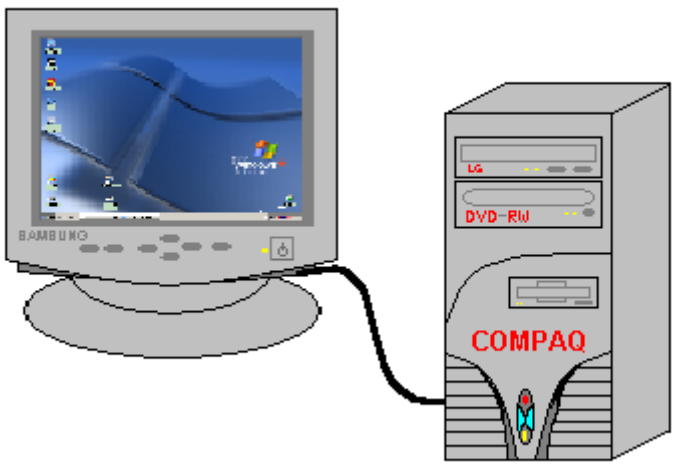
Л.М. Круподьорова

Алгоритмізація і основи програмування



Міністерство освіти і науки України
Вінницький національний технічний університет

Л.М. Круподьорова

Алгоритмізація і основи програмування

Затверджено Вченою радою Вінницького національного технічного університету як навчальний посібник для студентів напряму підготовки 0804 “Комп’ютерні науки” спеціальностей “Інтелектуальні системи прийняття рішень” та “Програмне забезпечення автоматизованих систем” і підготовчих курсів ВНТУ. Протокол № 5 від 30 грудня 2004 р.

Вінниця ВНТУ 2005

УДК 681.31
К 84

Р е ц е н з е н т и:
С.В.Юхимчук, доктор технічних наук, професор
О.М.Роік, доктор технічних наук, професор
О.В.Сілагін, к. т. н., директор ТОВ “ І Т І ”

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України

Круподьорова Л.М.
К 84 **Алгоритмізація і основи програмування.**
Навчальний посібник. – Вінниця: ВНТУ, 2005. - 168 с.

В навчальному посібнику розглянуті основні засоби складання алгоритмів, історія виникнення найбільш відомих мов програмування, викладені основні положення програмування мовою Паскаль, наведено багато прикладів і рекомендації для їх засвоєння. Приділена увага різноманітним задачам з поясненням щодо їх створення. Наведені лабораторні роботи та індивідуальні завдання, що можуть бути використані при проведенні як практичних, так і лабораторних занять.

Навчальний посібник призначений для студентів напряму підготовки 0804 “Комп’ютерні науки” спеціальностей "Інтелектуальні системи прийняття рішень" та "Програмне забезпечення автоматизованих систем".

УДК 681.31

ЗМІСТ

1 МЕТОДИКА ПІДГОТОВКИ І РОЗВ'ЯЗУВАННЯ ЗАДАЧІ НА ЕОМ

Розв'язування задачі на ЕОМ – складний і трудомісткий процес. Будь-яка задача починається з постановки задачі. На основі словесного формулювання задачі вибираються змінні, які підлягають визначенню, записуються обмеження, зв'язки між змінними, які створюють математичну модель вирішуваної проблеми. Аналізується метод розв'язування. На цьому етапі необхідно прийняти дуже важливе рішення – чи буде використано наявне готове програмне забезпечення, чи буде розроблятись власна програма. Дешевше і швидше використовувати готові розробки. Оновлення програмного забезпечення – задача програмістів. В цьому випадку традиційно виділяються такі основні *етапи розв'язування* задачі на ЕОМ:

– постановка задачі, розробка математичної моделі;
– вибір методу чисельного розв'язування;
– розробка алгоритму і структури даних;
– проектування програми;
– виробництво остаточного програмного продукту;
– розв'язування задачі на ЕОМ.

Постановка задачі – точний опис початкових даних, умов задачі і цілей її розв'язування. На цьому етапі деякі умови задачі, задані у формі різних словесних описів, необхідно сформулювати на точній мові математики. Часто задача програмування задається в математичному формулюванні, тому необхідність у виконанні етапів 1 і 2 відпадає. Для розв'язання досить складних задач етап формалізації може зайняти значних зусиль і часу. Серед досвідчених програмістів поширена думка, що виконати етап формалізації – це значить зробити половину всієї роботи для створення програми.

Вибір методу розв'язування тісно пов'язаний з постановкою задачі. На першому етапі задача зводиться до математичної моделі, для якої відомий метод розв'язування. Метод чисельного розв'язування зводить розв'язування задачі до послідовності арифметичних і логічних операцій. Проте можливо, що для одержаної моделі відомі декілька методів розв'язування і тоді потрібно вибрати кращий. Можна удосконалити існуючий або розробити новий метод розв'язування формалізованої задачі. Ця робота за своїм характером є науково-дослідною і може зажадати значних зусиль. Розробкою і вивченням таких методів займається розділ математики, що називається чисельним аналізом.

При виборі методу треба враховувати вимоги, що пред'являються постановкою задачі, і можливості його реалізації на конкретній ЕОМ: точність розв'язування, швидкість отримання результату, необхідні затрати оперативної пам'яті для зберігання початкових і проміжних даних і результатів.

Алгоритм встановлює точну послідовність певних дій, що приводять до розв’язування задачі. При цьому послідовність дій може задаватися за допомогою словесного або графічного описів. Якщо вибраний для розв’язування задачі чисельний метод реалізований у вигляді стандартної бібліотечної підпрограми, то алгоритм звичайно зводиться до опису і введення початкових даних, виклику стандартної підпрограми і виведення результатів на екран або на друк. Більш характерний випадок, коли стандартною підпрограмою розв’язують лише якусь частину задачі. Тут ефективним підходом є розділення складної початкової задачі на деякі підзадачі, що реалізовуються окремими модулями. Визначається загальна структура алгоритму, взаємодія між окремими модулями, деталізується логіка. Цей етап тісно пов'язаний з наступним етапом проектування програми.

Проектування програми включає декілька підзадач. По-перше, необхідно вибрати мову програмування. По-друге, визначити, хто буде використовувати розроблене програмне забезпечення і яким повинен бути інтерфейс (засіб спілкування з користувачем). По-третє, вирішити всі питання щодо організації даних. По-четверте, кодування, тобто опис алгоритмів за допомогою інструкцій вибраної мови програмування. Якщо задача, для якої розробляється алгоритм, складна, то не слід відразу намагатися вирішити всі проблеми. Підхід, що склався в даний час, до розробки складних програм полягає в послідовному використанні принципів проектування зверху вниз, модульного і структурного програмування.

Остаточний програмний продукт виходить після *налагодження і випробування* програми. При програмуванні і введенні даних з клавіатури можуть бути допущені помилки. Їх виявлення, локалізацію і усунення виконують на етапі налагодження і випробування (тестування) програми. Причому можуть бути допущені логічні помилки і на етапі постановки задачі, і на етапі алгоритмізації. В цьому випадку необхідно повернутися до попередніх етапів. Допрацьовувати і покращувати програму можна протягом всього життєвого циклу програмного продукту.

Розв’язування задачі на ЕОМ – виконання всіх передбачених програмою обчислень і виведення результатів розрахунку на екран дисплея або на друк.

Приклад. Побудувати траєкторію руху тіла, кинутого під кутом до горизонту, і визначити дальність кидка.

Постановка задачі, розробка математичної моделі. Для спрощення моделі приймемо такі припущення:

– нехтуємо кривизною Землі, вважаючи її поверхню площиною;
– прискорення вільного падіння вважаємо константою;
– нехтуємо опором повітря;
– нехтуємо рухом Землі.

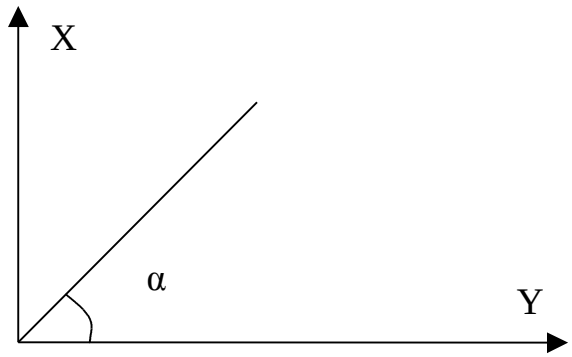


Рисунок 1

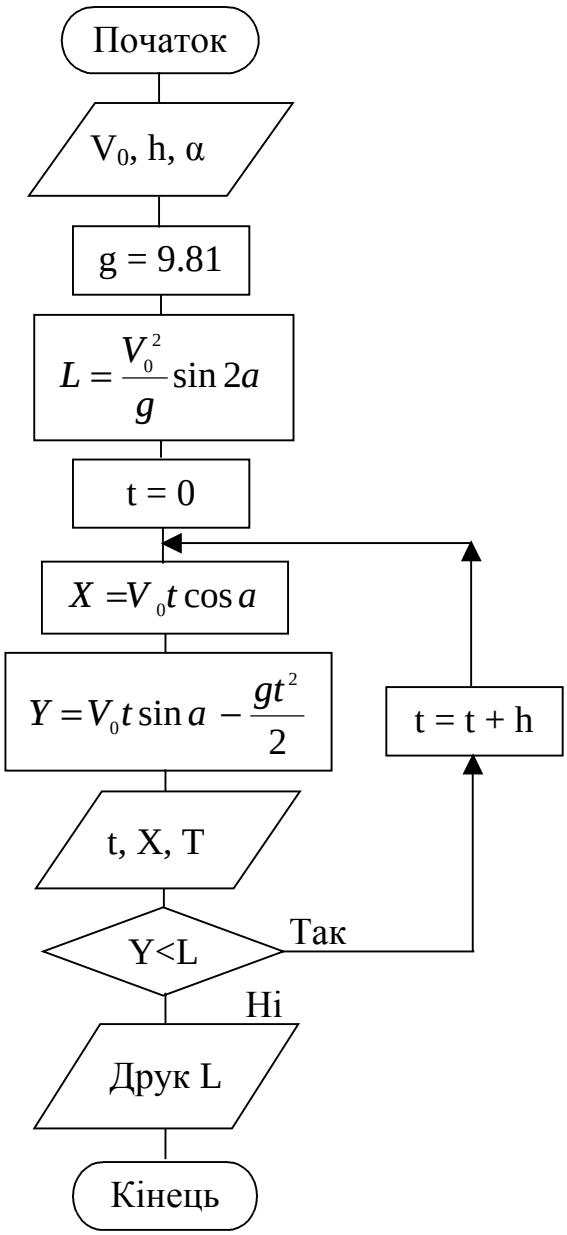


Рисунок 2

$$Y = V_0 t \sin a - \frac{gt^2}{2}$$

$$X = V_0 t \cos a \qquad (1)$$

Математичний опис об'єкта.
Рівняння руху тіла, кинутого під кутом до горизонту, в декартових координатах записуються таким чином:
Y, X – вертикальна і горизонтальна складові руху;
g – прискорення вільного падіння;
t – час.

Створення математичної моделі дозволяє поставити задачу математично.

Нехай рух тіла, кинутого під кутом α до горизонту з початковою швидкістю V_0 (рис. 1), описується системою рівнянь (1). α і V_0 задані. Потрібно визначити дальність польоту і положення тіла в кожен момент часу від t = 0 до t_k з кроком h.

Другим етапом розв'язування задачі можна нехтувати, оскільки ніякого особливого чисельного методу розв'язування в даному випадку застосовувати не потрібно. Точка падіння визначається з системи рівнянь при умові Y = 0.

$$X = \frac{V_0^2}{g} \sin 2\alpha.$$

Змінюючи значення t з кроком h, розраховуємо координати X, Y.

Алгоритм розв'язування краще всього подати у вигляді схеми (рис.2).

На наступному етапі алгоритм записується на мові програмування і текст програми переноситься на носій, з якого вона вводиться в ЕОМ. Часто програма вводиться з клавіатури дисплея в оперативну пам'ять

комп'ютера, а потім записується на диск.

За готовою програмою виробляються розрахунки і результати порівнюються з контрольним прикладом. Контрольним прикладом можуть служити експериментальні дані. Якщо в процесі налагодження програми виникають помилки, то їх необхідно виправити.

2 СПОСОБИ ЗАПИСУ АЛГОРИТМУ

2.1. Алгоритм і його властивості

У основі розв’язання будь-якої задачі лежить поняття алгоритму. Під **алгоритмом** прийнято розуміти "упорядкований скінченний набір чітко визначених правил для розв’язування задач за скінченну кількість кроків" (ДСТУ 2938-94, п.3.33).

При складанні алгоритмів слід враховувати ряд вимог, виконання яких приводить до формування необхідних властивостей.

Алгоритм повинен бути однозначним, що виключає довільність тлумачення будь-якого з розпоряджень і заданого порядку виконання. Ця властивість алгоритму називається **визначеністю**.

Реалізація обчислювального процесу повинна через певне число кроків привести до видачі результатів або повідомлення про неможливість розв’язання задачі. Ця властивість алгоритму називається **результативністю**.

Розв’язання однотипних задач з різними початковими даними можна здійснювати за одним і тим же алгоритмом, що дає можливість створювати типові програми для розв’язання задач при різних варіантах завдання значень початкових даних. Ця властивість алгоритму називається **масовістю.**

Визначений алгоритм обчислювального процесу можна розчленувати на окремі етапи, елементарні операції. Ця властивість алгоритму називається **дискретністю**.

Алгоритмізація – техніка складання алгоритмів і програм для розв’язання задач на ЕОМ.

2. 2 Образотворчі засоби для опису алгоритмів

До образотворчих засобів опису алгоритмів відносяться такі основні способи їх подання:

– словесний (записи на природній мові);
– структурно-стилізований (записи на мові псевдокоду);
– програмний (тексти на мовах програмування);
– графічний (схеми графічних символів).

Словесний спосіб запису алгоритмів є описом послідовних етапів обробки даних і задається в довільному викладі на природній мові. Спосіб заснований на виконанні загальноприйнятих засобів спілкування між людьми і, з погляду написання, труднощів для авторів алгоритмів не

представляє. Проте для виконавців такі описи алгоритмів часто неприйнятні. Вони строго не формалізуються, страждають багатослівністю записів, допускають неоднозначність тлумачення окремих розпоряджень. Тому такий спосіб опису алгоритмів не має широкого розповсюдження.

Приклад. *Записати алгоритм знаходження найбільшого загального дільника двох натуральних чисел (m і n) на природній мові*.

При такому словесному способі зміст алгоритму може бути таким:

- якщо числа рівні, то необхідно узяти будь-яке з них як відповідь, інакше – продовжити виконання алгоритму;
- визначити більше число з двох чисел;
- замінити більше число різницею більшого і меншого чисел;
- повторити алгоритм спочатку.

Структурно-стилізований спосіб запису алгоритмів заснований на формалізованому представленні розпоряджень, що задаються шляхом використання обмеженого набору типових синтаксичних конструкцій. Такі засоби опису алгоритмів часто називаються **псевдокодами**. Різновидом структурно-стилізованого способу опису алгоритмів є відома школярам алгоритмічна мова в російській нотації (АЯРН).

Приклад. Приведемо описаний на АЯРН алгоритм розв'язування задачі про визначення належності точки D трикутнику ABC.

```
алг Визначення належності точки трикутнику (дійсні xA, yA, xB, yB, .
xC, yC, xD, yD ціле z літ a);
арг xA,yA,xB,yB,xC,yC,xD,yD;
рез z,a;
        поч
        дійсні S1, S2, S3, S4
        обчислити значення S1, рівне площі трикутника ABC;
        обчислити значення S2, рівне площі трикутника ABD;
        обчислити значення S3, рівне площі трикутника ACD;
        обчислити значення S4, рівне площі трикутника CDB;
        якщо S1 = S2+S3+S4
        то  z := 1,
          a := "точка всередині трикутника",
        інакше z := 0,
          a := "точка поза трикутником",
        все
        надрукувати значення a:
кін
```

Програмний спосіб запису алгоритмів – це алгоритм, записаний на мові програмування, який дозволяє на основі певних правил формувати послідовність розпоряджень, які однозначно відображають значення і зміст різних частин алгоритму з метою подальшого їх виконання на ЕОМ.

Приклад. Програма на мові Бейсік переведення температури з градусів Цельсія в градуси Фаренгейта.

```
PRINT "Переведення температури з градусів Цельсія в градуси Фаренгейта"
   PRINT "Вкажіть температуру в градусах Цельсія"
   INPUT C
 6 IF C = 9999 THEN 7
   F = C*1.8+32
   PRINT C, F
   GOT0   6
 7 END
```

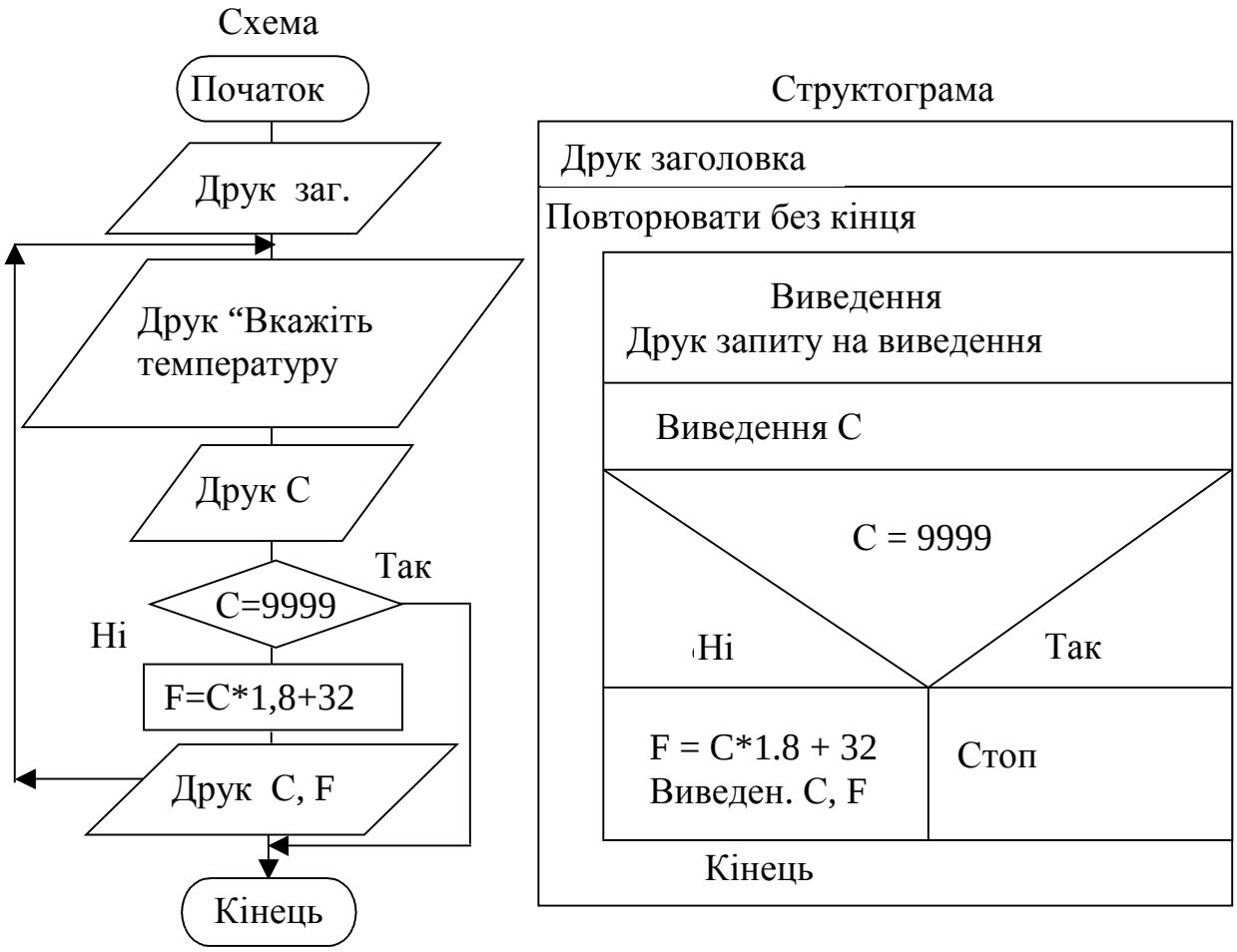


Рисунок 3 – Схема і структурограма переведення температури з шкали Цельсія в шкалу Фаренгейта

Для **графічного зображення алгоритмів** використовуються графічні символи. Найпоширенішими є блокові символи (блоки), що сполучаються лініями передач управління. Існує державний стандарт на виконання графічного запису алгоритму. Графічний запис алгоритму є найбільш наглядним. Перелік умовних графічних символів, їх найменування, форма, розміри і функції, що відображаються, встановлюються ГОСТом 19.003-80. Основні графічні символи, які використовуються для опису алгоритмів, приведені в літературі [1,2].

Схеми можуть бути представлені також у вигляді структурограм або

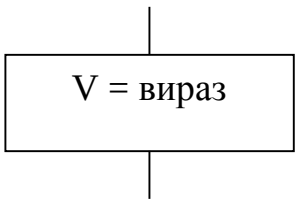
на ім'я їх авторів, діаграмами Нессі - Шнейдермана [7].

На рис. 3 приведена схема алгоритму переведення температури з шкали Цельсія в шкалу Фаренгейта. Переведення здійснюється за формулою:

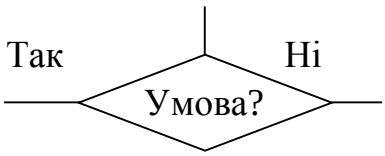
Температура за Фаренгейтом = (температура за Цельсієм)·180/100 +32.

2.3 Схеми алгоритмів

Використання схем дозволяє представити алгоритм в наочній формі, тому вони найчастіше використовуються.

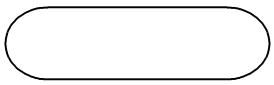


Обчислювальний блок є прямокутником, в якому записуються розрахункові формули. Причому формула повинна бути записана таким чином, що змінна, яка обчислюється, записується зліва, далі йде знак := рівності (в даному випадку цей знак називається привласненням), далі – розрахункова формула.



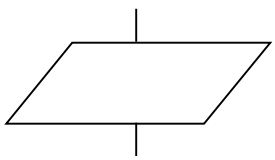
Перевірка умови зображається ромбом, усередині якого записується ця умова. В результаті перевірки вибирається один з двох можливих шляхів обчислювального процесу.

Якщо умова виконується, тобто має значення ТАК, то наступним виконується етап по стрілці ТАК. Якщо умова не виконується, то здійснюється перехід по стрілці НІ.



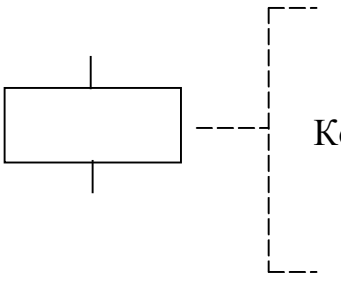
Початок і кінець обчислювального процесу зображаються овалом, в якому записуються слова "Початок" або "Кінець".

При розв'язанні задач на ЕОМ початкові дані задаються при введенні в машину. Введення даних може здійснюватися різними способами, наприклад, з клавіатури, перфострічки, з диска і т. ін. Завдання чисельних значень початкових даних ми називатимемо введенням, а фіксацію результатів розрахунку – виведенням. Введення початкових даних і виведення результатів зображаються паралелограмом. Усередині нього пишеться слово "Введення" або "Виведення" і перераховуються змінні, які підлягають введенню або виведенню.



Паралелограмом позначається введення - виведення, не прив'язане до якогось конкретного пристрою. Якщо введення або виведення здійснюється з виконанням конкретних пристроїв, то блоки введення-виведення зображаються за допомогою спеціальних фігур [2].

Коментар використовується в тих випадках, коли пояснення не поміщається усередині блоку.



За стандартом висота блоку дорівнює **а**, ширина 1.5×**а**, де розмір **а** вибирається з ряду 10, 15, 20 мм. Блоки "початок" і "кінець" мають висоту 0.5а.

Лінії потоку проводять паралельно зовнішнім краям рамки сторінки. Напрям ліній потоку зверху вниз і зліва направо приймають за основне; якщо лінії потоку основного напряму не мають зламів, то їх напрям стрілками можна не позначати. У решті випадків напрям ліній потоку позначати стрілкою обов'язково. Записи усередині символа або поряд з ним повинні виконуватися машинописом або креслярським шрифтом і повинні бути короткими. У лівому верхньому кутку в розриві ліній ставиться номер блоку.

В наш час основна тенденція у використанні схем алгоритмів полягає не стільки у вказівці послідовності операцій, скільки в групуванні блокових символів у вигляді базових керуючих конструкцій. До них відносяться лінійний, розгалуження і повторення. **Основні структури алгоритмів** – це обмежений набір блоків і стандартних способів їх з'єднання для виконання типових послідовностей дій. Такі структури рекомендуються при виконанні так званого структурного підходу до розробки алгоритмів і програм. Структурний підхід припускає виконання тільки декількох основних структур, комбінація яких дає все різноманіття алгоритмів і програм.

2.3.1 Алгоритми лінійної структури

Алгоритм лінійної структури (лінійний) – алгоритм, в якому всі дії виконуються послідовно один за одним. Такий порядок виконання дій називається природним.

Розглянемо декілька прикладів.

Задача 1. Визначити площу трикутника за формулою Герона

$$S = \sqrt{p \cdot (p-a) \cdot (p-b) \cdot (p-c)},$$

де a, b, c – довжини сторін;

p = (a + b + c)/2 – півпериметр трикутника.

Для того, щоб розрахувати S, необхідно мати чисельні значення p, a, b, c. Ми можемо розрахувати p за формулою, а ось значення a, b, c повинні бути задані наперед, інакше задачу розв’язати неможливо.

Запишемо словесний алгоритм.

1. Задати чисельні значення a, b, c.
2. Обчислити p за формулою:

$$p = (a + b + c)/2.$$

3. Обчислити S за формулою:

$$S=\sqrt{p\cdot(p-a)\cdot(p-b)\cdot(p-c)}.$$

4. Зафіксувати результат.

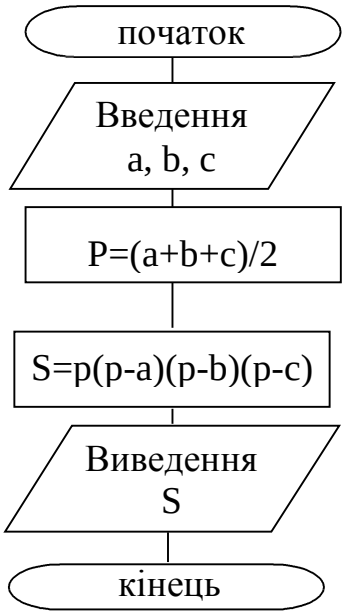


Схема є послідовністю блоків, сполучених лініями потоків. Напрям потоку задається стрілкою, але стрілка не ставиться, якщо напрям потоку зверху вниз і зліва направо. У лівому верхньому кутку в розриві ліній ставиться номер блоку.

Усередині блоку введення записується слово "Введення" і перераховуються початкові дані (імена змінних), які задаються ззовні. Усередині блоку виведення записується слово "Виведення" і перераховуються змінні, які є результатом розрахунку.

Приведемо ще один приклад схеми алгоритму лінійної структури.

$$Y = (e^{-x_1} + e^{-x_2}) \ /2;$$

$$Z = (a\sqrt{x_1} - b\sqrt{x_2}) \ /c;$$

$$X_1 = \left(b + \sqrt{|b^2 - 4 \cdot a \cdot c|}\right) /2a;$$

$$X_1 = \left(b - \sqrt{|b^2 - 4 \cdot a \cdot c|}\right) /2a.$$

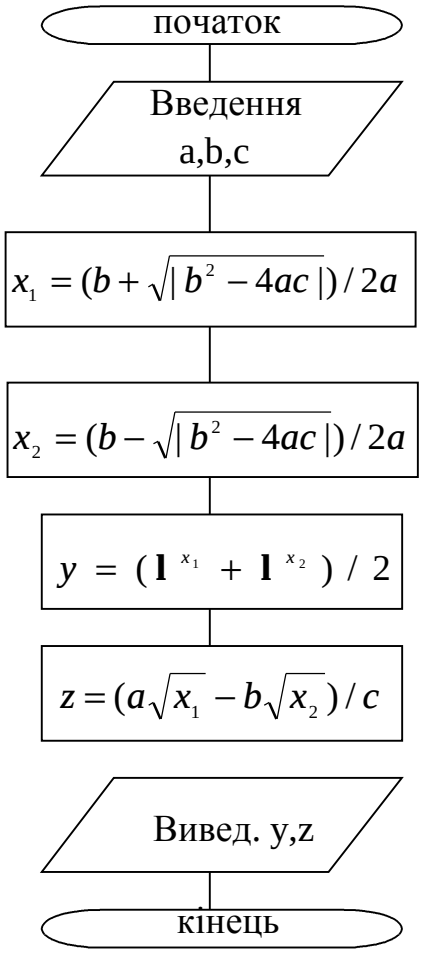


Рисунок 5 – Схема алгоритму лінійної структури

Розробка алгоритму (рис. 5).Початкові дані: a, b, c. Y і X повинні бути визначені раніше, ніж Y і Z оскільки входять в розрахункову формулу для Y і для Z.

