

```
type str = string[16];
     data = record
              number: 1..31;
              month: 1..12;
              year: 1900..1999
            end;
     form = record
              surname: str;
              sex: (female, male);
              birthday: data
            end;
var group: array [1..25] of form;
```

Розробити програму, яка б:

1) визначала найстаршого чоловіка у введеному списку (вважаючи, що такий є і він лише один);
2) друкувала всі прізвища людей зі списку, що починаються з указаної літери, та інформацію про їх дату народження;
3) обчислювала кількість людей у списку, що мають однакову дату народження.

551. Нехай програма містить описи:

```
type number = 1..31;
     month = 1..12;
     year = 1..2050;
     data = record
              n: number;
              m: month;
              y: year
            end;
weekdays = (sunday, monday, tuesday, Wednesday,
            thursday, friday, saturday);.
```

Вважаючи, що всі дати задаються за григоріанським календарем, визначити:

1) кількість днів у тому місяці, якому належить уведена дата;
2) коректність уведеної дати (наприклад, не може бути 30 лютого);
3) кількість днів, що пройшла між двома введеними датами;
4) день тижня, на який припадає введена дата, зважаючи на те, що 1 січня 1-го року було понеділком;
5) скільки повних тижнів міститься в році, якому належить уведена дата.

552. Опис типів користувача у програмі має вигляд:

```
type words = string[9];
     number_telephone = 100000..999999;
     friend = record
                surname: words;
                number: number_telephone
              end;
     page = array [1 ..20] of friend;
     notebook = array ['A'..'Z'] of page;
```

Вважаючи, що на кожній сторінці записника вказано прізвища, які починаються з однієї і тієї самої літери, що є індексом сторінки, скласти програму, яка могла б визначити:

1) чи є в записнику відомості про знайомого з указаним прізвищем і якщо є, то друкувала б номер його телефону;
2) наявність вказаного номера телефону і виводила б прізвище його власника;
3) на яких сторінках записника розташовано номери телефонів знайомих, прізвища яких починаються з указаної літери;
4) кількість записів про знайомих у записнику, прізвища яких починаються з указаної літери;
5) прізвища і ввві дані про знайомих, номери телефонів яких починаються з указаної цифри.

ПОКАЖЧИКИ. ПОСИЛАЛЬНІ ТИПИ. ДИНАМІЧНІ ЗМІННІ

Нам у житті завжди чогось бракує, а програмістам завжди, в першу чергу, бракує комп'ютерної пам'яті.

Давайте нагадаємо, яким чином зберігаються дані в пам'яті комп'ютера. Значення кожної змінної записане в деякій області пам'яті. Програмі відомо, за якою адресою записана ця змінна і скільки місця в пам'яті комп'ютера вона займає, оскільки кожна змінна має свій тип і довжину.

Отже, приходимо до ідеї роботи з безіменними величинами, звертаючись до них лише за адресами, з яких вони починаються.

Тому тепер треба знати не ім'я змінної, а її адресу. Величини, які містять адреси, мають тип **pointer,** тобто «точка входу». Це новий для нас тип. Величини цього типу займають в пам'яті 4 байти.

У чому полягає зручність використання змінних, які містять адреси замість імен змінних? Тепер можна, змінюючи значення змінної типу **pointer,** насправді змінювати адресу, яка там записана, і тим самим подорожувати пам'яттю, читаючи звідти інформацію або записуючи її туди. Звичайно, такі подорожі дуже ризиковані і можуть призвести до втрати дорогоцінної інформації. Тому до набуття певного досвіду варто обережно використовувати тип **pointer.**

Тип **pointer** указує на початок пам'яті, в якій записано певну інформацію, розміри якої невідомі. Ми їх можемо звідти читати щонайменше байтами. Але найчастіше в програмах маємо справу з типізованими змінними. Тому нам зручно було б, крім адреси початку значення змінної, вказати також програмі, скільки місця займає ця змінна. Для цього в Pascal існують різновиди типу **pointer,** які носять назву ***посилальних типів*** і позначаються символом «^» перед типом змінної. Тип самої змінної, на яку вказує змінна посилального типу, називається ***базовим.***

Перейдемо до прикладів:

```
type A = array [1 ..100] of real;
var P_int:  ^integer;
    P_real: ^real;
    P_A:    ^A;
    P:  pointer;
```

Змінна P_int міститиме адресу, що вказуватиме на комірку пам'яті, починаючи з якої записане значення цілої величини, тобто довжиною в 2 байти. При читанні цього значення Pascal розкодовуватиме послідовність 0 та 1, яка там міститься, за правилами кодування цілих чисел. Змінна P_real відповідно міститиме адресу початку величини типу **real.** Корисно розібрати змінну P_A. Вона вказуватиме на початок області пам'яті комп'ютера, з якої починається розташування елементів масиву А, кожний з яких є дійсним числом. Реальна кількість елементів масиву, з якою працюватиме користувач, поки що невідома. Відомо лише, що вона не може перевищувати числа 100.

У багатьох типів Pascal існує можливість присвоювання «нульових» значень:

```
v_integer:=0; v_string :=' '; v_set := [].
```

Схожа операція є і для посилальних типів:

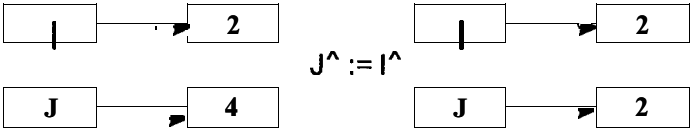
vpoint := nil.

У цьому випадку змінна vpoint вказуватиме **«в нікуди».**

Як же дістатися до значення, що записане за адресою, на яку вказує змінна типу «покажчик» або посилального типу? Для цього в Pascal існує операція ***розіменовування.***

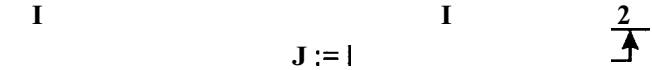
Якщо P вказує на адресу, за якою записане деяке значення, то P^ — це вміст області пам'яті комп'ютера, тобто значення змінної, записаної за адресою P.

Наприклад, результатом виконання операції J^ := I^ буде копіювання значення змінної, записаної за адресою *I*, в область пам'яті, на яку вказує змінна *J*. Ця операція можлива, якщо за обома адресами записані значення змінних однакового типу (мал. 26).



Мал. 26

Якщо виконати операцію J := I, то змінна *J* матиме таку саму адресу, як і змінна *I* (мал. 27).



Мал. 27

У результаті виконання цієї операції посилання на значення 4 втрачене. У пам'яті комп'ютера в цьому місці фактично лишилося **«сміття»,** бо цим значенням уже ніхто не зможе скористатися. Саме подібних ситуацій треба уникати.

Структури змінних, на які вказують величини посилальних типів, називають ще ***динамічними,*** оскільки ми самі розподіляємо їх у пам'яті та звільняємо від них пам'ять, коли необхідність у їх значеннях відпадає. Це класичний спосіб роботи з масивами невідомої наперед довжини. Бувають також задачі, коли кількість і розмірність масивів не дозволяє одночасно розташувати їх у пам'яті. Проте з алгоритму видно, що робота з усіма масивами одночасно не ведеться. Отже, резервувати для них заздалегідь пам'ять, як ми робили це досі, нераціонально.

Для розміщення динамічних змінних Pascal відводить спеціальну область пам'яті розміром в 64 Кбайти. Ця область носить назву *«купи»*.

Розглянемо процедурні та функціональні можливості Pascal, які допоможуть вам при використанні посилальних типів.

Нагадаємо, що в процедурах і функціях після їх назв подається список формальних параметрів із зазначенням їх типів. Параметри, перед якими є службове слово var, є результатами, а перед якими немає - аргументами.

1. Процедура New(var P: pointer) відводить місце для зберігання динамічної змінної P^ і присвоює її адресу покажчику P.

2. Процедура **GetMem(var** P: pointer; Size: integer) відводить місце в Size байт у «купі» і присвоює адресу його початку покажчику P. При цьому відводиться нерозривний блок пам'яті.

3. Процедура Dispose(var P: pointer) вивільняє пам'ять від динамічної змінної, на яку вказував покажчик P.

4. Процедура **FreeMem(var** P: pointer; Size: integer) звільняє Size байт у **«купі»,** починаючи з адреси, що записана у змінній P.

5. Функція MaxAvail повертає довжину в байтах найбільшого суцільного вільного блоку динамічної пам'яті.

6. Функція **Mem** Avail повертає суму в байтах усіх довжин вільних блоків у **«купі».**

А тепер розглянемо програму, що демонструє роботу з динамічною пам'яттю. Для цього використаємо задачу, подібну до тих, які розв'язували, вивчаючи масиви.

Нехай нам треба розташувати в динамічній пам'яті елементи цілочислового масиву, кількість яких наперед невідома, та порахувати кількість значень, з яких утворено цей масив, якщо відомо, що їх кількість не перевищує 256.

```
program dimension;
  type
    Dim = array [1 ..5000] of integer;
  var
    P:          ^Dim;
    i, n, L, m  integer;
    SL:         integer;
    PSize:      longint;
    S:          set of byte;
begin
  write ('Задайте кількість елементів масиву: ');
  readln (n);
  SL := SizeOf (integer);
  PSize := SL*(MaxAVail div SL);
  L:=n*SL;
  if (L > PSize) or (SizeOf(Dim) > PSize)
```

```
        then
          writeln ('Такий масив не може бути розміщений у пам''яті')
        else
          begin
            GetMem(P, L);
            S := []; m := 0;
            for i := 1 to n do
              begin
                readln (P^[i]);
                if not (P^[i] in S) then
                  begin
                    S := S + P^[i];
                    m := m + 1
                  end
                writeln (P^[i]);
              end;
            for i := 1 to n do
            writeln (P^[i]);
            writeln ('Кількість значень, з яких складається масив: ', m);
            FreeMem(P, L);
          end;
    end.
```

У цій програмі використано функцію **SizeOf,** яка визначає розмір у байтах одного елемента вказаного типу.

Використання динамічної пам'яті для розміщення даних різної структури — дуже потужна можливість. За допомогою динамічних змінних організовуються і розміщуються в пам'яті такі дані, як стеки, списки, черги, дерева.

Запитання для самоконтролю

1. Що розуміють під динамічним розподілом пам'яті комп'ютера?
2. Яка відмінність між типом «покажчиком» і посилальним типом?
3. Який розмір «купи» в Pascal?
4. Що таке операція розіменовування? Яким символом вона позначається?
5. Назвіть процедури, які розподіляють пам'ять під динамічні змінні.
6. Назвіть процедури, які звільняють пам'ять від динамічних змінних.
7. Назвіть функції, які визначають об'єм вільної області в «купі».

Не припускайтеся помилок!

Якщо розмір динамічного масиву перевищує 64 Кбайти, то на це ви отримаєте таку реакцію:

Error 29: Ordinal type expected.

Виконайте завдання[1]

Вправи

553. Нехай у програмі зроблено такі описи:

```
type r = ^integer;
var p, q: r;
```

Нехай змінні *p, q* набувають значень, зображених на малюнку (мал. 28).

Мал. 28

1) Чим є значення змінної p — посиланням на величину цілого типу чи самим цим значенням?
2) Що визначає запис p^ - посилання на величину цілого типу чи саму цю величину?
3) Яка різниця між записами p та p^ і які типи цих величин?
4) Які з операторів присвоювання коректні:
   a) p := q; c) q := p^;
   b) p^ := q; d) q^ := p^.
5) Які з наступних висловлювань хибні:
   a) значенням змінної p є число 5;
   b) змінна q вказує на число 3;
   c) змінна q посилається на область пам'яті, де записане число 3;
   d) типи змінних p і q однакові;
   e) типи змінних p і p^ однакові;
   f) величина, значення якої записане за адресою, що міститься у змінній p, не має імені;
   g) значення змінної p можна змінювати;
   h) після виконання оператора p := q значення 5 буде втрачене;
   i) щоб збільшити число 3 на 1, треба виконати оператор q := q^ + 1.
6) Що буде виведено на друк у результаті виконання таких операторів:

[1] Розв'язувати задачі з використанням динамічних змінних можна, скориставшись задачами з одновимірними і багатовимірними масивами. Якщо наперед невідома кількість елементів масивів, для економнішого використання оперативної пам'яті варто використовувати динамічну пам'ять.

```
p := q ;
if p = q
   then p := nil
   else if p^ = q^
        then q := p;
if p = q then q^ := 4;
writeln (p^)?
```

554. Якщо в програмі описані такі змінні

```
var p, q: ^integer; r:^char;,
```

то які з наступних операторів правильні і чому?

1) `p = q;`
2) `q = r;`
3) `r^ := p^;`
4) `p^ := nil;`
5) `q^ := ord(r^);`
6) `if r <> nil then r^ := nil^`
7) `q = p^;`
8) `p = nil;`
9) `r := nil;`
10) `if q > nil then q^ := p^`
11) `if q = p then write(q);`
12) `if q <> r then read(r^).`

555. Конструкцію даних, описану таким чином:

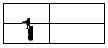
```
type chain = ^link;
     link = record
          n: integer;
          next: chain
          end;,
```

можна назвати ланцюгом (chain), який складається з окремих ланок (link) - саме число *n* та посилання на наступну ланку ланцюга next. Кінцем ланцюга вважатимемо ситуацію, коли на деякому кроці next = **nil.**

Нехай у програмі описано змінну

```
var p: chain;
```

і задано таке схематичне значення змінної, на яку вона вказує:



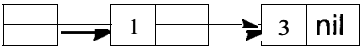
3 nil

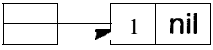
Перетворення змінної *p* до такого схематичного вигляду

можна виконати за допомогою оператора присвоювання

```
p := p^.next.
```

Які оператори присвоювання перетворять значення змінної *p* у такий вигляд:

1) 

2) 

Задачі

556. Використовуючи розташування масивів у динамічній пам'яті, розв'язати будь-які із задач № 311–342.

557. Під час розв'язування задач № 343-376 розмістити значення елементів таблиць у динамічній пам'яті і простежити за допомогою відповідних стандартних функцій, скільки вільного місця було в динамічній пам'яті до введення початкових даних та скільки його залишилося після введення.

558. Ввести *n* елементів одновимірного цілочислового масиву, розмістивши їх у динамічній пам'яті, та впорядкувати їх там за спаданням.

559. Дано непорожню послідовність натуральних чисел, яка завершується числом 0. Надрукувати порядкові номери тих чисел послідовності, які мають найбільше значення.

560. Дано послідовність з не менш ніж двох різних натуральних чисел, що завершується 0. Надрукувати у зворотному порядку всі числа, розташовані між найменшим і найбільшим членами цієї послідовності.

561. Для роботи деякої програми треба використати три однакові за довжиною одновимірні масиви. Визначити, скільки елементів масивів можна використовувати у програмі, якщо ці елементи:

1) цілочислові;
2) дійсні;
3) типу
```
record
    name:    string[10];
    surname: string[20];
    age:     byte;
end;
```

Скільки вільної пам'яті лишиться в «купі» після розташування там зазначених масивів даних?

562. Послідовність з *n* цілих чисел розташована в динамічній пам'яті парою таких даних: число і покажчик на область у динамічній пам'яті, де розташоване наступне число (якщо значення покажчика **nil**, то це означає, що дане число в послідовності останнє). Для цього в програмі є такий опис:
```
type nb = ^number;
     number = record
                numb: integer;
                next: nb;
              end;
```

Скласти програму, за допомогою якої можна було б ввести початкову кількість цілих чисел, а після цього виконати такі дії:

1) дописати ще одне ціле число після першого від'ємного значення;
2) вилучити перше додатне число;

3) поміняти місцями перше й останнє числа;

4) розташувати першим найменше число, останнім - найбільше число; решту елементів змістити всередину, не змінюючи при цьому їх взаємного розташування.

563. Організацію даних, описаних у попередній задачі, будемо називати списком. Щоб список даних був замкненим, останній елемент списку повинен указувати на перший елемент або перший - на останній. Нехай список складається з n елементів, значення яких збігаються з їх порядковими номерами. Визначити значення елемента замкненого списку, який стоїть на k-му місці $(k > n)$, почавши відлік від першого.

564. Використовуючи умову задачі № 563, список можна організувати як двонаправлений, тобто для кожного елемента ввести інформацію про розташування як наступного, так і попереднього елементів. Зокрема, для першого елемента попереднім, а для останнього наступним буде значення **nil** (якщо цей список не замкнений). Опис змінних виглядає так:

```
type nb = ^number;
     number = record
                numb:              integer;
                preceding, next: nb;
              end;
```

Скласти програму, яка могла б знайти даний елемент у списку і вивести сусідні елементи, врахувавши, що для першого елемента не існує попереднього, а для останнього - наступного.

ДОДАТКОВІ МОЖЛИВОСТІ ВВЕДЕННЯ ТА ВИВЕДЕННЯ ІНФОРМАЦІЇ

Файли

Ви вже навчилися вводити велику кількість початкових даних, задаючи їх у розділі констант. Але й це не завжди зручно. Адже для коригування даних треба мати початковий текст програми, в якому і робляться всі виправлення. А як бути, коли треба працювати з exe-файлом? Тут стануть у пригоді файли. Саме в них будемо тримати вхідні дані для роботи програми. Тепер треба, щоб програма вміла прочитати їх із файла. Обговорення цих питань і є метою цього розділу.

З поняттям файла можна ознайомитися в будь-якій літературі з основ інформатики.

У Pascal вводиться поняття логічного і фізичного файлів. Фізичний (з яким досі мали справу) - файл на зовнішньому носії, наприклад на дискеті. Такий файл ідентифікується іменем, доступ до нього визначається певним шляхом. Наприклад,

C:\PASCAL\my file.dat.

Логічний файл у Pascal - це змінна типу «файл», яка має такий загальний вигляд опису:

<ім'я змінної файлового типу>: **file of** <тип>.

Наведемо приклади:

```
var f_int:   file of integer;
    f_real:  file of record
                       name:  string;
                       marks: integer;
                     end;
    f_char: file of char;
    f_txt:  text;
    f:      file;
```

У цих оголошеннях змінних відсутнє справжнє ім'я файла на носієві. І справді, Pascal працює з файлами на рівні імен змінних, тобто в самій програмі звертаємося до файла як до змінної величини. Це і є логічним файлом. Проте як програма зрозуміє, де цей файл знаходиться насправді і як він називається? Цю інформацію ви визначите в Pascal-програмі лише один раз перед початком роботи з файлом за допомогою спеціальної процедури.

У прикладах, наведених вище, усі файли, крім останніх трьох, називаються *типізованими.* Адже для всіх цих файлів указаний тип елементів, з яких вони складаються. Два передостанні файли є *текстовими:* для їх опису можна використовувати будь-який з варіантів. Останній запис - це приклад опису *нетипізованого* файла, (для його елементів не вказано тип). У цьому разі Pascal вважатиме, що файл складається з окремих байтів. Тобто читання і виведення інформації йтиме блоками, мінімальний з яких становитиме один байт, а далі вже ваша справа, як ви будете опрацьовувати її. Звичайно, що використання такого варіанта опису файлів дещо специфічне.

У межах цього посібника розглядатимемо лише текстові й типізовані файли, які на практиці зустрічаються найчастіше.

І ще одне зауваження: тип «файл» є структурованим, тому що він складається з елементів іншого типу.

Дотепер ми вводили і виводили інформацію, не задумуючись над тим, як програма розуміє, що введення треба робити з клавіатури, а виведення - на екран монітора. Справа в тому, що, якщо не вказується додаткова інформація про введення/виведення даних, Pascal надає цим файлам системне ім'я CON (консоль), що означає для введення - клавіатуру, а для виведення - монітор. Для читання інформації з інших файлів необхідно використовувати такий різновид стандартної процедури читання

read (<ім'я логічного файла>, <список змінних>).

Для виведення даних у файл відповідно використовується процедура

write (<ім'я логічного файла>, <елементи виведення>).

Введення/виведення інформації в текстових і типізованих файлах трохи різниться, тому це питання розглядатиметься окремо.

Чому особлива увага приділяється саме введенню/виведенню інформації під час роботи з файлами? А тому, що саме це відрізняє програми, що працюють з файлами, від програм, що працюють з клавіатурою і монітором. Адже коли інформація потрапить у пам'ять комп'ютера, то її можна обробляти за визначеним алгоритмом.

Стандартні процедури і функції для роботи з файлами

1. Перш ніж розпочати роботу з файлом, необхідно логічний файл пов'язати з відповідним йому фізичним файлом. Для цього існує процедура

Assign (<файлова змінна>, <ім'я фізичного файла>).

Якщо файл знаходиться в тому самому каталозі, де й програма, яка його використовує, то достатньо вказати лише ім'я файла. В іншому випадку треба вказати шлях до цього файла.

Приклади:

```
Assign (f_1, 'my_file.dat');
Assign (fl, 'c:\data\my_file.dat');
Assign (f_str, str).
```

Дуже цікавий останній приклад, в якому ім'я фізичного файла міститься в рядковій змінній і це значення можна вводити з клавіатури, роблячи програму більш незалежною від конкретних файлів.

Процедура **Assign** безпосередньрго доступу до файла не дає. Вона лише повідомляє програмі, з яким файлом ви захочете працювати.

2. Наступні дві процедури дають необхідний доступ до файла, відкриваючи його для читання

Reset (<файлова змінна>)

та для запису

Rewrite (<файлова змінна>).

Їх використання для текстових і типізованих файлів дещо відрізняється.

Цікаво, що коли процедурою Rewrite буде відкрито вже існуючий файл, то інформація в ньому втрачається. Тому, перш ніж скористатися нею, треба бути дуже уважними!

3. Відповідно до попередніх процедур, існує процедура, яка припиняє доступ до файла, закриваючи його:

Close (<файлова змінна>).

Зв'язок із фізичним файлом при цьому не втрачається, лишається лише відкрити його ще раз.

Бажано ніколи не забувати закривати файли, з якими робота вже завершена. Особливо це важливо, коли ви здійснювали запис до файла. Якщо ви після цього його не закриєте, то вся робота може бути марною. Адже інформація записується на диск із буфера введення/виведення (область пам'яті комп'ютера, що відводиться при відкритті файла) лише тоді, коли він заповниться. Тому ви втратите останній буфер інформації, а при невеликій її кількості файл на диску ви знайдете, але його довжина буде нульовою.

4. Інколи виникає необхідність з працюючої програми знищити непотрібні файли. Користуйтеся для цього процедурою

Erase (<файлова змінна>).

5. Подумайте, може перед знищенням файла бажано зберегти його інформацію у файлі з іншим ім'ям? Якщо так, то для цього в Pascal передбачена процедура

Rename (<файлова змінна>, <нове ім'я фізичного файла>).

6. Функція

EOF (<файлова змінна>): **boolean**

повертає значення **True**, якщо при обробці файла було досягнуто його кінця, і **False** — навпаки.

Наприклад:

```
Assign (f, 'my_file.dat');
Assign (fnew, 'new_file.dat');
Reset (f);
  i := 0;
while not EOF (f) do
  begin
    i: = i + 1;
    read (f, a);
    write (a, ' ')
  end;
Rename (f, f_new);
Close (f_new);
```

Особливості роботи з текстовими файлами

Під час роботи з текстовими файлами програма читатиме з них інформацію так само, як і з клавіатури. Виведення даних у текстовий файл аналогічне виведенню їх на екран монітора.

Розглянемо детальніше використання стандартних процедур **read, readln, write, writeln** для текстових файлів.

Спочатку нагадаємо, як працюють процедури **write, writeln** під час виведення інформації на екран монітора. Під час використання процедури **write** інформація виводиться на екран монітора в один рядок і курсор залишається після останнього виведеного символу, чекаючи продовження роботи. Під час використання процедури **writeln** виведення рядка на екран завершується переведенням курсору на початок наступного рядка, тобто після коду останнього символу записується код клавіші **Enter** (код 13). Саме таким чином інформація записується і у файл.

Тепер під час читання даних з файла процедура **read** вводитиме з файла по одній порції інформації, ігноруючи код символа **Enter,** тобто якщо рядок «закінчиться», то вона перейде до читання інформації в наступному рядку. Під час використання процедури **readln** відбудеться одночасне читання цілого рядка інформації до коду символа **Enter.** При повторному використанні цієї процедури читатиметься вже наступний рядок. Якщо до символу **Enter** знаходитиметься більше значень, ніж вказано в процедурі **readln,** то всі зайві значення будуть проігноровані.

Слід зауважити, що саме читатиме й писатиме програма, працюючи з текстовим файлом. Це залежатиме від того, якими типами описані змінні в процедурах **read/readln** та **write/writeln.** Якщо змінні типу **integer** або **real,** то у файлі повинні бути записані числа цього типу через пробіл, а якщо типу **char,** то інформація буде читатися посимвольно. Наприклад, процедура **readln** (f, a, b, c) прочитає з файла / увесь рядок до символу **Enter** і присвоїть три числа відповідно змінним *a, Ь, с,* якщо вони описані як числові типи. Але все це за умови, що в цьому рядку є саме три елементи і вони є числовими значеннями.