2.3.2 Алгоритми структури, що розгалужується

На практиці рідко вдається подати схему алгоритму розв'язування задачі у вигляді лінійної структури. Часто залежно від яких-небудь значень проміжних результатів необхідно організувати обчислення або за одними, або за іншими формулами. **Розгалуження** – така схема, в якій передбачено розгалуження вказаної послідовності дій на два напрями залежно від підсумку перевірки заданої умови. У схемах такої структури використовується логічний блок (рис. 6).

Задача 2.

Розрахувати Y за умовами.

$$Y=\begin{cases} -X, & X<0; \\ X^2, & X\geq 0. \end{cases}$$

Розробка алгоритму. У цій задачі повинно бути задано X. Далі X аналізується. Якщо X<0, то обчислення виконується за першою формулою, якщо ця умова не виконується, то виконується друга умова X > 0, оскільки умови X<0 і X ≥ 0 взаємовиключені, і Y обчислюється за іншою формулою.

Словесний алгоритм розв'язування цієї задачі виглядатиме таким чином.

1. Задати чисельне значення для X.
2. Перевірити умову X<0:
   – якщо умова виконується перейти до п. 5;
   – якщо умова не виконується перейти до п. 3.
3. Обчислити Y за формулою $Y = X^2$.
4. Перейти до пункту 6.
5. Обчислити Y за формулою Y =-X.
6. Зафіксувати обчислене Y.

Схема даного алгоритму подана на рис. 6.

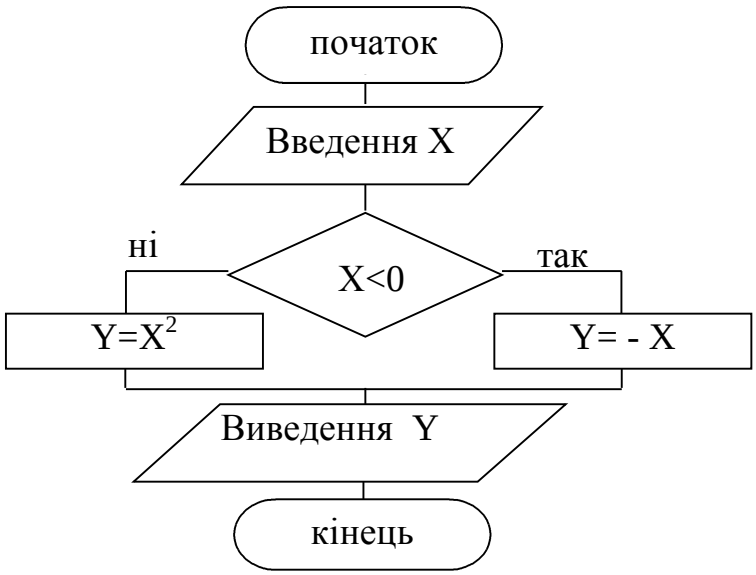


Рисунок 6 – Схема алгоритму структури, що розгалужується.

Рекомендується під словом "ні" записувати умову, протилежну тій, що перевіряється.

Задача 4. Обчислити значення функції Z = X /Y, де Y = sin(nX)+ 0.5.

Розробка алгоритму. Здавалося б вирішення цієї задачі можна описати алгоритмом лінійної, структури. Проте, для задоволення властивостей масовості і результативності алгоритму необхідно, щоб при будь-яких початкових даних був одержаний результат або повідомлення про те, що задача не може бути розв‘язана при заданих початкових даних. Дійсно, якщо Y = 0, задача не може бути розв‘язана. Тому в алгоритмі

необхідно передбачити цей випадок і видати як результат інформацію про те, що Y = 0. Даний обчислювальний процес повинен мати дві гілки: у одній, якщо Y ≠ 0, необхідно обчислити значення змінної Z, а в іншій дати

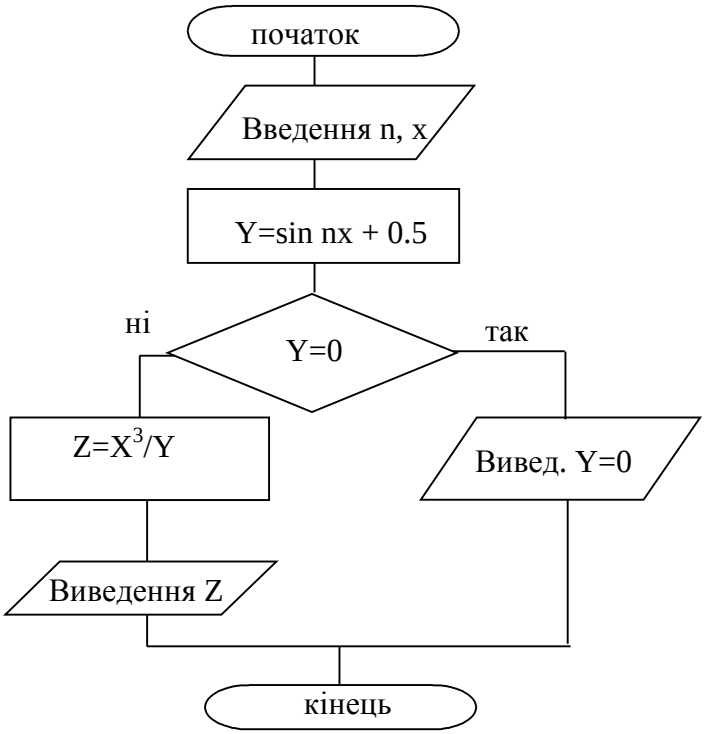


Рисунок 7 – схема алгоритму задачі 4

інформацію Y = 0 (рис 7).

Якщо умова Y = 0 виконується, то працює блок 6, тобто здійснюється виведення повідомлення про рівність Y нулю. Після блоку 6 зразу ж виконується блок 8 – розв'язання закінчується. Якщо умова Y=0 не виконується, це значить, що виконується умова Y ≠ 0 і ми повинні розрахувати Z і вивести його.

2.3.3 Алгоритми циклічної структури

Алгоритми, окремі дії в яких багато разів повторюються, називаються **алгоритмами циклічної структури (повторення)**. Сукупність дій алгоритму, пов'язану з повторенням, називають **циклом**. Приведемо приклад схеми алгоритму циклічної структури.

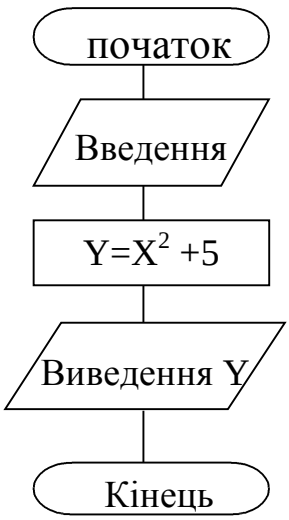


Рисунок 8

Задача 3. Обчислити множину значень функції $Y = X^2 + b$ для всіх значень X від -10 до 10 з кроком 2, при b = 5.

Розробка алгоритму. Значення Y необхідно обчислити 11 разів, тобто необхідно 11 разів виконати алгоритм лінійної структури (рис. 8).

Завдання Х можна автоматизувати, організувавши цикл. Для цього необхідно задати початкове значення Х, тобто Х = -10. Далі розрахувати Y за формулою, вивести чисельне значення Y, змінити Х і повернутися до розрахунку Y.

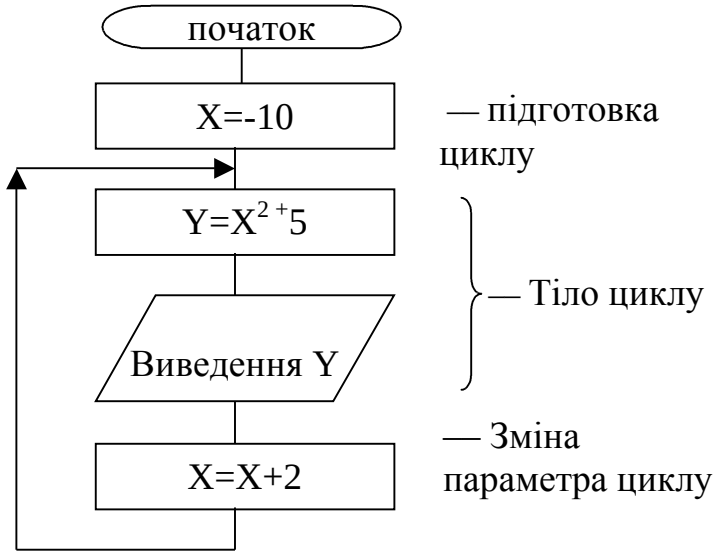


Рисунок 9

Тоді схема виглядатиме таким чином (рис 9).

На рисунку 10 подана схема алгоритму циклічної структури. Блоки 3, 4, які створюють тіло циклу, повторюються багато разів. Скільки разів? Нескінченна кількість. При кожному розрахунку до попереднього значення Х додається 2, далі йде повернення до розрахунку Y, виведення Y і знову Х змінюється на 2. За умовою задачі розрахунком Y при Х=10 потрібно обмежитися. Отже, необхідно включити умову закінчення розрахунків. До тих пір, поки Х≤10, розрахунки виконувати; як тільки Х стане більше 10 – обчислення закінчити. У схему включимо логічний блок.

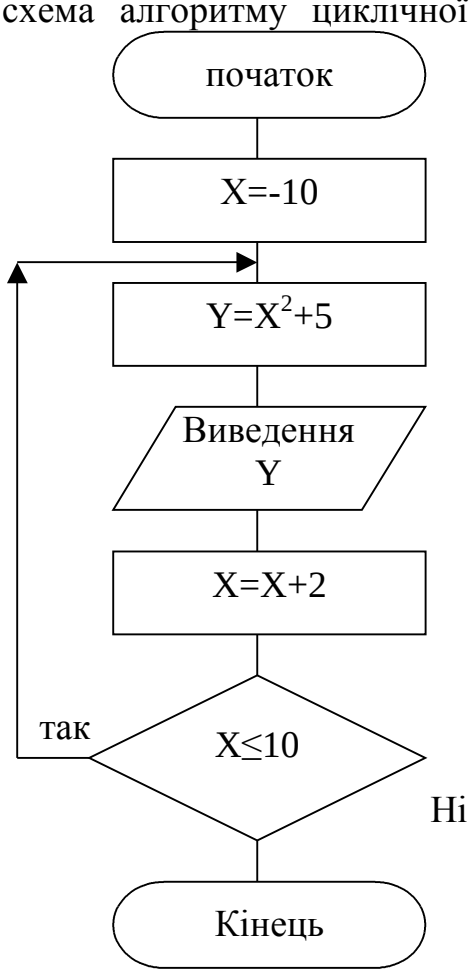


Рисунок 10– Схема алгоритму циклічної структури

У блоці 2 здійснюється завдання початкового значення для Х. У блоці 3 розраховуються значення Y. У блоці 5 фіксується поточне значення Х із заданим кроком. У блоці 6 аналізується величина Х. Якщо Х ще не перевищив свого кінцевого значення, та необхідно повернутися до блоку 3 і повторити обчислення. Якщо Х став більше граничного значення, розрахунки потрібно закінчити.

Щоб алгоритм став більш універсальним, початкове значення $X_н$

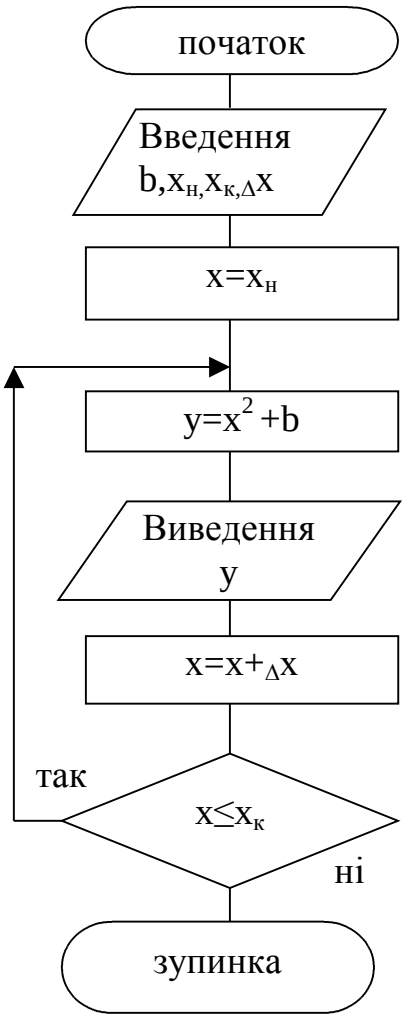


Рисунок 11 – Схема обчислення квадратного рівняння

кінцеве значення $X_к$ і крок зміни ΔX задамо в блоці введення (рис. 11).

Величина, із зміною якої пов'язане багаторазове виконання циклу, називається **параметром циклу**. У нашому прикладі це X. Блоки 4, 5 – тіло циклу. Блок 3 є підготовкою циклу. Блок 6 – зміна параметра циклу (підготовка чергового кроку), а блок 7 – умова продовження циклу.

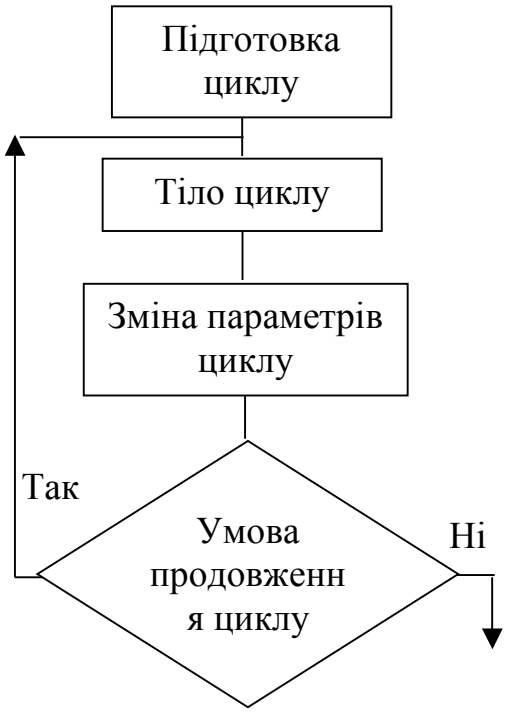


Рисунок 12 – Схема циклу з післяумовою

Така циклічна структура називається циклом "До" (рис. 12). Особливість цього циклу полягає у тому, що він виконується хоча б один раз, оскільки перша перевірка умови виходу з циклу відбувається після того, як тіло циклу виконане.

Існує ще цикл "Поки" (рис. 13). Цикл "Поки" відрізняється від циклу "До" тим, що тут перевірка умови проводиться до виконання тіла циклу. Якщо при першій перевірці умова виходу з циклу виконується, то тіло циклу не виконується жодного разу.

Використовуємо цикл типу ”Поки” для нашого прикладу (рис.14).

Для зображення алгоритмів циклічної структури використовується також блок "модифікація" (рис.15).

У блоці "модифікація" об'єднуються декілька блоків: підготовка циклу, перевірка (підготовка чергового кроку).

У блоці модифікації записується параметр циклу, знак рівності

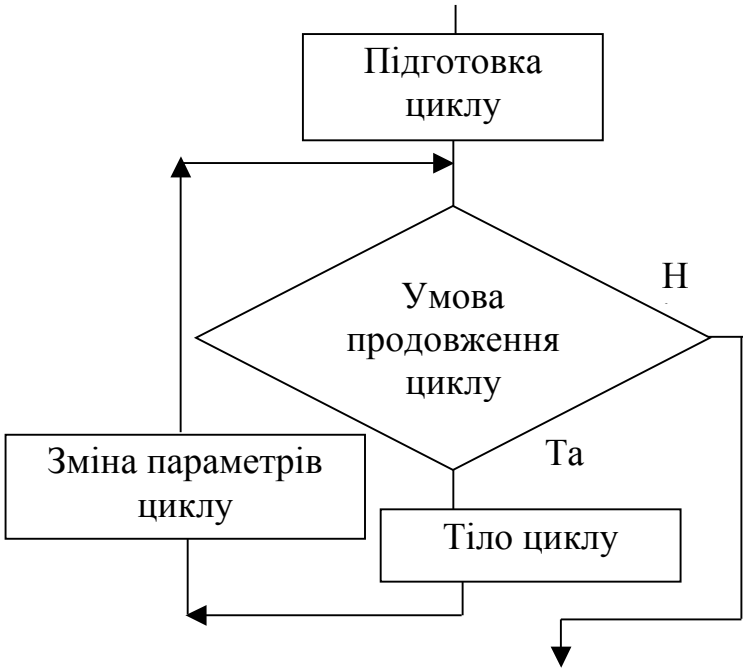


Рисунок 13- Схема циклу с передумовою

(привласнення), закінчення, зміна параметра цик-лу початкове значення параметра циклу, кінцеве значення параметра цик-лу і крок зміни параметра циклу. Для нашого прик-ладу схема алгоритму з використанням блоку модифікації виглядає як на рис.15.

Використаємо цикл типу "Поки" для нашого прикладу (рис.14).

Блок 3 можна прочитати таким чином: виконати для всіх X від і $X_н$ до $X_к$ з кроком ΔX. Тіло циклу виділено лінією потоку, замкнутою на блок модифікації.

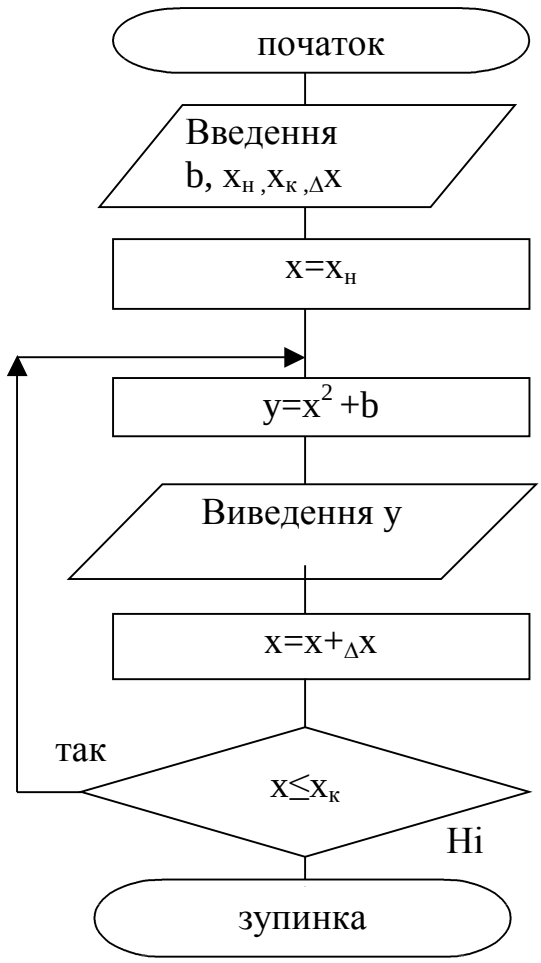


Рисунок 14 – Схема циклу з передумовою

Задача 6. Визначити середній ріст студентів в групі.

Розробка алгоритму. Початковою інформацією для вирішення цієї задачі є число студентів в групі і ріст кожного студента. Тут ми зустрічаємося з поши-реною задачею розрахунку додатку. Сума виходить шляхом накопичення доданків в якій-небудь змінній. Нако-пичення здійснюється в циклі. Почат-ковому значенню суми привлас-нюється значення нуль. У циклі до поточного значення суми додається значення чергового доданку S = S + R.

Число студентів в групі позначимо N. Це початкова величина. У змінній S накопичуватимемо суму. Задамо S значення нуль. Підрахунок номера студента здійснюватимемо в змінній i. Почнемо з першого студента i = 1. Вводимо ріст першого студента.

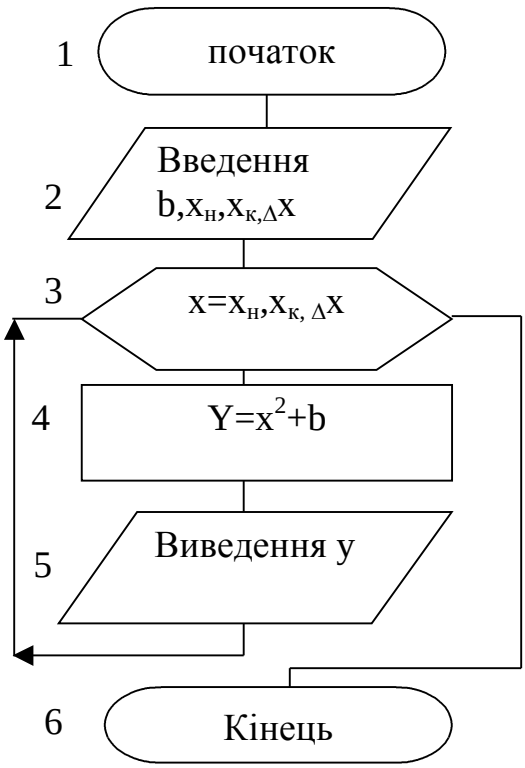


Рисунок 15

другого студента. Далі переходимо до наступного кроку. Цикл повториться N раз і в змінній S накопичиться сума всіх значень ростів студентів (рис. 16).

Середній ріст визначаємо за формулою S = S/N. Необхідно звернути увагу на форму запису даного виразу. У правій і в лівій частині цього виразу найменування однієї і тієї ж змінної. Такий запис означає: обчислити чисельне значення, вирази в правій частині і привласнити це значення змінній, що стоїть зліва.

Запишемо схему, використовуючи блок модифікації (рис. 17). Треба звернути увагу на те, що блок 4 - модифікація – об’єднав три блоки попередньої схеми – блоки 4,7,8. Не можна точно сказати, яка з типів циклічних структур ("До" або "Поки") приховується в блоці "модифікація". Наприклад, в мовах

До попереднього значення суми, тобто до нуля, додамо ріст першого студента і результат привласнимо змінній S. Перейдемо до наступного кроку

i = i+1 = 1+1 =2.

У змінної і тепер значення 2. Виконаємо перевірку виходу з циклу. Якщо і не перевищило ще значення N, то ми повертаємося до блоку 5 і вводимо ріст наступного студента.

До попереднього значення суми, а це ріст першого студента, додаємо ріст другого студента і результат записуємо в змінну S. У змінній S тепер зберігатиметься сума двох значень: ріст першого студента і ріст

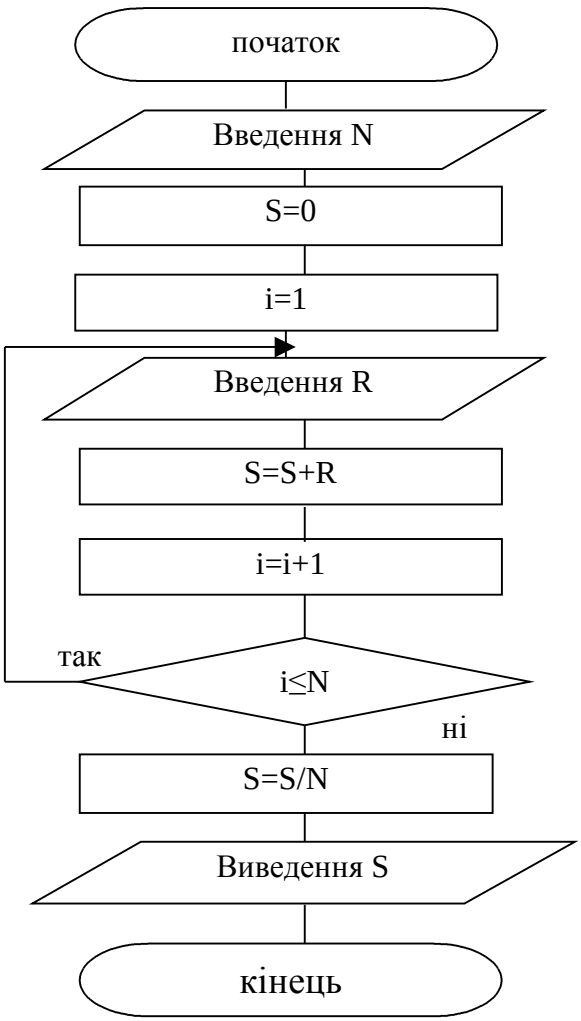


Рисунок 16

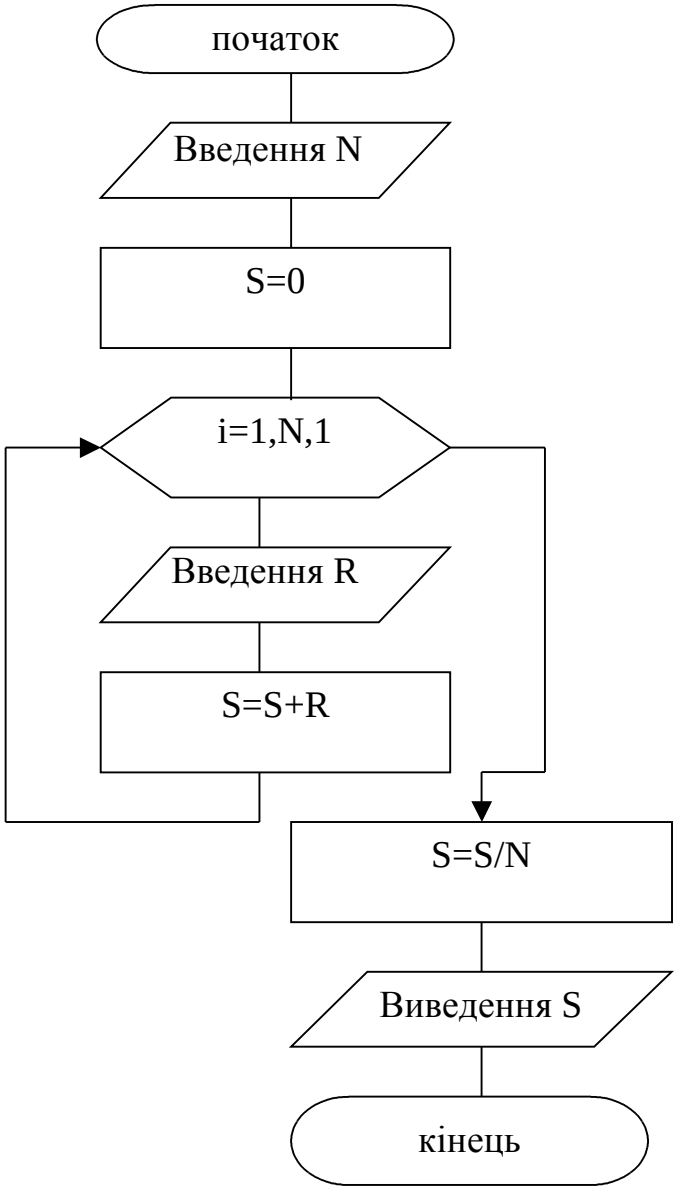


Рисунок 17

програмування ця структура організована по різному.

Задача 7. Визначити кількість студентів в групі, що мають ріст вище середнього.

Розробка алгоритму. Початкові дані: кількість студентів і ріст кожного студента. Перша половина алгоритму розроблена в попередній задачі – визначено середній ріст. Далі ріст кожного порівнюється з середнім.

Якщо значення росту студента, що вводиться, перевищує середнє значення, то в лічильник числа таких студентів необхідно додати одиницю К=К+1. Лічильник утворюється із змінної К, початкове значення якої вважаємо рівним нулю (блок 8), а підсумовування відбувається в блоці 12, якщо умова (блок 11) виконується. У схемі А (рис. 18) є один недолік: ріст студентів в групі вводиться двічі. Якщо уявити, що задача розв'язується на ЕОМ, то в цьому випадку доведеться вводити одну і ту ж інформацію двічі. Цей недолік можна усунути введенням змінної з індексом. У схемі В

(рис. 18) ріст студента позначений змінною R_i, де і – поточний номер студента. R – одновимірний масив з N елементів. Блок 10 з алгоритму виключений.