Нехай деякий текстовий файл містить таку інформацію:

```
1 2 3
4 5 6
7 8 9
```

Спробуйте відповісти на два запитання.

Яка інформація буде надрукована таким фрагментом програми:

```
for i := 1 to 3 do
  begin
    read (f, a);
    write (a, ' ')
  end?
```

А яку інформацію надрукує фрагмент програми, читаючи її з того ж самого файла:

```
for i := 1 to 3 do
  begin
    readln (f, a);
    write (a, ' ')
  end?
```

А тепер відповіді. У першому випадку на екрані монітора з'являться числа 1 2 3, оскільки числа читатимуться з одного рядка, а в другому - 1 4 7, оскільки після читання кожного першого числа в рядку процедура **readln** переходитиме до наступного рядка.

Для створення текстових файлів, з яких інформація вводитиметься у програму, можна використовувати звичайні текстові редактори. Для цього може бути використане і середовище Turbo Pascal: за допомогою меню **File - New** необхідно відкрити нове вікно, набрати через пробіл потрібні числа і зберегти цю інформацію у файлі. Зручно надавати файлам, що містять вхідну інформацію, імена з розширеннями *.dat або *.txt.

Для роботи з текстовими файлами додаються деякі процедури та функції.

1. Процедура

Append (<файлова змінна>)

дає змогу відкривати текстові файли для дописування інформації в кінець цих файлів (покажчик файла встановлюється в кінець уже існуючого файла). Ця процедура доповнює процедури **Rewrite,** яка відкриває для роботи новий файл, та **Reset,** яка відкриває текстовий файл, зберігаючи в ньому інформацію і встановлюючи покажчик на початок файла.

2. Функція

Eoln (<файлова змінна>): **boolean**

повертає значення **true,** якщо при читанні символів із текстового файла досягнуто код символу **Enter** (код 13), і **false** - у протилежному випадку.

Найкраще особливість роботи з текстовими файлами продемонструє приклад.

Нехай деякий текстовий файл містить такі дані:

10 -2 1 102 – 95 5 20 11.

Програма, що знаходитиме суму елементів цього файла, може мати такий вигляд:

```
program listing;
  var f, f_res:   text;
      S, a, i:    integer;
      st:         string[12];
  begin
    writeln ('Задайте ім"я вхідного файла:');
    readln (st);
    writeln ('Задайте ім"я результуючого файла:');
    readln (st_res);
    Assign (f, st);
    Assign (fres, st_res);
    Reset (f);
    S := 0;
    while not EOF (f) do
```

```
      begin
        read (f, a);
        S := S + a;
      end;
    Close (f);
    Rewrite (f_res);
    write (fres, S);
    Close (f_res);
  end.
```

У наведеному прикладі для читання числової інформації з текстового файла ми використали цикл **while ... do.** Оскільки ***текстові файли є файлами послідовного доступу,*** то визначити наперед кількість чисел, що містяться в них, неможливо. Але цю незручність текстових файлів можна обійти, записавши на початку такого файла кількість чисел, що записані далі.

Тобто вхідний текстовий файл може містити інформацію

5 1 3 4 0 2,

а фрагмент програми, що читатиме з нього числову інформацію, може мати такий вигляд:

```
read (f, n);
for i := 1 to n do
  read (f, a);
```

Отже, з текстового файла можна читати числові значення так само, як і з клавіатури, і записувати у файл так само, як і виводити на екран монітора. Звичайно, що це можна робити лише тоді, коли відомо, що дані у файлі є і саме такі, які вам потрібні. Це стосується і дійсних чисел.

Інша справа із звичайною текстовою інформацією. Її треба обробляти посимвольно. Вважатимемо, що в текстовому файлі є деякий рядок інформації. Це робиться так:

```
program words;
  varf: text;
      a: char;
      S: string;
begin
  writeln ('Задайте ім''я файла:');
  readln (str);
  Assign (f, str);
  Reset (f);
                        {перевірка кінця файла перед}
                        {читанням символу}
    if not EOF (f) then read(f, a);
                        {організація циклу для читання}
                        {символьної інформації до кінця файла}
    while not EOF (f) do
      begin
        S := ' ';       {підготовка змінної для накопичення слова}
        while a in ['a'..'z'] do   {цикл, що фіксує}
```

```
            begin                    {всі латинські літери}
              S := S + a;            {накопичення змінної-слова}
              read (f, a);
            end;
          writeln (S);  {виведення на монітор отриманого слова}
                        {цикл, що пропускає всі розділові знаки}
          while a in [' ', ',', '!', '?',      ';', '–']do
            read (f, a);
        end;
      Close (f);
  end.
```

Зрозуміло, що в текстовий файл дані можна записати не тільки з текстового редактора, а й програмно:

```
  program txt;
    var f:    text;
        str, t: string;
    begin
      writeln ('Задайте ім"я файла:');
      readln (str);
      Assign (f, str);
      Rewrite (f);
      writeln ('Введіть текст');
      readln (t);
      write (f, t);
      Close (f)
    end.
```

Наведений приклад має лише одну особливість - текст, який можна записати у файл, не перевищуватиме 255 символів, оскільки він записується у файл через рядкову змінну. Якщо вас це не влаштовує, то доведеться вводити інформацію з клавіатури і записувати у файл посимвольно або кількома рядковими змінними.

Особливості роботи з типізованими файлами

Типізовані файли носять саме таку назву, бо відразу обумовлюється, з елементів якого типу вони складатимуться. Файл розглядається як послідовність елементів указаного типу. За одну операцію введення або виведення обробляється лише один елемент указаного файла. Для цього під час відкриття файла створюється спеціальний *покажчик файла,* який указує на нульовий елемент (нумерація елементів файла починається з нуля). Після операції введення або виведення покажчик файла буде показувати перший елемент. І так далі.

Схематично цей процес можна зобразити так:

0	1	2	$N-1$
елемент 1	елемент 2	елемент 3	елемент N

Суттєва відмінність типізованих файлів від текстових (навіть тих, що містять числову інформацію) полягає в тому, що *типізовані файли є файлами прямого доступу.* Тобто програма знає кількість елементів у типізованому файлі і, відповідно, має доступ у будь-який момент до будь-якого з них.

Зверніть увагу на те, що процедури **readln** і **writeln** для типізованих файлів просто не мають змісту. Елементи в цих файлах нічим не відділяються один від одного. В них просто не існує такого поняття, як кінець рядка.

Дещо специфічно працюють для типізованих файлів процедури **Reset** і **Rewrite.** Вони обидві відкривають файли як для читання, так і для запису, встановлюючи покажчик файла на самий початок, але **Rewrite** відкриває новий файл або знищує вже існуючий, а **Reset** працює лише для існуючих файлів, зберігаючи в них інформацію.

Розглянемо процедури та функції, специфічні для роботи з типізованими файлами.

1. Процедура, яка дає змогу встановлювати покажчик файла на потрібний елемент.

Seek (<файлова змінна>, <номер елемента>).

Наприклад, фрагмент програми

```
for i := 1 to n do
  begin
    Seek (f, n - i);
      read (f, a);
      writeln (a);
  end;
```

дає змогу переглянути всі елементи файла в зворотному порядку.

2. Функція, яка повертає кількість елементів у вказаному типізованому файлі, виглядає так:

FileSize (<файлова змінна>): **integer.**

Тип, указаний у кінці опису функції, означає, що повернутий результат буде цілою величиною.

Наприклад: N := **FileSize** (f).

Тепер розглянемо конкретну програму, що працює з типізованим файлом. Нехай треба обчислити суму всіх цілих чисел, які записані на диску у файлі ту_ file.dat.

```
program number;
  var f:      file of integer;
      S, a, i: integer;
  begin
    Assign (f, 'my_file.dat');
    Reset (f);
    S := 0;
    for i := 1 to FileSize (f) do
```

```
        begin
          read (f, a);
          S := S + a;
        end;
      Close (f);
      writeln ('Сума елементів файла: ', S)
    end.
```

Не слід забувати, що перш за все дані у файл треба якимось чином занести. Для типізованих файлів це можна зробити лише програмно приблизно так:

```
program remember;
  var f:   file of integer;
      a, i: integer;
begin
  Assign (f, 'my_file.dat');
  Rewrite (f);
  writeln ('Задайте кількість елементів файла: ');
  read (n);
  for i := 1 to n do
    begin
      readln (a);
      write (f, a);
    end;
  Close (f);
end.
```

Переглядати файли можна безпосередньо в інтегрованому середовищі, відкриваючи їх за допомогою опції Open (F3). Проте якщо ви відкриєте таким чином типізований файл, навіть з цілими числами, то ніяких чисел там не побачите, бо Pascal усю цю інформацію кодує. Якщо ж цс буде текстовий файл, то ви побачите всю інформацію в звичному вигляді і зможете її коригувати безпосередньо в інтегрованому середовищі, не забуваючи при цьому натискати клавішу F2 для збереження її на диску. Тому зручно для наочності записувати числову інформацію в текстові файли. Хоча з точки зору швидкості обробки інформації це невигідно - постійно відбувається перетворення даних із текстового вигляду в числовий і навпаки.

Отже, порівняємо роботу з типізованими і текстовими файлами.

У типізованому файлі відома кількість його елементів, є можливість прямого доступу до будь-якого елемента. Але не можна переглядати вміст типізованого файла звичайними засобами. Запис і читання інформації в типізованому файлі можна здійснювати лише програмно.

У текстових файлах інформація відкрита. Тобто записувати інформацію в текстовий файл і переглядати її там можна в звичайному текстовому редакторі. Проте визначити, скільки чи-

сел або символів містить текстовий файл, перейти до необхідного елемента у текстовому файлі можна, лише переглядаючи його послідовно.

Запитання для самоконтролю

1. Поясніть відмінність між логічними і фізичними файлами.
2. Запишіть загальний вигляд опису змінної типу «файл».
3. Які види логічних файлів існують у Pascal?
4. У чому відмінність між типізованими й нетипізованими файлами?
5. Назвіть стандартні процедури для роботи з файлами.
6. Поясність, у чому відмінність роботи з числовою і символьною інформацією в текстових файлах.
7. Назвіть процедури і функції для роботи з текстовими файлами.
8. У чому полягає особливість роботи з типізованими файлами?
9. Як відбувається читання та запис у типізованих файлах?
10. Що таке покажчик файла? Яке його призначення?
11. Назвіть процедури і функції для роботи з текстовими файлами.
12. У якому вигляді зберігається числова інформація в типізованих і текстових файлах?

Не припускайтеся помилок!

Якщо ви використовуєте процедуру **Reset** для відкриття файла, якого ще не існує, вам буде видане повідомлення про помилку:

Error 15: File not found.

Якщо ви намагаєтеся записати дані у файл, відкритий процедурою **Reset,** то також буде видане попереднє повідомлення про помилку.

Якщо ви намагаєтеся створити файл на переповненому диску, то на це вкаже повідомлення про помилку:

Error 16: Disk full.

Якщо ви намагаєтеся прочитати значення з типізованого файла в змінну іншого типу, то буде видано повідомлення про помилку щодо незбігання типів **(Error 26).**

Якщо було створено файл типу **real,** а ви обробляєте його як **integer** або навпаки, то в результаті буде прочитано із файла зовсім не ті за значеннями числа.

Якщо, працюючи у програмі з файлами, ви отримали повідомлення про помилку

Error 13: Too many open files,

це означає, що кількість файлів перевищує ту кількість, яка дозволена операційною системою. Внесіть відповідні зміни у файл CONFIG.SYS.

Виконайте завдання

Вправи

565. Дайте відповіді на такі запитання.

1) Чи можуть бути елементи файла різнотипними величинами?

2) Який максимальний номер елемента типізованого файла, який складається з 50 елементів?
3) Як описати файл, якщо в ньому треба розмістити однакові групи елементів з величин різного типу?
4) Чи може у файлі існувати елемент з від'ємним порядковим номером?
5) Чи можуть два елементи одного і того самого файла мати однаковий порядковий номер?
6) Чи можна на місце одного елемента файла записати інший, але такого самого типу? А іншого типу?
7) Чи можна на місце одного елемента файла записати два такі самі елементи?
8) Чи можна після зчитування елемента із файла зразу ж зчитати наступний після нього елемент? А розташований через дві позиції за ним? А попередній елемент?
9) Чи можна одночасно читати з файла і записувати в нього інформацію?
10) В яке місце файла можна дописувати інформацію: на початок; у середину; в кінець?

566. Нехай файл з цілочисловими елементами містить два числа - 1 і 1. Визначити значення змінної *y* після виконання таких послідовностей операторів:

1)
```
reset (f);
if not eof (f) then read (f, y);
if not eof (f) then read (f, y);
```
2)
```
reset (f); y := 0;
while not eof (f) do
   begin
      read (f, x);
      y := y + x
   end;
```
3)
```
reset(f); y := 1;
repeat
   read (f, x);
   y := y*x;
until eof (f).
```

567. Визначити вміст цілочислового файла *f* після виконання таких дій:

1)
```
rewrite (f);
   if eof (f) then write (f, 1) else write (f, 2);
   if eof (f) then write (f, 3) else write (f, 4);
```
2)
```
rewrite (f);
   for i := 3 downto 1 do write (f, sqr(i)).
```

Задачі

568. Дано файл, який містить цілі числа. Визначити:
1) кількість парних елементів;
2) суму додатних елементів;
3) середнє арифметичне найменшого і найбільшого елементів;
4) добуток квадратів елементів;
5) суму модулів елементів з непарними номерами.

569. Дано файл із дійсними числами. Визначити:
1) найбільший і найменший елементи;
2) найменший елемент серед елементів з парними номерами;
3) різницю між першим і найбільшим елементами;
4) яких елементів більше - від'ємних чи додатних.

570. Дано файл, елементами якого є цілі числа. Обчислити:
1) кількість подвоєних непарних чисел серед елементів;
2) кількість елементів, значення яких збігаються з їх номерами;
3) кількість елементів, значення модулів яких є квадратами їх номерів;
4) кількість парних елементів, які мають парні номери.

571. Записати у файл результати обчислення за формулою $f(x) = X^{i}$ sin x - 5x для x = 1, 2, 30.

572. Нехай для x = 0, 1, 15 обчислення проводиться за формулою:

$$y = x\cos x + \frac{2\,|\sin x\,|}{x!}.$$

Усі додатні результати записати у файл f, а всі від'ємні — у файл g.

573. Таблицю значень y = sin x для x = - 10, −9.9, 10 записати у файл відповідними парами чисел (x; y).

574. Дано файл f, компонентами якого є дійсні числа. Отримати копію файла у файлі g.

575. Послідовність значень обчислюється за формулою:

$$\frac{}{i^2 + i + 6}, \text{ де } i = 1, 2, 3, \ldots .$$

У файл g записати ті члени послідовності, які передують першому члену, значення якого менше за є.

576. Дано файл f, компонентами якого є цілі числа. Переписати у файл g ті з них, які:
1) є парними числами;
2) менші за задане ціле число;
3) діляться на 3 і не діляться на 7;
4) мають парні номери;
5) є квадратами непарних чисел.

577. Дано два файли g_1 та g_2, компонентами яких є дійсні числа. Переписати із збереженням порядку компоненти файла g_1 у файл g_2 і навпаки - із g_2 у g_1, скориставшись для цього третім файлом g_3.

578. Дано файл f із цілих елементів. Переписати парні числа у файл g, а непарні – у файл А, зберігаючи при цьому їх заданий порядок.

579. Дано файл f, що складається з дійсних елементів. Переписати їх у файл g у зворотному порядку.

580. Дано файл f, що складається з дійсних елементів. Створити файл g, що містить елементи файла f без повторних входжень.

581. Дано два файли f і g, що містять цілі елементи. Створити файл А, переписавши в нього:

1) спочатку всі елементи файла f, а потім файла g;

2) чергуючи елементи файла f і g.

582. Дано файл f, який містить лише дійсні числа, відмінні від 0. Переписати у файл g лише додатні числа, а у файл А - від'ємні числа із файла f, зберігаючи при цьому їх порядок у файлі f.

583. До умови задачі № 582 додається обмеження - кількість додатних і від'ємних чисел у заданому файлі однакова. Переписати числа із файла f у файл g, чергуючи додатні й від'ємні числа.

584. Дано файл f, який містить лише цілі числа, відмінні від 0. Відомо, що числа в заданому файлі чергуються так: десять додатних, десять від'ємних, десять додатних, десять від'ємних і т. д. Переписати числа із файла f у файл g, змінивши чергування чисел:

1) п'ять додатних, п'ять від'ємних і т. д.;

2) двадцять додатних, двадцять від'ємних і т. д.

585. Дано файл f, який містить цілі числа. Кількість елементів файла кратна натуральному числу n. Для кожної групи з n чисел із файла f, зберігаючи їх порядок, записати у файл g:

1) мінімальні значення;

2) середні арифметичні значення;

3) півсуму найменшого і найбільшого чисел.

586. Припустивши в умові задачі № 585, що кількість елементів файла f довільна, виконати завдання:

1) записати у файл g максимальні значення з кожної групи n елементів заданого файла, а решту елементів файла f (якщо він менший за n) переписати у файл А;

2) записати у файл g впорядковані групи з n елементів даного файла, зберігаючи їх порядок, а решту елементів файла f (якщо він менший за n) дописати у цей файл без змін.

587. Дано два однотипні файли f і g. Визначити:

1) чи рівні ці файли, тобто чи збігаються послідовності їх елементів;

2) чи є один із цих файлів доповненням другого, тобто чи є один із них першою частиною іншого.

588. Дано файл f, який складається з цілочислових елементів. Змінити в ньому послідовність елементів, переписавши останній елемент на перше місце, передостанній - на друге і т. д., не використовуючи для цього додаткового файла.

589. Дано два символьні файли f і g. Злити вміст цих файлів у файл h, розташувавши спочатку інформацію з файла f, а потім з файла g.

590. Виконати № 574–579 за умови, що вказані файли є символьними.

591. Закодувати текст, що вводиться з клавіатури, записавши у символьний файл відповідні ASCII-коди цих символів.

592. У символьному файлі f розміщено цілі числа. Знайти їх середнє арифметичне значення.

593. Дано символьний файл, який містить десяткові цифри, розділені пробілами. У новий символьний файл записати відповідні текстові еквіваленти цих цифр (нуль, один, два і т. д.), розділяючи їх пробілами та зберігаючи при цьому їх послідовність.

594. У символьному файлі розміщені дійсні числа, що містять таку інформацію: перше число - кількість елементів послідовності, всі наступні - самі елементи послідовності. Вивести на екран монітора елементи послідовності, що містяться у файлі, і визначити кількість від'ємних елементів.

595. У символьному файлі розміщено інформацію про цілочислову матрицю таким чином: перший рядок файла - кількість рядків та стовпчиків матриці, наступні рядки - елементи матриці, задані по рядках. Вивести на екран монітора елементи матриці, розташувавши їх у вигляді таблиці (вважається, що розмірність матриці дозволяє вивести її на екран).

596. Дано натуральні числа n і m. Елементи прямокутної таблиці розміром $n \times m$ обчислюються за формулою $\frac{i+j}{nm}$. Отримані значення прямокутної таблиці записати по рядках у текстовий файл. На початку цього файла окремим рядком повинні бути записані значення чисел n і m.

597. Дано послідовність з n цілих чисел. Записати у символьний файл дану послідовність чисел та її упорядкований за зростанням варіант з відповідними текстовими коментарями: «дана послідовність:» та «упорядкована послідовність:».

598. Дано символьний файл, в якому міститься інформація про співробітників деякого підприємства за схемою:

прізвище␣ім'я␣по батькові, прізвище ім'я␣по батькові, ...

Перенести ці відомості в інший файл за такими зразками:

1) ім'я ␣ прізвище, ім'я ␣ прізвище, ...;

2) ім'я ␣ по батькові ␣ прізвище, ...;

3) прізвище ␣ і. п/б., прізвище ␣ і. п/б., ...

599. У типізованому файлі міститься інформація про учнів даної школи у такому вигляді: прізвище ␣ ім'я ␣ клас (рік навчання та літера).

Визначити:

1) учнів, які мають однакові прізвища;

2) учнів, які вчаться у даному класі;

3) кількість учнів, які носять вказане ім'я і навчаються в даному класі;

4) учнів школи, в яких однакові прізвища й імена;

5) учнів даного класу, в яких однакові прізвища;

6) кількість учнів у даному класі;

7) класи, в яких навчається найбільша кількість учнів;

8) кількість класів у кожній з паралелей (наприклад, кількість 1-х класів, 2-х класів і т. д.).

600. Використовуючи умову задачі № 599, переписати інформацію про учнів старших класів (9–11) із заданого файла:

1) у другий файл у такій послідовності: 9-А, 9-Б, 10-А, 10-Б, 11-А, 11-Б, ...;

2) переписати інформацію, зазначену у п. 1), в окремі файли по кожному класу.

601. У файлі *f* містяться дані, зазначені в двох попередніх задачах, і додатково бали, отримані учнями за останній семестр:

1) визначити кількість учнів, які вчаться на «7–9» та «10-12»;

2) середній бал учнів кожного класу;

3) записати у файл *g* інформацію про учнів, які вчаться на «10-12»;

4) записати в один файл інформацію про учнів, які мають усі бали не нижче як «7–9», а в другий - інформацію про учнів, які мають хоча б один з балів «4–6».

602. У файлі *f* міститься база даних про книжковий фонд шкільної бібліотеки в такому вигляді:

прізвище автора ␣ назва книжки ␣ рік видання.

Визначити:

1) наявність книжки за вказаними даними;

2) усі назви книжок указаного автора;

3) усі книжки за вказаним роком видання;

4) усі книжки, в назві яких присутнє дане слово.

Якщо ви зразу розгорнули книжку на цій сторінці, вас можна зрозуміти, проте одночасно слід застерегти. Цей розділ навмисне винесено на кінець посібника. Адже без хороших алгоритмічних навичок використання графічних можливостей буде неповним. Отже, почніть знайомство з посібником спочатку і скористайтеся всіма корисними порадами, які містяться в ньому.

Основні процедури і функції для роботи з графічними образами. Модуль GRAPH

Усі процедури і функції для роботи в графічному режимі містяться в модулі **Graph.** Тому його необхідно підключати на початку роботи з програмою.

Перш ніж розпочати роботу в графічному режимі, треба налаштувати програму на цей режим. Для цього необхідно виконати таку *процедуру ініціалізації графіки:*

InitGraph(GraphDriver, GraphMode, GraphPath),

де **GraphDriver** - ціле число, що визначає тип монітора, **GraphMode** - ціле число, що визначає режим його роботи, **GraphPath** - рядковий вираз, який визначає шлях до файла-драйвера з розширенням ***.bgi.**

Розглянемо детальніше призначення параметрів, які використовуються при ініціалізації графіки.

Параметр **GraphDriver** задається назвою або кодом і означатиме тип графічного адаптера. Наприклад, у вас установлено монітор типу VGA, але ви хочете працювати в режимі CGA – це можливо. Значення параметра **GraphDriver** можна взяти з такої таблиці:

Ім'я	Значення
Detect	0 (Автоматичний вибір драйвера)
CGA	1
MCGA	2
EGA	3
EGA64	4
EGAMono	5
IBM8514	6
HercMono	7
ATT400	8
VGA	9
PC3270	10
CurrentDriver	- 128 (Працює зі встановленим на даний момент драйвером)

Параметр **GraphMode** визначає, в якому режимі роздільної здатності працюватиме графічний адаптер. Його також можна задавати як назвою, так і кодом. Можливі режими лише найпоширеніших типів адаптерів наведено у таблиці:

Ім'я	Значення	Розмір поля	Палітра
CGAC0	0	320x200	co
CGAC1	1	320x200	C1
CGAC2	2	320x200	C2
CGAC3	3	320x200	C3
CGAHi	4	640x200	2 кольори
EGALo	0	640x200	16 кольорів
EGAHi	1	640x350	16кольорів
EGA64LO	0	640x200	16 кольорів
EGA64Hi	1	640x350	4 кольори
EGAMonoHi	0	640x350	2 кольори
VGALo	0	640x200	16 кольорів
VGAMed	1	640x350	16 кольорів
VGAHi	2	640x480	16 кольорів

де С0 - світло-зелений, рожевий, жовтий; C1 - світло-голубий, світло-фіолетовий, білий; С2 - зелений, червоний, коричневий; С3 - голубий, фіолетовий, світло-сірий.

Параметр **GraphPath** - це рядкова величина, яка визначає, де знаходиться драйвер графічного введення/виведення інформації. Для різних типів адаптерів існують різні драйвери. Наприклад, для моніторів EGA та VGA - egavga.bgi, для монітора CGA - cga.bgi. Якщо цей файл міститься в тому ж самому підкаталозі, що й exe-файл програми, то шлях не вказується (''),якщо ж в іншому місці, то його необхідно вказати (наприклад, 'c:\BGI').

Надалі використовуватимемо такий вигляд процедури ініціалізації графіки:

```
var GraphDriver, GraphMode, GraphPath: integer;

GraphDriver :=VGA;
GraphMode := VGAHi;
InitGraph(GraphDriver, GraphMode, GraphPath);
```

Процедура *закриття графіки*

CloseGraph

дає змогу програмі повернутися назад у текстовий режим. Бажано кожну програму, в якій передбачається робота з графікою, завершувати цією процедурою.

Тепер розглянемо основні процедури та функції модуля **Graph.**

1. Процедура очищення екрана монітора в графічному режимі:

ClearDevice.

2. Процедура, що виводить зображення пікселя заданого кольору:

PutPixel(x, y: **integer;** Color: **Word),**

де x, y - координати точки, Color - колір точки. Значення кольорів залишаються такими самими, що й у текстовому режимі.

3. Процедура виведення лінії:

Line(x1, y1, **x2, y2: integer),**

де **(x1; y1)** - координати початку, **(x2; y2)** - координати кінця лінії. Для задання кольору лінії необхідно скористатися процедурою **SetColor,** яка буде описана пізніше.

4. Для виведення зображення кола треба скористатися процедурою:

Circle(x, y: **integer;** Radius: **word),**

де (x; y) - координати центра кола, Radius – розмір радіуса цього кола. Колір кола задається так само, як і колір лінії.

5. Процедура

Ellipse(x, y: **integer;** StartAngle, EndAngle **,** XRadius, YRadius: **word)**

дає змогу вивести на екран монітора дугу еліпса від кута StartAngle до кута EndAngle. Центр еліпса міститься у точці (x; y), радіуси по горизонталі та вертикалі відповідно XRadius, YRadius.

6. Побудувати прямокутник можна за допомогою процедури:

Rectangle(x1, y1, **x2, y2: integer),**

де (x1; y1) - координати лівого верхнього кута прямокутника, **(x2; y2)** - координати правого нижнього його кута.

7. Щоб задати колір фігур для описаних вище процедур, необхідно скористатися процедурою:

SetColor(color: **word),**

де параметр color визначає потрібний колір ліній фігури.

8. Колір фону малюнка можна задати за допомогою процедури:

SetBkColor(color: **word).**

9. Колір зафарбовування (заливки) задає процедура:

SetFillStyle(Pattern: **word;** Color: **word),**

де Pattern визначає вид шаблону заливки, а Color - його колір. Модуль **Graph** пропонує великий вибір значень параметра Pattern, зазначених у таблиці **7.**

Таблиця 7

Ім'я	Значення	
EmptyFill	0	суцільна заливка кольором фону
SolidFill	1	суцільна заливка поточним кольором
LineFill	2	заливка типу = = =
LtSlashFill	3	заливка типу ///
SlashFill	4	заливка жирними лініями типу ///
BkSlashFill	5	заливка жирними лініями типу \\\
LtBkSlashFill	6	заливка типу \\\
HatchFill	7	заливка рідкою штриховкою
XhatchFill	8	заливка густою штриховкою
InterleaveFill	9	заливка переривчастою лінією
WideDotFill	10	заливка рідкими точками
CloseDotFill	11	заливка густими точками
UserFill	12	власна заливка

10. Зображення зафарбованого прямокутника здійснюється за допомогою процедури:

Bar (x1, y1, x2, y2: **integer).**

11. Для малювання стовпчастих діаграм (паралелепіпедів) зручно використовувати процедуру:

Bar3D (x1, y1, x2, y2: **integer;** D3: **word;** Top: **boolean),**

де глибина задається параметром D3, а параметр Top задає режим зображення верхньої площини (**true** — відображати, **false** - ні).

12. Для малювання кругових діаграм треба виділити деякий сектор заданого круга. Для цього можна скористатися процедурою:

PieSlice (x, y: **integer;** StAngle, EndAngl, Radius: **word),**

де (x; y) - координати центра кола, Radius - розмір радіуса цього кола, а сектор задається значеннями двох кутів у градусній мірі: StAngle - початкового кута, EndAngl - кінцевого кута.

13. Для зображення сектора еліпса існує процедура:

Sector (x, y: **integer;** StAngle, EndAngl, XRadius, YRadius: **word),**

де (x; y) - координати центра еліпса, XRadius - розмір радіуса еліпса по горизонталі, YRadius - розмір радіуса еліпса по вертикалі, StAngle - початковий кут сектора, EndAngl - кінцевий кут сектора.

14. Нарешті універсальна процедура

FloodFill (x, y: **integer;** Border: **word)**

заповнює всю область навколо точки (x; y), що обмежена лініями кольору Border. Колір і тип заповнення визначається процедурою **SetFillStyle.**

15. Функція **GraphResult: integer** повертає після ініціалізації графіки ціле число, яке є кодом завершення ініціалізації. Результат «0» означає коректне завершення ініціалізації.

16. Функція **GetMaxX: integer** визначає максимальне значення піксел я по горизонталі у вибраному графічному режимі. При цьому початок відрахунку пікселів (0; 0) знаходиться у верхньому лівому куті.

17. Функція **GetMaxY: integer** визначає максимальне значення пікселя по вертикалі у вибраному графічному режимі.

18. Функції **GetX: integer** та **GetY: integer** визначають поточне положення графічного курсора по горизонталі (X) та вертикалі (У).

Далі розглядатимемо процедури, що стосуються виведення текстової інформації в графічному режимі. Оскільки після ініціалізації графіки будь-які зображення виводяться по пікселях, то це стосується і текстової інформації. Таке виведення тексту відбувається набагато повільніше, ніж у текстовому режимі.

19. Для вибору певного шрифту існує процедура:

SetTextStyle (Font, Direction: **word;** CharSize: **word),**

де Font задає один із типів шрифтів, Direction - горизонтальне (0) чи вертикальне (1) виведення тексту, а CharSize - розмір символів шрифту (від 1 до 10).

Pascal дає змогу працювати зі стандартними шрифтами, перелік яких наведено в таблиці 8.

Таблиця 8

Ім'я	Значення	Вигляд
DefaultFont	0	Матричний шрифт 8x8 (по замовчуванню)
TriplexFont	1	Напівжирний шрифт
SmallFont	2	Світлий шрифт (тонке начерчення)
SansSerifFont	3	Книжна гарнітура (рубаний шрифт)
GothicFont	4	Готичний шрифт

Для роботи зі шрифтами в графічному режимі необхідно, щоб у поточному каталозі були файли відповідних штрихових шрифтів із розширенням *.chr. Шрифт **DefaultFont** встановлюється після ініціалізації графічного режиму автоматично і не потребує файла шрифту. Інші шрифти реалізовані відповідно у файлах trip.chr, small.chr, sans.chr, goth.chr.

20. Перш ніж скористатися в програмі додатковими шрифтами, їх треба зареєструвати. Для цього існує функція:

InstallUserFont(FontName: **string): integer,**

яка повертає числовий код, під яким буде зареєстровано **даний** шрифт. FontName – файл шрифту без розширення *.chr.

21. Безпосереднє виведення тексту можна здійснити за допомогою процедури:

OutTextXY (x, y: **integer**; TextString: **string**),

де (x; y) — координати лівого верхнього кута початку тексту, а TextString - безпосередньо текст або змінна, що його містить.

Спробуємо підібрати приклад програми, яка вмістила б усі описані вище графічні можливості. Під час виконання цієї програми ви побачите на екрані монітора покрокове виведення кіл різного кольору, зафарбовування їх, виведення зображення елемента стовпчастої діаграми, виведення тексту в графічному режимі:

```
program picture;
  uses Graph;
  var
    GraphDriver, GraphMode: integer;
  begin
    GraphDriver:=VGA;
    GraphMode := VGAHi;
    InitGraph (GraphDriver, GraphMode,'');
      SetTextStyle (1, 0, 4);
    OutTextXY (50, 400, 'Перші графічні етюди.');
    SetColor (red);
    Circle (50, 50, 20);
    readln;
    SetColor (green);
    Circle (50, 50, 40);
    readln;
    SetFillStyle (1, blue);
    FloodFill (50, 50, red);
    readln;
    SetFillStyle (11, magenta);
    FloodFill (50, 50, green);
    readln;
    ClearDevice;
    SetFillStyle (4, yellow);
    Bar3D (10, 10, 50, 60, 10, True);
    readln;
    CloseGraph;
  end.
```

Для перевірки отриманої інформації зробіть зміни в цій програмі і виконайте її. Наприклад, поміняйте розміщення і розміри кіл, кольори і тип їх зафарбовування, шрифт тексту, сам текст тощо.

Використання комп'ютера для побудови й дослідження графіків різних функцій доцільне особливо під час моделюван-

ня різних процесів. Спершу це здається простим: достатньо організувати цикл зі зміною значень аргументу, обчислити значення відповідної функції і виводити на екран монітора точки з цими координатами. Однак ми не враховуємо те, що координати точок на екрані монітора мають лише цілі невід'ємні значення, початок координат міститься в лівому верхньому куті, а вісь *Oy* напрямлена вниз. Щоб зображення графіка було максимально наближеним до реального, необхідно оцінити можливі зміни значень як аргументу, так і функції, що будується, і ввести масштабування. Розглянемо приклад побудови графіка функції $|\sin x|$:

```
program graphic;
  uses Graph;
  var
    i, GraphDriver, GraphMode: integer;
  begin
    GraphDriver:=VGA;
    GraphMode := VGAHi;
    InitGraph (GraphDriver, GraphMode,'');
    for i := 1 to 600 do
      PutPixel (i, 200-round(abs(sin(i/30)*50)), 4);
    readln;
    CloseGraph;
  end.
```

Щоб зручніше було розібратися з цією програмою, прокоментуємо деякі коефіцієнти у виразі K-round(abs(sin(i/L)*M)). Для отримання бажаної щільності точок, якими зображається графік функції, треба ввести відповідний коефіцієнт *L*. Амплітуда даної синусоїди залежить від коефіцієнта *M*, а значення *K* дає змогу змістити графік вниз по осі *Oy*.

Так само можна отримати й дослідити графік якоїсь іншої функції, змінюючи відповідні параметри. Наприклад, такі фрагменти програм:

```
for i := 1 to 200 do
  PutPixel (i, 400-round(exp(i/20)*7), 2);
```

та

```
for i := 1 to 600 do
  PutPixel (i, 200-round(ln(i*10)*7), 2);
```

дають змогу експериментувати з графіками функцій exp(x) та ln(x).

Дуже часто доводиться виводити на екран монітора геометричні фігури. Але при цьому слід ураховувати специфіку екранних координат: точка з координатами (0; 0) виводиться на екран монітора у лівому верхньому куті, збільшення по осі *Ox* іде зліва направо, а по осі *Oy* - зверху вниз. Тому виникає не-

обхідність у враховуванні масштабування в разі великих значень параметрів точок на координатній площині, що виходять за межі екранних координат, відсутності від'ємних і дробових значень при виведенні точок на екран монітора. Для прикладу розглянемо програму, що будує багатокутник, заданий координатами своїх вершин у порядку їх обходу:

```
program picture_1;
  uses Graph;
  var
    i, GraphDriver, GraphMode, n:   integer;
    a, b:  array [1..100]                  of real;
    point_a, point_b:  array [1..100] of integer;
    min a, max a, min b, max b:   real;
    kx, ky, k:                    real;
  begin
    writeln ('input n:');
    read (n);
    writeln ('input data:');
    for i := 1 to n do
      read (a[i], b[i]);
      {визначення максимальних значень по осі Ox та по осі Oy}
    max a := abs (a[1]);
    max b := abs (b[1]);
    for i := 2 to n do
      begin
        if abs (a[i]) > max a then max a := abs(a[i]);
        if abs (b[i]) > max b then max b := abs(b[i]);
      end;
    kx := 1; ky := 1;
    {визначення коефіцієнта масштабування}
    if max a > 319 then kx := 319/max a;
    if max b > 239 then ky := 239/max b;
    if kx > ky then k := ky else k := kx;
    {обчислення значень координат точок з урахуванням }
    {коефіцієнта масштабування}
    for i := 1 to n do
      begin
        point_a[i] :=round(a[i]*k);
        point_b[i] :=round(b[i]*k);
      end;
      {переміщення центра координат у центр екрана монітора}
    for i := 1 to n do
      begin
        point_a[i] := point_a[i]+320;
        point_b[i] := 240–point_b[i];
      end;
    {виведення зображення багатокутника на екран монітора }
    {в графічному режимі}
    GraphDriver :=VGA;
```

```
    GraphMode := VGAHi;
    InitGraph (GraphDriver, GraphMode,'');
    Line (320, 0, 320, 479);
    Line (0, 240, 639, 240);
    SetColor (2);
    for i := 1 to n-1 do
      Line (point_a[i], point_b[i], point_a[i+1], point_b[i+1]);
    Line (point_a[n], point_b[n], point_a[1], point_b[1]);
    readln;
    CloseGraph;
end.
```

Запитання для самоконтролю

1. У чому полягає особливість програмування з використанням графіки?
2. Що таке ініціалізація графічного режиму? Яким чином відбувається налагодження програми на роботу з необхідним типом монітора в певному режимі?
3. Які процедури модуля **Graph** дають змогу виводити на екран монітора прості геометричні фігури?
4. Які процедури модуля **Graph** дають змогу працювати з кольорами? Які їх особливості?
5. Які процедури і функції модуля **Graph** реалізують роботу з текстами?

Не припускайтеся помилок!

Якщо при звертанні до процедур або функцій модуля **Graph** ви отримаєте повідомлення про помилку:

Error 3: Unknown identifier,

то це означає, що ви забули підключити модуль **Graph.**

Якщо компілятор на назву модуля **Graph** у розділі **Uses** реагує виведенням повідомлення про помилку:

Error 15: File not found (GRAPH.TPU),

то це означає, що компілятор не може знайти файл graph.tpu.

Якщо в поточному підкаталозі відсутній файл із розширенням *.bgi, то ви отримаєте повідомлення про помилку на кроці виконання програми:

BGI Error: Graphics not initialized (use InitGraph).

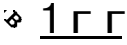
Якщо програма виконує виведення графічних образів, але їх не видно на екрані, то це може відповідати таким помилковим ситуаціям:

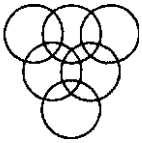
- колір графічного образу збігається з кольором фону;
- координати точок виходять за межі екрана.

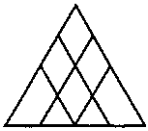
Задачі

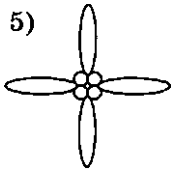
603. Вивести на екран монітора такі малюнки, створивши для цього необхідні процедури або функції:

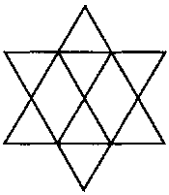
n разів











7)

8)

604. Скласти програму, яка при натисненні клавіші д (день) малює Сонце, а при натисненні клавіші н (ніч) - Місяць.

605. Розробити меню-орієнтовану програму, яка б за вибором користувача зображала на екрані:

1) «т» - трикутник;
2) «п» - прямокутник;
3) «к» - коло.

606. Зобразити на екрані три кола. При натисненні клавіші «ч» верхнє коло зафарбовується червоним кольором, при натисненні клавіші «**ж**» середнє коло зафарбовується жовтим кольором, а при натисненні клавіші «з» нижнє коло зафарбовується зеленим кольором.

607. «Зоряне небо». Заповнити екран монітора різнокольоровими точками, кількість яких, колір та координати визначаються випадковим чином.

608. Виконати завдання № 607 за умови, що точки замінюються лініями.

609. Зобразити на екрані дитячу пірамідку з n різнокольорових прямокутників, розміри яких крок за кроком зменшуються на 10 %.

610. Зобразити сімейство квадратів зі спільним центром, який збігається з центром екрана, і з довжинами сторін, що відрізняються на 10 пікселів.

611. Вивести на екран монітора веселку, яка опирається на нижню межу екрана.

612. Скласти програму руху по екрану зліва направо вертикальної лінії із заданими координатами. Імітація руху здійснюється повторним малюванням лінії на попередньому кроці кольором фону екрана і побудовою її на новому місці.

613. Зобразити на екрані хатинку з вікном і дверима, в якій при натисненні клавіші «пробіл» запалюється і гасне світло у вікні.

614. Зобразити на екрані монітора декартову систему координат на площині, початок якої збігається з центром екрана.

615. Зобразити на екрані монітора просторову декартову систему координат, початок якої збігається з центром екрана.

616. Стовпчаста діаграма (гістограма) - це послідовність прямокутників однакової ширини і висоти a_i, розташованих на одному горизонтальному рівні, де a_i - члени деякої послідовності. Для більшої наочності стовпчики зображають різними кольорами.

Побудувати стовпчасту діаграму за даними n цілими числами.

617. Виконати завдання № 616 для послідовності дійсних чисел.

618. Секторною діаграмою називають круг, площі секторів якого пропорційні відповідним числовим величинам, узятим з деякої послідовності.

Для заданої послідовності з n дійсних чисел побудувати секторну діаграму. Для більшої наочності сектори діаграми зафарбувати різними кольорами.

619. Використовуючи масштабування і особливості екранних координат, створити програму виведення графічних примітивів:

1) лінії; 2) кола; 3) прямокутника; 4) паралелепіпеда.

620. Зобразити на екрані монітора з урахуванням масштабування і особливостей екранних координат графік функції:

1) $\sin x$; 2) $|\sin x|$; 3) $\sin x + \cos x$.

621. N-кутник заданий координатами своїх вершин $(x_1; y_1)$, $(x_2; y_2), (x_n; y_n)$ в порядку обходу за годинниковою стрілкою. Враховуючи масштабування і особливості екранних координат, зобразити його на екрані монітора.

622. Зобразити на екрані сітку з квадратів розміром $n \times m$ та розташувати в центрі кожного квадрата:

1) випадкове двозначне ціле число;

2) числа від 1 до *n*m,* обчислені випадковим чином без повторень.

623. Розробити програму для введення тексту з клавіатури у графічному режимі та реалізувати такі можливості його редагування:

1) переміщення курсора по тексту;

2) знищення символу в позиції курсора;

3) знищення символу перед курсором;

4) вставлення символу перед курсором.

624. **«Кресляр».** За допомогою клавіш керування $\uparrow, \downarrow, \leftarrow, \rightarrow$ змоделювати рух точки по екрану в заданому напрямі, яка залишала б за собою слід.

625. Розробити власну програму малювання кола з центром у точці (а; ***Ь)*** і радіусом ***R.***

626. Скласти програму, яка виводитиме на екран монітора графіки траєкторій тіла, кинутого під кутом до горизонту в проміжку від 10° до 70° через кожні 20° з початковими швидкостями 20 м/с та 30 м/с, скориставшись для кращої порівняльної характеристики різними кольорами. Опором повітря знехтувати.

Зауваження. Для побудови графіка слід скористатися формулою:

$$\frac{v_{0y}}{v_{0x}} \quad \frac{g}{2v_{0x}^2}$$

де $v_x = v_{0x} = v_0 \cos a$, $v_y = v_{0y} = v_0 \sin a$.

627. За даними додатними значеннями а, *L* та *H* визначити початкову швидкість v_0, з якою необхідно кинути предмет з точки (0; 0) у точку *(L; H)* та зобразити траєкторію польоту на екрані монітора. Опором повітря знехтувати.

Зауваження. Скористатися формулами:

$$x = v_0 t \cos\alpha;\ y = y_0 t \sin\alpha - \frac{gt^2}{2},$$

звідки рівняння траєкторії визначатиметься так:

$$y = \frac{\sin\alpha}{\cos\alpha} x - \frac{g}{2v_0^2 \cos^2 a} x^2.$$

Формула для визначення початкової швидкості матиме вигляд $v_0 = \sqrt{2(L\operatorname{tg}\alpha - H)\cos^2\alpha}$.

628. «Баскетбол». Скориставшись формулами й умовами попередньої задачі, скласти програму, яка у діалоговому ре-

жим1 за даними значеннями кута а та початкової швидкості v_0 будувала б траєкторію польоту м'яча, кинутого з умовної точки (0; 0) у кільце, що знаходиться на відстані L та на висоті H від місця кидання. У програмі потрібно передбачити вдалі та невдалі кидки, накладаючи графіки всіх кидків один на один і зображуючи їх різними кольорами. Завершенням програми вважається вдалий кидок або натиснення клавіші, зазначеної користувачем.

Функції і процедури для організації рухомих графічних об'єктів

У Pascal передбачена можливість створення рухомих графічних об'єктів. Розглянемо групу процедур, які використовують для реалізації анімації. Спочатку ознайомимося з динамікою виконання руху графічних об'єктів.

Нехай на екрані зображено зафарбований круг. Тепер це зображення треба змістити. Це можна зробити трьома способами.

Варіант 1. Очистити екран, знищивши зображення, і вивести це зображення на новому місці. Недолік цього методу полягає в тому, що спостерігається миготіння всього екрана.

Варіант 2. Намалювати зображення ще раз, але кольором фону, а після цього перемалювати його на новому місці. Ефект такого руху буде вже трохи кращий, але буде відчуватися миготіння самого малюнка.

Варіант 3. Запам'ятати область екрана, де буде виведено малюнок; вивести малюнок; відновити на цьому місці область екрана, що була до виведення малюнка; зберегти наступну область екрана, де буде виведений малюнок; вивести на нове місце малюнок і т. д.

Підведемо підсумок: третій варіант найкращий. Але для його організації потрібно десь запам'ятовувати інформацію фрагментів екрана. Ця інформація запам'ятовується в динамічній пам'яті.

1. Першою розглянемо функцію, яка визначає розмір прямокутної області екрана в байтах:

ImageSize (x1, y1, x2, y2: **integer**): **word**,

де $(x_1; y_1)$ - координати лівого верхнього кута прямокутної області, $(x_2; y_2)$ - координати правого нижнього кута цієї області. Результатом виконання цієї функції буде ціле число.

2. Процедура, що зберігає образ вказаної прямокутної області екрана в динамічній пам'яті за вказаною адресою:

GetImage (x1, y1, x2, y2: **integer**; **var** BitMap),

де $(x_1; y_1)$ - координати лівого верхнього кута прямокутної області, $(x_2; y_2)$ – координати правого нижнього кута цієї

Зрозуміло, що ці параметри повинні бути такими ж самими, що й у функції **ImageSize**. ***BitMap*** - безтипова змінна, яка є буфером для зберігання інформації. Трохи пізніше все це буде пояснено на конкретному прикладі.

3. Процедура, що відновлює збережену в динамічній пам'яті прямокутну область:

PutImage (x, y: **integer; var** BitMap, Mode: **word**),

де ***(x; y)*** - координати верхнього лівого кута області екрана, куди буде поміщене зображення, ***BitMap*** - буфер, в якому зберігається образ відновлювальної прямокутної області, ***Mode*** - режим накладання зображення на екран. Можливі значення параметра ***Mode*** задаються таблицею 9.

Таблиця 9

Ім'я	Значення	Дія
CopyPut	0	Операція MOV (заміщення)
XORCopy	1	Операція XOR
ORPut	2	Операція OR
ANDPut	3	Операція AND
NOTPut	4	Операція NOT

Щоб розібратися з цими режимами, треба пригадати, що зображення на екрані монітора створюється за допомогою пікселів, які можуть набувати значень 1 або 0. Тобто зображення в точці екрана, що відповідає даному пікселю, є або його немає. Така сама ситуація і з фрагментом зображення, який збережений у буфері. При накладанні цих двох зображень виконуються логічні операції **AND, OR, NOT** над значеннями 1 та 0. Операція **MOV** заміщує новими значеннями старі, при цьому старе зображення на екрані «накривається» новим фрагментом. Цікавішою є операція **XOR**. Виконавши двічі процедуру **PutImage** з один і тим самим фрагментом, ми тим самим відновимо зображення на екрані.

Нехай треба зобразити на екрані рух круга з лівого верхнього кута у правий нижній. Це може виконати такий варіант програми:

```
program picture;
  uses Graph, Crt;
  var
    GraphDriver, GraphMode: integer;
    P:                      pointer;
    Size, i:                integer;
```

```
begin
  GraphDriver := VGA;
  GraphMode := VGAHi;
  InitGraph (GraphDriver, GraphMode,'');
  SetColo (red);
  Circle (50, 50, 20); {коло з центром (50; 50) та радіусом 20}
  SetFillStyle (1, blue);
  FloodFill (50, 50, red); {одержання круга червоного кольору}
  Size := ImageSize (30, 30, 70, 70);
    {квадрат, описаний навколо круга}
  {відведення динамічної пам'яті розміром Size байтів}
  GetMem (P, Size);
  GetImage (30, 30, 70, 70, P^);
    {збереження фрагмента зображення}
  PutImage (30, 30, P^, 1); {знищення фрагмента на екрані}
  for i := 1 to 250 do
    begin
    {виведення фрагмента на екрані на новому місці}
      PutImage (30+2*i, 30+2*i, P^,1);
      {виведення фрагмента на екрані для його знищення }
      {в режимі XORcopy}
      PutImage (30+2*i, 30+2*i, P^, 1);
    end;
  repeat until KeyPressed;
  CloseGraph;
end.
```

Запитання для самоконтролю

1. Які процедури і функції модуля **Graph** використовуються для створення рухомих графічних об'єктів?
2. Які існують режими використання процедури **PutImage?** Чим вони відрізняються?
3. Запропонуйте можливі варіанти реалізації рухомих об'єктів у графіці. Які процедури і функції модуля **Graph** можна при цьому використати?