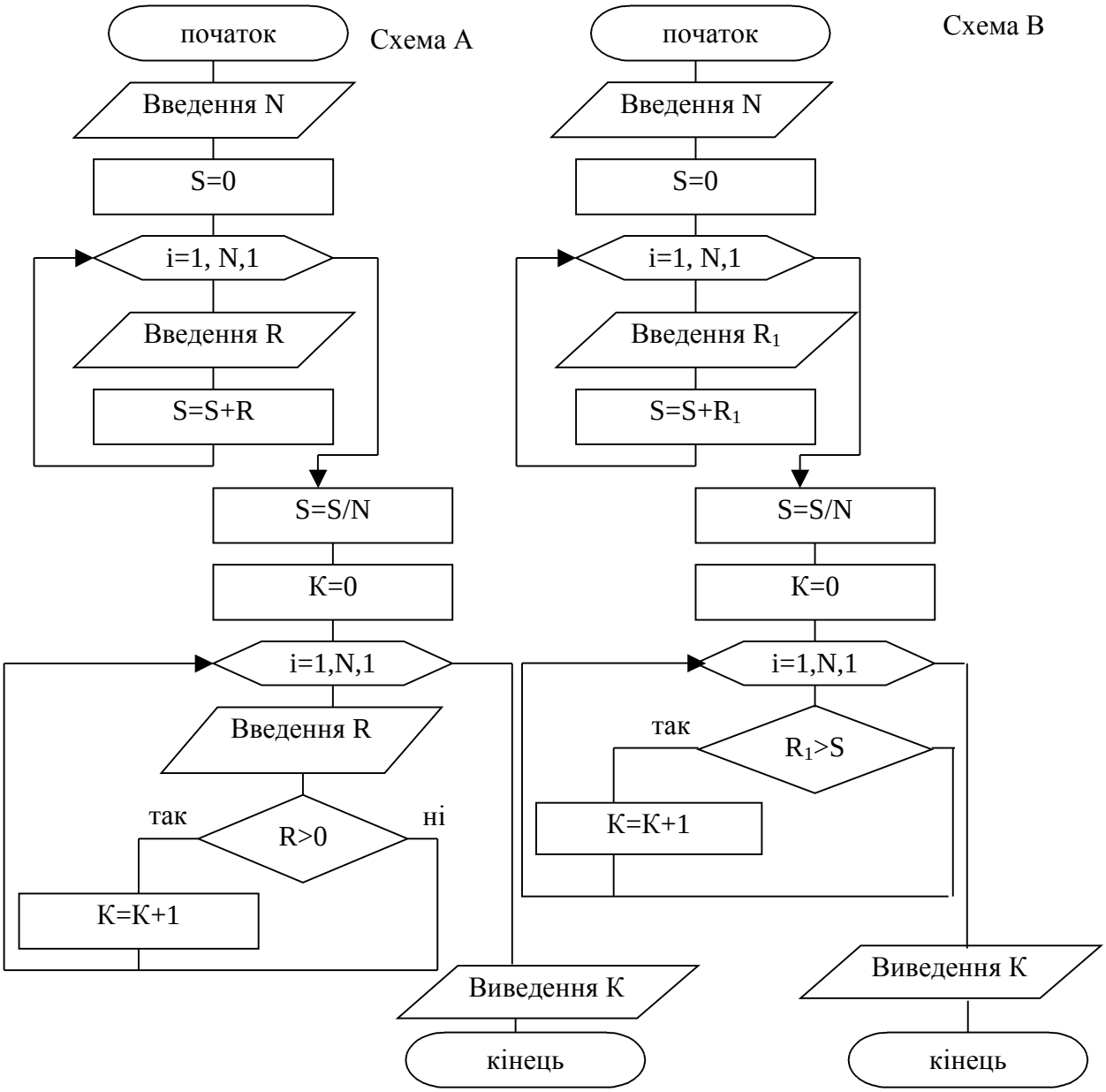


Рисунок 18 – Схема до задачі 7

Задача 8. Задана послідовність з n чисел. Визначити кількість додатних, від'ємних і нульових елементів.

Розробка алгоритму. Початкові дані – одновимірний масив з N елементів. Назвемо його А. Елементи масиву – a_1, a_2, a_3, ..., a_n. Ввести послідовність – це значить ввести всі елементи одновимірного масиву. У

схемі (рис. 19) присутній елемент

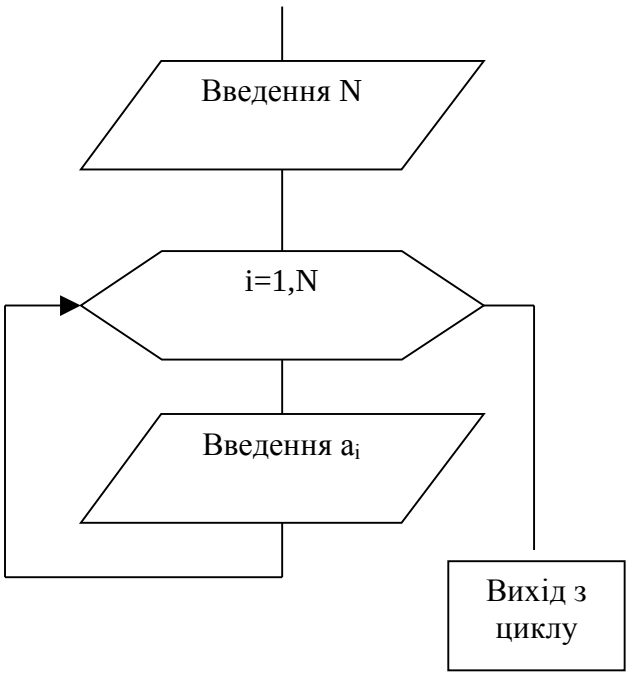


Рисунок 19

Цей елемент повинен повторитися для всіх і від 1 до N. Отже, необхідно організувати цикл. Число елементів послідовності повинно бути задане раніше, оскільки це кінцеве значення параметра в циклі введення елементів. Схема введення одновимірного масиву виглядає таким чином (рис. 19).

Якщо крок і параметра в циклі рівний одиниці (+1), то в блоці «модифікація" крок можна не вказувати. Крок 1 прийнято приймати за замовчуванням.

Щоб визначити кількість від'ємних елементів, потрібно перебрати всі елементи і перевірити умову $a_i<0$. Якщо умова виконується, то такий елемент потрібно порахувати, тобто, як в попередній задачі, відкласти 1 у відведену для підрахунку змінну. К – кількість від'ємних

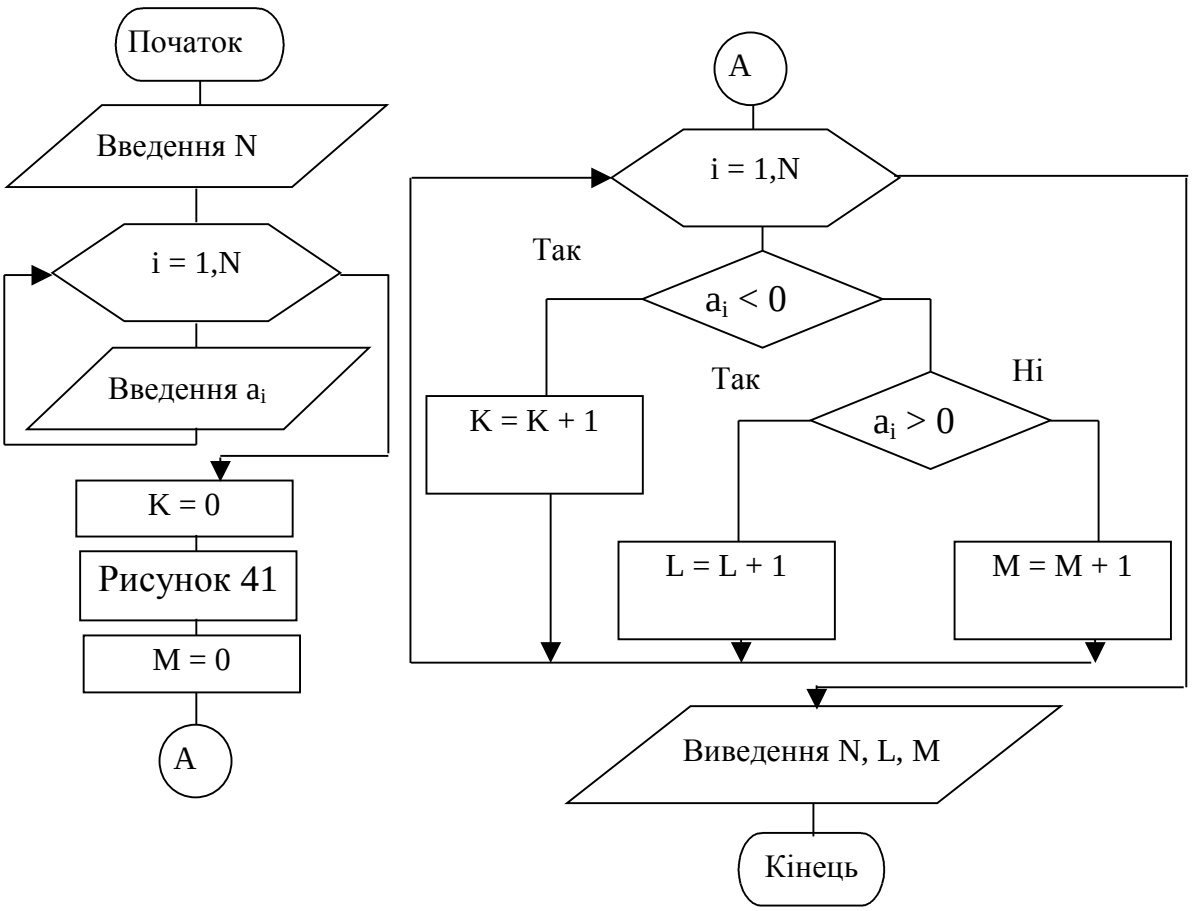


Рисунок 41

Рисунок 20

елементів, L – кількість додатних елементів, M – кількість нульових елементів. Для підрахунку L і M не потрібно організовувати нові цикли. Підрахунок кількості додатних, від'ємних, і нульових елементів організований в одному циклі.

(А) – з'єднувач. При великій насиченості схеми символами окремі лінії потоку допускається обривати. При цьому в кінці (початку) обриву повинен бути поміщений символ (А) – з'єднувач.

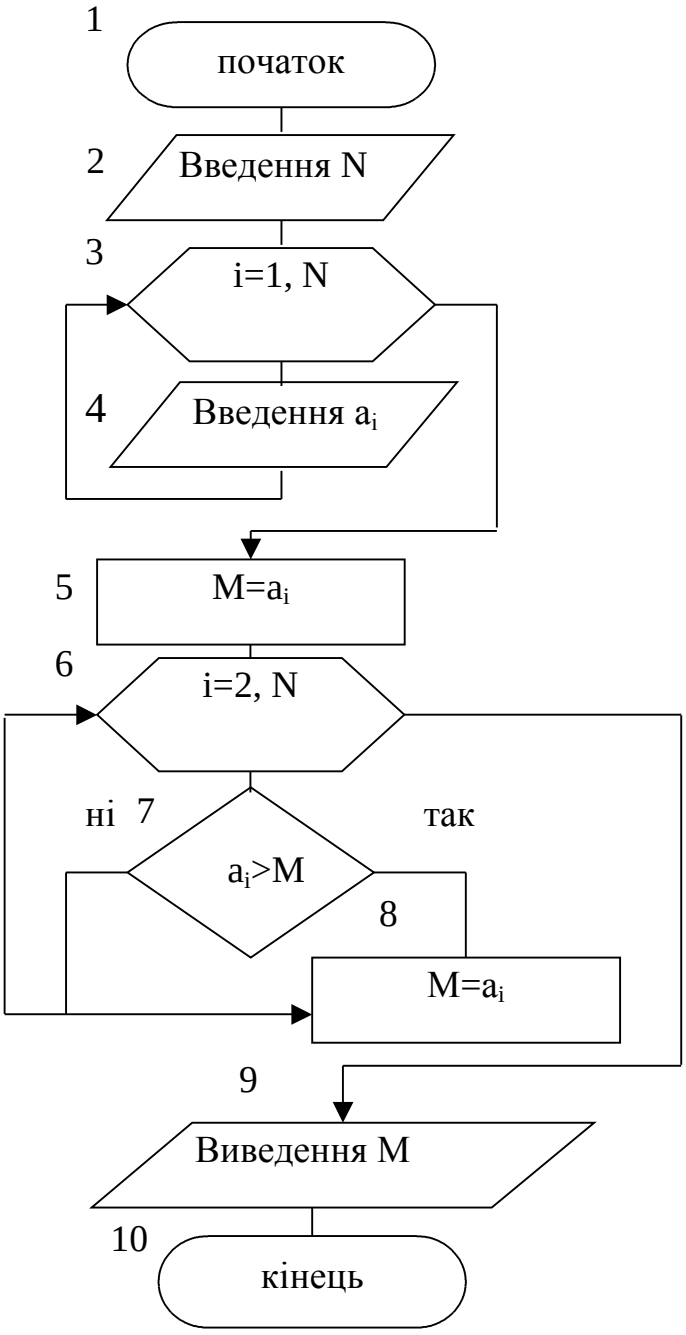


Рисунок 21

Задача 9. Знайти найбільший елемент послідовності з N елементів. Розробка алгоритму. Схема введення послідовності чисел (одновимірного масиву) розглянута в попередній задачі. Для пошуку максимального елементу введемо допоміжну змінну M (рис. 21).

Привласнимо M значення першого елементу і далі перебиратимемо всі елементи послідовності від 2 до N, порівнюючи a і M. Якщо a виявиться більше ніж M, то M потрібно привласнити значення a_i. У протилежному випадку здійснюється перехід до наступного елементу послідовності. Після закінчення циклу M прийме значення найбільшого елементу.

Задача 10. Задана послідовність $a_1,a_2,...,a_N$. Розташувати елементи в спадній послідовності. *Розробка алгоритму* (рис. 22). Скористаємося алгоритмом розробленим в попередній задачі. У початковій послідовності знайдемо найбільший елемент і запам'ятаємо його порядковий номер K. Нехай це буде a_k. Поміняємо a_k місцями з елементом a_1. Найбільший елемент виявився на

першому місці. Таку процедуру потрібно виконати з елементами, що залишилися, від 2 до N. У цьому випадку найбільший елемент а потрібно поміняти місцями з другим елементом. На другому місці виявиться другий за величиною елемент. Далі шукаємо найбільший елемент послідовності від 3 до N. Виходить, що процедуру пошуку найбільшого елементу потрібно повторити N-1 раз. Останній елемент сам з собою можна не порівнювати. Для запам'ятовування порядкового номера максимального елемнту використовуємо змінну К. Блоки 2, 3, 4 – введення одновимірного масиву. Блок 5– створюється цикл для визначення порядкового номера елементу, з якого починається пошук максимального. Блоки 6, 7–задаються початкові значення для М і К. У М записується значення найбільшого еле-менту, в К – порядковий номер цього елементу. Блоки 8, 9, 10 – алгоритм пошуку максимального елементу послідовності. Блок 11 – запам'ятовуємо порядковий номер елементу, записаного в М. Блок 12 – на місце максимального елементу записуємо і-ий елемент (з і-го елементу починається пошук максимального). Блоки 14, 15– виведення послідовності.

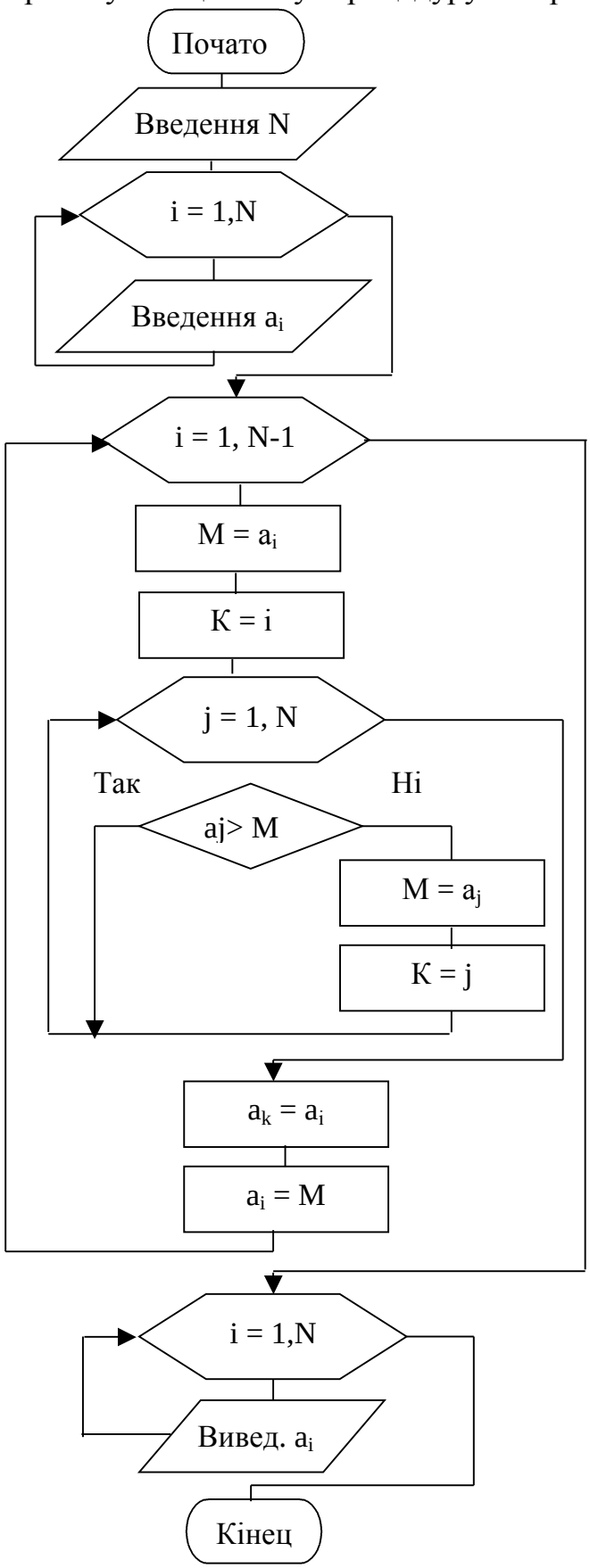


Рисунок 22

Слід звернути увагу на те, що блоки 12 і 13 не можна поміняти місцями. Якщо це зробити, то значення а буде втрачено.

Задача 11. Обчислити суму нескінченного ряду

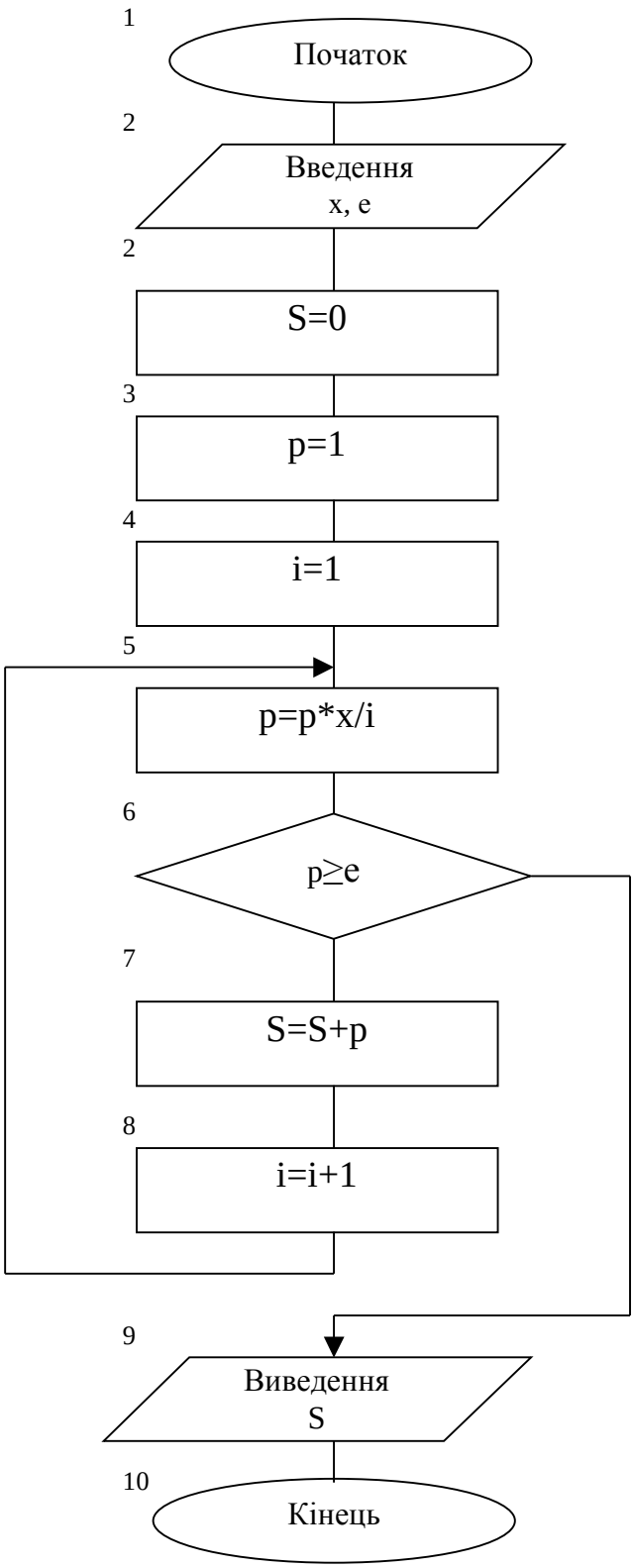


Рисунок 23 – Схема алгоритму циклічної структури типу “Поки”

$X, \frac{X^2}{2!}, \frac{X^3}{3!}, ..., \frac{X^n}{n!}$ з точністю ε.

Розробка алгоритму (рис.23). Початковими даними є X і ε. Дотепер цикли мали задане число повторень. У даній задачі число повторень невідоме, оскільки невідомий порядковий номер останнього доданку. Необхідно підрахувати суму членів послідовності, задовольняючих умову $\frac{X^n}{n!} \geq \varepsilon$. Кожен член послідовності повинен бути одержаний з попереднього члена цієї послідовності.

$$P_1 = \frac{X}{1};\ \ P_2 = \frac{X}{1} * \frac{X}{2};\ \ P_3 = \frac{X^2}{2} * \frac{X}{3}$$

Така форма запису, в якій кожен наступний елемент розрахунку може одержатись з попереднього, називається **рекурентною формою**.

Для даної задачі $P_i = P_{i-1} * \frac{X}{i}$... Для розрахунку величини поточного члена послідовності необхідно знати його порядковий номер і значення попереднього члена.

Блок 3 – обнулюємо початкове значення суми. Блок 4– допоміжна дія. Завдання початкового значення для P. Блок 5 – завдання порядкового номера члена послідовності. Блок 6 – розрахунок величини поточного члена послідовності. Блок 7 – порівняння поточного члена послідовності з заданою

точністю. Якщо $P \geq \varepsilon$, то такий елемент потрібно підсумовувати, якщо ні, то задана точність досягнута і розрахунок можна закінчити. Блок 8 – нагромадження суми. Блок 9 – перехід до наступного поточного номера і потім до блоку 6.

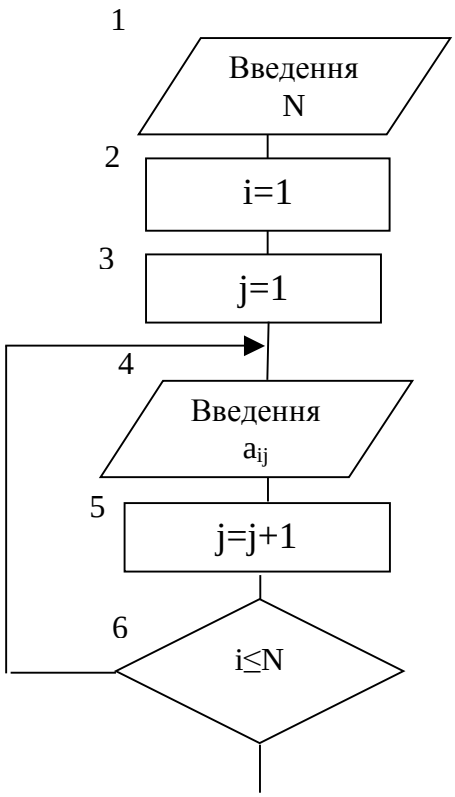
У даному прикладі організована циклічна структура типу "Поки".

Повторення послідовності операторів перевіркою умови на початку кожного проходу циклу називається **ітерацією.**

2.3.4 Алгоритми із структурою вкладених циклів

Будь-який цикл, що містить усередині себе один або декілька інших циклів, називається **вкладеним.** Цикл, що охоплює інші цикли, називається **зовнішнім** а решта циклів – **внутрішніми**. Правила організації як для зовнішнього, так і внутрішнього циклів такі ж, як і для простого циклу. Параметри цих циклів змінюються одночасно, тобто при одному значенні параметра зовнішнього циклу параметр внутрішнього циклу приймає по черзі всі свої значення.

Матриці – зручний приклад для демонстрації алгоритмів із структурою вкладених циклів.



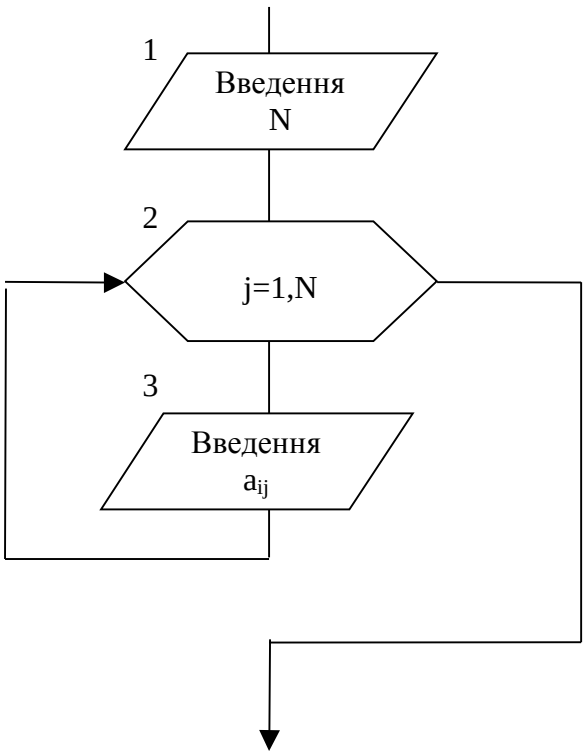


Схема без блоку модифікації Схема з використанням блоку

Рисунок 24

У матриці [A] M стовпців і N рядків. Кожен елемент матриці визначений двома індексами, які вказують положення цього елементу. На першому місці записується номер рядка, на другому – номер стовпця. Наприклад, a – елемент матриці [A], розташований в другому рядку третього стовпця.

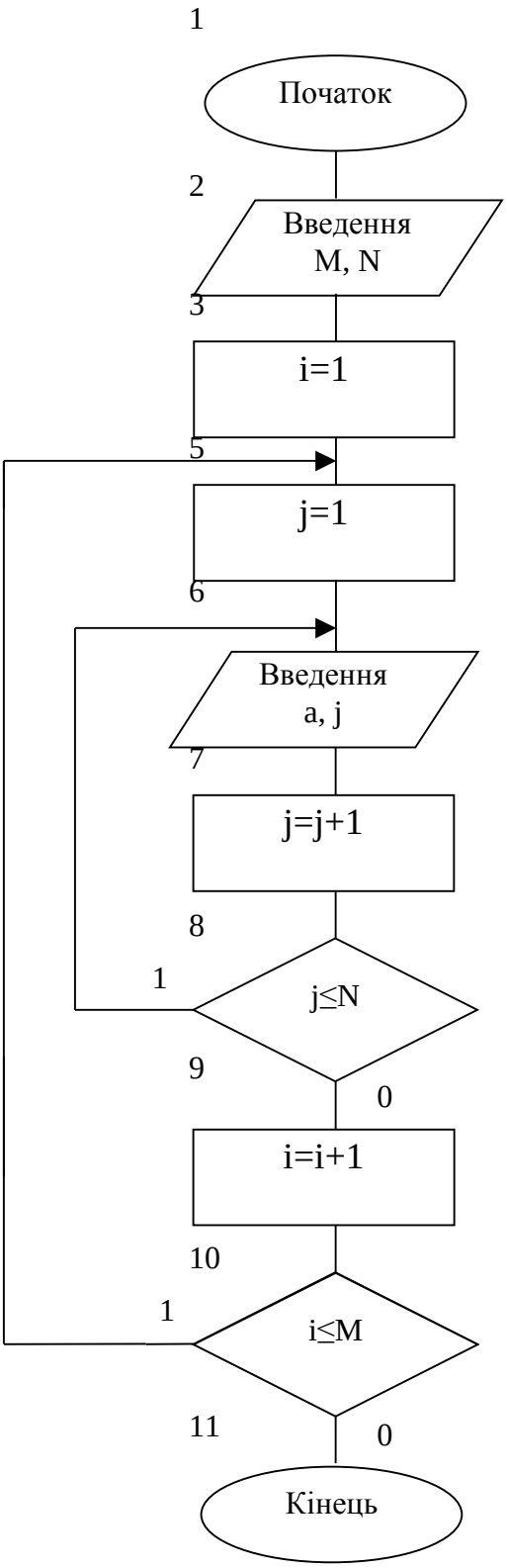


Рисунок 25 – Введення матриці

Якщо необхідно задати значення матриці, значить потрібно задати значення всіх її елементів. Назвемо елемент матриці a_{ij}. Для всієї матриці в цілому і пробігає значення від 1 до N, а j від 1 до M.

Кожен рядок матриці можна розглядати як одновимірний масив.

З введенням одновимірного масиву ми знайомі. Введення першого рядка матриці визначає наступна сукупність елементів схеми (рис. 24).

Така ж схема повинна бути виконана для другого рядка матриці, для третього і так далі до M–ого рядка. Тобто номер рядка може бути заданий циклом і цей цикл буде зовнішнім по відношенню до циклу по стовпцях. Схема введення матриці [A] без блоку модифікації подана на рис. 25.

Зовнішній цикл по i – блоки 3, 4, 5, 6, 7, 8, 9.

Блок 3 – підготовка циклу по i.

Блок 8 – зміна параметра циклу.

Блок 9 – перевірка умови закінчення циклу.

Блоки 4, 5, 6, 7, 8 – тіло циклу.

Внутрішній цикл по j – блоки 4, 5, 6, 7.

Блок 4 – підготовка циклу по j.

Блок 6 – зміна параметра циклу.

Блок 7 – перевірка умови закінчення циклу.

Блоки 5, 6 – тіло циклу.

При переході до наступного значення i (до наступного рядка) j – відновлює первинне значення рівне 1.

Задана матриця

$$[A]=\begin{vmatrix}1&0&2&3\\0&1&1&1\\5&4&6&0\end{vmatrix}$$

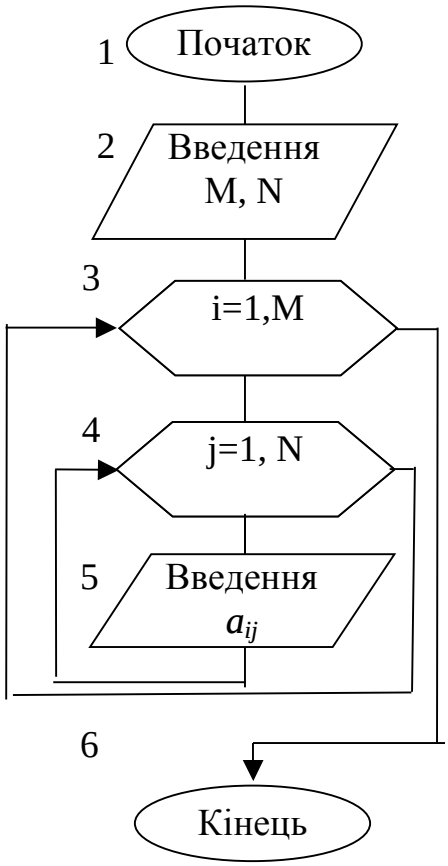


Рисунок 26 – Введення з блоком модифікації

При використанні розробленої схеми алгоритму елементи матриці повинні бути введені в такому порядку: 1, 0, 2, 3, 0, 1,1, 1, 5, 4, 6, 0.

Для того, щоб ввести елементи матриці по стовпцях (1,0,5,0,1,4,2,1,6,3,1,0), потрібно цикл по j зробити зовнішнім, а цикл по і внутрішнім. В цьому випадку другий індекс змінюється повільніше першого.

a_{11}, a_{21}, a_{31},…,a_{M1}, a_{12}, a_{22}, a_{32},…,a_{M2},…a_{MN}

Зустрічаються задачі, в яких не має значення, за яким параметром організувати зовнішній і внутрішній цикл. Але бувають випадки, коли це принципово.

При використанні блоку модифікації схема виглядає таким чином (рис. 26).

Задача 12. Задана квадратна матриця [A], що складається з N рядків і M стовпців. Знайти суму діагональних елементів.

Розробка алгоритму. Початкові дані: кількість рядків N, кількість стовпців N, елементи матриці.

Діагональні елементи матриці a_{11}, a_{22}, a_{33},..., a_{MN} задовільняють умову i = j.
Необхідно здійснити перебір елементів матриці і підсумувати ті, що відповідають умові i = j.

Блоки 2, 3, 4, 5 – введення матриці [A].

Блок 6 – завдання початкового значення для суми.

Блоки 7, 8 – цикли, що здійснюють перебір елементів матриці.

Блок 9 – перевірка умови для підсумовування.

Блок 10 – нагромадження суми.

Блок 11 – виведення суми.

Схема (рис. 27) значно спрощується, якщо для позначення індексу рядка і індексу стовпця використовувати одну і ту ж змінну. В цьому випадку можна обмежитися одним циклом при підсумовуванні.

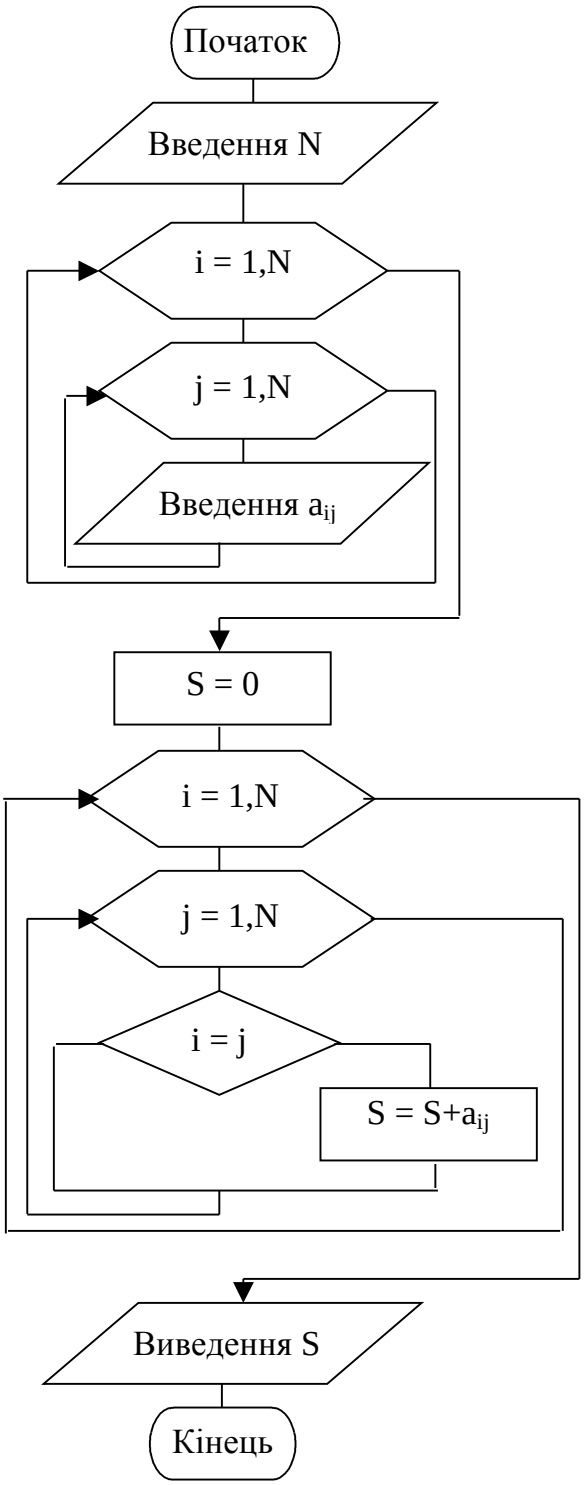


Рисунок 27 – Схема задачі 12

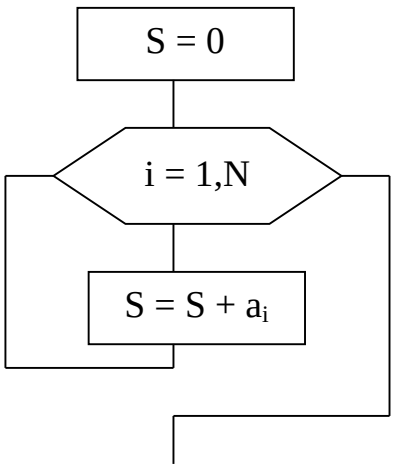
Задача 13. Група з N студентів здавала в зимову сесію M іспитів. Відомі результати іспитів. Підрахувати середній бал кожного студента в зимову сесію.

Розробка алгоритму. Результати іспитів є матрицею, що складається з N рядків і M стовпців. Рядок є оцінками кожного студента, одержані ним за предмети сесії. Номер стовпця – це номер предмета. Задача полягає в тому, щоб знайти суму елементів кожного рядка.

$$S_i = \sum_{i=1}^{M} a_{ij}.$$

Тоді середнє: $S_i = S_i / M$.

Фрагмент схеми



Результатом розрахунку буде одновимірний масив S, що складається з N елементів (рис. 28).

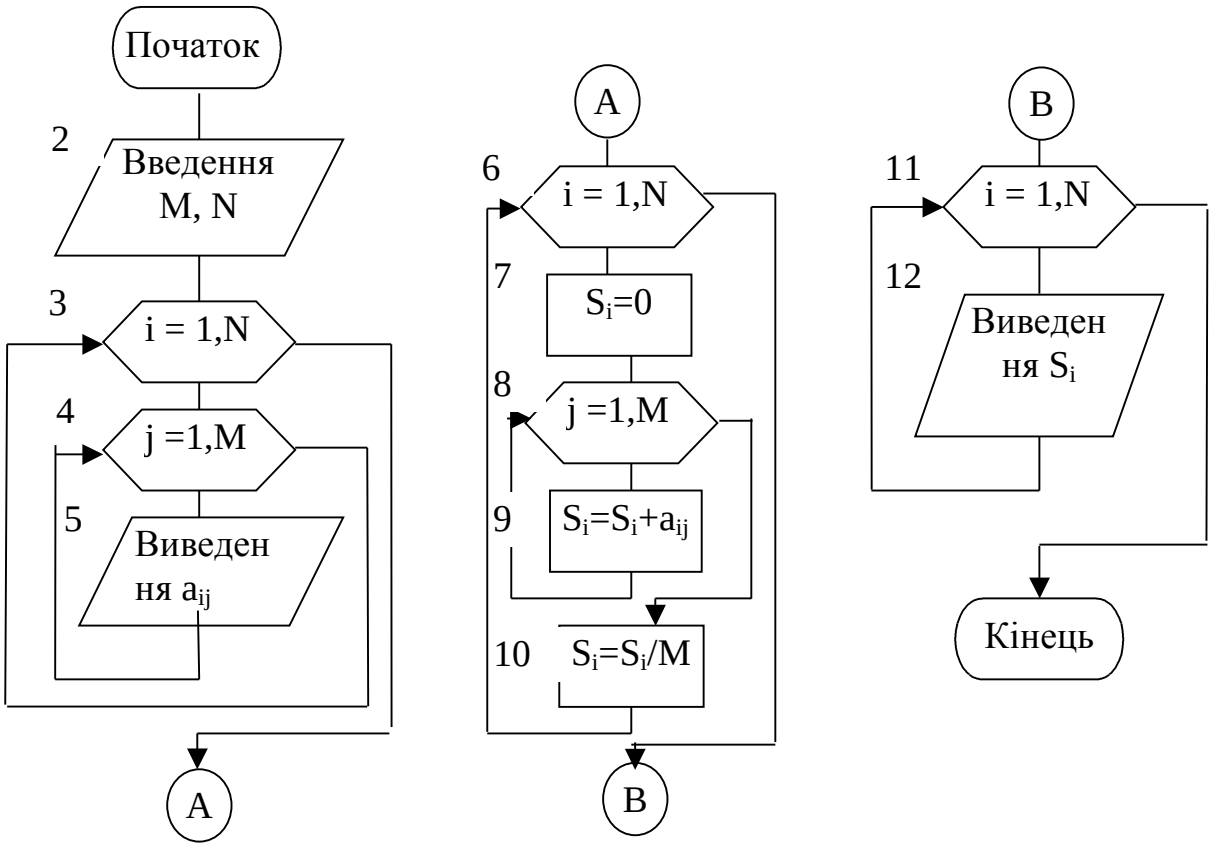


Рисунок 28 – Схема задачі 13

Блок 6 – цикл по рядках. У задачі i – це номер студента у відомості.
Блок 7 – занулення суми. S – змінна, в якій накопичується сума оцінок 1-го студента.
Блок 8 – перебір предметів від 1 до M.
Блок 9 – підсумовування оцінок.
Блок 10 – визначення середнього. Після того, як цикл по j відпрацював, одержана сума ділиться на кількість предметів.
Блоки 11, 12 – виведення одновимірного масиву.

Задача 14. Задана матриця [A], що складається з N рядків і M стовпців. Визначити кількість нульових елементів в парних стовпцях.

Розробка алгоритму (рис. 29). Щоб проглянути парні стовпці, необхідно організувати цикл з кроком 2, починаючи з другого.

Блоки 2, 3, 4, 5 – введення матриці. Блок 6 – початкове значення для змінної K. K – це кількість нульових елементів. Блок 7 – цикл для переглядання матриці по рядках. Блок 8 – цикл для переглядання матриці по стовпцях. Перебираємо тільки парні стовпці. Блок 9 – перевірка рівності нулю елементів матриці. Блок 10 підрахунок кількості, блок 11 – виведення результату.

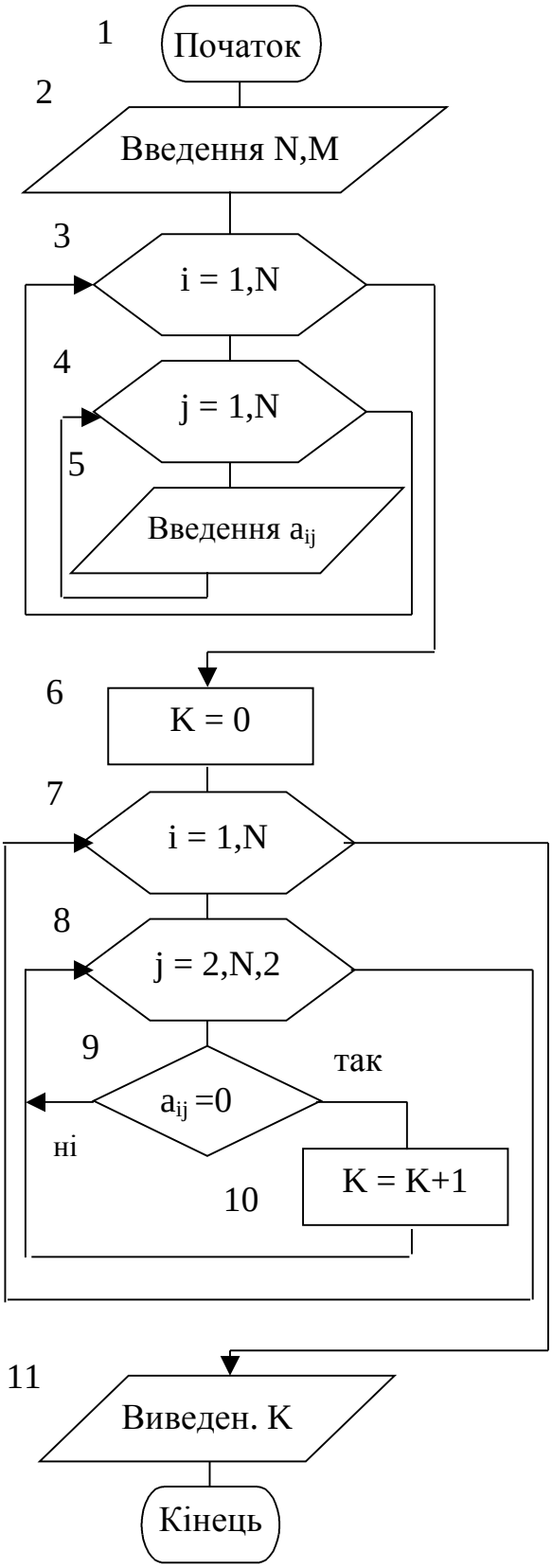
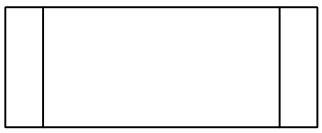


Рисунок 29 – Схема задачі 14

2.3.5 Підпорядковані алгоритми

При записі алгоритмів можуть використовуватися алгоритми, складені раніше. Алгоритми, цілком використані у складі інших алгоритмів, називаються **підпорядкованими алгоритмами** або **підалгоритмами**. Не виключено, що алгоритм, що містить в своєму описі підпорядковані алгоритми, сам може виступати в ролі підалгоритму. У схемах підалгоритм зображається символом:



Таким блоком позначаються в схемах виклики підпрограм, які використовуються в програмах.

Задача 15. Скласти алгоритм обчислення числа сполучень. Число сполучень розраховується за форму-лою:

$$C_n^m = \frac{n!}{m!(n-m)!}.$$

Обчислення факторіалу оформити у вигляді підалгоритму. У блоках "початок" і "кінець" в під порядкованому алгоритмі записуються слова "вхід" і "вихід" (рис.30 В).

У даній задачі необхідно тричі обчислити факторіал: n!, m! і (n-m)!. Для розрахунку факторіала роз-роблений підалгоритм. Замість повного запису послідовності блоків підалгоритму, який повинен повторюватися тричі в основній

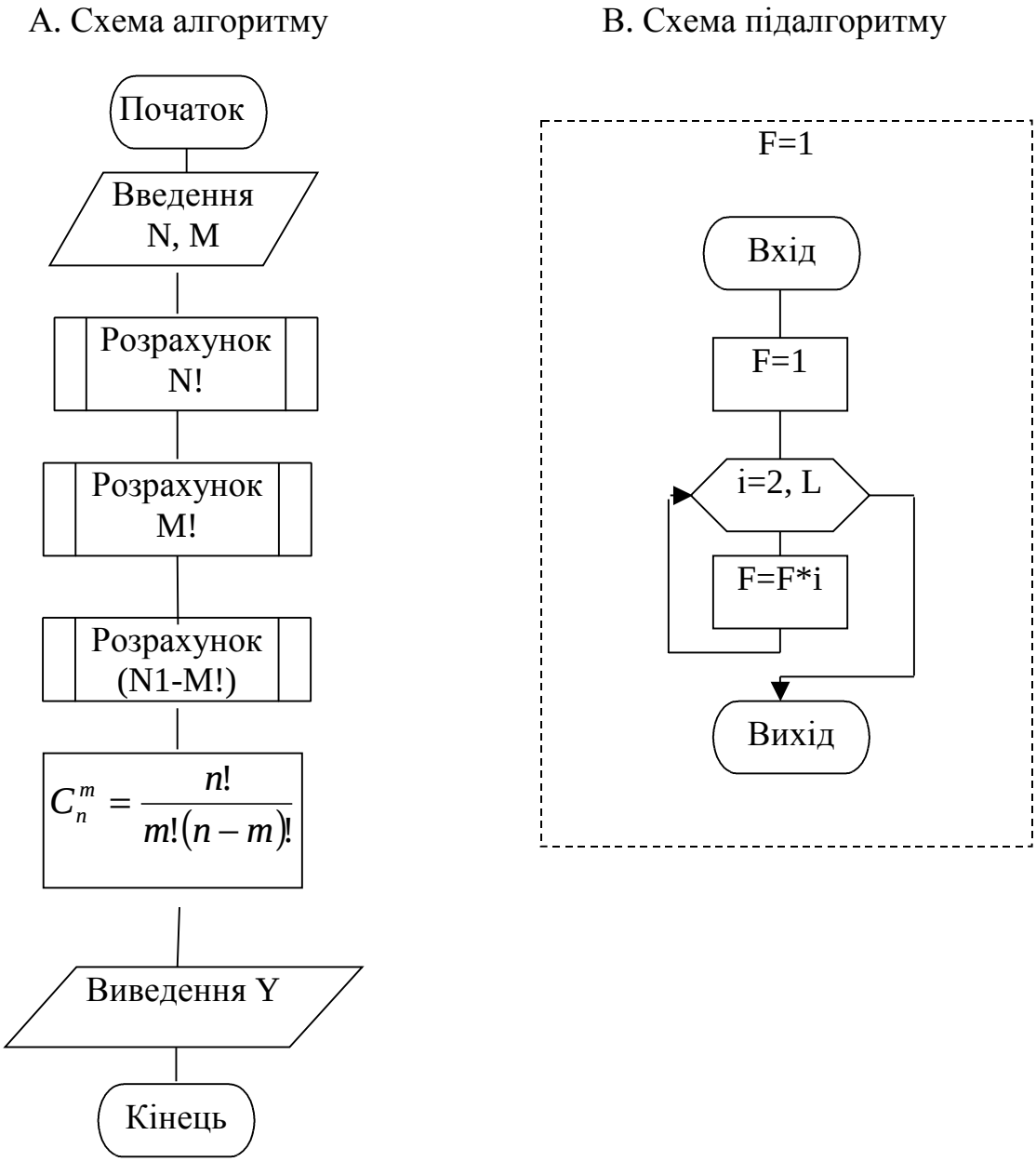


Рисунок 30

схемі, введені блоки звертання до підалгоритму.

Використання підалгоритмів знаходить широке застосування в практиці алгоритмізації і є одним з найзначніших і найцікавіших прийомів.

Одним з прийомів розробки алгоритму розв’язування складних задач є метод покрокової деталізації, коли спочатку продумується і фіксується загальна структура алгоритму без детального опрацьовування окремих його частин, але при цьому також використовуються лише основні структури алгоритмів. Блоки, що вимагають подальшої деталізації, позначаються пунктирною лінією. Далі опрацьовуються (деталізують) окремі блоки, не деталізовані на попередньому кроці. Повністю закінчивши деталізацію всіх блоків, одержимо вирішення всієї задачі в цілому.

3 ТУРБО ПАСКАЛЬ – МОВА ВИСОКОГО РІВНЯ

3.1. Системи програмування

За наявності десятків тисяч прикладних програм часто доводиться стикатися з ситуацією, коли існуючі програми не задовольняють або роблять щось недостатньо швидко, або неефективно. У цій ситуації єдиний вихід – написання власної програми.

За своєю природою комп'ютер може виконувати тільки прості операції, які можна вводити одну за одною в його пам'ять прямо в машинних кодах. Виснажлива монотонність такої роботи привела колись перших програмістів до єдиного рішення – створення асемблерів, тобто засобів, що спрощують підготовку машинних кодів програм користувачів за рахунок написання їх в деяких мнемонічних позначеннях з подальшим автоматичним перекладом. Подальший розвиток цих ідей привів до створення мов програмування високого рівня, в яких довгі і складні послідовності машинних операцій були замінені одним єдиним словом – оператор.

Спеціальні програми забезпечують опосередковане "розуміння" обчислювальною машиною інших мов програмування шляхом перекладу текстів, складених на цих мовах, в тексти на машинній мові.

Під **системою програмування** розуміють сукупність мови програмування і віртуальної машини, що забезпечує виконання реальною машиною програм, складених на цій мові.

Мовою програмування називають систему позначень для точного опису алгоритмів для ЕОМ. Ці мови є штучними мовами із строго певним синтаксисом (побудова речень і правила поєднання слів) і семантикою (смислове значення слів і зворотів мови), тому вони не допускають вільного тлумачення конструкцій, характерних для природної мови (мови спілкування між людьми).

Віртуальна машина – це програмний комплекс, що зв'язує вхідну мову ЕОМ з іншою, машинною мовою. Віртуальна машина містить *транслятор* та/або *інтерпретатор* і може включати бібліотеки підпрограм, налагоджувач, компонувальник і інші сервісні засоби.

Транслятор є програмою, що здійснює переклад текстів з однієї мови на іншу. У системі програмування транслятор перекладає програму з вхідної мови цієї системи на машинну мову реальної ЕОМ (на якій функціонує дана система програмування або функціонуватиме програма, що розробляється) або на проміжну мову програмування, вже реалізовану, або яка підлягає реалізації. Одним з різновидів транслятора є **компілятор**, **що** забезпечує переклад програм з мови високого рівня (наближеного до людини) на мову нижчого рівня (близький до ЕОМ), або машинозалежна мова. Інший різновид транслятора – **асемблер**, **що** здійснює переклад програм з мови низького рівня (мови асемблера) на машинну мову, що має

приблизно той же рівень. Деякі транслятори служать для перенесення програм з однієї машини на іншу. Програма, що подається на вхід транслятора, називається **початковою**, а результат трансляції – **об'єктною програмою**.

Діаметрально протилежними характеристиками володіє альтернативний засіб реалізації мови – інтерпретатор. **Інтерпретатор** є програмним продуктом, що виконує задану програму шляхом одночасного її аналізу і реалізації передбачених нею дій. При використанні інтерпретатора відсутнє розділення на дві стадії (переклад і виконання) і, більш того, відсутній явний переклад програми навіть по частинах перед черговим етапом виконання. Насправді ж розпізнається чергова конструкція програми і інтерпретатором виконуються визначувані нею дії. Після цього процес аналізу і реалізації вказаних дій циклічно повторюються.

Можливі і змішані стратегії реалізації мов програмування, наприклад, трансляція на проміжну мову з подальшою інтерпретацією проміжної програми.

Програма на мові програмування складається з послідовності операторів (інструкцій), які задають ті або інші дії. Основним є оператор привласнення, який використовується для зміни вмісту областей пам'яті.

Виконання програми зводиться до послідовного виконання операторів з метою перетворення початкового стану пам'яті (тобто значень змінних) в завершальне. Таким чином, з точки зору програміста є програма і пам'ять, причому перша послідовно оновлює зміст останньої.

Однією з найважливіших класифікаційних ознак процедурних мов є їх рівень. **Рівень мови програмування** визначається семантичною (смисловою) ємністю його конструкцій і його орієнтацією на програміста-людину. Мова програмування (частково) ліквідовує семантичний розрив між методами вирішення задач машиною і людиною. Мови програмування, не залежні від особливостей конкретної машини орієнтовані на широкий круг користувачів, вважаються *мовами високого рівня* (по відношенню до рівня машинних команд ЕОМ). Чим більше мова орієнтована на програміста, тим вищий його рівень.

Двійкова мова є не що інше, як безпосередньо машинна мова. В наш час такі мови програмістами не застосовуються.

Шістнадцяткова мова забезпечує деяке спрощення запису програми на машинній мові шляхом представлення чотирьох двійкових цифр однією шістнадцятковою. Ця мова використовується як доповнення до мов високого рівня таких, як Паскаль або СІ для програмування критичних до часу виконання фрагментів алгоритмів.

Мова Асеблер – це мова, призначена для подання в легкій для читання формі програм, записаних на машинній мові. Вона дозволяє програмісту користуватися мнемонічними кодами операцій, на свій розсуд привласнювати символічні імена регістрам ЕОМ і елементам пам'яті, а

також задавати найзручніші в тому або іншому контексті схеми адресації. Крім того, мова Асемблер забезпечує представлення констант в різних системах числення (наприклад, в десятковій або шістнадцятковій).

Мова Макроасемблер є розширенням мови Асемблер за рахунок включення макрозасобів і надає засоби визначення і використовування нових, могутніших команд як послідовностей базових інструкцій, які дещо підвищують її рівень.

Мови Асемблер і Макроасемблер застосовуються системними програмістами-професіоналами з метою використовування всіх можливостей устаткування ЕОМ і отримання ефективної програми, як за часом виконання, так і за потрібним об'ємом пам'яті.

3.1.1 Історія мов програмування

Приведемо короткий перелік найпоширеніших мов програмування високого рівня, які мають переважне розповсюдження в наш час, причому мають по декілька версій.

Бейсік є простою мовою програмування, розробленою в 1964 році для використовування новачками. Робота в середовищі Бейсіку спочатку велася тільки в режимі інтерактивної (діалогової) інтерпретації. В наш час є і компілятори з цієї мови. У цій мові широко використовуються різного роду умовчування, що вважається поганим тоном в більшості сучасних мов. Не дивлячись на це, Бейсік дуже популярний, особливо на ПЕВМ. Існує безліч його діалектів. Бейсік є однією з найдинамічніших мов. Не без підстав цю мову іноді порівнюють з пітоном, що заковтує і переварює все нове, що з'являється в інших мовах програмування. Рівень Бейсіку не можна визначити однозначно. Сучасні діалекти дуже розвинені і мало чим нагадують свого предка.