Виконайте завдання

Задачі

629. «Годинник». Змоделювати рух годинної та хвилинної стрілок.

630. Проілюструвати на екрані монітора зміну дня і ночі повільним переміщенням спочатку Сонця, а потім Місяця.

631. Розробити програму, яка імітує процес збільшення повітряної кульки під час її надування до певних розмірів, після чого вона лопає.

632. Змоделювати рух поршня під дією газу, що розширюється.

633. Розробити програму-модель коливання маятника на невагомому жорсткому підвісі з урахуванням опору повітря.

634. Створити комп'ютерну модель руху електронів навколо ядра.

635. Розробити програму, яка моделювала б зображення профілю паруса, що надувається зі збільшенням сили вітру (натиснення клавіші →) і опадає при її зменшенні (натиснення клавіші ←).

636. За допомогою клавіш керування **T** та ↓ змоделювати рух ртуті в термометрі відповідно вгору і вниз.

637. **«Постріл».** Змоделювати наведення хрестоподібного прицілу на центр мішені, розташованої в центрі екрана. Рух прицілу припиняється, коли він збігається з центром мішені, і супроводжується звуковим сигналом.

1) Розробити модель, яка керується випадковим чином.
2) Розробити модель, яка керується користувачем за допомогою курсорних клавіш.

638. **«Баскетбол».** Зобразити на екрані відбивання від підлоги (нижня частина екрана) м'яча, що зображується зафарбованим кругом.

1) Для спрощення алгоритму траєкторію руху м'яча вважати ламаною лінією.
2) Розробити максимально наближену до реальності модель відбивання м'яча від підлоги.

639. **«Олівець».** Розробити програму, яка зображає на екрані рух олівця, що вимальовує синусоїду.

640. Процес збільшення новорічної ялинки можна уявити собі таким чином: вже існуюча частина ялинки переміщується вище, а знизу з'являється її новий більший фрагмент. Зобразити цей процес на екрані монітора.

641. Змоделювати плавний рух об'єкта по екрану монітора. Позиція, в яку переміщується об'єкт на кожному наступному кроці, вказується «мишкою». Об'єктом вважати:

1) повітряну кульку;
2) білу хмарку;
3) різнокольорового метелика.

642. «Насадження дерев». Розробити програму, за допомогою якої користувач «мишкою» визначає об'єкт (дерево, кущ, квітку), що знаходиться в нижній частині екрана, і «мишкою» вказує місце на екрані, куди його необхідно перемістити.

643. Рукописний варіант літери «ш» представляється як послідовність ламаних, координати яких занесено у цілочисловий масив. Змоделювати «виписування» цієї літери, читаючи дані з даного масиву.

644. Змоделювати «виписування» ваших ініціалів, якщо дані (див. умову задачі № 643) містяться в текстовому файлі.

645. Розробити модель зіткнення двох куль:

1) однакової маси;

2) різних мас.

МОДУЛІ

Останній розділ цього посібника присвячений створенню власних модулів користувача. На випадок, якщо у вас накопичиться багато власних сервісних програм, якими ви дуже часто користуєтеся, помістіть їх у модуль і користуйтеся ними так само, як ви користуєтеся процедурами і функціями самого Pascal.

Структура модуля

Модуль (unit) можна розглядати як сукупність описів процедур і функцій, які потім можна використовувати в інших програмах. Для використання процедур і функцій деякого модуля в програмі в розділі Uses треба вказати його ім'я. Для того щоб модуль був готовий до використання в програмі, його необхідно скомпілювати. При цьому одержується файл з розширенням ***.tpu** (Turbo Pascal Unit).

У модулі можна виділити чотири частини:

Unit <ім'я модуля> - заголовок модуля;

Interfase - розділ оголошень або інтерфейс;

Implementation - розділ реалізації;

розділ ініціалізації **(begin...end).**

А тепер поговоримо детальніше про останні три розділи і про те, що в них може міститися.

Під терміном «інтерфейс» розуміється деякий зв'язок. Іти-меться про зв'язок даного модуля з програмою, яка буде його використовувати. Підключені в цьому розділі інші модулі, описані в ньому змінні, типи, константи будуть доступні як процедурам і функціям даного модуля, так і програмі, до якої вони підключені. У цьому розділі вміщуються також заголовки всіх процедур і функцій даного модуля. Можна сказати, що описи цього розділу є зовнішніми як для даного модуля, так і для програм, які його використовуватимуть.

Розділ реалізації містить перелік усіх модулів, що підключаються для виконання процедур і функцій даного модуля (вони недоступні зовнішнім програмам), типи, мітки, константи, змінні, якими зможуть користуватися всі процедури і функції даного модуля. Тобто всі описи розділу реалізації є зовнішніми тільки для процедур і функцій даного модуля. І основним моментом є те, що

цей розділ містить тіла процедур і функцій, які включені до даного модуля. Кожна процедура або функція повинна починатися із заголовка. Оскільки заголовки цих процедур і функцій разом із формальними параметрами вже були описані в розділі оголошень для даного модуля, то тут можна обмежитися лише іменами процедур і функцій без повторення списків формальних параметрів. Безпосередні описи процедур і функцій можуть містити свої розділи типів, констант, міток, змінних, процедур і функцій, які будуть доступні лише цим процедурам або функціям. Тобто іншим процедурам і функціям цього модуля, а тим більше програмам, до яких він підключається, вони не доступні.

Останнім у модулі може стояти розділ ініціалізації. Він містить ту частину модуля, яка завжди виконується при підключенні даного модуля до деякої програми. Ця частина модуля ніби приєднується до програми, що викликає цей модуль, і виконується на самому її початку. Найчастіше в цій частині змінні, які використовуються в модулі, заповнюються стартовими значеннями, і відбуваються будь-які одноразові дії, що повинні виконуватися саме на початку програми, яка викликає цей модуль.

Вміст усіх означених розділів у модулі необов'язковий. Може бути також відсутнім і розділ ініціалізації. Наведемо приклад порожнього модуля:

```
Unit empty;
Interface;
Implementation
end.
```

Особливості роботи з модулями

Послідовність підключення модулів до Pascal-програми є дуже суттєвою. Під час компіляції програми модулі підключаються саме в тій послідовності, в якій вони описані в розділі Uses. Тому спочатку треба вказувати системні модулі Pascal у порядку спадання необхідності в програмі, що їх викликає, які включені в бібліотечний файл turbo.tpl (наприклад, **CRT**), а вже потім ті, що знаходяться в tpu-файлах.

Наведемо приклад модуля, який містить лише розділ оголошень з описом типу й константи, які будуть доступні в програмах, що підключатимуть цей модуль:

```
Unit my_unit;
Interface
  type
    my_type = array [1..100] of byte;
  const
    my_const = 100;
Implementation
end.
```

А тепер розглянемо модуль, який міститиме функції пошуку максимального і мінімального з двох значень.

```
Unit optimum;
Interface
  function max (x, y: real): real;
  function min (x, y: real): real;
Implementation
  function max (x, y: real): real;
    begin
      if x > y then max := x
               else max := y
    end;
  function min (x, y: real): real;
    begin
      if x > y then max := x
               else max := y
    end;
  begin
    write ('Ви користуєтеся бібліотечними функціями ');
    writeln ('модуля optimum.tpu')
  end.
```

Запитання для самоконтролю

1. Чим модуль відрізняється від звичайної Pascal-програми?
2. З яких частин складається структура модуля?
3. Яке призначення розділу оголошень? Яким може бути його вміст?
4. Що може містити в собі розділ реалізації?
5. Яке призначення вмісту розділу ініціалізації?
6. Яка принципова відмінність змінних, типів, констант, описаних у розділі оголошень та реалізації?
7. Якою може бути мінімальна структура модуля?
8. Якою має бути послідовність підключення бібліотечних процедур і функцій до програм?

Не припускайтеся помилок!

Оскільки модулі це ті ж самі Pascal-програми, то їм притаманні всі типові помилки, розглянуті в попередніх темах.

Виконайте завдання

Задачі

646. Розробити модуль, в який увійшли б описи власних типів користувача для роботи з «календарними» програмами:

1) назви місяців;
2) назви днів тижня;
3) назви кварталів;
4) назви робочих днів;
5) назви вихідних днів.

647. Розробити модуль, що складається з процедур і функцій для роботи з масивами:

1) введення елементів одновимірного масиву з n елементів;
2) виведення елементів одновимірного масиву з n елементів;
3) введення елементів двовимірного масиву розмірністю $n \times m$;
4) виведення елементів двовимірного масиву розмірністю $n \times m$.

648. Створити модуль, до якого ввійшли б процедури і функції для пошуку найбільших і найменших значень:

1) для двох заданих елементів;
2) для трьох заданих елементів;
3) для послідовності з n елементів;
4) для таблиці розмірністю $n \times m$ елементів.

649. Створити модуль із процедур і функцій для впорядкування:

1) одновимірного масиву за зростанням;
2) одновимірного масиву за спаданням;
3) двовимірного масиву за зростанням (по рядках);
4) двовимірного масиву за зростанням (по стовпчиках);
5) двовимірного масиву за спаданням (по рядках);
6) двовимірного масиву за спаданням (по стовпчиках).

650. Розробити модуль, у який входили б процедури і функції для створення графічних пейзажів з такими елементами:

1) будиночок;
2) ялинка;
3) квіточка;
4) сонечко;
5) паркан.

Усі ці елементи повинні виводитися на екран з указаного місця і з заданим коефіцієнтом збільшення.

651. Створити модуль, в який увійшли б процедури і функції для роботи з «мишкою» в графічному режимі:

1) очікування натиснення будь-якої клавіші «мишки»;
2) керування переміщенням «мишки» по екрану монітора за допомогою курсорних клавіш із заданою швидкістю;
3) супровід натиснення кнопок «мишки» звуковим сигналом;
4) виведення деякого тексту на екран монітора, як реакція на натиснення кнопок «мишки».

652. Розробити модуль для створення власного графічного редактора з такими функціональними можливостями:

1) вибір кольору малювання;
2) малювання лінії;
3) малювання кола із заданим радіусом і місцеположенням центра;

4) малювання довільних контурів;

5) зафарбовування замкненої фігури вказаним кольором.

Зауваження. Використання процедур і функцій модуля Graph не забороняється.

653. Розробити модуль, який містив би процедури і функції для роботи з «вікнами» у графічному режимі (на зразок вікон в інтегрованому середовищі Turbo Pascal):

1) розкриття вікна стандартного розміру, прив'язаного до певного місця екрана;
2) зміна розмірів відкритого вікна;
3) переміщення відкритого вікна по екрану монітора;
4) зміна стану активності відкритих вікон (активним у даний момент часу може бути лише одне вікно);
5) згортання відкритого активного вікна.

654. Базуючись на умовах попередньої задачі, розробити модуль, який містив би процедури і функції для роботи з «вікнами» у текстовому режимі.

655. Удосконалити можливості двох попередніх задач, урахувавши можливість керування вікнами як за допомогою «мишки», так і за допомогою клавіатури.

656. Розробити модуль для роботи із вмістом файла з такими можливостями:

1) виведення вмісту файла на екран монітора;
2) виведення вмісту файла із зазначенням розмірів вікна, в яке він виводиться;
3) вирівнювання тексту по правому і лівому краях при виведенні його у вказане вікно за допомогою додаткових пробілів;
4) організація скролінга (зсуву) тексту у відкритому вікні.

657. Розробити модуль для роботи з графічними «кнопками», проімітувавши їх реальний ефект натиснення за допомогою «гри тіней». До складу модуля включити процедури і функції з такими можливостями:

1) створення «кнопки» заданого розміру, кольору (можна з урахуванням гами відтінків), місцеположення на екрані монітора;
2) підписування кнопки (виведення на неї необхідного тексту);
3) виведення графічної піктограми (малюнка) на створену кнопку;
4) натиснення віртуальної кнопки при натисненні кнопки «мишки»;
5) відпускання віртуальної кнопки при відпусканні кнопки «мишки»;
6) певна реакція на натиснення кнопки (наприклад, відкриття відповідного вікна).

Розділ IV

ЦІКАВА СУМІШ

Цей розділ містить задачі, розв'язання яких потребує застосування всього набутого досвіду. Під час добору задач не ставилася мета впорядкувати їх за складністю чи за типами. Тому розв'язуйте ті задачі, які, на вашу думку, найцікавіші та найдоступніші.

Більшість із них - задачі-проекти, розробка яких вимагає значного часу. Деякі задачі вимагають розробки зручного інтерфейсу для користувача, використання можливостей комп'ютера, тобто пристосування алгоритму до конкретної обчислювальної машини.

Суттєвим є також аналіз розроблених вами алгоритмів на ефективність як з точки зору часу їх виконання, так і з точки зору простоти та витонченості.

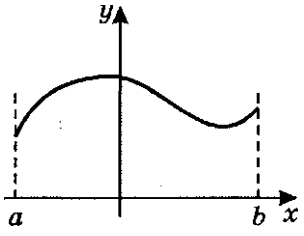
Виконайте завдання

658. Задані два натуральні числа a і *Ь*. Обчислити значення a/b з точністю до n знаків після коми.

659. На координатній площині задано коло з центром у точці $(x; y)$ і радіусом R, де x, y, R - дійсні числа. Визначити кількість точок з цілочисловими координатами, що потрапили всередину кола.

660. Дано натуральне n та дійсні числа $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$. Розглядаючи пари чисел $(x_i; y_i)$ як координати точок на площині, визначити радіус найменшого кола з центром у початку координат, яке охоплює всі дані точки.

661. «Метод Монте Карло»[1]. Помістимо фігуру у квадрат зі стороною a. Якщо «засівати» точками квадрат випадково, а в ньому відповідно і дану фігуру, то відношення кількості точок N, що потрапили у квадрат, до кількості точок M, що потрапили всереди-



[1] Назва зазначеного методу походить від назви міста, де розташована славнозвісна рулетка.

ну фігури, буде з певною точністю дорівнювати площі фігури, обчисленої за формулою Зрозуміло, що точність результату прямо пропорційно залежить від кількості точок, що згенеровані випадково.

Нехай дано деяку функцію $y = f(x)$. Обчислити площу фігури, обмежену кривою $y = f(x)$ та прямими $y = 0$, $x = a$, $x = b$.

662. Скориставшись ідеєю методу Монте Карло, визначити наближене значення числа π. Розглянемо чверть одиничного кола - його площа дорівнює $\pi/4$. Генеруючи випадково точки з координатами $0 \leqslant x \leqslant 1$ та $0 \leqslant y \leqslant 1$ у кількості N, визначимо ту кількість із них K, що потрапляє в одиничне коло з центром у початку координат. Чим більше число N, тим з більшою точністю відношення $\frac{AK}{N}$ дорівнюватиме числу π.

Розробити програму обчислення значення числа π з точністю до n знаків після коми.

663. Дано натуральне число n та цілочисловий одновимірний масив $a_1, a_2, \ldots a_n$. Визначити порядкові номери сусідніх елементів масиву з найменшою різницею значень.

664. Дано натуральне число n та послідовність $a_1, a_2, \ldots, a_n$. Визначити порядкові номери двох мінімальних членів послідовності, між якими знаходиться найменша кількість інших членів. Якщо мінімальний елемент один, то відстанню вважати 0.

665. Дано натуральне число n та цілочислову послідовність $a_1, a_2, \ldots a_n$. Визначити кількість розташованих підряд трійок членів послідовності, які можуть утворити неспадну підпослідовність цілих чисел, що йдуть підряд. Наприклад, для послідовності 3, 1, 1, 2, 1, 2, 3 відповіддю буде число 2.

666. Дано натуральне число n та послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Не впорядковуючи цю послідовність, визначити, на якому місці знаходитиметься число x з даної послідовності в її впорядкованому варіанті.

667. Дано натуральні числа n, k та послідовність дійсних чисел $a_1, a_2, \ldots a_n$. Використовуючи найменшу кількість порівнянь, визначити, яке число знаходитиметься на k-му місці в упорядкованому варіанті даної послідовності.

668. Визначити всі шестицифрові натуральні числа, які є номерами щасливих квитків (сума перших трьох цифр дорівнює сумі останніх трьох цифр), використавши для цього найменшу кількість циклів (класичний варіант - 6 вкладених циклів).

669. Два натуральні числа називаються дружніми, якщо кожне з них дорівнює сумі всіх дільників іншого, крім самого цього числа. Дано два натуральні числа 7Y і M. Знайти всі пари дружніх чисел у проміжку $[N; M]$.

670. «Обертове число». Знайти таке натуральне число, остання цифра якого 5, щоб при множенні його на 5 отримати нове число, яке при викреслюванні в ньому останньої цифри 5 і приписуванні її на початку, дорівнюватиме даному.

Виконати цю задачу, замінивши 5 на 2.

671. Дано натуральне число N. Отримати всі такі трійки чисел a_1, a_2, a_3 ($a_i \leqslant N$, $i = 1, 2, 3$), для яких одночасно виконуються дві умови:

1) $a_1 \leqslant a_2 \leqslant a_3$;

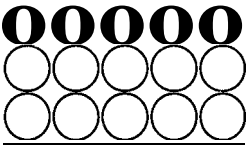
2) $a_1 + a_2 + a_3 = N$.

672. Дано натуральне число N. Представити це число всіма можливими сумами натуральних чисел.

673. Члени одновимірного цілочислового масиву $a_1, a_2, \ldots a_n$ містять цифри деякого n-значного двійкового числа. Визначити цифри наступного та попереднього за значенням двійкових чисел.

674. Нехай по деякому каналу зв'язку передається повідомлення, яке має вигляд послідовності нулів та одиниць. Через деякі перешкоди можливий помилковий прийом певних сигналів: нуль може бути сприйнятий як одиниця і навпаки. Можна передавати кожний сигнал тричі, замінюючи при цьому, наприклад, послідовність 1, 0, 1 послідовністю 1, 1, 1, 0, 0, 0, 1, 1, 1. Три послідовні цифри при розшифровуванні замінюються тією цифрою, яка трапляється серед них принаймні двічі. Таке потроєння сигналів значно підвищує якість передавання інформації. Написати програму розшифрування інформації.

675. Пластина, з якої штампують шайби, має форму прямокутника зі сторонами a і b, а діаметр шайби d. Існують такі варіанти штампування:

а) 

б) 

Яке штампування шайб дає економію металу?

676. Побудувати послідовність a_k, a_{k-1}, $\ldots$ a_0, де $0 \leqslant a_i \leqslant 9$, (тобто a_i - десяткова цифра), для якої виконується умова:

$$a_k * 10^k + a_{k-1} * 10^{k-1} + \ldots + a_1 * 10 + a_0 = 2^{2000}.$$

677. Використовуючи умову попередньої задачі, для заданого натурального числа n знайти n^{2000}

678. Дано натуральне число n ($1 \leqslant n \leqslant 2000$). Отримати послідовність десяткових цифр числа $n!$.

679. Дано натуральне число a_0 ($1 \leqslant a_0 \leqslant 10^{20}$). Відомо, що члени послідовності a_1, a_2, $\ldots$ a_i, ... утворюються за таким правилом:

$a_i/2$, якщо a_i - парне;

$3a_i + 1$, якщо a_i - непарне.

Визначити, для якого k член послідовності $a_k = 1$.

680. Дано натуральне число n ($1 \leqslant n < 10^{20}$). Якщо задане число не паліндром, то змінимо порядок його цифр на зворотний і додамо отримане число до заданого. Якщо одержане число знову не паліндром, то виконаємо над ним ту саму послідовність дій. Визначити, на якому кроці одержимо число паліндром.

Примітка. Паліндромом називається послідовність символів, яка зліва направо і справа наліво читається однаково.

681. Дано цілі числа n ($n > 0$) та P ($P > 1$). Скласти алгоритм переведення числа n у систему числення з основою P за умови:

1) число $n < 10^{256}$, $P < 10$; 2) число $n < 10^{256}$, $P \leqslant 32$.

682. Дано ціле число n ($n > 0$) у системі числення з основою P ($1 \leqslant P \leqslant 32$). Скласти алгоритм переведення числа n у десяткову систему числення.

683. Довільний опуклий багатокутник на площині заданий координатами своїх вершин (за годинниковою стрілкою). Визначити кількість прямих кутів багатокутника і номери вершин, в яких вони знаходяться.

684. Дано два натуральні числа n та m і дві дійсні матриці A_{ij}, де $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$ та B_{ij}, де $i = 1, 2, \ldots, m$; $j = 1, 2, \ldots, n$. Визначити значення елементів матриці C_{ij}, де $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, m$, що є результатом добутку матриць A та B:

$$C_{ij} = \sum_{}^{n} A_i \quad .$$

685. Дано текст-формулу, яка містить дужки. Вилучити із тексту всі символи, крім дужок. Вивести отриманий вираз і перевірити коректність даної формули щодо розташування дужок.

686. При створенні текстових редакторів виникає ситуація, коли під час набирання тексту користувачем у кінці рядка залишається кілька невикористаних позицій. Кількість таких невикористаних позицій змінюється від рядка до рядка і тому правий край створеного тексту виходить нерівним. Проблема вирішується рівномірним додаванням пробілів між словами кожного рядка.

Нехай дано символьну матрицю $n \times m$, у кожному з рядків якої є принаймні одна група пробілів, розташована всередині рядка. Розробити програму, яка за рахунок внесення змін у групи пробілів усередині рядків приводить до вирівнювання правих країв рядків, тобто в кінці кожного рядка матриці будуть відсутні пробіли. Кількість пробілів у різних групах, розташованих у межах одного рядка, повинна різнитися не більше ніж на одиницю.

687. Дано таблицю натуральних чисел розміром *nxn,* де *n* непарне і $3 \leqslant n \leqslant 100$. Обробка елементів таблиці виконується квадратними контурами, починаючи від зовнішнього контура і крокуючи до центра. Якщо найменше число в контурі на даному кроці є непарним, то всі елементи наступного контура збільшуються на 1, в протилежному випадку - залишаються без змін. Надрукуйте число, яке буде в центрі.

688. Дано послідовність з *n* цілих чисел, членами якої можуть бути числа 0, 1, 2. Упорядкувати цю послідовність за зростанням, використавши для цього якомога менше порівнянь.

689. У дитячому садку, де є три групи дітей А, *В, С* по 15 осіб у кожній, організовано змагання на кмітливість. Одне із завдань полягає в тому, щоб діти вишикувалися в ряди по троє так, щоб справа, зліва, позаду і попереду кожної дитини не стояла дитина з її групи. Допоможіть дітям.

690. У спортивних змаганнях беруть участь *n* команд по *m* учасників у кожній. В одному турі змагань учасникам треба по одному стати у клітини розкресленого на підлозі прямокутника розміром *nxm,* щоб у кожному ряду по горизонталі стояли учасники різних команд, а в кожній колоні не стояли підряд два учасники з однієї команди. Крім того, потрібно, щоб у кожній колоні стояло якомога менше учасників з однієї команди. Розробити програму, яка допоможе спортсменам.

691. Для пояснення теми «Арифметична прогресія» в учителя математики були заготовлені таблички з членами двох прогресій:

$a_1, a_2, \quad a_i,$ де a, = a, $+i*n$ та

$b_1, b_2, \quad b_i,$ де $b_i = b_1 + i*m.$

Кроки цих прогресій *n* та *m* - натуральні взаємно прості числа. Так сталося, що таблички перемішалися. Коли їх знову розклали в ряд, то вони утворили деяку послідовність з k членів, у якій виявилися дві картки з однаковими числами. Допоможіть вирішити проблему, визначивши члени обох прогресій, кількість членів у кожній з них та їх кроки n і m.

Пояснення. Арифметична прогресія - це послідовність чисел, кожне з яких отримується додаванням до попереднього одного й того самого числа, що називається кроком.

692. **«Лабіринт».** У деякому замку побудований лабіринт з однакових за розміром квадратних кімнат, схема якого утворює розкреслений у клітинку прямокутник розміром *nxm* ($n \leqslant 10$, $m \leqslant 100$), де по горизонталі знаходиться *m* кімнат, а по вертикалі - *n*. Усі кімнати пронумеровані послідовно по рядах, починаючи з верхньої лівої і завершуючи нижньою правою. Кімнаті: можуть мати або по двоє дверей, в одні з яких можна ввійти, а через інші вийти, або ж зовсім не мати дверей (немає замуровь них дверей, тобто таких, які в одній кімнаті є, а в сусідній чере:

стінку відсутні). Лабіринт має лише двоє дверей, через які можна як ввійти так і вийти. На схемі кожна кімната зашифрована лише одним числом, яке у двійковому зображенні визначає в ній положення дверей: північ - перша цифра зліва направо, схід - друга цифра, південь - третя, захід - четверта, причому 1 – у відповідному напрямку двері є, 0 - дверей немає. Розшифруйте і зобразіть графічно дану схему, відобразивши на ній розташування дверей, та роздрукуйте послідовність порядкових номерів кімнат для проходу через лабіринт.