Мова **Фортран** була розроблена в 1956 році, потім з'явилися нові його версії Фортран-2, Фортран-IV, Фортран-66, Фортран-77, Фортран-8X, Фортран-88. Свого часу ця мова була справді "робочим конем" науковців і широко використовується в наш час, не дивлячись на його обмеженість і корявість. Він надає користувачам великі можливості для обробки числових даних, особливо комплексних чисел. Ще у версії Фортран-2 вперше була реалізована ідея роздільної компіляції модулів, що дало можливість створювати бібліотеки наукових підпрограм.

Мова програмування **СІ** спочатку розроблена в 70-х роках. В наш час в СІ поєднуються переваги сучасних високорівневих мов і можливість доступу до апаратних засобів машини на рівні, який звичайно асоціюється з мовою Асемблер. СІ має синтаксис, що забезпечує надзвичайну стислість програм, а компілятори, внаслідок особливостей мови, здатні генерувати швидкі і ефективні програми на машинному коді.

Мова програмування **APL** створена в 1969 році. До числа її основних переваг відносяться багатий набір могутніх операторів, що дозволяють працювати з багатовимірними масивами як з єдиним цілим, а також

надання користувачу можливості визначати власних операторів. Основне його призначення – обробка масивів.

FORTH – гнучка і достатньо проста мова, розроблена в 1971 році. Важлива її особливість – відвертість (розширюваність). Програміст може додавати нові операції, типи даних і операторів. Останнє досягається шляхом скріплення будь-якого рядка програми із заданим програмістом словом, яке потім може використовуватися нарівні із стандартними операторами. Проте розширення мови веде до зниження ефективності.

В 1980 році з'явилася мова **Ада**. Названа вона на згадку про Аду Ловлейс – дочку англійського поета Лорда Байрона, першу програмістку в історії обчислювальної техніки. Вона була створена у Франції за замовленням американського міністерства оборони як універсальна мова програмування. Це найновіша та найсильніша з мов програмування, вона успадкувала якості мов Паскаль та Алгол-68 і додатково отримала багато інших якостей: **системне програмування, паралельність та інші.**

Мова **Ліспі** (list procssing language – мова обробки списків), розроблена американським професором Джоном Маккарті в 1961 році і мова **Пролог**(prolog – progranimetion en logigue – логічне програмування), розроблена Колмероє та іншими ученими університету Люмшини у Франції в 1973 році – це основні мови для задач, пов'язаних з **штучним інтелектом.**

Ліспі оперує списками (ланцюгова послідовність елементів), а Пролог – деревами (логічними розгалуженнями).

Працюючи в середовищі даних мов, комп'ютер розв'язує задачу, а програміст лише її формує. ЕОМ пробігає весь робочий простір в пошуках розв'язку спеціальним способом, зате повністю вичерпаним (можливий зворотний хід).

Існує велика множина спеціалізованих мов, які дозволяють ефективно розв'язувати задачі в деяких областях: моделювання (мови Симула, Симкрит, GPSS), управління апаратурою (ФОРТ), для написання системних програм (Сі), навчання програмуванню (Лого, Робік, алгоритмічна мова, А.П. Єршова) та інші.

На сьогодні у багатьох галузях почали широко використовуватись так звані системи програмування на відомих мовах.

Останнім часом почали з'являтися системи програмування на мові JAVA (symantec café, microsoft J++ та інші). Вони дозволяють створювати так звані JAVA – додатки (аплети) для WEB – сторінок в Інтернеті. Ці додатки можуть викликатись при перегляді WEB – сторінок і виконуватись на будь-якому комп'ютері, незалежно від операційної системи або типу мікропроцесора цього комп'ютера.

Частіше всього це робиться для "оживлення" WEB – сторінок, тобто введення в них елементів анімації, але можуть бути і інші застосування.

Особливим класом систем програмування є системи для створення додатків типу клієнт – сервер. Ці системи дозволяють швидко створювати

інформаційні системи для підрозділів і навіть великих підприємств. В них містяться засоби для створення інтерфейсу користувача, опис процедур обробки даних, заготовки для виконання типових дій для обробки даних і т.д. Ці системи, як правило, дозволяють працювати з різними СУБД. Серед найпоширеніших систем такого типу можна назвати POWERBUILDER фірми Sybase, Delphi фірми Borland, Visual Basic фірми Microsoft, SQL – Windows фірми Centura.

Але в цілому еволюція машинних мов відбувається у напрямку природної мови – це ідеальне рішення, яке полягає в тому, що користувачеві потрібно буде лише сформувати задачу на природній мові, а все інше зробить комп'ютер.

Звичайно, в наш час це всього лиш мрія, проте мови останніх поколінь більш близькі до мови людського мислення.

3.2 Характеристика мови програмування Паскаль

Однією з найпопулярніших мов програмування є мова **Паскаль.** Перша версія мови програмування Паскаль була розроблена на кафедрі інформатики Стенфордського університету швейцарським ученим Ніклаусом Віртом в 1968 році, і названа на честь французького ученого Блеза Паскаля. Пройшло багато часу з моменту появи Паскаля на ринку програмних продуктів, перш ніж він одержав загальне визнання внаслідок розробки мови програмування **Турбо Паскаль (ТП)** – діалекту мови, створеної американською фірмою Борланд. Ця фірма об'єднала дуже швидкий компілятор з редактором тексту і додала до стандартного Паскаля могутнє розширення, що сприяло успіху першої версії цієї мови. З тих пір Турбо Паскаль значно розширився. З'явилися нові графічні процедури, можливість використовування при написанні програм мови програмування низького рівня Асемблер, можливість створювати об'єктно-орієнтовані програми і багато іншого. У лінгвістичній концепції Паскаля пропагується системний підхід, що виражається, зокрема, в розкладанні крупних проблем на менші за складністю і розміром задачі, що легше розв'язуються. Набір операторів стандартного Паскаля відносно малий і легко вивчається. Але це породжує проблему розширення мови в додатках. У Турбо Паскалі ця проблема розв'язується за рахунок поставок великої кількості бібліотек різних процедур, готових до вживання в прикладних програмах.

Вплив Паскаля відчувається в наш час в різних мовах програмування. Так, серед нових діалектів Бейсіка є Паскаль з символікою Бейсіка. Навіть в мову СІ вбудовується все більше елементів, породжених Паскалем.

З моменту створення першої версії мови Паскаль пройшло багато часу і мова значно змінилася, але проте стандартний Паскаль є основою пізніших версій Турбо Паскаля. Надалі в описі мови зустрічатимуться обидва ці назви. Використовуватимемо назву Паскаль, якщо твердження

правильне і в стандартному Паскалі і в Турбо, а Турбо Паскаль, якщо в останніх версіях є відмінності. При вивченні складних конструкцій мови має сенс говорити тільки про Турбо Паскаль.

3.3. Алфавіт мови Паскаль

Будь-яка природна мова складається з декількох основних елементів: символів, слів, словосполучень і пропозицій. У алгоритмічній мові програмування є аналогічні структурні елементи: символи, слова, вирази (словосполучення) і оператори (пропозиції). При цьому слово утворюється з послідовності символів, вираз є групою слів, а оператор – певна комбінація слів і виразів.

Мова програмування Паскаль, як і будь-яка інша, має свій алфавіт. **Алфавітом мови програмування** називають набір символів (дозволений до використання і сприймання компілятором), за допомогою якого можуть бути утворені величини, вирази і оператори даної мови. Алфавіт мови Паскаль включає всі символи, представлені в кодовій таблиці, яка завантажена в оперативну пам'ять або зберігається в ПЗП комп'ютера. Кожному символу алфавіту відповідає індивідуальний числовий код від 0 до 255. Символи з кодами від 0 до 127 є так званою основною таблицею кодів ASCII. Їх склад і порядок визначені американським стандартом на коди обміну інформацією (ідентичні для всіх IBM-сумісних комп'ютерів).

Символи, які використовуються для складання ідентифікаторів:

– латинські рядкові і прописні літери;
– арабські цифри від 0 до 9;
– символ підкреслення (у Турбо Паскалі).

Символи розділювачі:

– проміжок, основне призначення якого розділення ключових слів і імен;

– керуючі символи (ASCII – коди від 0 до 31). Ці символи можуть застосовуватися при описі рядкових і символьних констант. Керуючі символи з ASCII – кодом 9 (табуляція), а також 10 і 13 (замикаючі рядок) використовуються як розділювачі при написанні програм.

Спеціальні символи, що виконують певні функції при побудові різних конструкцій мови:

+ - * / { } [ ] ( ) < > / ? ' : ; . ^ # @ $

Складові символи – група символів, які сприймаються компілятором як єдине ціле:

<= => := (* *) (. .) . .

"**Невживані" символи,** символи так званої розширеної таблиці ASCII, тобто символи, що мають коди від 128 до 255 (у цій області знаходяться символи алфавіту російської мови і символи псевдографіки на IBM-сумісних комп'ютерах), а також деякі символи з основної таблиці ASCII (наприклад,: &, !, %, " і інші). Їх можна використовувати в тексті коментарів і у вигляді значень констант рядків або констант символів.

Зарезервовані слова (BEGIN, END, PROGRAM і інші), що несуть певне смислове навантаження в мові програмування. **Зарезервоване слово** – це слово, яке в мові Паскаль має певне смислове значення. Ще говорять *службове слово* або *ключове слово* – це слова синоніми. Ім'я служить для позначення яких-небудь об'єктів. У мові Паскаль розрізняють два види імен: стандартні і ті, що даються користувачем ЕОМ.

3.4 Структура програми на мові Паскаль

Програми, написані на мові програмування Турбо Паскаль будуються відповідно до правил, що є дещо розширеними і "ослабленими" правилами синтаксису стандартного Паскаля. Приведемо приклад програми на Турбо Паскалі. Треба відразу підкреслити, що програма написана для процесу навчання, оскільки в життєвій ситуації вона не стане в нагоді.

```
PROGRAM Addition;
{ ADDITION.PAS - Програма підсумовування двох введених цілих чисел}
VAR
   Number_1, Number_2, Sum: INTEGER;
BEGIN
   Write ('Введіть перше число:');
   ReadLn (Number_1);
   Write ('Введіть друге число:');
   ReadLn (Number_2);
   Sum := Number_1 + Number_2;
   WriteLn (' Сума введених чисел рівна: ', Sum);
END.
```

Будь-яку програму, написану на Паскалі можна умовно розділити на дві основні частини:

- розділ оголошень і описів;
- розділ основного блоку.

У розділі оголошень і описів програміст повідомляє компілятор якими ідентифікаторами він позначає дані (константи і змінні, а також визначає власні типи даних, які він в майбутньому має намір використовувати в даній програмі. У Турбо Паскалі можливість підключати використовувані в програмі об'єкти, описані в іншому місці. Такі об'єкти називаються **модулями** і про них ми говоритимемо пізніше.

"Процедура" і "функція" – терміни, вживані в Паскалі для позначення спеціальним чином оформленої послідовності команд (підпрограми). Доступ до такої підпрограми може бути здійснений з будь-якого місця основного блоку програми, а також з будь-якої процедури функції, опис яких нижче. У розділі описів міститься опис процедур і функцій у вигляді тексту процедур і функцій, який будується за правилами аналогічним правилам побудови програми.

Основний блок програми складається з послідовності операторів,

причому робота програми починається саме з першого оператора основного блоку програми. Тіло основного блоку програми обмежене словами BEGIN і END.

Структура розглянутої програми має такий вигляд:

```
 PROGRAM Addition;
{Розділ описів}
BEGIN
{ Розділ операторів}
END.
```

Необхідно звернути увагу на наявність крапки після службового слова END. Після останнього оператора END завжди ставиться крапка, тим самим компілятор одержує інформацію про закінчення тексту програми.

Слово PROGRAM зарезервоване в Паскалі і означає початок програми. Далі записується ім'я програми (у приведеному прикладі – Addition). У Турбо Паскалі можна опускати оголошення імені оператором PROGRAM без яких-небудь наслідків для програми.

Рядки програми звичайно виділяють деякі смислові фрагменти тексту і можуть не зв'язуватися з конкретними діями в програмі. Програма записується у довільній формі, оператори не прив'язані до певної позиції рядка на відміну від інших мов програмування. Розташування тексту програми по рядках – справа смаку програміста, а не вимога синтаксису мови. В той же час рекомендується програму записувати в такій зовнішній формі, щоб її можна було легко читати і розуміти. Для цього широко використовуються проміжки, порожні рядки і коментарі. Рекомендується смислові частини виділяти однаковими відступами від початку рядка.

Проміжок в Паскалі використовується як розділювач окремих конструкцій мови, отже, необхідно уважно стежити за його наявністю як розділювача.

Відповідні рядкові і великі букви є еквівалентними, якщо тільки це не пов'язано з текстовими константами.

Розділювач відзначає кінець оператора або опису. Використовування особливого розділювача дозволяє розташовувати декілька операторів на одному рядку.

Після заголовка програми йде текст, укладений у фігурні дужки. Це коментар. Коментар – виділена фігурними дужками інформація для пояснення, яка не виконується програмою. Окрім фігурних дужок { }, можуть використовуватися також пари символів (*і*) зліва і праворуч від коментарю відповідно.

У нашому прикладі в розділі описів оголошені три змінні Number_1, Number_2, Sum як змінні цілого типу. Імена змінних записані через кому, а перед службовим словом INTEGER стоїть двокрапка.

Зарезервоване слово BEGIN в наступному рядку сигналізує компілятору про початок іншої частини програми – розділу операторів. Write, WriteLn, ReadLn служать для виведення і введення інформації і є

стандартними процедурами. У Турбо Паскалі є можливість використовування деяких стандартних процедур без передуючого опису цих процедур (загальне правило свідчить: всі процедури повинні бути описані). За своєю суттю оператор

Write ('Введіть перше число:');

є оператором звернення до вбудованої процедури виведення даних. Свою назву він одержав від write – записати, а writeln – записати рядок.

Процедура write здійснює вивидення об'єктів перерахованих в дужках через кому. В даному випадку виводиться текстова константа 'Введіть перше число:'.

ReadLn (Number_1); – стандартна процедура введення чисельного значення змінної з ім'ям Number_1. При виконанні програми машина надасть можливість ввести з клавіатури чисельне значення цієї змінної. Два наступні оператори аналогічні.

Оператор

Sum := Number_1 + Number_2;.

– оператор привласнення, один із основних операторів мови програмування. У його лівій частині вказується ім'я змінної, права частина є виразом того ж типу, що і змінна. Виконується оператор так: обчислюється чисельне значення виразу в правій частині і результат записується в змінну зліва. Іншими словами, оператор привласнення – обчислювач. Змінна Sum приймає значення суми двох змінних Number_1 і Number_2.

Оператор

Writeln (‘ Сума введених чисел дорівнює: ‘, Sum);

виводить два об'єкти – текстову константу і змінну. Оператор write виводить рядок на екран і залишає курсор в кінці тільки що виведеної трійки. Оператор writeln після виведення встановлює курсор в початок наступного рядка.

Для опису синтаксису мови часто використовуються синтаксичні діаграми. **Синтаксична діаграма** є направленим графом, при проходженні якого автоматично будується синтаксично правильна програма. У діаграмі слід розрізняти два типи символів:

Синтаксичні діаграми відображають лише невелику частину синтаксису. Кожне поняття, укладене в прямокутник, вимагає, у свою чергу, деякого визначення. Структуру програми на Паскалі можна представити такою схемою (рис 31):

На діаграмі видно, що заголовок може бути, а може ні. Після заголовка йде власне програма у вигляді деякого блоку. Поняття блок вимагає розшифровки (рис. 32).

Дамо ще діаграму для опису (рис. 33).

Всі визначення надалі вивчатимуться.

З синтаксичної діаграми видно, що не всі перераховані розділи обов'язкові в кожній програмі. У приведеному прикладі присутній тільки

опис змінних.

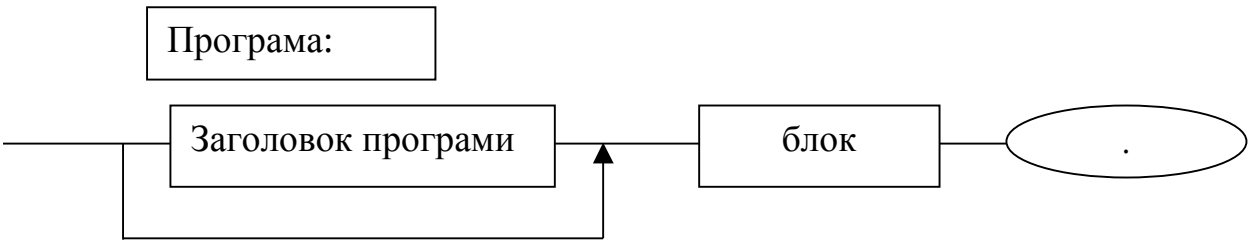


Рисунок 31

Синтаксичні діаграми

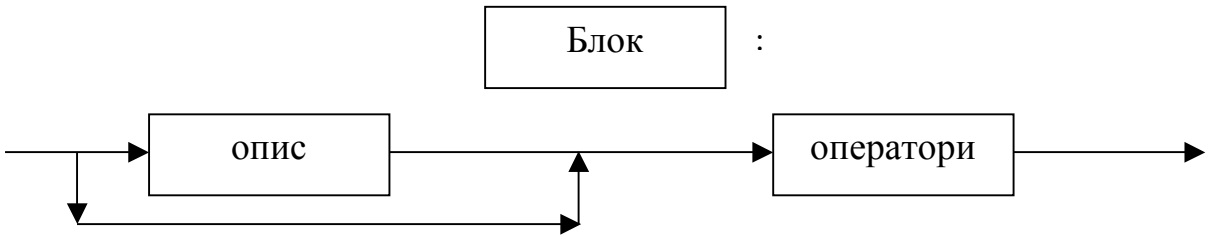


Рисунок 32

Докладніше див. [1-4,6,7].

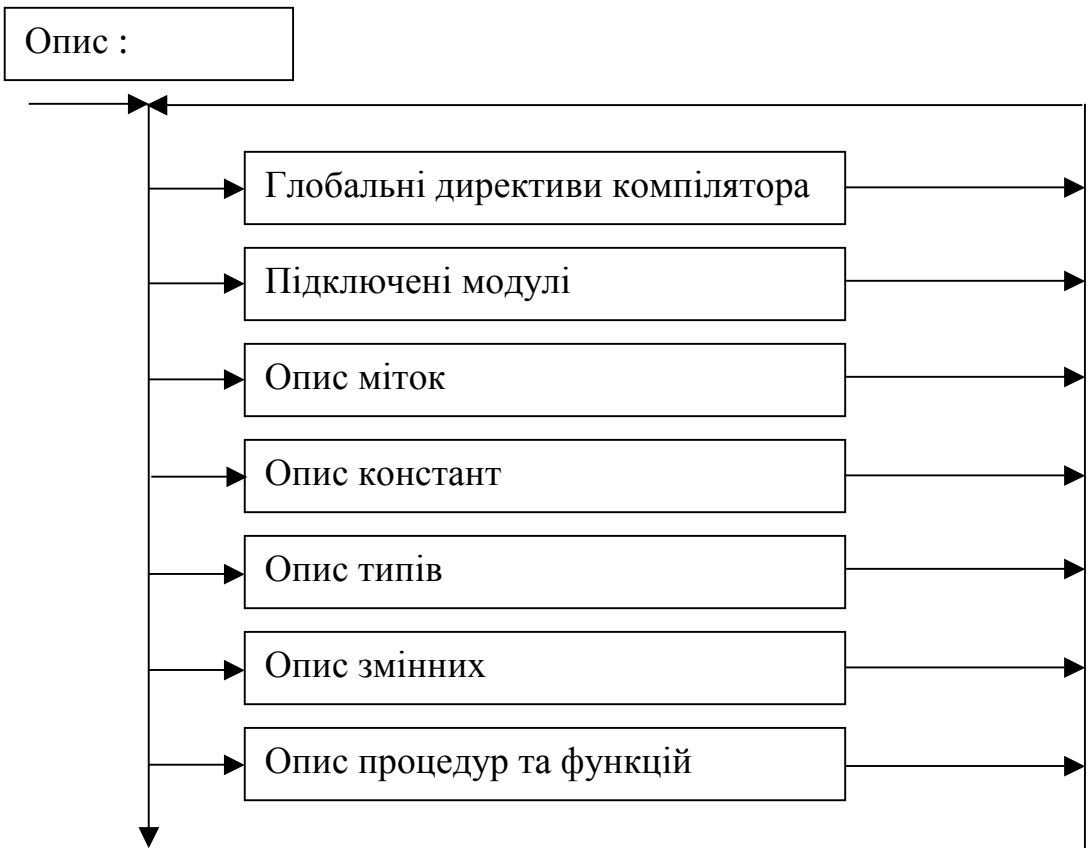


Рисунок 33 – Структура програми на мові Паскаль

4 СТАНДАРТНІ ТИПИ ДАНИХ

4.1 Загальні відомості

Одна з характерних особливостей сучасних ЕОМ полягає в можливості одержувати, накопичувати і зберігати великі об'єми інфор-- мації, що містять формалізовані відомості про навколишній світ.

Інформація, що поступила в ЕОМ, може бути оброблена з метою отримання з неї відомостей, що необхідні користувачу, в залежності від його інтересів і потреб практики.

Інформація, що зберігається в пам'яті ЕОМ і доступна для обробки, складається з даних. Дані певною мірою є наближеним віддзеркаленням дійсності, оскільки в них ігноруються деякі властивості і характеристики реальних об'єктів, неістотні для вирішуваної задачі.

Застосування ЕОМ для зберігання і обробки даних приводить до розділення даних і їх смислового змісту. ЕОМ, як правило, мають справу з даними як такими, а велика частина інтерпретуючої інформації в явній формі не фіксується. Її повинен знати користувач і застосовувати при аналізі даних, одержаних від ЕОМ.

Розглянемо, наприклад, програму чисельного аналізу, призначену для розв'язування систем звичайних диференціальних рівнянь. Ця програма одержує як початкові дані деякі числа і виробляє в результаті інші числа. Вона "не знає" чи описують диференціальні рівняння механічні або електромагнітні явища. Відповідальність за інтерпретацію результатів в контексті їх застосування лежить на користувачеві програми.

Кожний з параметрів для завдання властивості повинен бути поданий або окремим значенням, або сукупністю значень. Значення можуть характеризувати як кількісну, так і якісну сторону об'єкта. Так, параметр маса може задаватися кількісно (наприклад, 1275), параметр колір – якісно (наприклад, червоний); одні параметри – маса, довжина, ширина, висота, вартість, колір – відносяться до об'єкта в цілому, інші – характеризують різні властивості деталей, з яких складається виріб (масу, вартість, матеріал і т. ін.). Параметр може мати і складнішу структуру, відображаючи одночасно і назву деталі, і її серійний номер, і масу, і об'єм, і положення в просторі (координати). Це вже агреговані параметри.

У наукових і інженерних розрахунках найпоширенішим параметром є окреме вимірювання. Такі дані є **скалярами** і є окремими числами і словами (символьні послідовності).

Відомим прикладом скаляра є константа. Це елемент даних, який має фіксоване ім'я, фіксований тип і фіксоване значення. Для позначення константи використовується її явний запис або вибраний ідентифікатор. Наприклад, позначення 3.141592 задає константу дійсного типу, значення якої фіксоване як число 3.141592, а ім'я (зовнішнє уявлення для користувача ЕОМ) зображається її значенням. Така інтерпретація

константи загальноприйнята в математиці. Розробник алгоритму може побажати пов’язати з константою дійсного типу, представленої значенням 3.141592, ім'я P, яке є символічною константою.

Скаляром може бути також і рядок символів, утворений послідовністю літер. Наприклад, слово "ВОЛОДИМИР" задає константу літерного типу, значення якої фіксоване як ланцюжок літер "ВОЛОДИМИР", а ім’я представлене її ж значенням. При необхідності цій константі можна поставити у відповідність унікальний ідентифікатор (ім'я) і використати його як символьну константу. Отже, **константа** – деяка незмінна величина. Константа може задаватися числом або ідентифікатором.

Разом з константами широко використовуються і змінні. **Змінною** називається об’єкт, що має фіксоване ім’я, фіксований тип і значення, що змінюється залежно від вживаних дій.

4.2 Типи даних

Слід мати на увазі, що при розв’язуванні задач ЕОМ оперує не з значеннями, а з їх позначеннями, якими є конфігурації бітів, байтів або слів. Щоб ЕОМ могла при виконанні операцій розпізнавати приналежність цих конфігурацій до того або іншого типу даних, необхідне при розробці алгоритмів, і особливо програм, прямо вказувати цю приналежність. Досягається це, шляхом явного опису типів використовуваних даних. Залежно від типу, заданого в описі змінної, вона може приймати поточні значення тільки вказаного типу. Наприклад, якщо тип змінної A вказаний як "цілий", то вона в даний момент часів може мати будь-яке значення з допустимої множини цілих чисел {...-3, -2, -1,0,1,2,3,...}; якщо тип змінної B вказаний як "логічний", то поточне значення може бути одним з двох {істина, хибність}.

Кожен тип даних характеризується так званим кардинальним числом – кількістю різних значень, що належать типу. Для кожного типу даних пови-нен бути строго визначений набір операцій, які можна застосовувати при обробці даних цього типу.

Спочатку певні типи даних називаються простими, причому ті з них, які безпосередньо "вбудовані" в ЕОМ, носять назву **стандартних типів.** Кожна алгоритмічна мова програмування надає користувачу набір різних типів елементарних даних, засоби їх опису і операторів обробки, які забезпечують виконання над даними тих або інших дій. Компілятор зв’язує ім'я елементу даних з певною адресою пам'яті ЕОМ, за яким в процесі виконання програми зберігається значення іменованого елементу даних, що звільняє програміста від необхідності знати цю адресу.

Найважливіший принцип Паскаля: всі використовувані в програмі імена повинні бути описані до їх вживання. Описати ідентифікатор – це значить вказати тип пов'язаного з ним об’єкта програми.

У кожній мові програмування є свої правила запису ідентифікаторів.

Найчастіше це послідовність латинських букв і цифр, що починається з букви. У Турбо Паскалі правила запису ідентифікаторів такі:

- ідентифікатор може складатися з букв латинського алфавіту, цифр, знака підкреслення;
- ідентифікатор не може починатися з цифри;
- ідентифікатор не може збігатись ні з одним із зарезервованих слів;
- довжина ідентифікатора може бути довільною, але значущими вважаються перші 63 символи.

Наприклад:

A, B, M, N. SUMMA, Z1, Z2, Z3, PRIMA14, RRST_VALUE.

У стандартному Паскалі знак підкреслення не використовується.

До **простих типів** відносяться: цілочисловий, логічний, символьний, перелічний, інтервальний, дійсний. На основі простих типів даних можна будувати різні структуровані типи даних будь-якого ступеня складності.

Всі, використовувані в програмі об'єкти, зв'язуються з існуючими в мові типами даних в спеціальному описовому блоці програми. Для опису типів даних використовуються спеціальні службові слова.

Тип цілий містить підмножину цілих констант, при цьому кардинальне число підмножини розрізняється для різних ЕОМ. Для ЕОМ з двобайтним словом числа найчастіше знаходяться в діапазоні допустимих значень від -32768 до 32767. Такий тип змінної описується службовим словом INTEGER. До цілочислових також відносяться типи: BYTE, SHORTINT, WORD, LONGINT. Ці дані розрізняються внутрішнім уявленням і діапазоном можливих значень (-128... 127 для SHORTINT і -2147483648 ... 2147483647 для LONGINT). Приклади цілих чисел:

0, -3, 17, 193, -10000, 5.

Для даного типу INTEGER запис 50000 невірний, оскільки це число виходить за межу допустимих значень.

Якщо *i* і *j* ідентифікатори змінної цілого типу, то в описовій частині програми повинен бути присутнім запис:

i, j : integer.

Стандартні операції для цілих– це чотири дії арифметики: додавання, віднімання, множення і ділення без остачі. Остання операція повинна давати цілий результат, опускаючи можливий залишок. Ці операції над цілими числами виробляються абсолютно точно, і результатами цих операцій знову є цілі числа. У Паскалі є ще дві операції над цілими числами: div і mod. Ці операції мають по два цілих операнди (аргументи): якщо значення a і b невід'ємне і $b \neq 0$, то a div b і a mod b – це ціла частина від ділення і залишок, виникає при діленні a на b. Наприклад,

17 div 3 = 5, 8 mod 2 = 0,
17 mod 3 = 2, 1 div 5 = 0,
8 div 2 = 4, 1 mod 5 = 1.