Приклад:

$n = 2$; $m = 2$ - розміри лабіринта;

5 3 - зашифрована схема;

0 10

Відповідь: 1 2 4 або 4 2 1 та малюнок.

693. «Біжуча хвиля». Розробити діючу максимально наближену до реальності модель «розгойдування» мотузки за законом синусоїди (початковий стан - горизонтальна пряма). Врахувати, що розгойдування мотузки починається з лівого кінця, а кінці її не закріплені. Вагою мотузки, опором повітря та іншими фізичними параметрами - знехтувати.

694.[1] Дано текст, який завершується крапкою (в сам текст крапка не входить). Визначити, чи є цей текст правильним записом «формули»:

<формула>: := <терм>|(<формула><знак><формула>)
<знак>: := +|–|*
<терм>: := <ім'я>|<ціле>
<ім'я>: := <літера>|<ім'я><літера>|<ім'я><цифра>
<ціле>: := <цифра>|<ціле><цифра>
<літера>: := а|б|в|г|д|е|ж
<цифра>: := 0|1|2|3|4|5|6|7|8|9

У записі цієї задачі використаний символ «|», який треба читати як «або».

695. На лісовій галявині вночі білим кольором засвітилися світлячки. Але з'явився злий чарівник і почав закидати їх шапками-невидимками чорного кольору.

Перелякані світлячки, яких має накрити кинута шапка, змінивши колір на червоний, тікають на будь-яке нове безпечне місце, не зайняте в даний момент іншими світлячками і не накрите шапками. Заспокоївшись, вони знову світяться білим кольором. Ця картина повторюється щоразу, як тільки з'являється нова шапка-невидимка, кинута чарівником.

Зобразіть цю сцену на екрані монітора, який відіграватиме роль галявини, в такому діалоговому покроковому режимі:

[1] Див. № 501

<задайте кількість світлячків на галявині>;
Оадайте координати світлячків>;
<задайте кількість шапок-невидимок>;
<задайте діаметр 1-ї шапки та місцеположення її центра на галявині>;
(виконання алгоритму)
<задайте діаметр 2-ї шапки та місцеположення її центра на галявині>;
(виконання алгоритму)

(виконання алгоритму)
Оадайте діаметр n-ї шапки та місцеположення її центра на галявині>;
(виконання алгоритму).

696. Дано деякий текст, що складається зі слів та розділових знаків «!», «?», «;», «:», «,», «.». Розділові знаки в тексті слідують за словами, слова відокремлені пробілами. Введений текст необхідно розмістити в стовпчик, завширшки n символів, вирівнюючи його по правій та лівій межах, рівномірно додаючи необхідні пробіли. Довжина слів у тексті менша за ширину стовпчика.

697. Трикутником Паскаля називається числовий трикутник такого вигляду:

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
```

По боках цього трикутника розміщено одиниці, а кожне число всередині дорівнює сумі двох інших, що стоять над ним у найближчому рядку вгорі.

Дано натуральне число n. Вивести перші n рядків трикутника Паскаля.

698. Трикутник Лейбніца є дещо «протилежним» трикутнику Паскаля. Кожне число трикутника Лейбніца є сумою своїх південно-західного та південно-східного сусідів. Фрагмент трикутника Лейбніца має такий вигляд як показано на малюнку.

```
          1
        1   1
      2   2
    1   1   1
  3   6   3
1   1   1   1
```

Дано натуральне число n. Вивести перші n рядків трикутника Лейбніца.

699. Магічним квадратом називається квадратна таблиця розміром $N \times N$, елементами якої є натуральні числа від

1 до N^2, що жодного разу не повторюються, і при цьому в ній суми по будь-якій горизонталі, вертикалі та двох діагоналях дорівнюють одному і тому самому числу. Наприклад,

16	2	3	13
5	11	10	8
9	7	6	12
4	14	15	1

Магічний квадрат називається *симетричним,* якщо при обміні місцями симетричних рядків, стовпчиків або діагоналей властивості його не змінюються.

Дано натуральне число N та квадратна матриця розміром $N \times N$. Визначити, чи дана матриця є:

1) магічним квадратом;

2) симетричним магічним квадратом.

700. Деяка сума представлена натуральним числом N. Отримати всі способи виплати цієї суми за допомогою монет вартістю 1, 5, 10 та 25 копійок.

701. Деяка сума представлена натуральним числом M. Отримати всі способи виплати цієї суми за допомогою монет вартістю $a_1, a_2, \ldots a_n$ копійок.

702. Дано натуральне число n $(n > 1)$ та послідовність $a_1, a_2, \ldots a_n$. Визначити найдовшу зростаючу підпослідовність

$$a_i \leqslant a_{i+1} \leqslant a_{i+2} \leqslant \ldots \leqslant a_k \quad (i \geqslant 1,\ 2 \leqslant k < n)$$

цієї послідовності. Якщо їх декілька, то вивести елементи першої з них і останньої.

703. Нехай є 10 гир масою $a_1, a_2, \ldots a_{10}$, де a_i - натуральні числа. Через c_k позначено кількість способів, якими можна скласти масу k, тобто c_k - це кількість розв'язків рівняння $a_1x_1 + \ldots + a_{10}x_{10} = \kappa$, де x_i $(i = 1, \ldots 10)$ може набувати значення 0 або 1. Знайти $c_0, \ldots c_{10}$.

704. Дано натуральне число n. Вставити між деякими цифрами 1, 2, 3, 4, 5, 6, 7, 8, 9, записаними саме в такому порядку, знаки арифметичних операцій «+», «-» так, щоб значенням утвореного виразу було задане число n. Якщо жодне розставлення знаків неможливе, повідомити про це.

705. На екрані монітора зображені n ліній. Кожна з ліній задається своєю послідовністю точок $(x_{0i}; y_{0i})$ і $(x_{1i}; y_{1i})$, $(i = 1, \ldots n)$, що її утворюють.

Написати програму зафарбовування області, утвореної деякими даними лініями, якщо відомі координати точки (a; b), що належить цій області, колір ліній та колір зафарбовування. Межі екрана монітора також вважаються лінією, що обмежує

всю область. Користуватися стандартними можливостями мови програмування забороняється.

Дані вводяться в такій послідовності:

<колір ліній>;
<кількість ліній>;
<координати точок 1 -ї лінії>;
<координати точок 2-ї лінії>;

<координати точок N-ї лінії>;
<колір зафарбовування>.

706. Дано скінченну множину імен мешканців деякого міста. Для кожного з мешканців перераховані імена його дітей. Мешканці X та У вважаютьсяріднею, якщо:

1) або X - дитина У,

2) або У - дитина X,

3) або існує деякий Z такий, що X єріднею Z, а Z єріднею У.

Перерахувати всі пари мешканців міста, які єріднею.

707.[1] Побудувати синтаксичний аналізатор для поняття *константний-вираз.*

цифра {*цифра*}*

константнии-вираз ::= *константнии-вираз* { + } *знак-операцп*

708. Скласти програму-аналізатор для перевірки поняття *просте-логічне,* де

просте-логічне ::= *TRUE* | *FALSE* | *простий-ідентифікатор* | *NOT* (*просте-логічне знак-операції просте-логічне*)

простий-ідентифікатор ::= *літера*

знак-операцп ::= { *AND* }.

709. У вхідному файлі дано текст, який закінчується крапкою. Перевірити, чи задовольняє його структура таке визначення:

<текст> ::= <елемент> | <елемент> <текст>
<елемент> ::= a | b | (<текст>) | [<текст>] | {<текст>}.

710. Корені $y = \sqrt[k]{x}$ можна обчислювати із заданою точністю ε за такою ітераційною формулою:

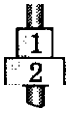
$y_0 = 1;$ = + $- y_n)/k$ $(n = 0,$ 1, 2, ...).

За даним дійсним числом $a > 0$ обчислити вираз

[1] Про умовні позначення цієї задачі та двох наступних задач див. № 501

$$1+\sqrt[7]{3}+a$$

711. «Ханойські башти». Є три стержні A, B, C та n дисків різного розміру, пронумерованих від 1 до n в порядку зростання їхніх розмірів. Спочатку всі диски насаджені на стержень A так, що на кожному диску зверху лежить менший за розміром диск. Треба перенести всі диски із стержня A на стержень С, дотримуючись таких умов: диски можна переносити лише по одному і більший диск не можна ставити на менший. Для цієї операції треба скористатися стержнем B. Надрукувати послідовно всі кроки, вказуючи пари стержнів, які беруть у них участь (першим - стержень, з якого знімається диск, другим — стержень, на який він перекладається).



$n-2$
$n-1$

A *B* *C*

712. Розробити програму-редактор «Павучок» для малювання графічних зображень, яка розумітиме і виконуватиме такі команди:

1) *Forward* - перемістити на вказану відстань;
2) ***Left*** - повернути наліво на заданий кут;
3) *Rigth* - повернути направо на заданий кут;
4) *Up* - не залишати за собою павутинки;
5) *Down* - залишати за собою павутинку.

Виконання відповідних команд закріпити за клавішами **F, L, R, U, D,** після натиснення яких треба задати відповідний параметр.

713. **«Життя».** У 1970 р. математик Джон Конвей запропонував чудовий алгоритм клітинного автомата, який можна вважати грою, що описує популяцію стилізованих організмів, яка розвивається у часі під дією протиборства тенденцій розмноження і вимирання. Правила цієї гри такі.

Дано поле розміром $n \times m$ клітинок. На клітинках з фішками є життя, на інших - немає. Сусідами кожної клітинки вважаються всі вісім клітинок, які її оточують.

Ця система функціонує за такими «генетичними законами»:

- «виживання»: кожна фішка, в якої є дві або три сусідні фішки, виживає і переходить у наступне покоління;

- «загибель»: кожна фішка, в якої виявляється більше як три сусіди, гине, тобто зникає з дошки, через перенаселення; кожна фішка, біля якої вільні всі сусідні клітини або зайнята лише одна, гине через самотність;
- «народження»: якщо кількість фішок, з якими межує деяка порожня клітинка, дорівнює трьом (не більше і не менше), то на цій клітинці відбувається народження нового «організму», тобто на цій клітинці з'являється фішка.

Процес існування популяції вважається завершеним у трьох випадках:

1) усі фішки з поля зникли, тобто система вимерла;
2) на полі немає порожніх клітин, тобто система перенаселена;
3) на полі не відбувається ніяких змін станів клітин, тобто система перейшла у стан статики.

Початкове розміщення фішок на полі - довільне.

Написати програму, яка б моделювала цю гру.

714. **«Мікроорганізми».** Розробити програму існування мікроорганізмів за таких умов. На прямокутному полі $n \times m$ клітинок за законом випадковості в будь-якому напрямі в межах заданого поля стрибають мікроорганізми двох видів. Початкова кількість і вага мікроорганізмів обох видів однакова. Якщо мікроорганізм стрибнув на клітину, яка вже зайнята іншим мікроорганізмом, то відбувається аналіз ситуації, що склалася, за такими правилами:

1) якщо в клітинці виявився мікроорганізм того самого виду, то вони разом утворюють новий мікроорганізм із заданою початковою вагою, і всі перестрибують на нові місця;
2) якщо в клітинці виявився мікроорганізм іншого виду, то їхня вага змінюється за такими правилами:
   a) якщо їхня вага на даний момент однакова, то той мікроорганізм, який стрибнув у клітинку, поглинає половину ваги того мікроорганізму, що її займав, зменшуючи відповідно вдвічі його вагу;
   b) якщо вага мікроорганізмів, що потрапили в одну клітинку, різна, то більший мікроорганізм поглинає половину ваги меншого, зменшуючи його вагу відповідно вдвічі;
3) мікроорганізм гине у двох випадках:
   a) якщо його вага стала меншою, ніж 10 % від початкової ваги;
   b) якщо його вага стала більшою, ніж 200 % від початкової ваги.

Існування популяції мікроорганізмів завершується, коли загине один вид.

715. «Вісім ферзів». На шаховій дошці розміром 8x8 розташувати вісім ферзів так, щоб вони не загрожували один одному.

716. Дано цілочисловий одновимірний масив з n елементів, значеннями якого є лише додатні та від'ємні числа. Використовуючи найменшу кількість порівнянь і не використовуючи додаткові масиви, перетворити цей масив так, щоб спочатку були розташовані від'ємні числа, а потім - додатні. Порядок чисел в додатній та від'ємній групах - довільний.

717. У двовимірному цілочисловому масиві розміром $n \times m$ елементи набувають лише натуральних значень. Прямокутником у масиві вважатимемо групу сусідніх елементів одного значення, що разом утворюють прямокутник розміром $k \times l$ ($k > 1$, $l > 1$). Прямокутником не вважається група елементів, що належить іншому прямокутнику. Обчислити кількість прямокутників у заданому масиві.

718. На одній прямій вулиці, де з одного боку розташовано n будинків, встановлюють телефони. Відстань між будинками задається дійсним масивом $d_1, d_2, \ldots, d_{n-1}$, а кількість телефонів у кожному будинку - цілочисловим масивом $k_1, k_2, \ldots k_n$. Визначити, в якому з будинків необхідно встановити АТС, щоб використати для цього найменшу сукупну довжину телефонних дротів, враховуючи, що для встановлення кожного телефону необхідно тягнути окрему лінію від АТС.

719. Міста деякої держави побудовані таким чином: у місті є кілька майданів, кожний з яких попарно з'єднаний вулицями з іншими майданами. З кожного майдану виходить парна кількість вулиць. З будь-якого майдану можна пройти на інший ланцюжком вулиць. Для кожного майдану відомий список майданів, які зв'язані з ним вулицями. Написати алгоритм, що будує маршрут, який дає змогу побувати на всіх майданах, обійшовши всі вулиці міста, проходячи кожною з них лише один раз.

720. У п'ятизірковому готелі є N номерів. Адміністратор має інформацію про кількість місць у кожному номері, перелік зайнятих номерів, кількість вільних місць у них на даний момент, стать клієнтів. Вважатимемо, що разом можуть бути поселені клієнти однієї статі.

У готель прибула група туристів - K жінок і L чоловіків. Скласти алгоритм, за яким можна розселити клієнтів у найменшу кількість номерів за таких додаткових умов:

1) група заїхала вдень на кілька годин, тому розселення може відбуватися в будь-які номери, незважаючи на стать клієнтів, з можливим підселенням;
2) група заїхала вночі, тому необхідно враховувати стать клієнтів. Можливе підселення в зайняті номери, але без переселення раніше розташованих клієнтів;

3) група заїхала на кілька днів, тому при розселенні враховується стать клієнтів, можливе підселення в зайняті номери та переселення раніше розташованих клієнтів в інші номери.

Вхідний текстовий файл має таку структуру:

перший рядок - кількість номерів у готелі (N), кількість жінок у групі *(K),* кількість чоловіків (L);

другий рядок - кількість місць у номерах готелю (за зростанням номерів готелю);

третій рядок - номер кімнати готелю, кількість зайнятих у ньому місць, стать клієнтів (« 0 » - чоловіки, « 1 » - жінки).

Вихідний текстовий файл повинен містити інформацію про розселення клієнтів у готелі після розміщення групи туристів, який складатиметься з трьох рядків відповідно до пунктів задачі. Структура рядків цього файла така сама, як третього рядка вхідного файла.

721. У старовинному саду росли N рідкісних плодових дерев. Цим садом опікується один літній досвідчений садівник. Він акуратно записує врожайність кожного дерева, доглядає і поливає їх. Складіть алгоритм за такими умовами.

1) Визначити середню врожайність саду, якщо відома врожайність кожного дерева ($t_i, i = 1, 2, \quad N$).
2) Зобразити схему саду на екрані монітора, якщо відомі цілочислові координати кожного дерева ($x_i; y_i$), i = 1, 2, N, ввівши, за потреби, масштаб.
3) Визначити, де садівникові варто встановити колонку для поливання дерев (у цілочислових координатах) щоб, поливаючи дерева, проходити найменшу відстань.
4) Обчислити довжину огорожі саду, яка пройде по крайніх деревах.
5) Визначити, у якій послідовності садівнику необхідно обходити дерева, щоб затрачувати при цьому найменше часу.

722. На прямокутному аркуші в клітинку намалювали лабіринт, позначивши «прохідні» клітинки - 0, а «непрохідні» $--1$. Дві клітинки на межі поля визначені як вхід у лабіринт та вихід із нього.

1) Організувати введення початкових даних з текстового файла INPUT.DAT.
2) Визначити, чи є лабіринт «прохідним», тобто чи існує хоча б один шлях між вхідною і вихідною клітинками, і якщо так, то знайти його.
3) Знайти найкоротший шлях проходження лабіринту.
4) Знайти всі шляхи проходження лабіринту.

Вхідний текстовий файл містить дані про лабіринт через пробіл по таких рядках:

перший рядок - розмір лабіринту (кількість рядків і стовпчиків);

другий рядок - координати клітинок входу і виходу з лабіринту (номери рядка і стовпчика).

У кожному наступному рядку вказані параметри (0 або -1) клітинок відповідного рядка лабіринту, починаючи з верхнього.

Результати роботи програми вивести на екран монітора.

723. На екран монітора виводяться відрізки, розміщені на одній горизонтальній прямій. Відрізки задаються двома координатами X LEFT та X RIGHT. Відрізки можуть перекриватися. Знайти сумарну довжину видимих відрізків на екрані.

724. Скласти програму, яка за точним квадратом цілого числа знаходить саме це число.

Вхідний текстовий файл з назвою SQRT.IN містить кілька вхідних чисел, розміщених в окремих рядках, довжина яких не перевищує 30 символів.

Вихідний текстовий файл з назвою SQRT.OUT містить по одному результуючому числу в окремих рядках.

725. Нехай n книжок мають відповідно товщину $a_1, a_2, \ldots, a_n$. Книжки треба розкласти на два стоси так, щоб різниця між їх висотою була найменшою.

726. Відомо, що розміри деякої прямокутної кімнати становлять nxm, де n, m - цілі числа. На складі в достатній кількості є квадратні килими, розміри яких (довжини сторін) визначаються цілими числами $a_1, a_2, \ldots, a_n$. Визначити, скільки і яких килимів потрібно, щоб повністю покрити кімнату за таких умов:

1) килими можна накладати один на одний;

2) килими не можна накладати один на одний, (кількість килимів не має значення);

3) килими не можна накладати один на одний і кількість їх при цьому повинна бути мінімальною.

727. У приміщенні дослідницької лабораторії стоїть клітка з мавпою. За межами цієї клітки на відстані l від мавпи на підлозі є кнопка, натиснувши на яку, вона отримає банан. Дістати кнопку, не сходячи з місця, мавпа може, тільки склавши із заготовок палицю. Всі заготовки пронумеровані від 1 до n і мають свої довжини x_i ($i = 1, 2, \ldots, n$). Під час натискання на кнопку мавпа може тримати палицю за один кінець, натискаючи в цей час на кнопку іншим кінцем. На цю процедуру мавпі відведено обмежений час, тому кількість заготовок повинна бути мінімальною. Довжина лапи мавпи - m. Допоможіть мавпі скласти палицю, вказавши номери необхідних заготовок. Як-

що є кілька варіантів - запропонуйте їх. Усі вхідні дані вважати цілими числами.

728. На підлозі прямокутного приміщення від однієї стіни до іншої лежить пряма паперова смужка, паралельна довшій стіні. Над підлогою паралельно їй є кілька перекриттів. У кожному з цих перекриттів над смужкою зроблені прямокутні прорізи різної ширини, проекції яких на підлогу перпендикулярні до смужки. Крізь таку систему перекриттів на смужку падають паралельні промені світла. Вказати координати освітлених частин смужки, якщо для кожного перекриття даної конструкції відомі кількість прорізів і координати кінців їх проекцій на смужку. Розсіюванням світла та шириною смужки знехтувати.

729. Відомий усім форт Буайяр пропонує своїм відвідувачам таке випробування: піднятися досить нахиленою слизькою площиною вгору, щоб дістати ключ. На площині розміщені невеличкі виступи, кожен з яких настільки малий, що поміститися на ньому може лише одна нога. Домовимося, що під час пересування площиною гравець не може перехрещувати ноги (ліву за праву і навпаки). Нехай l - довжина кроку гравця, n - кількість виступів, $(x_i; y_i)$, де $i = 1, 2, \ldots n$ - послідовність координат виступів на площині.

Вважатимемо, що початкове положення гравця:

$x_0 \geqslant 0$, $y_0 = 0$ (стартує з будь-якого місця на підлозі).

Визначити всі можливі найменші за кількістю кроків варіанти (якщо вони існують) послідовностей номерів виступів, по яких необхідно рухатися гравцю, чергуючи ноги, щоб дістатися самого верхнього виступу. Вкажіть, з якої ноги він починає рух. Усі вхідні дані вважати цілими числами.

730. Розробити алгоритм синтаксичного аналізу виконуваної частини тексту Pascal-програми. Вважатимемо, що на алфавіт мови програмування накладено такі обмеження:

1) ідентифікатором може бути лише латинська літера або латинська літера і цифра;
2) текст програми містить лише стандартні процедури введення/виведення інформації, оператори присвоювання, складені оператори та оператори **IF, CASE, FOR, WHILE, REPEAT...UNTIL;**
3) виразом в операторі присвоювання може бути лише ціла константа типу **Shortint** або ідентифікатор;
4) складені умови в операторах розгалуження та циклу можуть містити лише два операнди;
5) оператори розгалуження та циклу не можуть бути вкладеними.

Результатом виконання програми-аналізатора повинно бути виведення тексту програми, що аналізується, із зазначен-

ням місць, де зроблені помилки, та покроковий аналіз цих помилок. Аналіз помилок повинен здійснюватись за схемою:

a) «помилка в операторі»;

b) не вистачає ... («**begin**», «**end**», «;», «'», «)», «(», «,»);

c) «це не ...» (ідентифікатор, число, логічна операція).

731. На прямокутному більярдному столі розміром $n \times m$ кием під кутом а вдаряють кулю, яка лежить біля однієї зі сторін стола в позиції (x; *y*). Вважатимемо, що борти стола паралельні осям координат, а точка (0; 0) знаходиться у верхньому лівому куті цього стола. Визначити координати точки на більярдному столі, в яку вдариться куля:

1) на першому кроці;

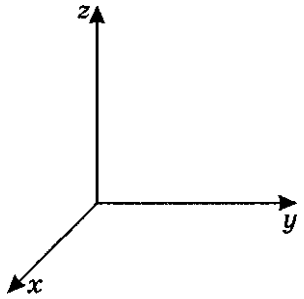
2) на другому кроці;

3) на k-му кроці.

Опором повітря, розміром кулі та тертям знехтувати. Відобразити процес відбивання кулі графічно, враховуючи масштабування.

732. Розробити програму, яка виконує синтаксичний аналіз функції $f(x, y)$, яка містить бінарні операції «+», «–», «*», «/», «^» (піднесення до степеня) та функції sin, cos, sqrt.

733. Розробити програму, яка за введеною функцією *f(x, y)* аналізує її синтаксис, будує її графік на екрані монітора у тривимірному просторі (мал. 30) з урахуванням прихованих ліній і обертає цей графік функції навколо осей *Oy* та *Oz* за допомогою курсорних клавіш «↑», «↓», «→», «←».



Мал. 30

734. Розробити середовище «База даних», яке передбачає за введеною кількістю полів та записів виконання автоматичних розрахунків у вказаних полях за заданими формулами. Середовище повинно надавати користувачу можливість редагування введених даних, зміни розмірів самої бази даних.

735. У даному двовимірному масиві визначити максимальний серед мінімальних і мінімальний серед максимальних елементів у всіх рядках.

736. Відомо, що на один носій інформації можна записати m байт інформації. N файлів користувача займають відповідно $k_1, k_2, \ldots k_N$ байт інформації. Визначити:

1) яку найменшу кількість носіїв інформації можна при цьому використати;

2) групи, на які треба розбити всі файли для наступного перенесення інформації у файлах на носії;

3) скільки вільного місця залишиться на кожному з використаних носіїв.

737. «Дзеркальне **відображення».** Дано послідовність з n чисел: k_1, k_2, k_n. Ця послідовність продовжена таким чином: спочатку до неї в дзеркальному відображенні дописані ті ж самі числа, потім до послідовності, яка була отримана на попередньому кроці, знову в дзеркальному відображенні дописані всі числа і т. д. Визначити, яке число буде знаходитися на m-**му** місці.

Приклад побудови **послідовності:**

$k_1 k_2 \ldots k_{n-1} k_n \; k_n \ldots k_2 k_1 \mid k_1 k_2 \ldots k_{n-1} \; k_n \; k_n$ **fc„_, ... $k_2 k_1$ ...**

738. «Числа Каталана». Перші сім чисел Каталана мають такий вигляд: 1, 2, 5, 14, 42, 132, 429. Для визначення загального члена послідовності чисел Каталана використовується **формула:**

$$_n = \frac{(2n)!}{n!(n+1)!}.$$

Обчислити n-**ний** елемент чисел Каталана, не використовуючи тип double, якщо $n \leqslant 68$.

739. «Числа Бернуллі». Програма, створена у першій половині XIX ст. першою програміcткою у світі леді Адою Лавлейс для аналітичної машини Чарльза Беббіджа, мала назву «Список операцій для обчислення чисел **Бернуллі».** Для отримання чисел Бернуллі можна використати таку рекурентну залежність:

$$B_0 = 1; \quad C_k^0 B_0 + C_k^1 B_1 + C_k^2 B_2 + \ldots C_k^{k-1} B_{k-1} = 0.$$

Дано натуральне число N. Визначити числа Бернуллі B_1, B_2, B_N.

740. «Числа Мерсенна». Прості числа виду $2^n - 1$ названі на честь фразцузького вченого Марселя Мерсенна. Дано натуральне число M. Розробити програму, що генерує числа Мерсенна, менші за M.

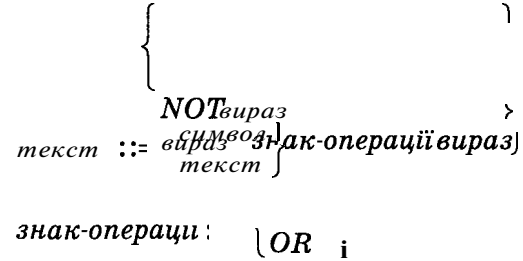
741. **«Трафарет».** Для багаторазового нанесення зображення на поверхню використовується трафарет із цим зображенням. Головна умова виготовлення трафаретів така: після вирізання нанесеного зображення його елементи не повинні випадати. Скласти алгоритм, який за даною таблицею розміром *nxm* визначить, чи є на даному трафареті частини, що випадають, та їх кількість. У таблиці, що містить лише значення 0 та 1, зображення закодовано цифрами 1. Вважається, що сусідні елементи таблиці «зливаються» в єдине ціле, якщо вони містяться на одній горизонталі чи вертикалі. Елементи зовнішнього контуру закодовано значеннями 0.

Приклад.

0 0 0 0 0	0 0 0 0 0
0 0 1 0 0	0 1 1 1 0
0 1 0 1 0 - елемент трафарету	0 **1 0 1 0** - елемент трафарету
0 0 1 0 0 не випадає	0 **1 1 1 0** випадає
0 0 0 0 0	0 0 0 0 0

742. Заповнити прямокутник розміром $n \times m$ числами таким чином. У верхньому лівому куті таблиці ставиться довільне натуральне число. Кожний наступний елемент таблиці дорівнює сумі всіх елементів прямокутного фрагмента даної таблиці, верхній лівий елемент якого збігається з верхнім лівим елементом даної таблиці, а правий нижній елемент є шуканий елемент.

743. Розробити програму **зпошуку** текстових файлів на зовнішньому носії, елементи вмісту яких визначаються за таким шаблоном-виразом:



Як і в будь-якому логічному виразі, в шаблоні-виразі також можуть використовуватися дужки.

744. Розробити діючу модель кидання і перекочування кубика по гральній дошці з бар'єрами, враховуючи його відбивання від бар'єрів та фізичні закони гальмування руху. Початковий стан кубика і місце його падіння на гральний стіл визначається випадково.

745. Під час друкування книжок за один прохід на великому аркуші паперу друкується зразу кілька сторінок книжки. Після цього спеціальна машина складає надруковані великі аркуші у так звані зошити, з яких потім формується книжка. Аркуш складається завжди вдвоє, причому спочатку знизу вгору, а потім зліва направо доти, поки не утвориться зошит.

Скласти програму, яка виведе на екран монітора послідовність номерів сторінок зошита відповідно до розташування на друкованому аркуші зліва направо і згори вниз (спочатку для тієї сторони аркуша, що містить 1-шу сторінку, потім для тієї, що містить 2-гу).

Врахувати, що кожний зошит містить 2^{k+1} сторінок книжки, де $k = 0, 1, 2, \ldots 9$ - кількість згинань аркуша паперу.

Зразок виведення результатів:

к = 0: 1,2

k = 1: 1, 4, 2, 3

к = 2: 1, 8, 4, 5, 7, 2, 6, 3

к = 9: .....

746. **«Фонтан».** У місті Енську на центральному майдані побудували новий фонтан, у центрі якого на підвищенні містить-

ся прямокутна конструкція, що складається з однакових блоків - прямокутних паралелепіпедів з квадратами в основі. Висоту кожного окремого блока можна регулювати. Після вимкнення фонтана в деяких місцях конструкції залишається вода. Дані про конструкцію оформлені у вигляді таблиці, кожен елемент якої містить висоту відповідного блока (найнижчий можливий рівень відповідає числу 0). Визначити:

1) чи залишаться всередині конструкції (не враховуючи крайніх) блоки, не залиті водою;
2) найбільшу глибину калюжі;
3) які блоки залишаться покритими водою (1), а які ні (0) після відключення фонтану.

Приклади.

а) *Вхідні дані:*	*вихідні дані:*
5 5	залишаться; найбільша глибина - 0
1 1 1 1 1	0 0 0 0 0
1 2 2 2 1	0 0 0 0 0
1 2 3 2 1	0 0 0 0 0 - елемент трафарету
1 2 2 2 1	0 0 0 0 0 випадає
1 1 1 1 1	0 0 0 0 0

б) *Вхідні дані:*	*вихідні дані:*
5 5	не залишаться; найбільша глибина - 1
3 3 3 3 3	0 0 0 0 0
3 2 2 2 3	0 1 1 1 0
3 2 1 2 3	0 1 1 1 0
3 2 2 2 3	0 1 1 1 0
3 3 3 3 3	0 0 0 0 0

747. «Дипломатичні переговори». У міжнародних переговорах на високому дипломатичному рівні беруть участь N представників різних країн. У залі, де проходитимуть переговори, встановлено круглий стіл. Ставлення одних представників країн до інших оцінюється коефіцієнтом сприятливості, що визначається числом із проміжку $[-1; 1]$.

Як розсадити учасників переговорів за круглий стіл, щоб атмосфера переговорів була найсприятливішою? Коефіцієнтом сприятливості між двома сусідами вважається середнє арифметичне значення коефіцієнтів сприятливості один до одного.

748. «Казка». У відомій казці Шарля Перо Червона Шапочка зустрічає Сірого Вовка в лісі та розповідає йому про хворобу бабусі, після чого Сірий Вовк намагається випередити Червону Шапочку і добігти до будиночка бабусі першим.

Нехай схемою частини лісу, якою після зустрічі будуть переміщуватися наші герої, є поділений на клітинки прямокутник розміром $n \times m$. Місце зустрічі Червоної Шапочки і Сірого Вовка позначено у верхньому лівому куті, а будинок бабусі знаходиться на межі правої сторони прямокутної схеми. На схемі

повні клітинки позначені числом 1 (можна пройти) та числом 0 (пройти не можна).

Обидва наші герої можуть рухатися лише в двох напрямах до будинку бабусі - на схід і на південь, рух навкіс їм заборонений. Маленька дівчинка може переходити або на сусідню клітинку, або перестрибувати через одну. Сірий Вовк може переміщуватися на сусідні клітинки або стрибати максимум через дві клітинки. Одне переміщення вважається одним кроком і виконується за однаковий час. Коли в результаті одного з кроків наші герої опиняються на одній клітинці, то Сірий Вовк ховається за кущ і пропускає на один хід уперед Червону Шапочку (на першому кроці також).

Визначити, чи зможуть наші герої дістатися до будиночка бабусі і якщо так, то хто першим добереться до нього за умови, що кожен обирає найкоротший шлях. Обчислити, скільки кроків зробить і скільки кроків пропустить кожний, та вказати шлях як послідовність клітинок.

Приклад.

Вхідна інформація:

залишається; найбільша глибина - 0

```
5 8                 {розмір схеми}
4 8                 {розташування будиночка бабусі}
1 1 1 0 0 0 0 0     {схема лісу}
0 1 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 1 1 0 1 0 1 1
0 0 1 0 1 0 0 1
```

вихідна інформація:

шлях Червоної Шапочки - 6 кроків

(1; 1); (1; 3); (2; 3); (4; 3); (4; 5); (4; 7); (4; 8)

шлях Сірого Вовка -

1) 4 кроки + 2 рази чекає

(чекає) (1; 1); (1; 3); (4; 3) + 1 (чекає); (4; 5); (4; 8)

2) 5 кроків + 1 раз чекає

(чекає) (1; 1); (1; 3); (2; 3); (4; 3); (4; 5); (4; 8)

Коментар. У другому варіанті відповідні герої прибувають одночасно, але Сірий Вовк чекає, тому Червона Шапочка перша.

749. **«Багатокутники».** Дано послідовність цілих чисел α_1, α_2, α_n ($0° < \alpha_i < 180°$), члени якої утворюють множину кутів. Визначити всі підмножини кутів, з яких можна утворити опуклі n-кутники (порядком кутів у n-кутнику можна знехтувати). Вивести кількість можливих варіантів.

Приклад.

Вхідна інформація:

```
6    {кількість кутів}
90 30 120 30 90 60
```

вихідна інформація:

ЗО 120 30 -трикутник
90 ЗО 60 - трикутник
90 120 90 60 - чотирикутник

750. **«Склад».** На складі встановлено контейнери для зберігання готової продукції. Кожний контейнер поділений на бокси, куди складається продукція. Звільнення й заповнення контейнерів відбувається таким чином: на замовлення (за датою виготовлення) видається партія продукції, і при цьому вивільняються відповідні бокси у контейнерах; для нової продукції послідовно (з 1-го до останнього контейнера) відшукуються і заповнюються вільні бокси. Якщо заповнилися вільні бокси в i-му контейнері, то переходять до пошуку вільних боксів у наступних контейнерах. Останній заповнюваний бокс може залишитися частково порожнім.

На складі ведеться журнал обліку наявності виготовленої продукції за такими параметрами:

дата виготовлення;

місце зберігання - $M_1, K_1, N_2, M_2, K_2, \quad N_l, M_l, K_l,$

де - номер контейнера, M_i - номер початкового боксу з даною продукцією, K_i - кількість зайнятих підряд боксів даною партією продукції в цьому контейнері ($i = 1, \quad l$).

Скласти програму, яка в діалоговому режимі вводила б інформацію про надходження й видачу продукції, вела облік наявності продукції на складі, видавала місце розташування продукції за датою виготовлення, таблицю розподілу вільних боксів по контейнерах і сумарну кількість вільного місця на складі (з урахуванням недозаповнених боксів).

Приклад.

Вхідна інформація:

4 10 10 {кількість контейнерів, кількість боксів у контейнерах, об'єм одного боксу}
1 1 5 {зайняті бокси}
10.04.98 1 6 1 2 3 5 4 1 2

вихідна інформація (діалог):

1 - надходження продукції
2 - видача продукції
3 - місце розташування продукції за датою
4 - таблиця розподілу вільного місця
5 - сумарна кількість вільного місця на складі
6 - завершення роботи
?4
1 контейнер: 7, 8, 9, 10 - бокси
2 контейнер: 1, 2, 8, 9, 10 - бокси
3 контейнер: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 - бокси
4 контейнер: 3, 4, 5, 6, 7, 8, 9, 10 - бокси

```
?1
дата?  23.01.99
об'єм?  115
?4
3 контейнер: 4, 5, 6, 7, 8, 9, 10 - бокси
4 контейнер: 3, 4, 5, 6, 7, 8, 9, 10 - бокси
?2
дата?  04.04.98
?4
1 контейнер: 1, 2, 3, 4, 5 - бокси
3 контейнер: 4, 5, 6, 7, 8, 9, 10 - бокси
4 контейнер: 3, 4, 5, 6, 7, 8, 9, 10 - бокси
?3
дата?  04.04.98
продукція відсутня
?5
205
?6
кінець програми
```

751. Задано натуральне число N. Знайти всі цілі додатні числа, що не перевищують N та діляться на кожну зі своїх цифр.

752. Визначити довжину й координати інтервалу з цілочисловими координатами кінців, на який припадають значення елементів дійсної послідовності $x_1, x_2, \ldots x_n$ ($n>1$, ціле).

753. Дано деякий текст і словник з N слів. Визначити, які слова і в якій кількості можна утворити з літер заданого тексту.

754. Повітряні кульки зчеплені мотузками в групи довільним чином. У кожній групі знаходиться більше як дві кульки. Всі групи кульок у свою чергу також зчеплені мотузками. Для того щоб кульки здійнялися в повітря, треба перерізати мотузки, які з'єднують групи.

Дано n кульок і m мотузок, що їх з'єднують. Усі кульки пронумеровані та відомі їх пари, що з'єднані між собою мотузками. Визначити номери кульок, між якими треба перерізати мотузки, щоб вони здійнялися в повітря.

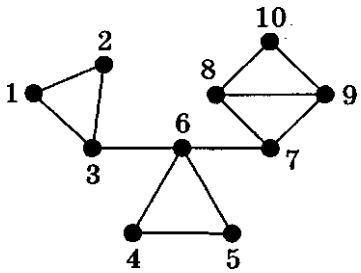
Приклад.

Вхідні дані:

```
10 {кількість кульок}
13 {кількість мотузок}
1  3      7  8    {пари з'єднань}
2  3      7  9
3  6      8  9
4  6      8  10
5  6      9  10
4  5      1  2
6  7
```

вихідні дані:

```
3 6
6 7
```



755. Дано квадратний масив A розміром n $(2 \leqslant n \leqslant 250)$, елементи якого дорівнюють 0, 1, 5 або 11. Підрахувати кількість четвірок *A[i,* y], *A[i* $+1$, y], *A[i, j* $+1$], *A[i* $+1$, y $+1$], в кожній з яких всі елементи різні.

756. Скласти програму обчислення значення $\sqrt{x}$ $(0 \leqslant x \leqslant 65\ 535)$ із заданою точністю *eps (eps* $\leqslant 1 \cdot 10^{10}$), скориставшись такою рекурентною послідовністю:

$$y_X = \frac{x+1}{\cdot} \qquad y_i = \frac{}{2}\ y_{i-1} + \frac{\cdot}{y_{i-}} \qquad (i = 2, 3, \ldots).$$

757. Задане натуральне число n $(n \leqslant 32767)$, кратне 3. Якщо скласти куби цифр заданого числа, то отримаємо нове. Застосуємо до отриманого числа таке саме перетворення. Цей процес продовжуватимемо доти, поки отримуване число не буде сталим. Визначити це стале число й порядковий номер перетворення, на якому воно отримане.

Вхідний файл number.in	*Вихідний файл* number.out
3	153 3

758. Нехай задано деяке натуральне число n $(n \leqslant 32\ 767)$. Кожне нове число утворюється додавання усіх цифр заданого. Цей процес продовжується доти, поки в результаті не утвориться однозначне число (цифра), яке називається коренем початкового заданого числа. Визначити корінь заданого натурального числа.

Вхідний файл root.in	*Вихідний файл* root.out
10	1

759. Сторони прямокутника задані натуральними числами *N та M* $(N, M \leqslant 65\ 535)$. Визначити, на скільки квадратів, сторони яких є натуральними числами, можна розрізати даний прямокутник, якщо від нього кожний раз відрізається квадрат максимальної площі.

Вхідний файл square.in	*Вихідний файл* square.out
3 6	2

760. Задано масив з N цілих чисел x_i, число M $(M \leqslant N,\ 1 \leqslant N,\ M \leqslant 64\ 000,\ 0 \leqslant x, \leqslant 255)$. Визначити мінімальне значення серед сум розташованих поруч M елементів масиву.

Вхідний файл sum.in	*Вихідний файл* sum.out
5 3 1 2 3 4 5 6	6

761. Задана послідовність цифр десяткової системи числення в кількості n $(n \leqslant 1\ 000\ 000)$. Кодом називається число, що не містить цифру 0 і цифри якого утворюють зростаючу послідовність. Визначити максимальний за значенням код.

Вхідний файл code.in	*Вихідний файл* code.out
12012	12

762. Для освітлення прямокутного майдану, де проходило Новорічне дійство, встановили потужний прожектор. Сторони майдану були паралельні осям координат, а площа поділена на сектори в кількості n секторів по горизонталі та m секторів по вертикалі ($3 \leqslant n,\ m \leqslant 100$). Кожний сектор мав подвійний номер, що складався з номера рядка по горизонталі та по вертикалі (нумерація починалася з північно-західного кута майдану). Промінь прожектора був спрямований перпендикулярно до північної сторони майдану і освітлював майдан під кутом 90°. Визначити координати двох точок на кордоні майдану, між якими знаходиться освітлена частина майдану, якщо відомо, що прожектор міститься в точці $(x; y)$ ($1 \leqslant x,\ y \leqslant 100$).

Вхідний файл light.in	*Вихідний файл* light.out
5 5 {розміри майдану} 3 3 {положення прожектора}	1 1 {перша освітлена точка} 1 5 {друга освітлена точка}

763. Задана квадратна матриця A порядку n ($1 \leqslant n \leqslant 8000$, $0 \leqslant A_i \leqslant 65\ 535$). Переставити рядки таким чином, щоб елементи в першому стовпчику були впорядковані за неспаданням. Вивести порядкові номери рядків, які в результаті утворюють необхідну послідовність.

Вхідний файл order.in	*Вихідний файл* order.out
3 2 3 3 2 1 2 3 1	1 3 2

764. Кожний звичайний дріб $\frac{m}{n}$ $(m < n)$ можна подати у вигляді ланцюгового дробу $\frac{m}{n} = \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots \cfrac{1}{a_k}}}$. Нехай задані натуральні числа $a_1, a_2, \ldots a_k$ $(a_i \leqslant 20,\ k \leqslant 20)$, що описують ланцюговий дріб. Визначити відповідні їм значення m і n.

Вхідний файл fraction.in	*Вихідний файл* fraction.out
1 2 2	5 7

765. «Лічилка». N ($N \leqslant 35\ 000$) дітей розташовані по колу. Відлік починають від першої дитини, надалі вилучають кожну k-ту дитину, зменшуючи при цьому коло, і продовжують лічбу. Визначити порядок вилучення дітей з кола, надрукувавши їх початкові порядкові номери.

Вхідний файл children.in	*Вихідний файл* children.out
3 1	1 2 3

766. На прямій лінії розташовано N $(1 \leqslant N \leqslant 15\ 000)$ відрізків, координати лівих і правих кінців яких задаються парою цілих чисел $(x_i; y_i)$, де $-32\ 768 \leqslant x_i, y_i \leqslant 32\ 767$, $i = 1, 2, \ldots, n$. Визначити найбільшу кількість та координати тих відрізків, що не перетинаються. Вважається, що відрізки перетинаються, якщо вони мають хоч одну спільну точку.

Вхідний файл segment.in	*Вихідний файл* segment.out
3 1 3 3 5 6 7	2 1 3 6 7

767. Для здійснення ефективної діяльності виробничі об'єднання повинні періодично проводити заміну обладнання, що використовується. При цій заміні враховуються прибуток від використання обладнання протягом одиниці часу, витрати, пов'язані з його зберіганням, вартістю закупівлі нового обладнання. Нехай на початку діяльності підприємства встановлено нове обладнання, що дає змогу на i-й рік виручити суму в R_i тисяч гривень, а щорічні витрати, пов'язані з його зберіганням, дорівнюють Z_i тисяч гривень $(0 < i \leqslant N)$. На момент заміни обладнання виручка становить R_0, а витрати на зберігання - Z_0. Витрати, пов'язані з придбанням та налагодженням нового обладнання, ідентичного встановленому, складають M тисяч гривень $(1 \leqslant R_i, Z_i, M \leqslant 200)$. З урахуванням усіх цих факторів знайти оптимальний план заміни обладнання, тобто план, що забезпечує максимальний прибуток від заміни обладнання протягом N років $(1 < N \leqslant 30)$, де 0 означає продовження роботи на старому обладнанні, а 1 - заміну на нове в i-й рік. Визначити також з урахуванням оптимального плану прибуток підприємства на кінець N-го року.

Вхідний файл change.in							*Вихідний файл* change.out
5	40					$\{N, M\}$	0 0 1 0 0
80	75	65	60	60	55	$\{R_i\}$	190
20	25	30	35	45	55	$\{Z_i\}$	

768. Декілька павуків сплели кожен своє павутиння. Кожен павук закріпив вузликом перетини ниток тільки свого павутиння. У деякі з вузликів попалися мухи-жертви. Вузлики всіх павутинь пронумеровані натуральними числами від 1 до N. M пар вузликів $(x_i; y_i)$ з'єднані між собою нитками павутиння $(1 \leqslant N, K \leqslant 65\ 535, 1 \leqslant M \leqslant 10\ 000)$, а в деяких з вузликів знаходиться K мух-жертв. Порядок задання номерів вузликів $x_i \Rightarrow y_i$ задає напрям руху павука по своєму павутинню до своїх мух-

жертв. Визначити (в порядку зростання) номери вузликів, з яких павуки почали плести своє павутиння, та номери вузликів (у порядку зростання), де знаходяться мухи-жертви кожного з них.

Вхідний файл spiders.in	
8 6 6	{N, M, K}
2 3 4 6 7 8	{номери вершин з мухами}
1 2	{пари вузликів $(x_i; y_i)$,}
1 3	{між якими є нитка павутиння}
1 4	
5 6	
5 7	
5 8	
Вихідний файл spiders.out	
1 5	{номери вершин з павуками}
1 2 3 4	{номери вершин з мухами для 1}
5 6 7 8	{номери вершин з мухами для 5}

769. Готуючись до Новорічної ялинки, Дід Мороз повинен був розкласти N ($1 \leqslant N \leqslant 30$) подарунків у два мішки. Для того щоб вміст мішків був максимально рівноцінним, він порахував кількість цукерок у кожному подарунку. Необхідно скласти програму, яка допоможе Дідові Морозу вирішити цю задачу і знайде один із можливих варіантів такого розкладу подарунків у два мішки, щоб модуль різниці між сумами цукерок у них був мінімальним. Вхідні дані містять у першому рядку інформацію про загальну кількість подарунків, у другому - послідовність кількості цукерок у кожному подарунку, заданих через пробіл. Вихідні дані містять два рядки із значеннями кількості цукерок у кожному подарунку першого і другого мішків, виведених через пробіл.

Вхідний файл dress.in	*Вихідний файл* dress.out
4	31 44
31 40 35 44	40 35

770. Одягаючись на Новорічне свято, Дід Мороз має надіти шкарпетки, штани, рукавиці, валянки, сорочку, шубу, шапку тощо. Заклопотаний старий дідусь забув послідовність, за якою йому слід надівати предмети свого гардероба. Нехай усі предмети гардероба Діда Мороза пронумеровані послідовністю натуральних чисел N ($1 \leqslant N \leqslant 100$). Послідовність одягання предметів гардероба задається M ($0 \leqslant M \leqslant N - 1$) парами чисел (наприклад, шкарпетки одягаються лише до валянок, а шапку можна надіти в будь-який момент). Допоможіть старенькому одягнутися, вказавши через пробіл послідовність одягання предметів гардероба за їх номерами. Для того щоб відповідь бу-

ла однозначною, одягайте на дідуся спочатку в порядку зростання ті предмети, від яких не залежать інші, потім наступні за зростанням їх порядкових номерів і т. д.

Вхідний файл dress.in	*Вихідний файл* dress.out
5 {кількість предметів} 4 {кількість зв'язних пар предметів} 1 2 {предмет 2 одягається після предмета 1} 1 4 2 4 3 4	1 3 5 2 4

771. Для визначення учасника змагань на Новорічному ранку в дитячому садочку Дід Мороз запропонував таку лічилку. Учасникам змагань дали номери, що утворюють послідовність натуральних чисел від n до m ($n < m < 10^{255}$). На першому кроці виходять із гри учасники, що стоять на непарних місцях. Потім з учасників, які залишились, виходять із гри всі ті, що стоять на парних місцях. Ці дії повторюються доти, поки не лишиться один учасник, що братиме участь у змаганнях. Визначити початковий номер учасника змагань, якщо вхідні дані містять n та m.

Вхідний файл count.in	*Вихідний файл* count.out
1 {n} 4 {m}	2

772. Для об'їзду всіх будинків по кільцевій дорозі Дід Мороз використав свій автомобіль. На кільцевій дорозі розташовано N заправок ($N \leqslant 1\ 000\ 000$). На кожній заправці є певна кількість пального. Загальної кількості пального вистачає для повного об'їзду дороги. Автомобіль має необмежений бак, який у початковий момент часу порожній. З деякої заправки починає рухатись автомобіль Діда Мороза, заправившись усім пальним з цієї заправки. Якщо йому вистачає пального, щоб доїхати до наступної заправки, то він заправляється всім пальним, що є на цій заправці, і т. д. Скласти програму, яка визначить заправку, починаючи з якої автомобіль проїде всю дорогу і повернеться до заправки, з якої він почав рух. Автомобіль рухається за годинниковою стрілкою.

Вхідні дані містяться у файлі car.in, де в першому рядку міститься кількість заправок, в другому рядку - довжина дороги. Починаючи з третього, в кожному наступному рядку міститься два числа: відстань у кілометрах від цієї заправки до першої заправки та кількість літрів бензину, що є на цій за-

правці. Вважатимемо, що одного літра пального вистачає на один кілометр руху.

Файл car.out повинен містити одне число, що відповідає номеру заправки, з якої треба починати рух.

Вхідний файл car.in	*Вихідний файл* car.out
6 12 0 1 2 1 4 1 6 4 8 2 10 3	4

773. Дід Мороз прибув на Землю для святкування Нового Року на космічному кораблі. Зореліт має квадратну форму і поділений на квадратні кімнати, згруповані в рядки і стовпчики з номерами від 1 до n ($n \leqslant 100$). Усі кімнати в межах одного стовпчика з'єднані між собою звичайними дверима, а для переміщення між різними стовпчиками існує система двосторонніх телепортаторів (із кімнати, яка знаходиться в i-му рядку та k-му стовпчику можна телепортуватися в кімнату на k-му рядку та i-му стовпчику і навпаки). Дід Мороз отримав перепустку першого рівня, яка дає йому змогу використовувати кожен телепортатор лише один раз, після чого він деактивується разом із тим телепортатором, на який перемістився Дід Мороз. Допоможіть дідусеві використати всі наявні телепортатори, якщо відомі розмір корабля та його карта (0 - немає телепортатора, 1 - є телепортатор). Відповіддю має бути послідовність номерів стовпчиків, розділених пробілами, якими слід переміщатися Дідові Морозу, або число 0, якщо він не може використати всі телепортатори.

Вхідний файл teleport.in	*Вихідний файл* teleport.out
4 0 0 1 0 0 0 0 1 1 0 0 1 0 1 1 0	2 4 3 1

774. **«Освітлення».** Деяке багатокімнатне приміщення освітлює m світильників. Вони перемикаються за допомогою пристрою керування, що складається з n перемикачів, кожний з яких під'єднаний до всіх світильників. Початковий стан усіх світильників задається послідовністю двійкових цифр a_i ($i = 1, 2, 3, \ldots m$), де 0 - вимкнені світильники, а 1 - ті,

що світяться. Принцип роботи кожного перемикача задається послідовністю двійкових цифр b_{ij} ($i = 1, 2, 3, \ldots n$; $j = 1, 2, 3, \ldots m$), де 1 – світильники, які даний перемикач переводить у протилежний стан, 0 - лишає без змін. Якою найкоротшою послідовністю перемикачів треба скористатися, щоб вимкнути світло в усьому приміщенні? Якщо така послідовність відсутня, то повідомити про це.

Вхідний файл ligth.in		*Вихідний файл* ligth.out
6 3	{m n}	1 3
0 1 1 0 1 0	{a}	
0 1 0 0 1 0		
0 0 0 1 0 0	{k_2}	
0 0 1 0 0 0	{k_3}	

775. Для нумерації N ($N \leqslant 255$) палат у деякому лікувальному закладі використовувався однаковий набір цифр a_i, де $a_i \in [0, 1, 2, \ldots 9]$. Отримані номери були надані палатам у порядку їх зростання. Які номери були на K палатах, розташованих до ординаторської, і на L палатах - після неї ($1 \leqslant K < L \leqslant N$), якщо на самій ординаторській стояв номер $a_{i1}a_{i2}a_{i3} \ldots a_{in}$ де ij - порядкові номери цифр із заданого набору.

Вхідний файл number.in		*Вихідний файл* number.out
5	{N}	132
1 2 3	{набір цифр для нумерації}	231
213	{номер ординаторської }	
1	{K}	
1	{L}	

776. Після оцифрування відсканованої схеми виявилося, що вона складається з декількох деталей, які, у свою чергу, утворюються за допомогою горизонтальних смуг різної довжини. Одну деталь утворюють смуги, що мають між собою хоча б по одному спільному елементу з однаковою вертикальною координатою (тобто торкаються одна одної). Всі елементи деталей схеми позначені 1, решта - 0, лівий верхній кут схеми має координати (1; 1). Розміри схеми задаються параметрами $n \times m$ ($n \leqslant 150$, $m \leqslant 150$). Навколо кожної деталі схеми можна побудувати прямокутник, сторони якого паралельні осям координат і проходять по її зовнішніх елементах, і який не містить жодної іншої деталі схеми. Визначити координати лівого верхнього і правого нижнього кутів найбільшого за площею прямокутника, що утвориться після виділення елементів схеми вказаним методом. Якщо таких прямокутників декілька, то визначити всі.

Вхідний файл rectangle.in	*Вихідний файл* rectangle.out
4 6 0 0 0 0 0 1 0 1 1 0 1 1 0 0 0 1 0 0 0 0 0 1 1 0	1 5 2 6 3 4 4 5

777. За допомогою зубчастої передачі, що складається з n ($1 < n \leqslant 100$) послідовно з'єднаних шестерень різних діаметрів $d_1, d_2, \quad d_n$ ($1 \leqslant d_i \leqslant 255$), через спеціальні пристрої приводяться в рух поршні, що перекачують воду з верхніх циліндричних резервуарів у нижні і навпаки. При цьому кожна пара резервуарів верхнього і нижнього рівнів, які за умовою мають однакову висоту, з'єднана з однією шестірнею зубчастої передачі. Принцип перекачування води в спарених резервуарах побудований на тому, що коли вода з одного з них буде повністю перекачана в другий, то пристрій перемикає перекачування води у зворотному напрямі. Зубчаста передача приводиться в рух обертанням першої шестірні зі швидкістю k обертів в одиницю часу. Решта шестерень обертається за рахунок обертання першої, а їх рух, у свою чергу, спричиняє рух приєднаних поршнів та перекачування води в резервуарах. Швидкість руху поршнів пропорційна швидкості обертання шестерень: одиниця висоти циліндричного резервуара дорівнює одному оберту шестірні.

Визначити, в якому порядку слід з'єднати шестірні в зубчастій передачі, щоб через t ($1 \leqslant t \leqslant 100$) одиниць часу у верхніх резервуарах знаходилася найбільша сумарна кількість води, якщо відомо, що на початку заповнені водою лише верхні резервуари висотою $h_1, h_2, \quad h_n$ ($h_i \leqslant 255$).

Вхідний файл reserv.in	*Вихідний файл* reserv.out
35 1 {n, t, k} 1 1 1 {діаметри шестерень} 2 2 2 {висоти спарених резервуарів}	3 1 2 3

***

Перш ніж перегорнути останню сторінку підручника, підіб'ємо підсумок усьому сказаному.

Отриманий досвід і набуті навички дозволяють говорити мовою професіоналів, а деяка нова термінологія з гідністю поповнить ваш програмістський словник. Цінуйте будь-яку корисну інформацію і вмійте нею скористатися.

Розділ V

ІСТОРІЯ РОЗВИТКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Перші кроки розвитку програмного забезпечення

У 40-х роках XX ст. почала розвиватися нова наука - кібернетика, батьком якої став американський учений Норберт Вінер. У 1948 році він уперше опублікував свою працю «Кібернетика», в якій визначив основні положення цієї науки, а саме науки про загальні закони одержання, зберігання, передачі й обробки інформації. Основними технічними засобами для розв'язування задач кібернетики є електронно-обчислювальні машини, роботи над конструюванням яких ішли у цей самий період.

На жаль, на той час у Радянському Союзі кібернетика, так само як і генетика, була проголошена псевдонаукою. Саме тому поява першої вітчизняної електронно-обчислювальної машини затрималася майже на десять років порівняно з американською. Слід відзначити, що роботи над ЕОМ в Америці та Радянському Союзі на той час ішли незалежно одна від одної. Пізніше, враховуючи суттєве запізнення в розробці нових типів ЕОМ, була прийнята стратегія адаптації нових закордонних ідей у наших вітчизняних ЕОМ для того, щоб швидше ліквідувати відставання. Але зрозуміло, що його неможливо було подолати, оскільки вдавалося «позичити» лише ті ідеї, що вже побачили світ. «Головне», чого було досягнуто, так це того, що власні розробки були припинені. Але історія лишається такою, якою її зробили ми самі, тому повернемося до фактів.

Усі ЕОМ, що були створені за весь час їх розвитку, поділяються на покоління. Разом із розвитком комп'ютерної техніки розвивалося і програмне забезпечення. Адже якими б досконалими не були фізичні пристрої, комп'ютер буде німий без необхідних програм. Саме програми надихають комп'ютер на роздуми, аналіз, вибір рішення. Неосяжний набір програмних продуктів дозволяє розгорнутися людській фантазії до невидимих вершин. Дивлячись на комп'ютер, ми в першу чергу сприймаємо його як набір фізичних пристроїв, а потім уже як пристрій, який уміє рахувати, грати, малювати. Тому слід розглядати комп'ютер з таких двох позицій:

- *технічні засоби* (Hardware - «твердий» товар) - узагальнююче поняття для всіх фізичних частин комп'ютера, в тому числі і додаткових пристроїв, кабелів;

- *програмне забезпечення* (Software - «м'який» товар) - операційна система, сервісні та прикладні програми.

Цікаво, що співвідношення між вартістю апаратних та програмних засобів не завжди говорило на користь других. Можна навести таку таблицю динаміки змін відносної вартості програмних і апаратних засобів по роках:

50-ті роки: 30 % — програмне забезпечення,
70 % - апаратні засоби;
60-ті роки: 65 % - програмне забезпечення,
35 % - апаратні засоби;
70-ті роки: 80 % - програмне забезпечення,
20 % - апаратні засоби;
80-ті роки: 85 % - програмне забезпечення,
15 % - апаратні засоби.

Програми для машин *першого покоління* (40-50-ті роки XX ст.) писалися мовою машинних кодів. При цьому програміст сам розподіляв комірки пам'яті під програму, вхідні дані й отримані результати. Писати програми машинною мовою було досить складно. Ці програми були дуже громіздкими, складними і ймовірність помилок у них була дуже великою. Тому на початку 50-х років була створена спеціальна система позначень - команд, яку назвали мовою Assembler. Для перекладу програми, написаної мовою Assembler, на мову машини складалися таблиці відповідності команд Assembler й їх машинних кодів. Спочатку такий переклад здійснювався вручну, а згодом цей процес автоматизували та доручили виконувати самій ЕОМ. У наш час мовою Assembler пишуть програми у разі необхідності ефективного використання всіх машинних ресурсів і досягнення високої швидкості виконання програм.

Наприкінці 50-х років з'явилися нові засоби спілкування з обчислювальною машиною - мови програмування високого рівня. Саме в цей час стали розроблятися методи, що полегшували спілкування користувачів з ЕОМ, велися розробки перших мов програмування, стандартних підпрограм, бібліотек програм з інструкціями по їх використанню тощо. Саме на цей час припадає розробка таких мов програмування, як Algol (Algorithmic Language), що використовувалася при програмуванні обчислювальних задач у математиці, фізиці, та Fortran (Formula Translation), що був орієнтований на створення програм проведення розрахунків при розв'язуванні задач, які використовували математичне моделювання.

ЕОМ першого покоління працювали в однопрограмному режимі, при цьому автори програм - програмісти - мали доступ до обчислювальної системи і керували нею з пульта управління ЕОМ. Таким чином, процес роботи користувачів-програмістів, якими були переважно інженери-електронщики, за пультом ЕОМ був послідовним. Для їхньої роботи відводився певний час за графіком. У подальшому таке керування перейшло до оператора - людини, що мала спеціальну професійну підготовку. З часом з'явилися спеціальні керуючі програми, що отримали назву *пакетні монітори,* призначенням яких було керування потоком програм, що збиралися у пакет. Програми набивалися на перфокарти, з яких формувався пакет завдань. Кожна програма повинна була починатися перфокартою, що містила інформацію про назву програми, вимоги до ресурсів для її виконання, і закінчуватися перфокартою, що містила ознаку кінця програми. Монітор здійснював уведення інформації з перфокарт чергової програми і передавав керування на її початок. Після завершення роботи програми монітор знову брав керування на себе і здійснював уведення наступної програми. Після завершення роботи всіх програм пакету монітор видавав повідомлення оператору. Під час роботи ЕОМ оператор стежив за її роботою і втручався у випадку зациклення програм, а також виконання ними несанкціонованих дій. Оскільки машини першого покоління не мали системи переривань, то виконувані програми могли пошкодити і саму керуючу програму монітора.

У період, на який припадає розвиток ЕОМ *другого покоління* (50-60-ті роки XX ст.), з'являються нові мови програмування високого рівня: Cobol (Common Business Oriented Language), що був орієнтований на обробку економічної інформації, Lisp (List Processing), який у наш час широко використовується для створення систем машинного інтелекту, та Basic (Beginner's All-purpose Symbolic Instruction Code), який використовується для написання програм розв'язування прикладних задач інженерами і популярність якого серед непрофесійних користувачів персональних комп'ютерів на сьогодні незаперечна.

Щодо організації роботи на ЕОМ на цей час, то слід сказати, що у програмістів з'явилася можливість спілкування з машиною не тільки в однопрограмному, а й у мультипрограмному режимі. Цю можливість спричинила поява переривань і захисту пам'яті в ЕОМ другого покоління. Керування потоком програм і ресурсами ЕОМ тепер виконується за допомогою програм *супервізорів* (supervisor - наглядач), які мали значно більші можливості, ніж пакетні монітори. У мультипрограмному режимі в пам'ять завантажується декілька задач із пакета, які нібито од-

ночасно виконуються. Насправді ця ілюзія одночасності роботи може бути реалізована завдяки неодночасному використанню різними програмами одних і тих самих ресурсів машини. Наприклад, одна програма виконує зовсім невеликі розрахунки, але виводить дуже багато інформації на друк, займаючи при цьому переважно пристрої виведення, а друга програма в основному виконує лише складні розрахунки, максимально використовуючи процесор і отримуючи невеликий числовий результат. У функції мультипрограмного режиму роботи машини входив постійний підбір з пакета саме тих програм, які повинні виконуватись одночасно. Від правильного вибору таких програм залежала ефективність використання обладнання машини. Протягом роботи машини в мультипрограмному режимі цей підбір програм постійно поповнювався з пакету програм, що формувався оператором на пристрої введення.

З часом функцій супервізора, який керував роботою машин другого покоління, стало недостатньо. Виникла необхідність створення цілого комплексу програм, пов'язаних у єдину систему, що отримала назву *операційна система.* Найбільш повний розвиток операційні системи отримали саме в машинах *третього покоління* (60-70-ті роки XX ст.). Крім цього, розвинувся режим колективного користування. Так само, як і в мультипрограмному режимі, у режимі колективного користування на ЕОМ одночасно обробляється декілька програм. Але різниця полягає в тому, що з кожною з таких програм у режимі діалогу працює окремий користувач. Це досягається за рахунок того, що спілкування користувачів із машиною організовується зразу з декількох терміналів, підключених до ЕОМ. Операційна система організовує одночасне обслуговування всіх користувачів, виділяючи кожному терміналу визначений квант часу роботи процесора. Насправді термінали обслуговуються послідовно, але швидкодія ЕОМ дозволяє забезпечити мінімальну затримку при виконанні замовлення, що поступає з кожного термінала. Тому в користувача створюється повна ілюзія того, що він один має в розпорядженні всі ресурси машини. Для ще ефективнішого використання ресурсів ЕОМ режим колективного доступу суміщається з режимом пакетної обробки завдань.

У цей час стали інтенсивно розвиватися мови програмування високого рівня. У 70-х роках з'являється нова мова програмування PL-1 (Programs Language), яка використовується для розв'язування великого класу наукових задач і обробки економічної інформації. З машинами третього покоління пов'язана ще одна важлива подія - користувач отримав можливість користуватися не лише цифровою інформацією, а й графічною.

Четверте покоління ЕОМ, що розпочалося в 70-ті роки XX ст., відзначилося великою подією в історії розвитку електронно-обчислювальної техніки. Влітку 1976 року сусіди шановного Пола Джобса, що мешкав на околиці невеличкого міста Лос-Альтос штату Каліфорнія, навіть не здогадувалися, що в його гаражі народжується чудо комп'ютерної техніки - перший персональний комп'ютер. Його 21-річний син Стівен Джобс і його товариш 25-річний Стів Возняк працювали над складанням перших комп'ютерів Apple. Лише через п'ять років фірма Apple стала легендою, її річний обіг коштів досяг 335 мільйонів доларів на рік. На сьогодні це одна з провідних фірм. Усьому світові відомі моделі комп'ютерів цієї фірми Apple-I, Apple-II, Liza, Macintosh, що одержали назви від різних сортів яблук в Америці.

У цей же період удосконалюється програмне забезпечення, інтенсивно розвиваються автоматизовані інформаційно-пошукові системи, бази даних, починають вести розмову про банки знань. З'являються такі відомі і популярні на сьогодні мови програмування, як C, Pascal, Ada. Характеризуючи ці мови програмування, можна відзначити, що якщо в мові Pascal особливе значення надається надійності, то в мові C — гнучкості. У свою чергу мова Ada є результатом довготривалої дискусії серед спеціалістів стосовно того, якою повинна бути сучасна мова програмування загального призначення. Таким чином ця мова служить відображенням більш пізньої точки зору спеціалістів на мови програмування.

Масове розповсюдження персональних комп'ютерів привело до створення програмних засобів (операційних систем, пакетів прикладних програм і т. ін.), які дозволили спілкуватися з комп'ютером широкому колу користувачів. Графічний інтерфейс персональних комп'ютерів зробив їх доступними будь-кому. З'явилися нові версії операційних систем для IBM-сумісних комп'ютерів: MS-DOS, UNIX, OS/2, WINDOWS, LINUX.

Сучасні операційні системи дозволяють працювати користувачам персональних комп'ютерів у мультипрограмному режимі, тобто одночасно розв'язувати на комп'ютері декілька задач. Мультимедійні можливості персональних комп'ютерів, а саме наявність потужних відео- та звукових карт, дозволяють реалізовувати користувачам самі шалені бажання: переглядати кінофільми та слухати музику, записану на компакт-дисках, користуватися мультимедійними енциклопедіями, що використовують відео- та звукові ефекти, самим створювати подібні програми, використовуючи сучасні об'єктно-орієнтовані середовища програмування Visual Basic, Delphi, Builder.

На сьогоднішній день вважається, що програмне й апаратне забезпечення комп'ютерів поновлюються кожні щопівроку. Тому давайте помріємо про те, якими будуть комп'ютери та програмне забезпечення для них уже найближчим часом.

Сучасні технології програмування

Розглянемо *метод покрокової деталізації*, який застосовується для створення складних алгоритмів.

Створивши вже не один алгоритм, ви, напевно, саме таким чином доводили його до остаточного «ідеального» варіанта. Нам тільки залишилося все назвати своїми іменами.

При покроковій деталізації на кожному етапі складання алгоритму формуються окремі його частини, що є актуальними на даний момент, а решта заміняється, наприклад, текстовим варіантом опису. На наступних етапах така сама методика застосовується до наступної групи фрагментів алгоритму. Такий підхід приводить до поступової деталізації алгоритму, уточнення як виконуваної, так і інформаційної його структури.

Розглянемо вже знайомий нам приклад алгоритму Евкліда. На першому кроці покрокової деталізації сформуємо дію повторення, яка дасть змогу визначити результат алгоритму:

```
while n <> m do
    (заміняємо одне з чисел на відповідну різницю цих чисел);.
```

На наступному кроці оформимо в термінах Pascal-програми вибір числа, яке необхідно замінити відповідною різницею:

```
while n <> m do
   if n > m
      then (замінюємо число n)
      else (замінюємо число m);
```

Тепер залишилося зробити останній крок у деталізації нашого алгоритму - записати дії заміни відповідних значень *n* та *m*:

```
while n <> m do
   if n > m
      then n := n - m
      else m := m - n;
```

Звичайно, наведений приклад є одним з найпростіших, на яких продемонстровано принцип покрокової деталізації.

Зусилля щодо покращання якості алгоритмів, ефективності праці програмістів знайшли реалізацію в структурному методі побудови алгоритмів.

Структурний підхід до побудови алгоритмів - це спосіб створення алгоритму з широким використанням допоміжних алгоритмів.

У складних і громіздких задачах такий підхід дає змогу розбити алгоритм на окремі частини - модулі, кожен з яких розв'язує свою самостійну підзадачу. Це дає можливість сконцентрувати зусилля на розв'язуванні кожної підзадачі, яка реалізується у вигляді окремої процедури. Для кожного такого модуля визначаються свої методи реалізації алгоритму та структура даних, якими він оперує.

Останнім кроком у модульній побудові алгоритмів є об'єднання окремих модулів у єдине ціле. Для цього між модулями повинні встановлюватися зв'язки для передачі інформації від одних модулів до інших: результати виконання одних модулів є вхідною інформацією для інших.

Основною метою структурного підходу до побудови алгоритмів є розробка алгоритму з мінімальними взаємозв'язками між його модулями.

Якщо поставлена перед вами задача є серйозним великим завданням, то важливість розбиття алгоритму на окремі модулі неможливо недооцінити. Особливо ефективна така методика під час розробки складних алгоритмічних систем колективами програмістів. У цьому разі кожному розробнику дається своя цілісна частина алгоритму, яка реалізується ним у вигляді окремого модуля. Завершенням колективної роботи є зведення цих модулів у єдине ціле.

А тепер розглянемо глобальніші питання, які постають при масовому застосуванні комп'ютерів. Це, в першу чергу, розробка нового програмного забезпечення.

Сьогодні вже можна говорити про сформовану індустрію виготовлення програм. Для цієї мети створено різноманітні *технології програмування.* Ознайомимося з найпоширенішими з них.

Низхідне програмування, або програмування «зверху-донизу», базується на ідеї поступової деталізації задачі на низку окремих підзадач. Спочатку глобальна задача представляється у вигляді набору підзадач і будується програма, в якій ці підзадачі виступають як деякі іменовані процедури, до яких можна організувати звернення. Ті підзадачі, які ще програмно не реалізовані, тимчасово заміняються так званими програмними заглушками, що дає змогу отримати перший варіант програми, готової для першого кроку налагодження і пошуку поліпшених варіантів. Після цього такий самий прийом застосовується до підзадач.

Висхідне програмування, або програмування «знизу-вгору», засноване на протилежному процесі. Спочатку пишуться і налагоджуються програми найнижчого рівня, а потім поступово з них збираються крупніші блоки. Цей процес завершується тоді, коли вся програма буде повністю зібрана і налагоджена.

З часом з'явилася ще одна технологія програмування, що дістала назву технології ***пакетів прикладних програм.*** Ідея її полягає в тому, що кожна нова задача розв'язується шляхом комбінації пакетів програм, якими є готові налагоджені автономні блоки.

Пакетна технологія знайшла своє продовження і розвиток в ***об'єктно-орієнтованому програмуванні.*** Об'єктно-орієнтовані засоби являють собою інструментарії мови дуже високого рівня, побудовані з інтелектуальних модулів, тобто таких, які виконують певні змістовні функції. Існують інтелектуальні модулі обслуговування екрана, формування меню, підготовки відповідей, контролю помилок при введенні даних користувачем і т. ін. Крім того, в інтелектуальні модулі можуть входити об'єкти, створені самим користувачем або програмістом для подальшої їх обробки і представлення. Наприклад, поширеність програмних продуктів фірми Microsoft привело до певної стандартизації інтерфейсних засобів прикладних програм у вигляді вікон, ієрархічної структури меню користувача тощо.

На сьогодні розроблені й продовжують розроблятися нові об'єктно-орієнтовані інструментальні засоби - середовища, в яких реалізовані мови програмування, що, в свою чергу, містять можливості об'єктного програмування. Методика об'єктно-орієнтованого програмування лежить в основі розробки нових прикладних систем та додатків. Зараз майже кожна традиційна мова програмування розробляється з підмножиною або надмножиною засобів підтримки об'єктного програмування. Це, наприклад, такі нові мови програмування і середовища, як Visual Basic, Visual C++, Visual FoxPro, середовища Delphi, Power Builder.

Класифікація обчислювальних машин

Поява та широке використання персональних комп'ютерів і різноманітних операційних систем спонукала до більш широкого діапазону їх використання, застосування для розв'язування різних типів задач. Саме ці основні моменти характеризують поділ сучасних комп'ютерів на такі класи.

Персональний комп'ютер (ПК) зобов'язаний своїм існуванням мікропроцесору. Ці комп'ютери, вартістю від декількох сотень до декількох тисяч доларів, здатні зберігати і обробляти досить великі об'єми інформації, вести серйозні математичні розрахунки. Персональний комп'ютер можна сьогодні побачити не тільки в офісах, учбових закладах, але й у багатьох вдома.

Основними фірмами-виробниками персональних комп'ютерів є фірма IBM і фірма Apple. Отже, можна говорити про існування двох основних технологічних напрямів по виробництву комп'ютерної техніки. Саме за цією ознакою персональні комп'ютери на сьогоднішній день поділяються таким чином: комп'ютери IBM, IBM-сумісні комп'ютери, що випускаються за технологією фірми IBM різними фірмами по всьому світу, та комп'ютери Apple.

Однак є ще й інші ознаки, що спонукають поділ комп'ютерної техніки на класи в залежності від її застосування. Можливості сучасних персональних комп'ютерів настільки великі, що межі між цими класами стають все більш розмитими, і, подекуди, буває дуже важко визначити, до якого класу відноситься той або інший комп'ютер.

Поява і розвиток операційної системи UNIX дозволили об'єднувати персональні комп'ютери в локальні мережі. Об'єднання комп'ютерів у локальну мережу вимагає наявності потужного «головного» комп'ютера, до якого під'єднані решта робочих місць. Такі комп'ютери отримали назву *сервери*. Прикладні комерційні та бізнес-системи, що передбачають роботу з базами даних, потужними видавничими системами, мережевими системами та системами обслуговування комунікацій, вимагають переходу до моделі обчислень «кліснт-сервер». У такому варіанті частину роботи може виконувати сервер, а частину - комп'ютер користувача цієї мережі.

Залежно від інформації, яка міститься на сервері, вони поділяються на такі типи: файл-сервер, сервер бази даних, принт-сервер, обчислювальний сервер, сервер додатків. Таким чином, тип сервера визначається типом ресурсу, яким він володіє - файлова система, база даних, принтери, процесори, прикладні пакети програм.

З іншого боку існує класифікація серверів, яка визначається масштабом мережі, в якій вони використовуються: сервер робочої групи, сервер відділу, сервер масштабу підприємства (корпоративний сервер). Оскільки кількісне поняття групи та відділу для різних підприємств різне, то такий поділ є досить умовним.

Використання локальних мереж і наявність у них робочих місць, що працюють під керуванням сервера, привело до виділення такого класу комп'ютерів, як *робочі станції*. За технічними якостями робочі станції можуть значно поступатися серверам, оскільки саме ті беруть на себе основне навантаження під час сеансу роботи. Прабатьками робочих станцій були «міні-комп'ютери». Початкова орієнтація робочих станцій на професіональних користувачів спонукала до того, що ці комп'ю-

тери на сьогодні є прекрасно збалансованими системами, в яких висока швидкодія суміщається з великим об'ємом оперативної та зовнішньої пам'яті, високоякісною та швидкодіючою графічною підсистемою та різноманітними пристроями введення/виведення.

Найновітніші технології виробництва ПК і розробка так званих робочих станцій «початкового рівня» останнім часом значно зблизила ці два класи комп'ютерів, що привело до появи нового поняття - «персональна робоча станція».

Робочі станції використовуються в невеликих організаціях, таких як дослідницькі лабораторії й офіси підприємств, і розраховані на використання окремими користувачами і невеликими групами.

Мейнфрейм — це синонім поняття «велика універсальна ЕОМ». Мейнфрейми і до сьогоднішнього дня вважаються найпотужнішими (не рахуючи суперкомп'ютерів, про які мова піде далі) обчислювальними системами загального призначення, що забезпечують безперервний цілодобовий режим експлуатації. Вони можуть містити один або декілька процесорів, кожний із яких може в свою чергу мати сопроцесор, тобто прискорювач операцій із дробовими числами. У нашій уяві мейнфрейми все ще асоціюються з великими за габаритами машинами, що вимагають спеціально обладнаних приміщень з системами водяного охолодження і кондиціювання. Однак це не зовсім так. Прогрес у області елементно-конструкторської бази дозволив суттєво скоротити габарити основних пристроїв. Поряд із надпотужними мейнфреймами, що вимагають організації складної водяної системи охолодження, існують менш потужні моделі, для охолодження яких достатньо повітряної вентиляції, та моделі, побудовані за блочно-модульним принципом, що не вимагають спеціальних приміщень і кондиціонерів.

Основними виробниками мейнфреймів є відомі комп'ютерні компанії IBM, Amdahl, ICL, Siemens Nixdorf та деякі інші, але провідна роль належить все ж таки компанії IBM. Саме архітектура системи IBM/360, що була випущена в 1964 році, і її наступні покоління стали зразком для наслідування. У СРСР протягом багатьох років випускалися машини ряду «ЕС ЭВМ», що були вітчизняним аналогом цієї системи.

Але все ж таки потужнішими на сьогоднішній день залишаються *суперкомп'ютери*. Їх відрізняє швидкодія, складність конструкції та висока вартість. Вони можуть виконувати мільярди операцій за секунду та зберігати величезні об'єми інформації. Підвищення швидкодії в них досягнуто за рахунок як

використання декількох процесорів, що працюють паралельно, так і більш швидкодіючих інтегральних мікросхем на основі нових матеріалів. Але основне, що виділяє суперкомп'ютери серед інших у окремий клас, - це задачі, які вони виконують. Суперкомп'ютери використовуються для керування різними найскладнішими системами, для швидкого розв'язування таких комплексних задач, як, наприклад, передбачення погоди або моделювання складних хімічних і біологічних процесів. На сьогоднішній день у світі налічується трохи більше десятка суперкомп'ютерів.

***

Якщо вам вистачило терпіння прочитати посібник від початку до кінця і ви при цьому не заглядали, скільки сторінок вам ще залишилося, якщо ви відшукали в ньому корисну інформацію і збираєтеся повертатися до неї ще не раз - значить не дарма витрачено стільки часу. Для поглиблення своїх знань і навичок в алгоритмізації та програмуванні ви можете звертатися як до наукових робіт класиків, так і до сучасної комп'ютерної літератури.

Перш ніж перегорнути останню сторінку підручника, підіб'ємо підсумок усьому сказаному.

Отриманий досвід і набуті навички дозволяють говорити мовою професіоналів, а деяка нова термінологія з гідністю поповнить ваш програмістський словник. Цінуйте будь-яку корисну інформацію і вмійте нею скористатися.

ДОДАТКИ

«Гарячі клавіші» середовища Turbo Pascal 7.0

«Гаряча клавіша»	Дія
F1	Одержання довідки (допомоги)
Ctrl+F1	Активізація довідки про оператор, на який указує курсор
F2	Збереження у файлі тексту з активного вікна
Ctrl+F2	Закриття всіх відкритих програмою файлів і встановлення програмного лічильника на початок програми
F3	Відкриття нового вікна і завантаження в нього вказаного файла
Alt+F3	Закриття активного вікна
F4	Запуск на виконання програми до позиції курсора
Ctrl+F4	Аналіз і зміна значень змінних
F5	Збільшення/зменшення розміру активного вікна
Alt+F5	Перемикання на екран користувача
Ctrl+F5	Зміна положення і розмірів активного вікна
F6	Перехід до наступного вікна
Shift+F6	Перехід до попереднього вікна
Alt+цифра	Перехід до вікна з указаним номером
F7	Виконання програми пооператорно з пооператорним виконанням усіх підпрограм
F8	Виконання програми пооператорно з виконанням усіх підпрограм без операторної деталізації
Ctrl+F7	Доповнення списку змінних у Watch-вікні
Ctrl+F8	Встановлення/скасування контрольної точки на рядку програми, що вказана курсором
Alt+F9	Компіляція програми з активного вікна
Ctrl+F9	Компіляція і запуск програми на виконання з активного вікна
F10	Активізація головного меню
Ctrl+Y	Вилучення рядка, на який указує курсор
Ctrl+N	Вставлення нового рядка після того, на який указує курсор
Shift+стрілки	Розширення блоку, що відмічається, від положення курсора в напрямі стрілки

Розширення блоку, що відмічається, від положення курсора до кінця рядка
Розширення блоку, що відмічається, від положення курсора до початку рядка
Вказання початку блоку, що відмічається
Вказання кінця блоку, що відмічається
Зняття/відновлення відміченого блоку
Копіювання відміченого блоку в те місце, де встановлено курсор
Перенесення відміченого блоку в те місце, де встановлено курсор
Знищення відміченого блоку
Копіювання відміченого блоку в буфер проміжного зберігання
Перенесення відміченого блоку в буфер проміжного зберігання
Копіювання відміченого блоку з буфера проміжного зберігання в те місце, де встановлено курсор
Завершення сеансу роботи інтегрованого середовища Turbo Pascal із збереженням (у разі підтвердження) змінених файлів

Основні помилки компіляції середовища Turbo Pascal 7.0

1 Out of memory - вихід за межі пам'яті
2 Identifier expected - очікується ідентифікатор
3 Unknown identifier - невідомий ідентифікатор
4 Duplicate identifier - повторний опис ідентифікатора
5 Syntax error - синтаксична помилка
6 Error in real constant - помилка в дійсній константі
7 Error in integer constant - помилка в цілій константі
8 String constant exseeds line - рядкова константа перевищує розміри рядка
9 Too many nested files - забагато вкладених файлів
10 Unexpected end of file - несподіваний кінець файла
11 Line too long - рядок надто довгий
12 Type identifier expected - очікується ідентифікатор типу
13 Too many open files - надто багато відкритих файлів
14 Invalid file name - помилкове ім'я файла
15 File not found - файл не знайдено
16 Disk full - диск заповнений
17 Invalid compiler directive - невірна директива або ключ компілятора
18 Too many files - надто багато файлів
19 Undefined type in pointer definition - невизначений тип у визначенні посилання
20 Variable identifier expected - потрібен ідентифікатор змінної
21 Error in type - помилка у визначенні типу
22 Structure too lange - надто велика структура
23 Set base type out of range - базовий тип множини порушує дозволені межі
24 File components may not be files or objects - компоненти файла не можуть бути файлами або об'єктами
25 Invalid string length - неправильна довжина рядка
26 Type mismatch - невідповідність типів
27 Invalid subrange base type - неправильний базовий тип для діапазону
28 Lower bound > then upper bound - нижня межа більша за верхню
29 Ordinal type expected - потрібен зчислений тип
30 Integer constant expected - очікується ціла константа
31 Constant expected - очікується константа
32 Integer or real constant expected - очікується ціла або дійсна константа
33 Pointer type identifier expected - очікується ім'я типу «покажчик»
34 Invalid function result type - неправильний тип результату функції
35 Label identifier expected - потрібен ідентифікатор мітки
36 BEGIN expected - очікується **BEGIN**
37 END expected - очікується **END**
38 Integer expression expected - очікується вираз цілого типу
39 Ordinal expression expected - очікується вираз зчисленого типу
40 Boolean expression expected - очікується логічний вираз
41 Operand types do not match operator - типи операндів не відповідають оператору
42 Error in expression - помилка у виразі

43 Illegal assignment - хибне присвоєння
44 Field identifier expected - очікується ім'я поля запису
45 Object file too large - об'єктний файл занадто великий
46 Undefined EXTERN - невизначена зовнішня процедура
47 Invalid object file record - неправильний запис об'єктного файла
48 Code segment too large - сегмент коду надто великий
49 Data segment too large - сегмент даних надто великий
50 DO expected - очікується слово **DO**
51 Invalid PUBLIC definition - неправильне визначення **PUBLIC**
52 Invalid EXTRN definition - неправильне визначення **EXTRN**
53 Too many EXTRN definition - надто багато визначень типу **EXTRN**
54 OF expected - очікується слово **OF**
55 INTERFACE expected - очікується інтерфейсна секція
56 Invalid relocatable reference - неприпустиме переміщуване посилання
57 THEN expected - очікується слово **THEN**
58 TO or DOWNTO expected - очікується слово **TO** або **DOWNTO**
59 Undefined forward — невизначений випереджаючий опис
60 Too many procedures - дуже багато процедур
61 Invalid typecast - неправильне приведення типу
62 Division by zero - ділення на нуль
63 Invalid file type - неправильний файловий тип
64 Cannot Read or Write variables of this type - немає можливості читати або записати змінні даного типу
65 Pointer variable expected - очікується змінна-показчик
66 String variable expected - очікується рядкова змінна
67 String expression expected - очікується вираз рядкового типу
68 Unit not found - програмний модуль не знайдено
69 Unit name mismatch — невідповідність імен програмних модулів
70 Unit version mismatch - невідповідність версій програмних модулів
71 Internal stack overflow - переповнення внутрішнього стека
72 Unit file format error - помилка формату файла програмного модуля
73 Implementation expected - очікується секція реалізації
74 Constant and case types don't match - типи константи і тип виразу оператора **case** не відповідають одне одному
75 Record or object variable expected - очікується змінна типу «запис»
76 Constant out of range - значення константи виходить за межі допустимих значень
77 File variable expected — очікується файлова змінна
78 Pointer expression expected - очікується вираз типу «покажчик»
79 Integer or real expression expected - очікується вираз цілого або дійсного типу
80 Label not within current block - мітка не знаходиться в середині поточного блоку
81 Label already defined - мітка вже визначена
82 Undefined label in proceding statement part - невизначена мітка у виконуваному розділі операторів
83 Invalid @ argument - неправильний аргумент оператора @
84 Unit expected - очікується слово **Unit**
85 «;» expected - очікується «;»

86 «:» expected - очікується **«:»**
87 «,» expected - очікується **«,»**
88 «(» expected - очікується **«(»**
89 «)» expected - очікується **«)»**
90 «=» expected - очікується **«=»**
91 «:=» expected - очікується **«:=»**
92 «[» or «(.» expected - очікується **«[»** або **«(.»**
93 «]» or «.)» expected - очікується **«]»** або **«.)»**
94 «.» expected - очікується **«.»**
95 «..» expected - очікується **«..»**
96 Too many variables - забагато змінних
97 Invalid FOR control variable - недопустима змінна циклу **FOR**
98 Integer variable expected - очікується змінна цілого типу
99 File types are not allowed here - тут не допускаються файлові типи
100 String length mismatch - невідповідність довжини рядка
101 Invalid ordering of fields - неправильний порядок полів
102 String constant expected - очікується константа рядкового типу
103 Integer or real variable expected - очікується змінна цілого або дійсного типу
104 Ordinal variable expected - очікується змінна зчисленого типу
105 INLINE error - помилка в операторі **INLINE**
106 Character expression expected - очікується вираз символьного типу
107 Too many relocation items — забагато переміщуваних елементів
108 Overflow in arithmetic operation - переповнення під час виконання арифметичної операції
109 No enclising For, While or Repeat statement - не включений в **For, While** або **Repeat**-конструкцію оператор
110 Cannot run a unit - модуль виконувати не можна
111 Compilation aborted - компіляція перервана
112 CASE constant out of range - константа оператора **CASE** виходить за допустимі межі
113 Error in statement - помилка в операторі
114 Cannot call an interrupt procedure - неможливо викликати напряму процедуру переривання
115 Must have an 8087 to compile this - для компіляції необхідна наявність сопроцесора 80x87
116 Must be in 8087 mode to compile this - для компіляції необхідний режим використання 80x87
117 Targer address not found - адресу призначення не знайдено
118 Include files are not allowed here - у такій ситуації вкладення файлів неприпустиме
119 TP file format error - помилка формату файла **.TP**
120 NIL expected - очікується **NIL**
121 Invalid qualifier - неправильний кваліфікатор
122 Invalid variable reference - неправильне посилання на змінну
123 Too many symbols - забагато символів
124 Statement part too large - завеликий розділ операторів
125 Module has no debug information - у модулі немає інформації для налагодження

126 Files must be var parameters - файли повинні бути описані як **VAR**-параметри
127 Too many conditional symbols - забагато умовних символів
128 Misplaced condotional directive - пропущена умовна директива
129 ENDIF directive missing - пропущена директива **ENDIF**
130 Error in initial conditional defines - помилка в початкових умовних визначеннях
131 Header does not match previous definition - заголовок не відповідає попередньому визначенню
132 Critical disk error - критична помилка диска
133 Cannot evaluate this expression - неможливо обчислити даний вираз
134 Expression incorrectly terminated - некоректне завершення виразу
135 Invalid format specifier - неправильний специфікатор формату
136 Invalid inderect reference - неприпустиме непряме посилання
137 Structured variable are not allowed here - некоректне використання структурної змінної
138 Cannot evaluate without System unit — неможливо обчислити без модуля **System**
139 Cannot access this symbol - доступ до даного символу відсутній
140 Invalid floating-point operation - неприпустима операція з плаваючою комою
141 Cannot compile overlay to memory - не можна виконувати компіляцію оверлеїв у пам'ять
142 Procedual or function variable expected - очікується змінна-процедура або змінна-функція
143 Invalid procedure or function reference - неприпустиме посилання на процедуру або функцію
144 Cannot overlay this unit - цей модуль не може бути оверлейним
145 Too many nested scopes - забагато вкладених контекстів програми, де використано деяке ім'я
146 File access denied — доступ до файла заблокований **DOS**
147 Object type expected - очікується тип «об'єкт»
148 Local object types are not allowed - неприпустимі локальні описи типів об'єктів
149 VIRTUAL expected - необхідне слово **VIRTUAL**
150 Method identifier expected - очікується ідентифікатор методу
151 Virtual constructors are not allowed - віртуальні конструктори неприпустимі
152 Constructors identifier expected - очікується ідентифікатор конструктора
153 Descructor identifier expected - очікується ідентифікатор деструктора
154 Fail only allowed within constructors - виклик **Fail** припустимий тільки з конструктора

Таблиця ASCII кодів

Клавіша	Нормальне натиснення		+CTRL	+ALT
1	2	3	4	5
A	65	97	1	0 30
B	66	98	2	0 48
C	67	99	3	0 46
D	68	100	4	0 32
E	69	101	5	0 18
	70	102	6	0 33
	71	103	7	0 34
н	72	104	8	0 35
I	73	105	9	0 23
J	74	106	10	0 36
к	75	107	11	0 37
L	76	108	12	0 38
м	77	109	13	0 50
N	78	110	14	0 49
0	79	111	15	0 24
р	80	112	16	0 25
	81	113	17	0 16
R	82	114	18	0 19
	83	115	19	0 31
т	84	116	20	0 20
U	85	117	21	0 22
v	86	118	22	0 47
W	87	119	23	0 17
X	88	120	24	045
Y	89	121	25	0 21
Z	90	122	26	0 44
[{	91	123	27	
\\|	92	124	28	
]}	93	125	29	
	96	126		0 120
1!	49	33	03	0 121
2@	50	64		0 122
3#	51	35		0 123
4$	52	36		0 124
5%	53	37	30	0 125
6^	54	94		0 126
7&	55	38		0 127
8*	56	42		0 128
9(	57	40		0 129
0)	48	41	31	0 130
-	45	95		0 131
=+	61	43		
,<	44	60		
.>	46	62		

Продовження таблиці

1	2	3	4	5
/?	47	63		
	59	58		
, „	39	34		
Вліво	0 75	52	0 115	4
Вправо	0 77	54	0 116	6
Вгору	0 72	56		8
Вниз	0 80	50		2
Home	0 71	55	0 119	7
End	0 79	49	0 117	1
PgUp	0 73	57	0 132	9
PgDn	0 81	51	0 118	3
Ins	0 82	48		
Del	0 83	46		
Esc	27	27	27	
Backspace	8	8	127	
Tab	9	0 15		
Сіра /	47	47		
Сіра *	42	42		
Сіра -	45	45		
Сіра +	43	43		
Enter	13	13	10	
Пробіл	32	32	32	32
F1	0 59	0 84	0 94	0 104
F2	0 60	0 85	0 95	0 105
F3	0 61	0 86	0 96	0 106
F4	0 62	0 87	0 97	0 107
F5	0 63	0 88	0 98	0 108
F6	0 64	0 89	0 99	0 109
F7	0 65	0 90	0 100	0 110
F8	0 66	0 91	0 101	0 111
F9	0 67	0 92	0 102	0 112
F10	0 68	0 93	0 103	0 113
F11	0 133	0 135	0 137	0 139
F12	0 134	0 136	0 138	0 140

Таблиця ***скен-кодів***

Клавіша	Скен-код		Клавіша	Скен-код	
	(16)	(10)		(16)	(10)
1	2	3	4	5	6
Esc	$01	1	Z	$2C	44
!1	$02	2	X	$2D	45
@2	$03	3	C	$2E	46
#3	$04	4	V	$2F	47
$4	$05	5	B	$30	48
%5	$06	6	N	$31	49
^6	$07	7	м	$32	50
&7	$08	8	<,	$33	51
*8	$09	9		$34	52
(9	$0A	10	?/	$35	53
)0	$0B	11	Shift (правий)	$36	54
-	$0C	12	PrintScreen	$37	55
+=	$0D	13	Alt	$38	56
Backspace	$0E	14	Пробіл	$39	57
TAB	$0F	15	CapsLock	$3A	58
Q	$10	16	F1	$3B	59
W	$11	17	F2	$3C	60
E	$12	18	F3	$3D	61
R	$13	19	F4	$3E	62
T	$14	20	F5	$3F	63
Y	$15	21	F6	$40	64
U	$16	22	F7	$41	65
I	$17	23	F8	$42	66
0	$18	24	F9	$43	67
P	$19	25	F10	$44	68
{[	$1A	26	NumLock	$45	69
}]	$1B	27	ScrollLock	$46	70
Enter	$1C	28	7 Home	$47	71
Ctrl	$1D	29	8 Вгору	$48	72
A	$1E	30	9 PgUp	$49	73
S	$1F	31	Сірий -	$4A	74
D	$20	32	4 Вліво	$4B	75
F	$21	33	5	$4C	76
G	$22	34	6 Вправо	$4D	77
H	$23	35	Сірий 4-	$4E	78
J	$24	36	1 End	$4F	79
K	$25	37	2 Вниз	$50	80
L	$26	38	3 PgDn	$51	81
.,	$27	39	0 Ins	$52	82
	$28	40	. Del	$53	83
	$29	41	F11	$D9	217
Shift (лівий)	$2A	42	F12	$DA	218
I\	$2B	43			

ЛІТЕРАТУРА

1. *Абрамов С.Л., Гнездилова Г.Г., Капустина Е.Н., Селюн М.И.* Задачи по программированию. - М.: Наука, 1988. - 224 с.
2. *Бондарев В.М., Рублинецкий В.И., Качко Е.Г.* Основы программирования. - Харьков: Фолио; Ростов н/Д.: Феникс, 1997. - 368 с.
3. *Бурсиян Э.В.* Задачи по физике для компьютера: Учеб. пособ. для студентов физ.-мат. фак. пед. ин-тов. - М.: Просвещение, 1991. - 256 с.
4. *Бухтияров А.М., Фроло Г.Д., Олюнин В.Ю.* Сборник задач по программированию на языке ПЛ/1: Учебн. пособ. для вузов. - М.: Наука, 1988. - 320 с.
5. *Вирт Н.* Алгоритмы и структуры данных: Пер. с англ. - М.: Мир, 1989. – 360 с.
6. *Гейн А.Г., Житомирский В.Г.* Основы информатики и вычислительной техники. - М.: Просвещение, 1993. - 254 с.
7. *Довгаль С.И., Сбитнева А.И.* Паскаль, ТурбоПаскаль, многооконная среда на персональных ЭВМ. - К.: Информсистемасервис, 1992. - 181 с.
8. *Епанешников А., Епанешников В.* Программирование в среде TURBO PASCAL 7.0. - М.: Диалог-МИФИ, 1993. - 288 с.
9. *Карасева Т.В.* Сборник задач и упражнений по основам информатики и вычислительной технике. - М.: Колледж, 1995. - 128 с.
10. *Касьянов В.Н., Сабельфельд В.К.* Сборник заданий по практикуму на ЭВМ. - М.: Наука, 1986. - 272 с.
11. *Кнут Д.* Искусство программирования для ЭВМ: Т. 1. Основные алгоритмы. - М.: Мир, 1976. - 735 с.
12. *Окулов С.М.* Основы программирования. - М.: ЮНиМЕДиА-СтАЙЛ, 2002. - 424 с.: илл.
13. *Окулов С.М.* Программирование в алгоритмах. - М.: БИНОМ. Лаборатория знаний, 2002. - 341 с: илл.
14. *Остер Г.* Задачник. - М.: Росмэн, 1998. - 128 с.
15. *Пильщиков В.Н.* Сборник упражнений по языку Паскаль: Учеб. пособ. для вузов. - М.: Наука, 1989. - 160 с.
16. *Поляков Д.Б., Круглое Л.Ю.* Программирование в среде Турбо Паскаль (версия 5.5). - М.: Из-тво МАИ А/О «Росвузнаука», 1992. - 576 с.
17. *Прайс Д.* Программирование на языке Паскаль: Практическое руководство. – М.: Мир, 1987. - 232 с.
18. *Рюттен Т., Франкен Г.* Турбо Паскаль 6.0. Основы языка программирования. - К.: Торгово-издательское бюро BHV, 1992. - 236 с.

ЗМІСТ

Навчальне видання

КАРАВАНОВА Тетяна Петрівна

За загальною редакцією
академіка НАН України Згуровського М.З.

ІНФОРМАТИКА

ОСНОВИ АЛГОРИТМІЗАЦІЇ ТА ПРОГРАМУВАННЯ

777 задач з рекомендаціями
та **прикладами**

Навчальний посібник
для 8-9 класів із поглибленим
вивченням інформатики

Рекомендовано Міністерством освіти і науки України

Редактори *Н. Дашко, М. Зубченко*
Обкладинка і макет *П. Машкова*
Технічні малюнки, художнє оформлення *О. Безо*
Технічний редактор *В. Олійник*
Коректори *С. Романичева, М. Рубан*
Комп'ютерна верстка *О. Безобчука*