Ці операції одного приорітету з множенням і діленням, що важливо мати на увазі при обчисленнях виразів.

Тип дійсний позначає підмножину дійсних констант. Тоді як арифметичні дії з цілими дають точні результати, для арифметичних дій над дійсними числами (операції додавання, віднімання, множення, ділення) допускається неточність в межах помилок округлення. У цьому і полягає явна відмінність між типами "цілий" і "дійсний", характерне для більшості мов програмування. Для чисел дійсного типу в мові Турбо Паскаль визначено п'ять стандартних дійсних типів: дійсний (REAL), з одинарною точністю (SINGLE), з подвійною точністю (DOUBLE), з підвищеною точністю (EXTENDED) і складний (COMP). На перших порах обійдемося типом REAL Діапазон допустимих значень для типу REAL від 2.9×10^{-39} до 1.7×10^{38}, область, пам'яті для розміщення – 6 байтів, точність 11-12 знаків.

До цього типу відноситься підмножина дійсних чисел, які можуть бути подані у форматі з фіксованою точкою і з плаваючою десятковою точкою. **Числа з фіксованою точкою** записуються у вигляді цілої і дробової частин числа. (Наприклад: 5.45, -0.001, 17.0, -19.1919, 0.143. Запис числа не може починатися або закінчуватися точкою. Числа з плаваючою точкою використовуються для запису чисел, що змінюються в широкому діапазоні значень (від дуже маленьких до дуже великих). Десятковий порядок числа записується буквою Е. Наприклад, 65.4E22 відповідає $65.4^* 10^{22}$. Числа з плаваючою точкою: 0.547E+3, 5.47E+2, 54.7E+1, 547.0E0, 5470E-1, 54700E-2 представляють одне і теж число 547.

Для обробки дійсних (речовинних) чисел передбачені такі операції: додавання (+), віднімання (-), множення (*), ділення (/). Операції піднесення до степеня в Паскалі не існує. Якщо $C = a^b$, то С розраховують за формулою

$$C = e^{(b*\ln a)}.$$

Якщо a і b змінні дійсного типу, то в описовій частині програми повинно бути присутнім

```
a, b: real;
```

Як вже говорилося, тип змінної дозволяє не тільки встановлювати довжину її внутрішнього подання, але і контролювати ті дії, які виконуються над нею в програмі. Контроль за використовуванням змінних – важлива перевага Паскаля перед іншими мовами програмування, в яких допускається автоматичне перетворення типів. У Паскалі майже неможливі неявні перетворення типів. Виключення зроблене тільки відносно констант і змінних типу INTEGER (цілі), які дозволяється використовувати у виразах типу REAL.

Тип логічний містить всього два значення, які позначаються як істина і хибність (TRUE і FALSE). Слово BOOLEAN описує логічні змінні. Логічні змінні використовуються для зберігання результатів логічних обчислень. Значення TRUE і FALSE є за своєю суттю ідентифікаторами констант. Для булевих змінних дозволені тільки порівняння ">" (більше), "<" (менше), " =" (дорівнює) і "<>" (не дорівнює). Іншими допустимими

операціями є: логічне складання (AND), логічне множення (OR), заперечення (NOT), логічне виключення (XOR). Змінні типа BOOLEAN займають 1 байт пам'яті.

Тип літерний (символьний) включає множину друкованих символів. Символьний тип CHAR – є типом даних, призначеним для зберігання одного символа (букви, знаку або коду). У змінну цього типу на комп'ютері IBM може бути поміщений будь-який з 256 символів розширеного коду ASCII. Це букви ['A'...'Z', 'a'...'z'], цифри ['0'...'9'], розділові знаки і спеціальні символи. Змінна типу CHAR в пам'яті займає 1 байт. Значення для змінних типу CHAR задаються в апострофах. Крім того, є можливість задавати значення вказівкою числового значення ASCII-коду. В цьому випадку перед числом, що позначає код ASCII символа, ставиться знак (#). Наприклад, CH:= #65 – привласнення змінної CH символу з ASCII кодом 65, тобто символу 'A'. Опис символьної змінної:

u, v : char;

Якщо константа в програмі позначена ідентифікатором, то її необхідно оголосити в описовій частині за допомогою службового слова CONST.

Наприклад,

CONST year = 2005; time = 12.05;

year – константа цілого типу, оскільки не має в записі числа десяткової точки; time – константа дійсного типу.

Розділ оголошення змінних починається зарезервованим словом VAR, вслід за яким розташовуються конкретні змінні. Для оголошення змінної необхідно вказати ім'я змінної і її тип. Наприклад,

```
VAR
a : INTEGER;
d, c : REAL;
b, e, f, g : CHAR;
```

Заборона на автоматичне перетворення типів ще не означає, що в Паскалі немає засобів перетворення даних. Для перетворення даних в мові існують вбудовані функції, які набувають як параметр значення одного типу, а повертають результат у вигляді значення іншого типу. Для перетворення даних типу CHAR (символ) в ціле число призначена функція ORD, зворотне перетворення INTEGER в CHAR здійснює функція CHR.

Зокрема, для перетворення REAL в INTEGER є навіть дві вбудовані функції такого роду: ROUND округляє REAL до найближчого цілого, а TRUNC відсікає REAL шляхом відкидання дробової частини.

```
trunc(3.14)= 3,   trunc(-3.14)= -3,   trunc(3.7)= 3.
round(3.14)= 3,   round(3.7)= 4,     round(-3.14)==-3.
```

4.3 Структури даних

При описі алгоритму важливо однозначно визначити не тільки властивості, але і структурні особливості використаних в алгоритмі об'єктів, над якими виконуються перетворення в ході вирішення задачі. Дотепер розглядалися скалярні дані: константи і змінні. Змінні, які мають як поточне значення тільки одну величину, називаються **скалярними змінними.**

Різновидом змінної може бути і змінна "з індексом", яка є елементом масиву. Для її позначення використовують ім'я масиву і перелік (список) індексів:

A[1], G[1, 5], RAD[K, L], S[3, 4, 5].

При обробці даних широке розповсюдження має і загальніше поняття, таке, як **структурована змінна,** тобто змінна, яка складається з декількох елементів або компонент, і на яку можна посилатися як на єдиний об'єкт. Наприклад, побудова календаря дозволяє вказувати конкретний день, але при цьому існує і спосіб посилання на місяць і рік. У опис типу структурованої змінної повинно входити число його складових і характеристики їх типів.

Якщо всі елементи об'єкта відносяться до одного і того ж типу, то така структурована змінна є однорідною і може бути представлена у вигляді деякого масиву.

Масив – це регулярна структура з так званим випадковим доступом, що означає: всі компоненти масиву однорідні, можуть вибиратися довільно і є однаково доступними.

Для позначення окремого елементу масиву, як вже згадувалося, до імені масиву додається список індексів, що дозволяє здійснювати доступ до конкретного елементу.

Список індексів – це впорядкована множина цілих чисел або змінних цілого типу, яка однозначно визначає місцеположення окремого елементу масиву. Кожен індекс має свій діапазон змін, так звану граничну пару.

Так, масив A цілого типу, впорядкований по двох вимірюваннях, можна подати як матрицю з n рядків і m стовпців:

$$\begin{matrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} & a_{27} & a_{28} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} & a_{37} & a_{38} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & a_{47} & a_{48} \end{matrix}$$

В даному прикладі n = 4, m = 8.

Доступ до елементу масиву задається списком з двох індексів:

a [2, 4] або A(2,4) – визначає елемент 2-го рядка 4-го стовпця;

a [l, k] або A(l, k) – визначає елемент l-го рядка k-го стовпця.

Індекс l має діапазон змін від 1 до 4, а індекс k – відповідно від 1 до 8.

При роботі з масивами, особливо великого розміру, звичайно

вибірково змінюють окремі компоненти. При цьому змінна масиву розглядається як сукупність змінних "з індексами", які її складають, і допускається привласнення значень кожному з компонентів.

Хоча при вибірковому привласненні змінюється значення окремого компоненту, з точки зору концепції структурованої змінної слід вважати, що змінюється все складове значення.

Загальний метод отримання структурованих змінних – це об'єднання компонентів, що належать до довільних (можливо складових) типів, в один складовий тип. Прикладами є:

– комплексні числа, що складаються з двох дійсних констант;
– координати точок, що складаються з двох дійсних чисел або залежно від розмірності простору, заданого системою координат;
– опис характеристик людей за допомогою декількох істотних відмінних ознак, таких, як прізвище, ім'я, по батькові, рік народження, стать, сімейний стан.

При обробці даних комбіновані типи такі, як опис людей або матеріальних об'єктів, часто зустрічаються у файлах (або наборах даних) і є записами істотних характеристик людини або об'єкта. Тому термін **запис** став широко використовуватися для позначення подібної сукупності структурованих даних. Окремі компоненти запису називаються **полями.** Наприклад, запис, призначений для зберігання інформації про міста складається з п'яти полів: назва міста, його географічні координати (довгота, широта, висота) і кількість населення. До цього запису, як до змінної, звертаються на ім'я змінної МІСТО, а до окремих полів шляхом використання складового імені: МІСТО**.**ІМ’Я або МІСТО.НАСЕЛЕННЯ.

Запис – більш універсальна структура, ніж масив. Вона не вимагає, щоб типи всіх її компонент були однаковими. Проте масив надає великі можливості, оскільки індекси його компонентів можуть обчислюватися, якщо вони представлені виразами, тоді як імена компонентів запису – це фіксовані ідентифікатори, які повинні задаватися в описі їх типу.

Ще одним типом структурованих змінних є **множина**. Цей тип використовується в тих випадках, коли інтерес має не значення якого-небудь елементу, а лише його наявність або відсутність. Якщо описати змінну з деяким ім'ям N як деяку множину натуральних чисел, то операція належності цій множині дасть логічне значення “істина“, якщо число є елементом множини, і значення “хибність“ – в протилежному випадку. Множини можна ефективно реалізовувати і обробляти.

До множин застосовуються такі основні операції: перетин множин, об'єднання множин, різниця множин, належність множині.

Масиви, записи і множини називаються **базисними структурами**. Для них характерно, що в процесі виконання алгоритму або програми вони залишаються структурно не змінними. У багатьох випадках потрібні складніші структури, які допускають такі зміни, як нарощування, скорочення або заміну в процесі виконання зв'язків між компонентами.

Створення динамічних структур зводиться до генерації основних компонентів, названих вузлами, і встановленню зв'язків між ними. Вузли звичайно є записами, зв'язки визначаються змінними, названими покажчиками. До таких динамічних структур даних відносяться списки, черги, стеки, дерева, орієнтовані графи.

Мова програмування Паскаль характеризується розгалуженою структурою типів. У Паскалі передбачено механізм створення нових типів даних, завдяки чому загальна кількість типів, використовуваних в програмі, може бути скільки завгодно великою.

5 ПОДАННЯ ОСНОВНИХ СТРУКТУР ПРОГРАМУВАННЯ МОВОЮ ПАСКАЛЬ

5.1 Операції і вирази

Змінні і константи – прості окремі випадки виразу. **Вирази** складаються з операндів, знаків операцій і круглих дужок. **Операндом** може бути константа, змінна, межа параметра-масиву (про це пізніше) або позначення функції. Значення виразу в тому, щоб пасивні сладові (операнди) зв'язати через активні складові (+, -, *, / і інші) і отримати деяке нове значення.

Вираз не просто має деяке значення, але і володіє абсолютно певним типом, який залежить від операндів і операцій.

Для того, щоб описати послідовність, в якій повинні стояти операнди у виразах, доцільно упорядкувати операції по рівнях.

1. Операції нижчого рівня виконуються раніше, ніж операції вищого рівня.
2. Операції одного рівня виконуються по черзі зліва направо.
3. Операції, укладені в круглі дужки, виконуються раніше операцій, записаних за дужками.

Ці правила діють для всіх типів виразів.

5.1.1 Арифметичні операції і вирази

Арифметичні вирази мають тип real або integer, причому ми завжди під real розумітимемо також single, double, extended і comp, а під integer – byte, word, shortint і longint.

Приклад арифметичного виразу:

(- b + sqrt (sqr(b) - 4 * a * c)) / (2 * a)

При складанні виразів слід виконувати такі правила:

1. Записувати всі складові частини виразів в один рядок.

Операція	Операнд 1	Операнд 2	Тип результату
*	real	real	real
*	integer	integer	integer
*	real	integer	real
/	real	real	real

/	integer	integer	real
/	real	integer	real
div	real	real	integer
mod	integer	integer	integer
+	real	real	real
+	integer	integer	integer
+	real	integer	real
-	real	real	real
-	integer	integer	integer
-	real	integer	real

2. Використовувати дужки тільки одного типу – круглі. Застосування фігурних квадратних дужок у виразах забороняється, оскільки вони, мають особливе призначення.

$$\frac{x}{\left(\frac{y}{z}\right)} = x / y / z.$$

$$\frac{x}{yz} = x/(y * z).$$

$$a(b+c) = a*(b+c).$$

$$\frac{\frac{a}{x}+\frac{2b}{5y}}{1+\frac{4z}{-2}} = (a/x + 2*b/(5*y))/(1+4*z/(-2)).$$

3. Не можна записувати підряд два знаки арифметичних операцій.

До арифметичних операцій відносяться: перший рівень *, /, mod і div; другий рівень +, -. Тобто при обчисленні арифметичного виразу діють звичні правила старшинства операцій: спочатку виконуються множення, ділення, ділення без остачі і знаходження залишку від ділення без остачі в тому порядку, в якому вони входять у вираз, а потім додавання і віднімання. Між знаком div (або mod) і числами, що беруть участь в діленні, повинно знаходитися хоча б по одному проміжку.

Декілька прикладів:

5 + 2 * 10 = 25

10.2*5-7 + 8.6/2 = 22.3;

(5+105) div 7 = 5

Тип результату залежить від типу операндів.

Запишемо декілька звичних математичних виразів на Паскалі.

Якщо при розстановці дужок виникнуть сумніви, пригадайте правило:

“Зайві дужки не заважають".

Не можна розміщувати два знаки операцій поряд в послідовності символів

3 * - 2, x1 / - x2 – це не вирази, виразами будуть 3 * (-2), x1 / (-x2).

У виразах може бути присутнім виклик функцій. Функція передає своє значення у вираз. У Паскалі є безліч стандартних функцій, про яких йтиметься майже в кожному розділі.

Правила запису стандартних функцій:

1. Ім'я функції записується буквами латинського алфавіту.
2. Аргумент функції записується в круглих дужках після імені функції.
3. Аргументом функції може бути константа, змінна або арифметичний вираз.

Наведемо ряд стандартних математичних функцій.

abs(x)	x
exp(x)	e^x
cos(x)	cos x
sin(x)	sin x
arctan(x)	arctgx
ln(x)	ln x
sqr(x)	x2
sqrt(x)	√x
random(x)	Якщо x відсутній, значенням функції є випадкове число типу real з діапазону 0<= ...<1. Якщо задається параметр x, значенням функції буде випадкове число з діапазону 0<=...<x.

5.1.2 Логічні операції

Логічні вирази мають значення типу boolean, тобто true або false.

Вираз, що служить для обчислення логічного значення, називається **логічним виразом або логічною умовою**.

Одним з видів логічного виразу є відношення. **Відношення** – це два вирази, об‘єднані операцією відношення. Наприклад,

y <0, a > b, x =y, x < a- b.

Операції відношення: > (більше), < (менше), > (не менше), < (не більше), = (дорівнює), ≠ (не дорівнює) – на мові Паскаль записуються відповідно: >, <, >=, <=, =, < > і мають нижчий пріоритет в порівнянні з арифметичними операціями. Іншими словами, спочатку виконуються арифметичні операції, а потім операції відношення.

Умова a + b ≠ c + d на мові Паскаль записується так: a + b <> c + d. Вислови про значення змінних можуть бути істинними або помилковими залежно від самих значень змінних. Так, якщо s = 5, t = 6, вислів s > t – помилковий, вислів s < t + 12 – істинний.

Логічні значення впорядковані. Вираз true > false є істинним.

З простих висловлень в Паскалі дозволяється будувати складніші. Хай A і B – деякі висловлення, тоді A and B – це нове висловлення, що затверджує істинність обох висловлень A і B; A or B – це нове

висловлення, що затверджує істинність хоча б одного з висловлювань A і B. Якщо C – висловлення, то not C – це нове висловлення, стверджуюче, що C – помилкове висловлення.

A	b	a and b	a or b		c	not c
False	false	false	false		true	False
False	true	false	true		false	True
True	false	false	true			
True	true	true	true			

Рівень операції	Операція	Тип операнда	Тип результату
0	not	boolean	boolean
1	and	boolean	boolean
2	or	boolean	boolean
3	<>	простий тип	boolean
3	=	простий тип	boolean
3	<	простий тип	boolean
3	<=	простий тип	boolean
3	>	простий тип	boolean
3	>=	простий тип	boolean

Операції над висловлюваннями (логічні операції) and, or, not називаються відповідно кон'юнкцією, диз'юнкцією і запереченням. Наступні таблиці уточнюють словесні пояснення.

Приклад:

1) a and b or c;
2) a and (b or c);
3) i<10;
4) (i<10)and (k<=0).

При визначенні істинності висловлення, побудованого з відносин за допомогою знаків логічних операцій і круглих дужок, діють такі правила старшинства операцій: найстарша операція заперечення; наступна кон'юнкція, потім – диз'юнкція.

Першою з двох операцій одного старшинства виконується та, знак якої у виразі зустрічається раніше. Круглі дужки змінюють цей природний порядок. Для висловлення

(x > y) or (y > z) and not ((x > 0) or (z > x))

встановлюється такий порядок логічних операцій:

4 3 2 1

(x > y) or (y > z) and not ((x > 0) or (z > x))

При x = -1, z = -2, y=1 результатом буде значення true.

(x > 0) – false, (z > x) – false;

(x > 0) or (z > x) – false;
not ((x > 0) or (z > x)) – true;
(x > y) – false, (y > z) – true;
(y > z) and not ((x > 0) or (z > x)) – true.
Зрештою false or true дає true.

5.2 Оператор присвоювання

Окремі інструкції, що входять в програму, прийнято називати **операторами**. У розділ операторів (виконуваної частини) програми

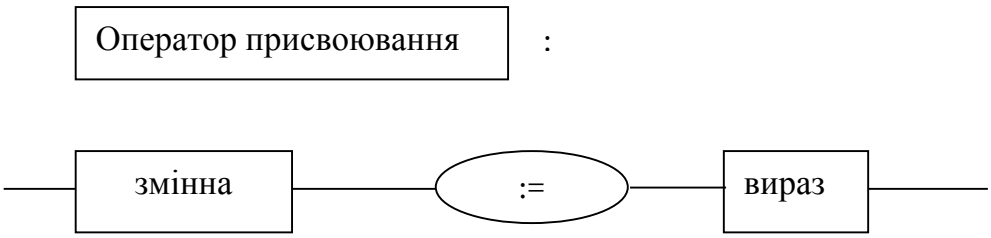


Рисунок 34

поміщаються оператори, які повинні бути виконані з визначеними в описовій частині даними.

Арифметичний вираз не є оператором, а є правилом (формулою), відповідно до якого може бути обчислено деяке значення. Воно може використовуватися як складова частина різних операторів, зокрема, оператора присвоювання.

Оператор присвоювання є найважливішим оператором будь-якої мови програмування. З його допомогою можна присвоїти змінній значення виразу. Вираз оцінюється, тобто визначається його значення, і це значення присвоюється змінній. В результаті колишнє значення змінної перезаписується, а тому старе значення втрачається. Знаком присвоювання в Паскалі є сукупність знаків (:=). наприклад, в операторі присвоювання
Y := (A * X + Y) * X + C праворуч від знаку := записаний вираз
(A * X + Y) * X + C, а зліва – змінна Y. Потрібно запам'ятати: зліва не може бути вираз.

Нехай i = 1. Після виконання оператора присвоювання i := i + 1, змінна i матиме значення 2 (рис. 34).

При використанні оператора присвоювання v:= е слід враховувати, що змінна v і вираз е повинні мати однаковий тип. Є лише одне виключення з цього правила: змінна може мати тип real, а вираз значення типу integer.

Приклад:

Нехай
var i, j: integer; x, y: real; a, b : char; p, q: boolean;
Наступні оператори присвоювання правильні:

p : =i<5;

```
a := '+';
x := i + j mod 7;
y := 275;
y :=x*(sqr(2));
```

Наступні оператори неправильні, оскільки не враховують типу:

```
i :=  3.678
b := 'stroka';
a := +;
```

Задача 1. Скласти програму обчислення площі трикутника за трьома сторонами a, b, c.

Алгоритм розв’язування задачі подамо у вигляді схеми(рис.35).

Схема алгоритму

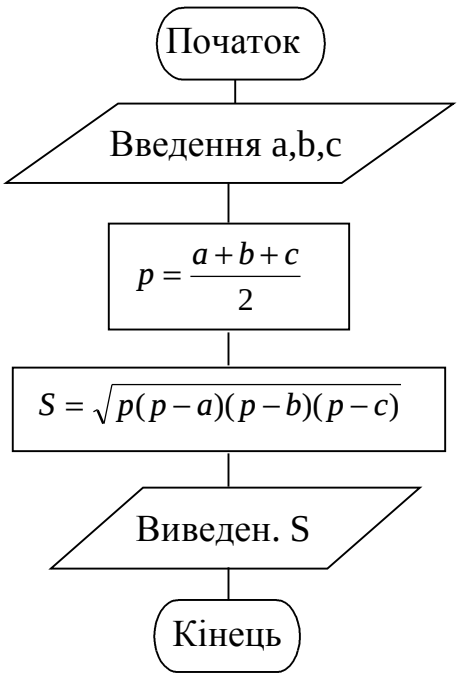


Рисунок 35

Програма на Паскалі

```
Prograrn  taskl;
var a, b, c, p, s: real;
begin
   readln(a, b, c);
   p := (a + b + c)/2;
   s := sqrt(p*(p -a)* (p -b)*(p- c));
   write(s)
 end.
```

Як вже говорилося раніше, абсолютно не обов'язково записувати кожний оператор в новому рядку. Вірним був би і такий запис:

```
program taskl; var a, b, c, p, s: real; begin readln(a, b, c); p := (a + b + c)/2;
s:= sqrt(p*(p -a)* (p - b)*(p - c)); write(s) end.
```

Важливо, щоб після кожного оператора стояла ;.

Прокоментуємо кожний оператор програми.

`program taskl;` Заголовок програми з ім’ям програми taskl. Ще раз нагадаємо, що це не обов'язковий оператор, його може не бути.

`var a, b, c, p, s: real;` Розділ описів. Всі змінні в Паскалі повинні бути описані, a, b, c, p, s – це все змінні, які зустрічаються в програмі і всі вони дійсного типу.

`begin` Оператором begin відкривається основний блок програми – блок операторів.

`readln(a, b, c);` Функція readln вводить чисельні значення змінних a, b, c. Ідентифікатори цих змінних записуються в дужках і через кому.

`p := (a + b + c)/2;` Оператор присвоювання. Ті чисельні значення, яке одержали змінні a, b, c при введенні, підсумовуються; одержана сума ділиться на 2; результат присвоюється

змінній с ім’ям p.

s := sqrt(p*(p -a)* (p - b)*(p - c)); Оператор присвоювання

write(s); Функція виведення. На екран дисплея виводиться чисельне значення змінної s.

end. Останній оператор програми. Закриває розділ операторів. Після останнього оператора end ставиться крапка.

Після того, як програма написана і для неї вибрані початкові дані, потрібно, щоб ця програма була виконана. Програма вводиться в оперативну пам'ять ЕОМ (наприклад, з клавіатури), дається команда на компіляцію і виконання. Процесор виконує програму. Оператор readln зажадає введення трьох чисельних значень. Оператор write виведе чисельне значення змінної s.

5.3 Складений оператор

Складений оператор – це послідовність довільних операторів програми, укладена в операторні дужки – зарезервовані слова begin ... end. Оператори розділяються крапкою з комою;

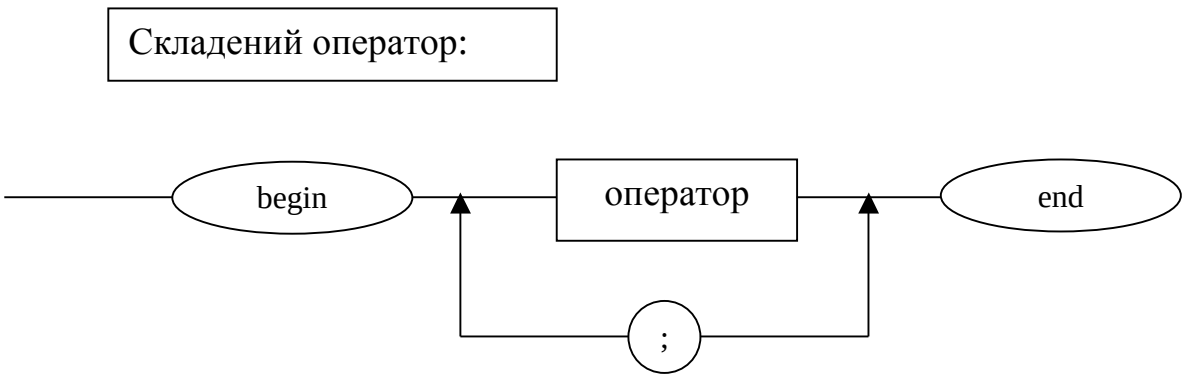


Рисунок 36

Виконувана частина програми є складовим оператором такого роду.

Складений оператор служить, в першу чергу, для того, щоб декілька операторів синтаксично об'єднати в один. Це часто потрібне там, де потрібно виконати декілька операторів, коли допустимий лише один. Поняття складеного оператора дозволяє за допомогою команд begin і end об'єднати декілька операторів і розглядати їх з точки зору синтаксису як один оператор. На характер операторів, що входять в складений оператор, не накладається ніяких обмежень. Серед них можуть бути і інші складені оператори.

```
Begin
 ……
   begin
```

```
…….
      begin
       …….
       …….
       …….
     end;
……..
end;
……..
end;
```

При виконанні операторів крапка з комою служить роздільником для двох операторів. Крапкою з комою перед завершальним end можна нехтувати.

```
begin                    begin
  read(i);                 read(i);
  write(i);                write(i)
end.                     end.
```

Обидва записи правильні, оскільки можна вважати, що між write(i); і end ; знаходиться порожній оператор. Порожній оператор – оператор, який не виконує ніяких операцій і нічого не змінює в даних і в програмі. Порожньому оператору відповідає відсутність запису на тому місці, де за правилами повинен бути який-небудь оператор. Після нього можна ставити символ крапки з комою, наприклад:

```
A := Y;
R := 2;
;
K := + 7.2;
```

У програмістів-початківців, часто виникає питання: де правильно поставити знак крапки з комою? Щоб на нього відповісти, звернемося до звичної природної мови. У будь-якому переліку елементів між ними ставиться кома, наприклад:

А, В, С, Д.

Якщо ці елементи об'єднати в одну групу, уклавши їх в круглі дужки (А, В, С, Д), то кома ставиться знову-таки між елементами: після відкриваючої і перед закриваючою дужками кома не вказується. Якщо ця група елементів входить до складу іншої групи, то кома ставиться і між групами, наприклад:

((А, В, С, Д), (К, М), Е, (Х, У))

Подібна система введена і в мові Паскаль, тільки в ньому роль круглих дужок виконують операторні дужки BEGIN – END, замість коми ставиться крапка з комою, а замість елементів – оператори.

5.4 Умовний оператор

Умовний оператор дозволяє перевірити певну умову і залежно від

результату виконати ту або іншу дію. За допомогою цього оператора програмуються алгоритми структури, що розгалужується.

Структура умовного оператора:

IF <умова>THEN <оператор1> ELSE < оператор2>,

де IF, THEN, ELSE – зарезервовані слова (якщо, то, інакше);

<умова> – довільний вираз логічного типу;

<оператор1>,<оператор2> – будь-які оператори мови.

Умовний оператор працює за таким алгоритмом. Спочатку обчислюється

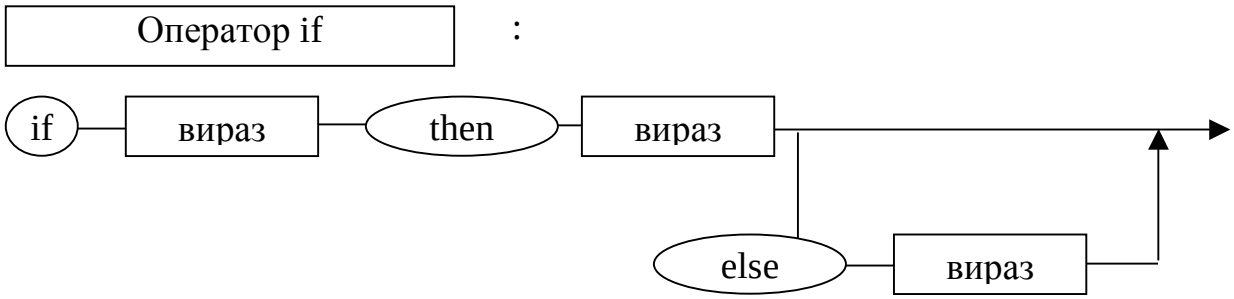


Рисунок 37

умовний вираз <умова>. Якщо результат є TRUE (істина), то виконується <оператор1 >, а <оператор2> пропускається; якщо результат є FALSE (хибність), навпаки, <оператор1> пропускається, а виконується <оператор2>.

Наприклад:

```
Var
    x, y, max: real;
    ……
    if x > max
        then y := max
    else
        y:=x;
```

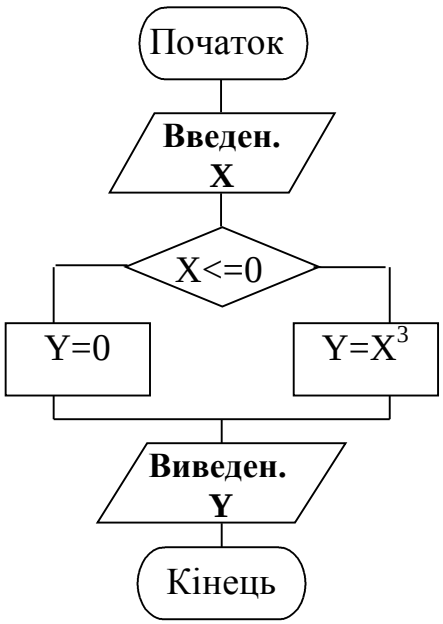


Рисунок 38

Цей умовний оператор читається: якщо умова x > max виконується, то y := max, інакше y := x.

Частина ELSE <оператор2> умовного оператора може бути опущена. Тоді при значенні TRUE умовного виразу виконується <оператор1>, інакше цей оператор пропускається (рис. 37).

Якщо потрібно виконати після then або else декілька операторів, вони обрамляються командами begin і end, утворюючи тим самим складений оператор.

Перед else ніколи не ставиться крапка з комою (оскільки це ще не кінець

оператора if).

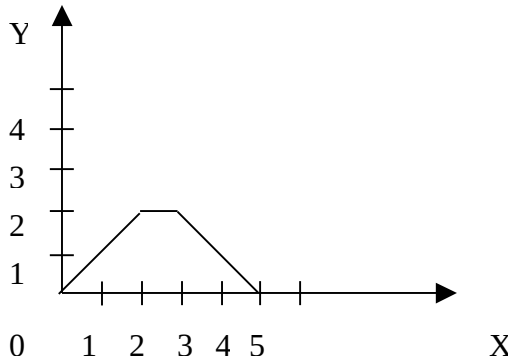


Рисунок 39

Задача 2. Нехай задана функція

$$Y==\begin{cases}0, & якщо \quad X<0;\\ X^3, & якщо \; X>0.\end{cases}$$

Написати програму обчислення значення Y за значенням X. Програма може виглядати так:

```
program task2;                    { заголовок програми}
var  x, y : real;                 { опис змінних}
begin                             {початок виконуваної частини}
    readln(x);                    {       -//-            }
    if x <= 0 then  y:= 0 else y := x*x*x; {  -//-   }
    write(y)                      {      -//-             }
 end.    {кінець виконуваної частини}
```

<Оператор1> і <Оператор2> за визначенням будь-який оператор мови, отже і умовний.

Задача 3. Нехай значення Y залежить від значення X.
Скласти програму обчислення значення Y за значенням X (рис. 40).

Програма.

```
program taskS;
var x, y: real;
begin
  read(x);
  if x < 2 then   y:=x
       else
            if x < 3 then   y:= 2
            else
                y:=-x+5;
           write(y)
end.
```

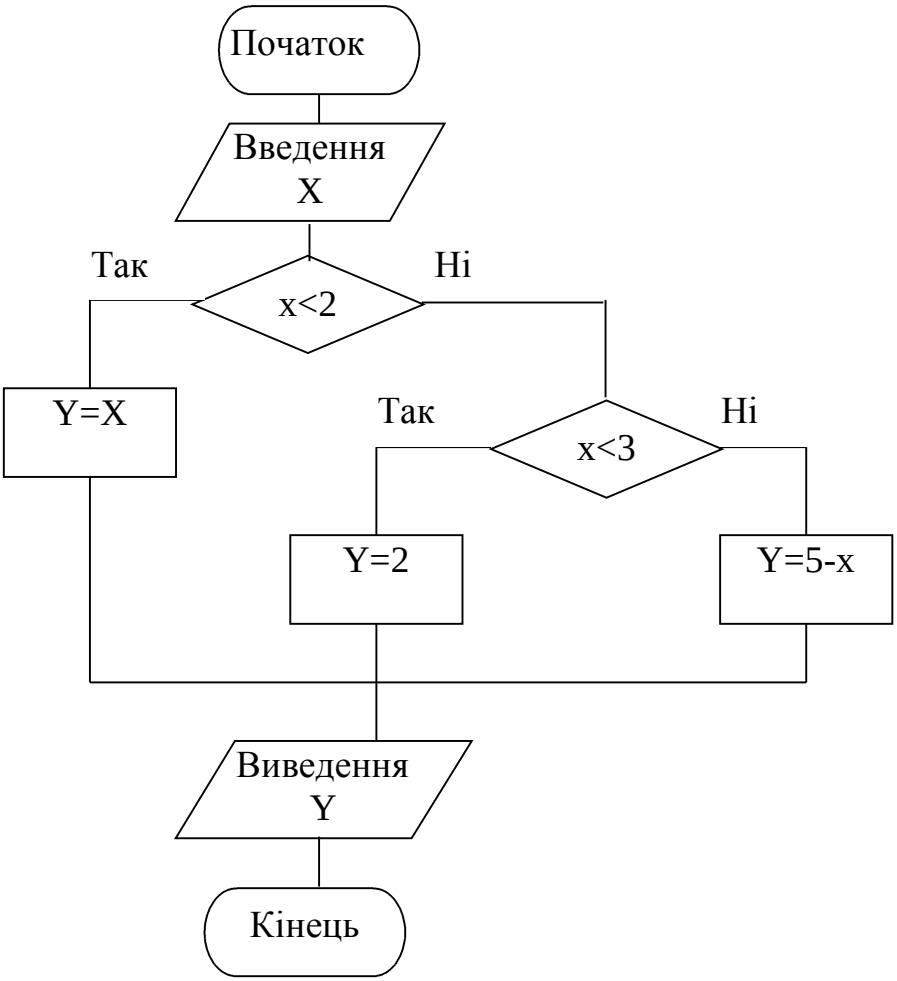


Рисунок 40

Якщо виконана умова x < 2, то y набуде значення, рівне значенню x, це значення потім буде виведено. Якщо умова x < 2 не задовольняється, то значення y буде визначено виконанням умовного оператора

```
if x < 3 then y := 2
        else y := -x + 5 .
```

Якщо else відсутній, а після оператора then знову стоїть оператор if, то виникає неоднозначність трактування умов. Ця неоднозначність розв'язується таким чином: будь-яка частина else, що зустрілася, відповідає найближчій до неї “зверху" частині then умовного оператора. Приведений вище вираз треба розуміти так:

```
if <вираз 1> then
      begin if <вираз 2>
                then   <вираз 1 >
                     else
                       < вираз 2>
      end;
```

Задача 4. Написати програму обчислення функції Z. (рис. 41).

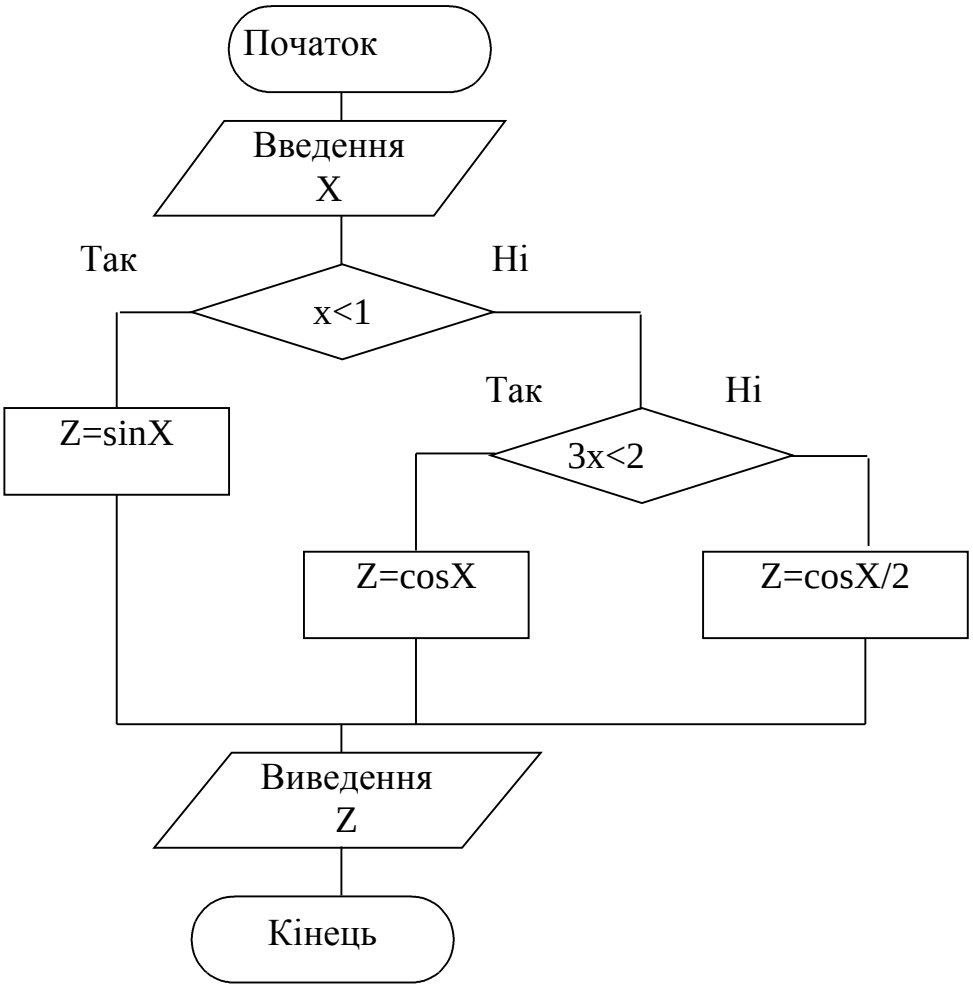


Рисунок 41

$$Z=\begin{cases}\sin(x), \text{ якщо } X<1;\\ \cos(x), \text{ якщо } 1<X<2;\\ \sin(x)/2, \text{ якщо } X>2\end{cases}$$

Програма.

```
program task4;
    var X, Z: real;
    begin
       readln (x);
         if X<1 then   Z := sin(x)
           else
              if X < 2 then
                  Z := cos(x)
```

```
            else
                Z := cos(x) /2;
        Write ('Z=',Z:5:3)
    end.
```

Для того, щоб виділити три гілки розв’язку, досить двох умовних операторів if X < 1 then Z := sin(x) else <умовний оператор 2>.
<умовний оператор 2>: if X <2 then Z := cos(x) else Z:= cos(x) /2. Слід звернути увагу на оператор (називатимемо так умовно процедуру) виведення write. У операторі write ('Z=', Z:5:3) дана команда на виведення константи 'Z= ' і змінної Z з форматом виведення: п'ять позицій на все число, з них три (на дробову частину).

5.5 Про процедуру write

Оператори виведення допускають використання вказівки про ширину поля, що відводиться під значення величини, що виводиться, в явному вигляді. Ширина поля виведення визначається типом пристрою, що використовується в даній ЕОМ. Форма подання змінних, що виводяться, визначається типом змінних: значення величин цілого типу виводяться в звичній формі; значення величин дійсного типу – у вигляді нормалізованого числа дійсного типу з порядком: значення логічного типу – у вигляді логічних значень TRUE і FALSE; значення символьних змінних – у вигляді відповідних символів.

Загальний вид запису операторів при виведенні значень цілого типу:

write(b:m); writeln(b:m);,

де b – ім'я змінної, що виводиться; m – константа або вираз цілого типу, що відводиться під значення .

Наприклад:

```
Write (dd : 6, ir : 8);
```

Значення змінних dd і ir розміщуються в одному рядку і займають відповідно шість і вісім позицій. Кожне значення розміщується у відведеному полі і займає крайні праві позиції. При цьому незаповнені позиції залишаються вільними, утворюючи проміжки. Якщо значення змінної не розміщується у відведеному полі, то збільшується число позицій.

При виведенні значень дійсного типу з фіксованою точкою вказується ширина поля, що відводиться під все значення і під дробову частину числа. Загальний вид запису операторів виглядає таким чином;

write(b : m : n); writeln(b : m : n);

де m – поле, що відводиться під запис значення; n – частина поля, що відводиться під дробову частину числа.

Наприклад:

```
write( a : 8 : 3);
```

В даному випадку під значення a виділяється вісім позицій, три з яких відводиться під дробову частину числа.

Якщо при виведенні дійсних значень не вказується кількість позицій, відведених під дробову частину числа, то результат виходить в нормалізованому вигляді з десятковим порядком.

Можна задавати кількість пропусків. Для цього необхідно записати оператор виведення у вигляді

wrlte(’ ’:q);

де q – константа цілого типу, яка вказує число пропусків.

Приклад розміщення інформації при виведенні:

```
Program  aaa;
const
  pi = 3.141592;
  t = 401;
  w = true;
  sim = 'D';
begin
    writeln('pi=', pi:8:6);
    writeln(t:6,’ ‘:6, w:4,’ ‘:6, sim:1)
end.
```

Інформація виводиться в два рядки у вигляді:

```
 pi=3.141592
401␣␣␣␣␣true␣␣␣␣␣␣d
```

␣ позначення проміжку.

5.6 Процедура введення

Для введення даних в мові Паскаль передбачені стандартні вбудовані програми (процедури) – READ і READLN. Оператор введення служить для введення даних в процесі виконання програми. Процедура READ використовується у вигляді:

– read (a1, a2, a3, ... an) – кожне значення, що вводиться, присвоюється послідовно змінним a1, a2, a3,..., an;

– readln (a1, a2, a3,..,an) – кожне значення, що вводиться, присвоюється послідовно змінним a1 a2, a3,..., an, після чого відбувається перехід на новий рядок (наступний оператор введення вводитиме дані з нового рядка);

– readln – перехід на новий рядок при введенні даних. Логічні дані в Паскалі вводити не дозволяється.

При введенні числові дані повинні розділятися проміжком або символом закінчення введення (клавіша Enter).

Приклад введення:

```
var a, b, c : real;
    t : integer;
    ……
    read (a, b, c);
```

```
readln;
read(к, t);
```

Після набору на екрані дисплея всієї програми і запуску її на виконання відбувається зупинка машини при зустрічі read(a, b, c). На клавіатурі ЕОМ необхідно набрати три дійсні числа, потім з нового рядка (виконується оператор readln) – два цілі числа відповідно оператору read(к, t).

Наприклад.

```
0.5 ␣ 6.23 ␣-7.1

 3 ␣ 48
```

При цьому змінні набувають такі значення: a = 0.5, b = 6.23, c = -7.1, к = 3, t =48, і виконання програми продовжиться.

Числа можна відділяти один від одного не тільки проміжками, але і символом закінчення введення (клавіша enter), тобто кожне число вводиться з нового рядка. Введення символьних даних має особливості. Оскільки проміжок, як і будь-який символ мови Паскаль, відноситься до символьних даних, символьні дані вводяться суцільним рядком відповідно до оператора введення. Нагадаємо, що одній змінній можна присвоїти значення тільки одного символа. Нехай є фрагмент програми

```
var a, b, c : char;
read( a, b, c);
```

Дані вводяться у вигляді: SNR. Змінні a, b, c набувають такі значення:

a = 'S', b = 'N', c = 'R'.

Для правильного введення символьних даних рекомендується перед кожним оператором введення символьних даних ставити оператор переходу на новий рядок readln, щоб введення здійснювалося завжди з нового рядка.

З введенням даних пов'язана стандартна функція EOLN. Вона приймає значення true, якщо досягнутий кінець рядка; інакше – значення false.

5.7 Оператор вибору

За практичними міркуваннями існує ще один тип умовного оператора. Оператор вибору дозволяє вибрати одне з декількох можливих продовжень програми. Структура оператора вибору:

CASE <ключ вибора> OF <список вибора> ELSE <оператор> END

Тут CASE, OF, ELSE, END – зарезервовані слова (випадок, із, інакше, кінець); <ключ выбора> – вираз порядкового типу (будь-якого з розглянутих, окрім типів REAL і STRING); <список вибора> – одна або більше конструкцій вигляду: <константа вибора> : <оператор>; <константа вибора> – константа того ж типу, що і вираз <ключ вибора>; <оператор> – довільний оператор.

Пояснимо загальний вид оператора. У цьому записі c – ключ вибору; ni : pi

–список вибору; ni – константа вибору; pi – оператор.

```
CASE c OF
   n1 : p1;
   n2 : p2;
   n3 : p3;
   ……..
  nn : pn
ELSE
   p
end;
```

Оператор працює таким чином. Спочатку обчислюються значення виразів <ключ вибору>, а потім в послідовності операторів <список вибору> відшукується такий, якому передує константа, яка дорівнює обчисленому значенню. Знайдений оператор виконується, після чого оператор вибору завершує свою роботу. Якщо в списку вибору не буде знайдена константа, відповідна обчисленому значенню ключа вибору, управління передається оператору, що стоїть за словом ELSE.

Частину ELSE <оператор> можна не писати, тоді за відсутності в списку вибору потрібної константи нічого не відбудеться і оператор вибору просто завершить свою роботу.

Задача 5. Залежно від ознаки q розрахувати значення функції Z.

$$Z==\begin{cases} 0, & якщо\ q=0; \\ \sin(x), & якщо\ q=1; \\ e^{x}, & якщо\ q=2; \\ \log(x), & якщо\ q=3; \\ x*x & в\ інших\ випадках \end{cases}$$

На рис. 42 приведена структура, яка називається "множинний вибір". Залежно від значення q, що вводиться, обчислення здійснюються за однією гілкою з п'яти.

Програма

```
Program task4;
 var X, Z: real; q : integer;
 begin
   write('Введіть X'); readln(x);
   writeln('Введіть ознаку: 0, 1, 2, 3, 0 – 0, 1 – sin(x), 2 – exp(x), 3 – log(x)');
   readln(q);
   case q of
     0 : Z :=0;
```

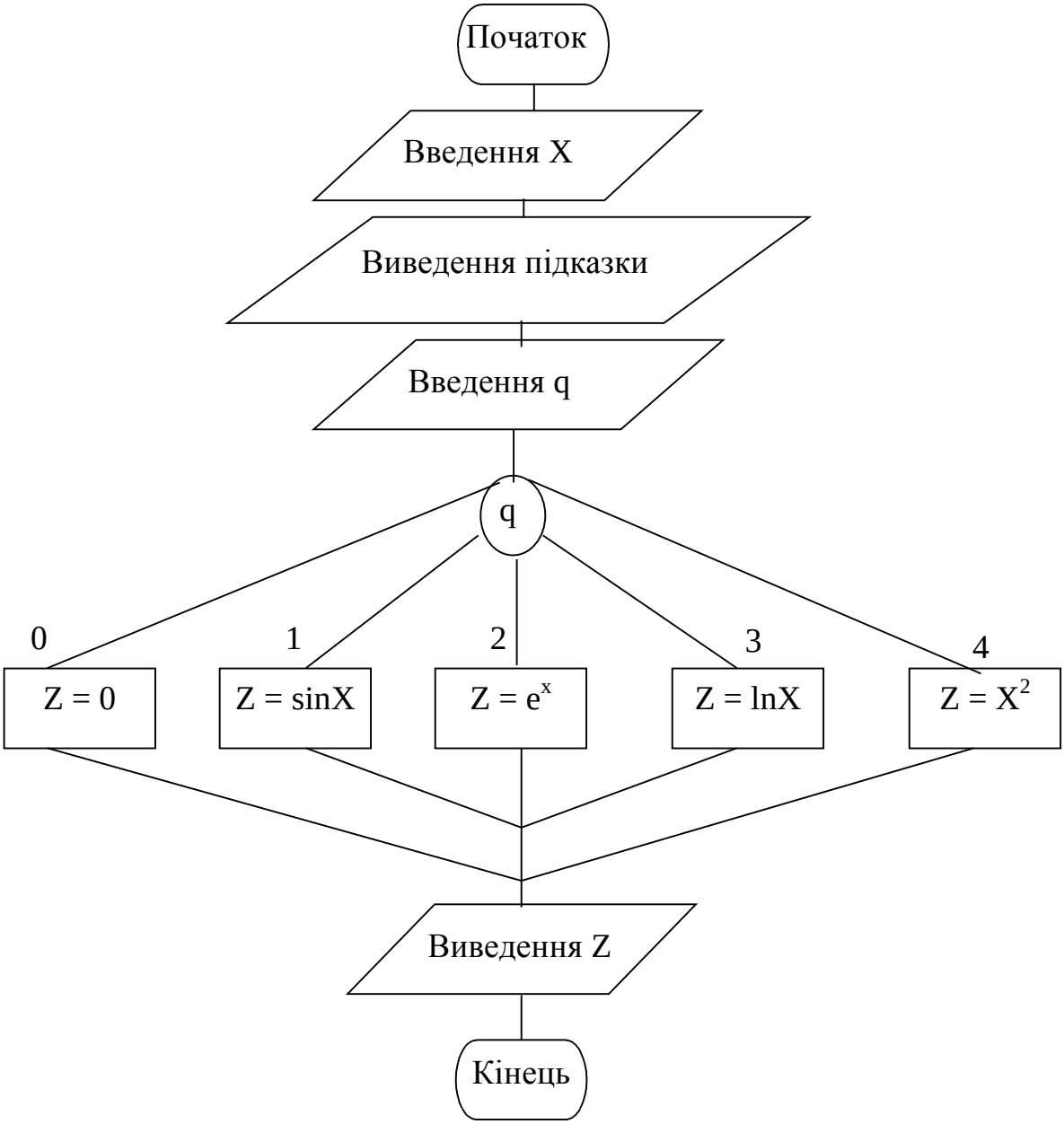


Рисунок 42 – Схема алгоритму задачі 5

```
    1 : Z := sin(x);
    2 : Z := exp(x);
    3 : Z:= ln(x)
    else
        Z := x*x
  end;
    write('Z = ',Z)
end.
```

Задача 6. Ввести два числа, знак арифметичної дії і вивести на екран результат відповідної дії.

Програма.

```
program task6;
var
    op :char;  x, y, z : real;
 begin
     write('x, y ='); readln(x.y);
     write('операція:'); readln(op);
      case op of
       '+' : z:= x + y;
        '-' : z:= x - y;
        '*' : z:= x * y;
        './': z:= x / y;
     else
        write('Таку дію не передбачено!');
     end;
      write('Результат = ', z)
end.
```

У програмі ключем вибору є змінна літерного типу. Залежно від введеного значення символа виконається одна з арифметичних дій: додавання, віднімання, множення або ділення. Якщо буде введений який-небудь інший символ, то результатом буде повідомлення: "Таку дію не передбачено!".

Програма вводить два рядки: перша містить два довільні числа, розділені пропуском, друга – символ арифметичної дії. На наступному рядку буде виведений результат.

Діаграма:

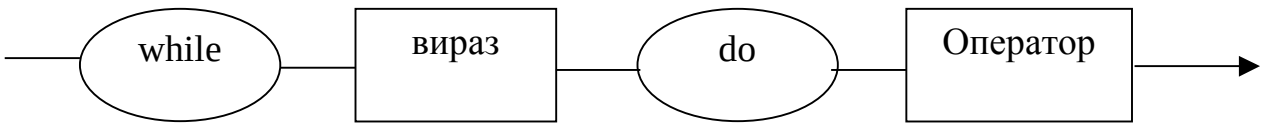


Рисунок 43 – Діаграма циклу з передумовою

Наприклад:

```
2 2
*
```

Результат = 4 або

```
18.5   + 0.12
```

Результат = 18.62

5.8. Оператори повторення

У мові Паскаль є три різні оператори, за допомогою яких можна запрограмувати фрагменти програм, що повторюються. Оператори повторення (цикли) передбачають виконання деяких операторів кілька разів. Якщо число повторень відоме наперед (до початку повторень), то в такій ситуації краще скористатися оператором циклу з параметром. У інших випадках слід використовувати операторів циклу з передумовою і післяумо-вою.

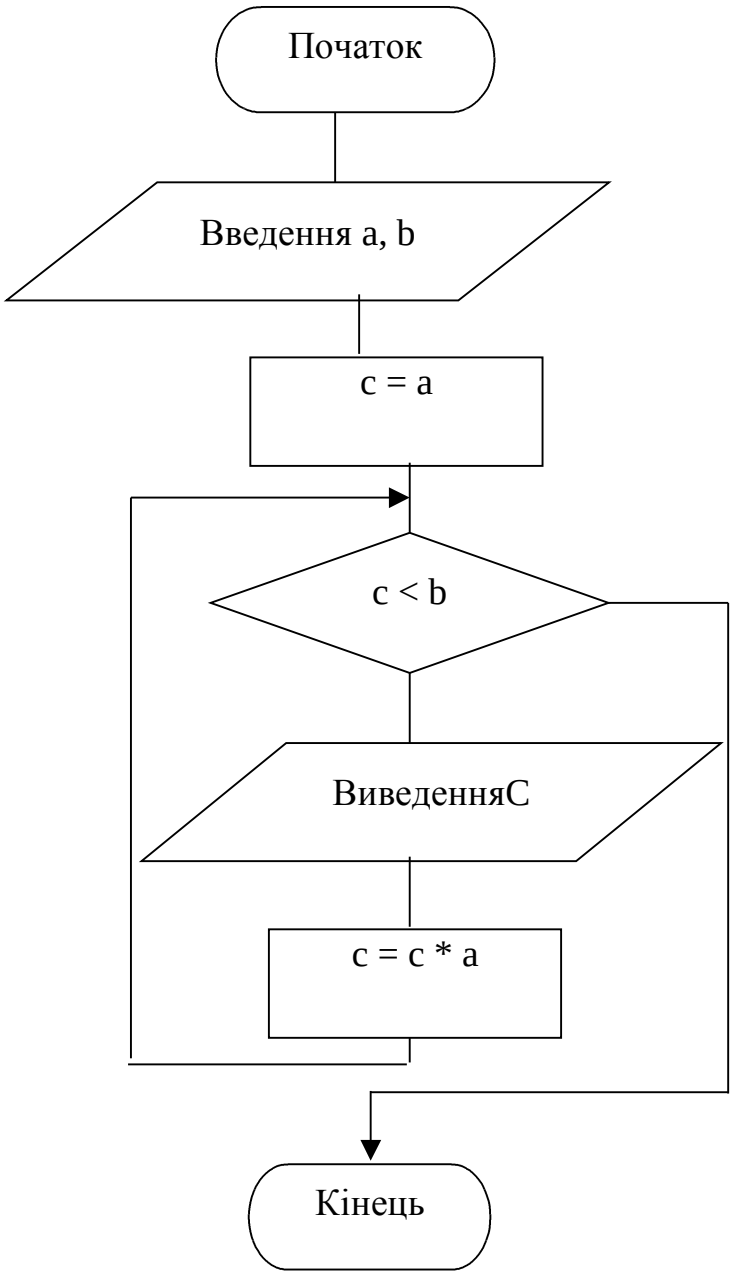


Рисунок 44

5.8.1. Оператор циклу з передумовою

Структура оператора(рис. 43):

WHILE <умова> DO <оператор>.

WHILE, DO – зарезервовані слова (поки виконується умова, робити);

<умова> – вираз логічного типу;

<оператор> – довільний оператор.

Якщо вираз <умова> має значення true, то виконується <оператор>, після чого обчис-лення виразу <умова> і його перевірка повторюються. Як-що <умова> має значення false, оператор while припиняє свою роботу.

Якщо в циклі необхідно виконати декілька операторів, то використовується складений оператор.

Задача 7. Нехай дані числа a, b (a>1). Одержати всі члени нескінченної по-слідовності $a,a^2,a^3,...$ менші числа b (рис.44).

```
program task7;
var a, b, c : real;
```

```
begin
```

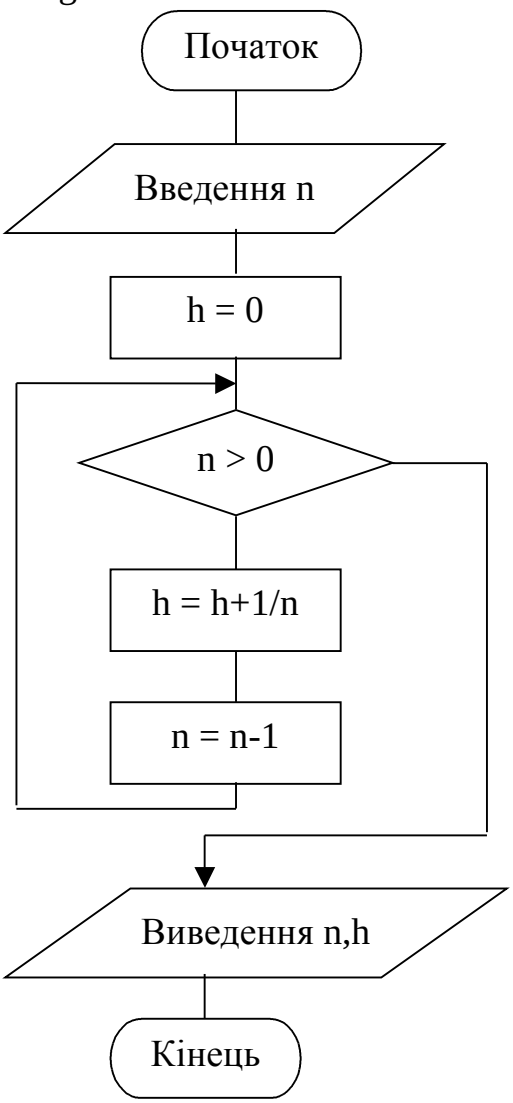


Рисунок 45

```
        readln(a, b); c:=a;
while c < b do
      begin writeln(c);
            c := c*a;
     end
  end.
```

При виконанні цієї програми змінна **c** послідовно приймає значення a, a^2, a^3, ... Зміна значення **c** відбувається до тих пір, поки воно не стане більше або рівне значенню b. Якщо a > b, то не буде виведено жодного члена послідовності.

Задача 8. Обчислити суму гармонічного ряду h = 1+1/2+1/3+...+1/n Підсумовування починаємо з n - ого члена. Перший доданок 1/n. У циклі n зменшується на одиницю; n=n-1. Підсу-мовування h = h + 1/n виконується до тих пір, поки n >0 (рис. 45).

```
      program taskS;
     var n: integer; h:real;
      begin readln(n); h:=0;
         while n > 0 do
              begin
                   h := h + 1/n; n := n - 1
              end;

            writeln('n =', n, 'h =', h)
end.
```

5.8.2 Оператор циклу з післяумовою

Структура оператора (рис. 46):

REPEAT <тіло циклу> UNTIL <умова>.

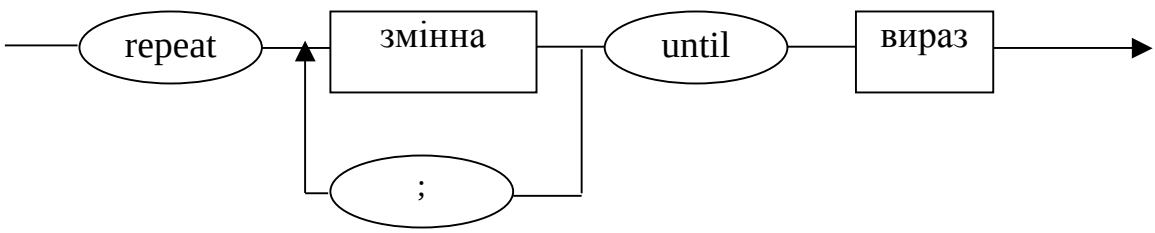


Рисунок 46 – Діаграма циклу з післяумовою

REPEAT, UNTIL – зарезервовані слова (повторювати доти, поки не буде виконана умова);

<тіло циклу> – довільна послідовність операторів,;

<умова> – вираз логічного типу.

Тіло циклу виконується хоча б один раз, після чого обчислюється вираз <умова>: якщо його значення є false, оператори тіла циклу повторюються, інакше оператор REPEAT ... UNTIL завершує роботу.

Продовження задачі 8. Напишемо програму для задачі 8 з використанням оператора циклу з післяумовою (рис. 47).

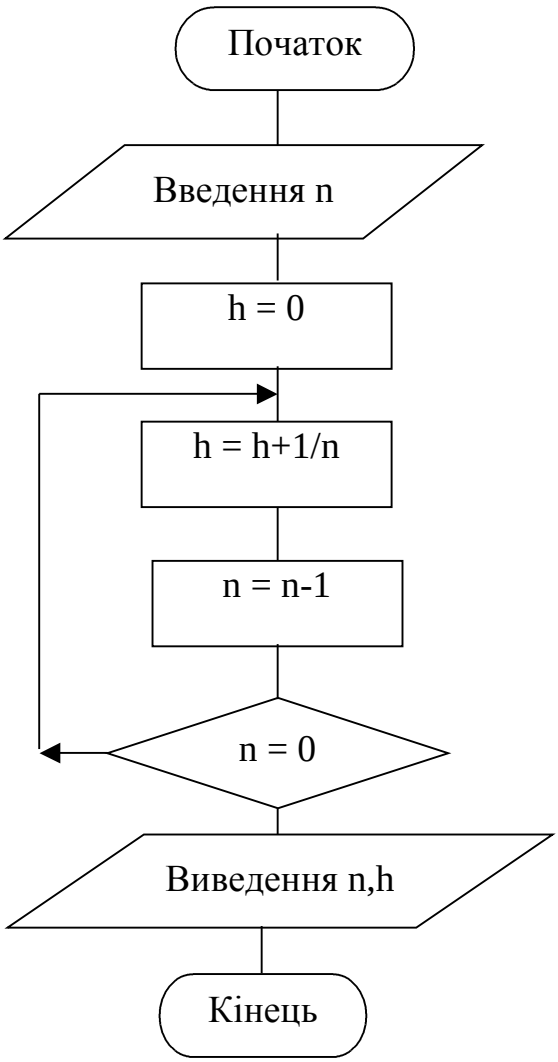


Рисунок 47

```
program task8;
 var n: integer; h:real;
  begin
      readln(n);
      h:=0;
      repeat
      h := h + 1/n;
      n := n – 1;
      until n = 0;
          write('n = ',n,'h = ',h)
          end.
```

5.8.3 Оператор циклу з параметром

Структура оператора:

FOR <парам. цикла>:=<поч. знач.> TO <кін. знач.> DO <оператор>

FOR, TO, DO – зарезервовані слова (для, до, виконати).

<парам. циклу> – параметр циклу (змінна типу integer);

<поч. знач.> – початкове, значення (вираз типу integer);

<кін. знач.> – кінцеве значення (вираз типу integer);

<оператор> – довільний оператор.

Передбачається, що у вирази <поч. знач.> і <кін. знач.> не входить параметр циклу.

Наприклад, for i:= 1 to n do s := s + i* i* i

Читається це так: для і від 1 до n виконати s := s + i^3.

При виконанні оператора FOR на початку обчислюється вираз, який задає значення параметру циклу, потім це значення присвоюється параметру циклу.

Після цього:

– проводиться перевірка умови <парам. циклу> <= <кінцеве. знач.>; якщо умова виконується, то перейти до пункту 2, інакше до пункту 4;
– виконується оператор <оператор>;
– нарощується змінна <парам. циклу> на одиницю і перехід до пункту 1;
– завершується роботи.

Задача 9. Написати програму розрахунку середнього росту студентів в групі.

Схема алгоритму наведена на рис.17.

Запишемо програму.

```
program task9;{Розрахунок середнього росту студентів в групі.}
var
        n,i: integer;
        s,r: real;
begin
        readln(n)
        s:=0;
        for i :=1 to n ,
          begin
                readln(r);
                s := s+r;
          end;
        s := s/n; write('Середній ріст =', s)
end.
```

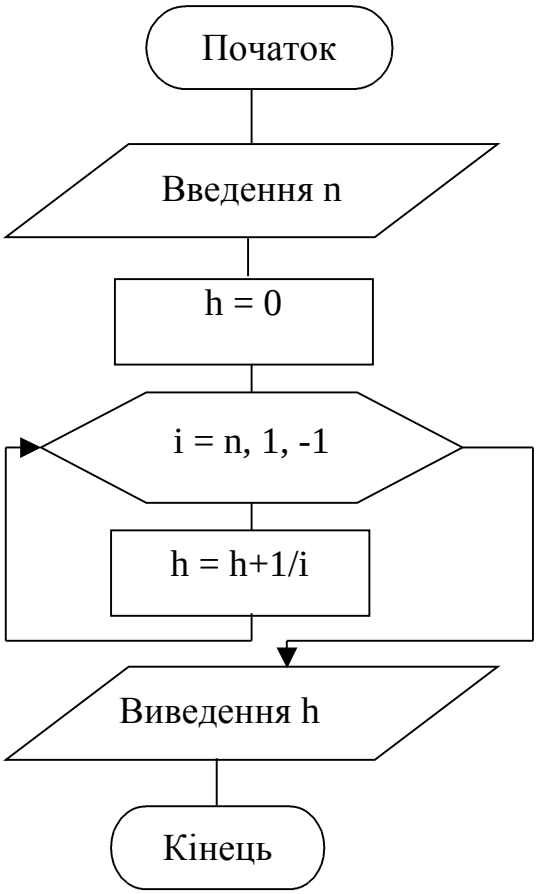


Рисунок 48

У цій програмі: n – кількість студентів в групі; i – номер студента; r – ріст студента; s – змінна, в якій накопичується сума, а потім в цю ж змінну записується середній ріст студентів, розрахований як середнє арифметичне. Ці змінні описані в програмі: i та n як цілі, а s і r як дійсні. У програмі використовується оператор циклу з параметром, оскільки наперед відома кількість повторень. В циклі повинні виконуватися два оператори, тому слід використовувати складений оператор begin - end.

Крок нарощування параметра циклу строго постійний і рівний (+1). Існує інша форма оператора FOR, де крок (-1)

FOR <парам. циклу> := <поч. знач.>DOWNTO <кін. знач.> DO <оператор>

Задача 10. Обчислити суму гармонічного ряду h = 1 + 1/2 +1/3 + ...+ 1/n. Програму написати використовуючи оператор FOR.

У задачі 7 програму записано з використанням операторів while і repeat. Для зображення схеми алгоритму скористаємося блоком "модифікація"

(рис. 48).

```
 program task10;
{Задача 8, але з використанням оператора циклу FOR.}
 var i, n: integer; h: real;
 begin
   readln(n);
   h := 0
  for i := n downto 1 do
          h := h + 1/i;
    Writeln(h)
end.
```

У цій програмі в циклі виконується лише один оператор, у зв'язку з цим складений оператор опущений.

Діаграма для оператора циклу з параметром:

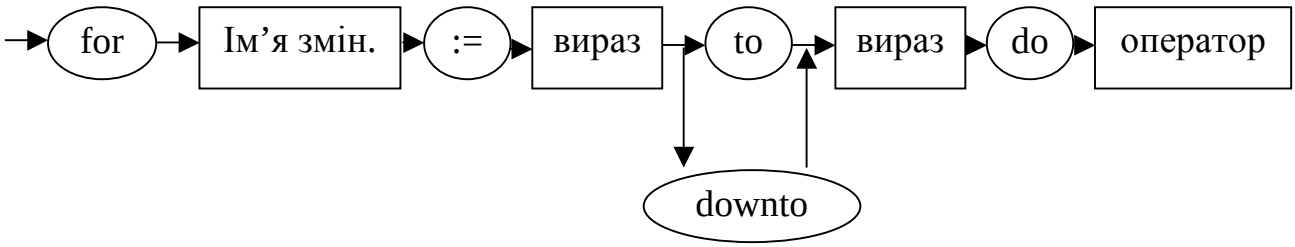


Рисунок 49 – Діаграма циклу з параметром

5.9 Мітки і оператори переходу

Розглянутих операторів цілком достатньо для написання програм будь-якої складності. Проте, в деяких випадках використання операторів переходу може спростити програму.

Оператор переходу має вигляд:

GOTO<мітка>.

Тут GOTO – зарезервоване слово (перейти на мітку).

<мітка> – мітка.

Мітка в Турбо Паскалі – це довільний ідентифікатор, що дозволяє іменувати деякий оператор програми і таким чином посилатися на нього. В цілях сумісності із стандартною мовою Паскаль в мові Турбо Паскаль допускається як мітки використання цілих чисел без знака. У стандартному Паскалі мітка – це ціле без знака. Мітка розташовується безпосередньо перед оператором, що позначається, і відділяється від нього двокрапкою. Перед тим, як з'явитися в програмі, мітка повинна бути описана. Опис міток складається із зарезервованого слова LABEL (мітка), за яким йде список міток.

label 222, a1, met;

Описаною міткою повинен бути помічений тільки один оператор

програми.

Оператор переходу вказує, що подальша робота повинна продовжуватися в іншій частині тексту програми, а саме з того місця, де знаходиться мітка.

Нехай програма містить послідовність операторів:

x := 2; a := b; goto 99;

14 : a := 0; x := b; 99 : y := x; write(x)

В цьому випадку спочатку виконуються опе-ратори x := 2 і a := b, потім йде перехід до оператора, поміченого міткою 99, тобто до оператора y := x. Після оператора y:= x буде виконано оператор write(x).

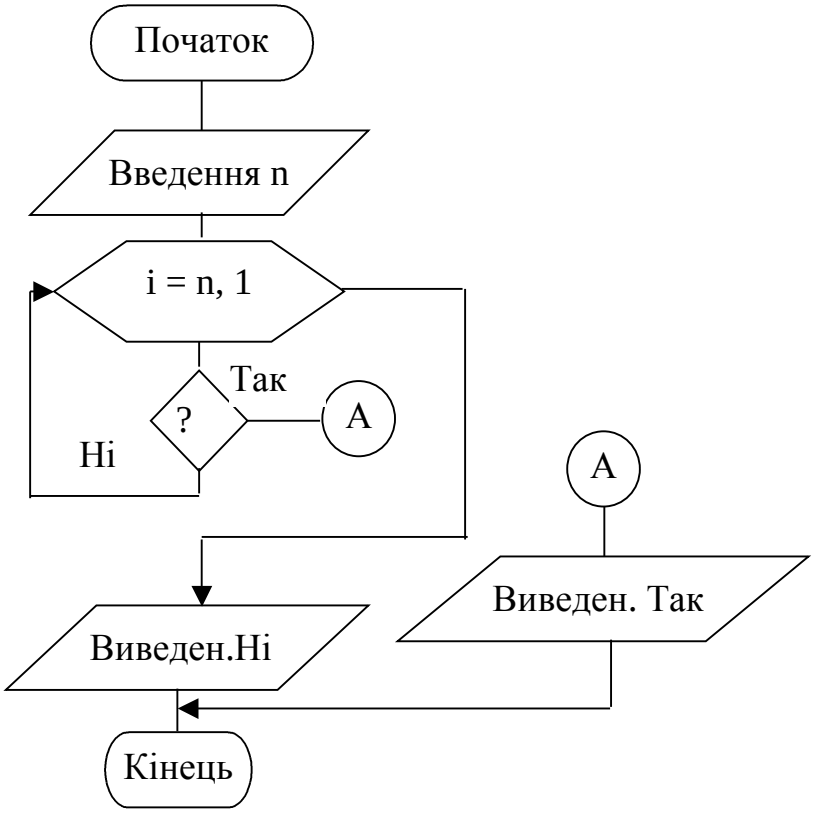


Рисунок 50

Задача 11. З'ясувати, є або немає серед чисел $\cos(i^3)\sin(i\cdot n)$, i=1,...,n, менших 0.0001. Якщо є, то виводиться "є", якщо ні – "ні" (рис. 50).

Програма,

```
program task11;
 label ml;
 var i, n : integer;
 begin
  readlh(n);
  for i := 1 to n do
     if cos(i*i*i)*sin(i*n) < 0.0001 then
       begin
            write('існує');
            goto ml
       end;
      write('Ні');
ml : end.
```

Якщо виявляється, що деяке число менше 0.0001, то наступні числа вже не розглядаються, йде висновок "існує" і перехід на кінець програми. Перед end розташований помічений міткою ml порожній оператор. Основне призначення порожнього оператора – дати можливість виходу з середини програми або складеного оператора. Знак питанння в логічному блоці на рисунку 50 означає, що в цьому блоці перевіряється умова, написана в програмі.

Програму можна написати з двома операторами переходу:

```
program task11;
  label m1, m2;
  var i, n : integer;
  begin   readln(n);
          for i := 1 to n do
          if  cos(i*i*i)*sin(i*n) < 0.0001  then goto ml;
          write('Hi');   goto m2;
 ml:  write('існує');
 m2: end.
```

За допомогою оператора переходу, розташованого поза умовним оператором або оператором циклу, не можна перейти всередину цього умовного оператора або оператора циклу. Часте використання оператора GOTO вважається поганим тоном. Це ускладнює читання програми і порушує принципи структурного програмування.

6 РОБОТА З МАСИВАМИ

При описі масиву необхідно вказати загальне число вхідних в масив елементів і тип цих елементів. Наприклад:

```
Var
a: array [1 ..10] of real;
b: array [0..50] of char;
c: array [-3..4] of boolean;
```

При описі масиву використовуються зарезервовані слова ARRAY і OF (масив із). За словом ARRAY в квадратних дужках указується тип-діапазон, за допомогою якого компілятор визначає загальне число елементів масиву. Тип-діапазон задається лівою і правою межами зміни індексу масиву, так що масив a складається з 10 елементів, масив b – з 51, а масив c – з 8.

Доступ до кожного елементу масиву в програмі здійснюється за допомогою індексу – цілого числа (точніше, виразу порядкового типу), що служить своєрідним ім'ям елементу в масиві (якщо ліва межа типу-діапазону рівна 1, індекс елементу збігається з його порядковим номером). При згадці в програмі будь-якого елементу масиву відразу за ім'ям масиву повинен слідувати індекс елементу в квадратних дужках, наприклад:

```
b[17]:='f';
c[-2]:=a[1]>a[2];
for k:= 1 to 10 do a[k]:= 0;
```

У правильно складеній програмі індекс не повинен виходити за межі, визначені типом-діапазоном. Наприклад, можна використовувати елементи a[1], b[38], c[0], але не можна a[0] або c[38]. Турбо Паскаль може контролювати використання індексів в програмі на етапі компіляції і на етапі рахунку.

Для ілюстрації роботи з масивами складемо програми для задач,

алгоритми яких розроблені раніше.

Задача 12. Визначити кількість студентів в групі, що мають ріст вище середнього. Відповідає задачі 7 в темі 2.

```
program taskl 2;
const
  m = 40;
var
  i, n, : integer;
  s: real;
  r: array [1..m] of  real;
begin
   readln(n);
   s:=0;
   for i:= 1 to n do
      begin
         readln(r[i]);
        s:=s+r[i]
    end;
   s:=S/h;
   k:=0;
   for i:= 1 to n do.
   if r[i] > s then k = k+i;
  write (‘Кількість студентів ростом вище середнього = ', k)
end.
```

m – константа цілого типу. Максимальна кількість студентів в групі може бути 40. Для величин ростів студентів використовується одновимірний масив з m елементів, n – кількість студентів в кожній конкретній групі, для якої виконується розрахунок, i – номер чергового студента, k – лічильник кількості студентів, що мають ріст вище середнього, r – масив росту студентів.

Задача 13.

Задана послідовність з n чисел. Визначити кількість додатних, від’ємних і нульових елементів. Це задача 8 з теми 2, де розроблений алгоритм і приведена схема алгоритму. Запишемо програму,

```
program task13;
const   n=20;
var
    i, k, l, m,n: integer;
    a: array [1..h] of real;
begin
    write('Введіть n, n <= 20');
    readln(n);
    for i :=1 to n do
      begin
```

```
        write('Елемент ',i); readln(a[i])
    end;
    e:= 0; r:= 0; m := 0
  for i := 1 to n do
     if a[i]<0 then
     k:=k+1
  else
     if a[i]>0 then
     e:=e+1
  else
     m := m + 1;
  write('Від'ємних ',  k,  'Додатних ',  e,  ' Нульових', m)
end.
```

У програмі введені допоміжні оператори виведення write('Введіть n, n <= 20'), write('Елемент ', i). Ці оператори служать для пояснення початкової інформації. При виконанні програми на екран виводиться "введіть n, n <= 20" перед зупинкою в очікуванні введення. Це є підказкою для людини, що використовує програму. Виведення – підказку перед оператором введення слід робити завжди.

Задача 14. Група з N студентів здавала в зимову сесію M іспитів. Відомі результати іспитів. Підрахувати середній бал з кожного предмета. У задачі 13, приведеної в темі 2, розроблений алгоритм розрахунку середнього бала для кожного студента. У нашій задачі відмінність полягає у тому, що необхідно найти суму елементів кожного стовпця Σa_{ij} тоді середнє $S_j = S/N$. Результатом розрахунку буде одновимірний масив S, що складається з M елементів.

Початкові дані: N – кількість студентів в групі, M – число іспитів в сесію, матриця A – оцінки, a[i,j] – оцінка i - го студента з j - го іспиту. Масив A –двовимірний масив, що складається з N рядків і M стовпців. Розмір масиву – N×M. Розмірність масиву A рівна двом, тобто дві координати (номер рядка і номер стовпця) визначають положення кожного елементу.

Програма.

```
Program task14;
Const  n = 30;m = 5;
var
        i, j: integer;
        s : array [1..m] of real;{Опис одновимірного масиву s}.
        a: array [1..n, 1..m] of integer;{Опис двовимірного массива}
begin
        for i := 1 to n do
        for j := 1 to m do
        readln a[i,j]; {Введення елементів матриці}
        for j := 1 to m do
```

```
        begin     s[j]:=0;
            for i := 1 to n do s[j]:= s[j]+ a[1,];
        {Підсумовуються елементи}
        { кожного стовпця матриці
          s[j]:=s[j]/n
        end;
for j := 1 to m do write (s[j])   {Виведення одновимірного масиву}
end.
```

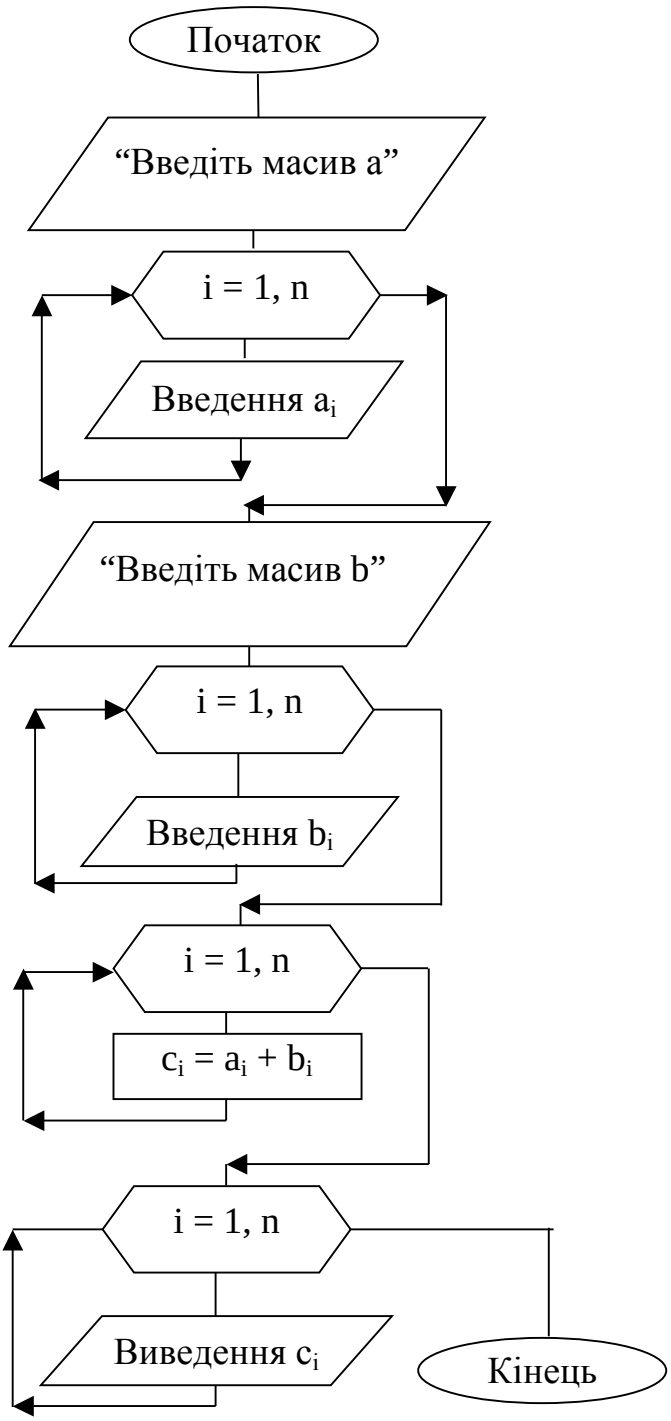


Рисунок 51 – Схема задачі 15

6.1 Робота з одновимірними масивами

Задача 15. Створити програму додавання двох векторів.

Вектор в програмі може бути представлений у вигляді одновимірного масиву. Нехай задані два одновимірних масива a і b, які складаються з n елементів (рис.51). В результаті додавання двох векторів вийде масив такого ж розміру, кожний елемент якого дорівнює $c_i = a_i + b_i$. В програмі повинні бути організовані цикли для введення одновимірного масиву a, для введення одновимірного масиву b, для розрахунку кожного елементу результуючого масиву c і для виведення результату.

Програма:

```
Program slogen;
{Додавання 2-х одновимірних масивів}
uses crt;
   const n = 10;
    var
        i, sum : integer;
       a, b, c : array [ 1..n ] of integer;
   begin
        clrscr;
        gotoxy(3,1);
```

```
  write (‘Введіть масив a’);
    for i := 1 to n do{Введення масиву ai}
      begin
       gotoxy(1+5*i, 2);
       readln(a[i])
      end;
    gotoxy(3, 3);
    write(‘Введіть масив b’);
    for i := 1 to n do
    {Введення масиву bi}
       begin
        gotoxy(1+5*i, 4);
        readln(b [i])
       end;
    for i := 1 to n do c[i] := a[i] + b[i];
{Розрахунок елементів масиву ci}
    gotoxy(3,9);
    writeln(‘Сумарний вектор’);
    for i := 1 to n do
     {Виведення масиву ci}
         begin
                 gotoxy(1+5*i, 10);
                 write(c[i]);
          end
     end.
```

Одновимірні масиви a, b, c описані в розділі описання змінних. Для звертання до кожного елементу масиву використовувався оператор циклу з параметром i, де i – перебирає всі значення від i до n, визначаючи цим кожен елемент масиву. Процедури clrscr і gotoxy(i,k) описані в модулі crt, тому за заголовком програми слідує uses – фаза. Clrscr – процедура гасіння екрана. Процедура gotoxy(i,k) переміщує курсор в i-ту позицію k-го рядка екрана. Параметри процедури i та k – величини типу integer. В цій програмі процедура gotoxy(i,k) використовується для наглядного введення і виведення масивів. Елементи масиву a і b: a[1], a[2], a[3], a[4], … , a[10] і b[1], b[2], b[3], b[4], …, b[10], вводяться в позиціях 6, 11, 16, 21, ..., 51 рядків 2 і 3 відповідно. Результат – масив c – виводиться в позиціях 6, 11, 16, 21, ..., 51 рядка 10.

Задача 16. Визначити чи містить заданий масив число рівне s.

Необхідно розробити програму, яка забезпечить введення з клавіатури десяти чисел і збереже їх в деякому одновимірному масиві. Потім програма повинна зробити запит в користувача введення ще одного числа. Після введення даного числа програма здійснить перевірку елементів масиву на наявність в них числа, рівного введеному і, якщо таке є, видасть

відповідне повідомлення на екран. Тобто користувач отримає інформацію

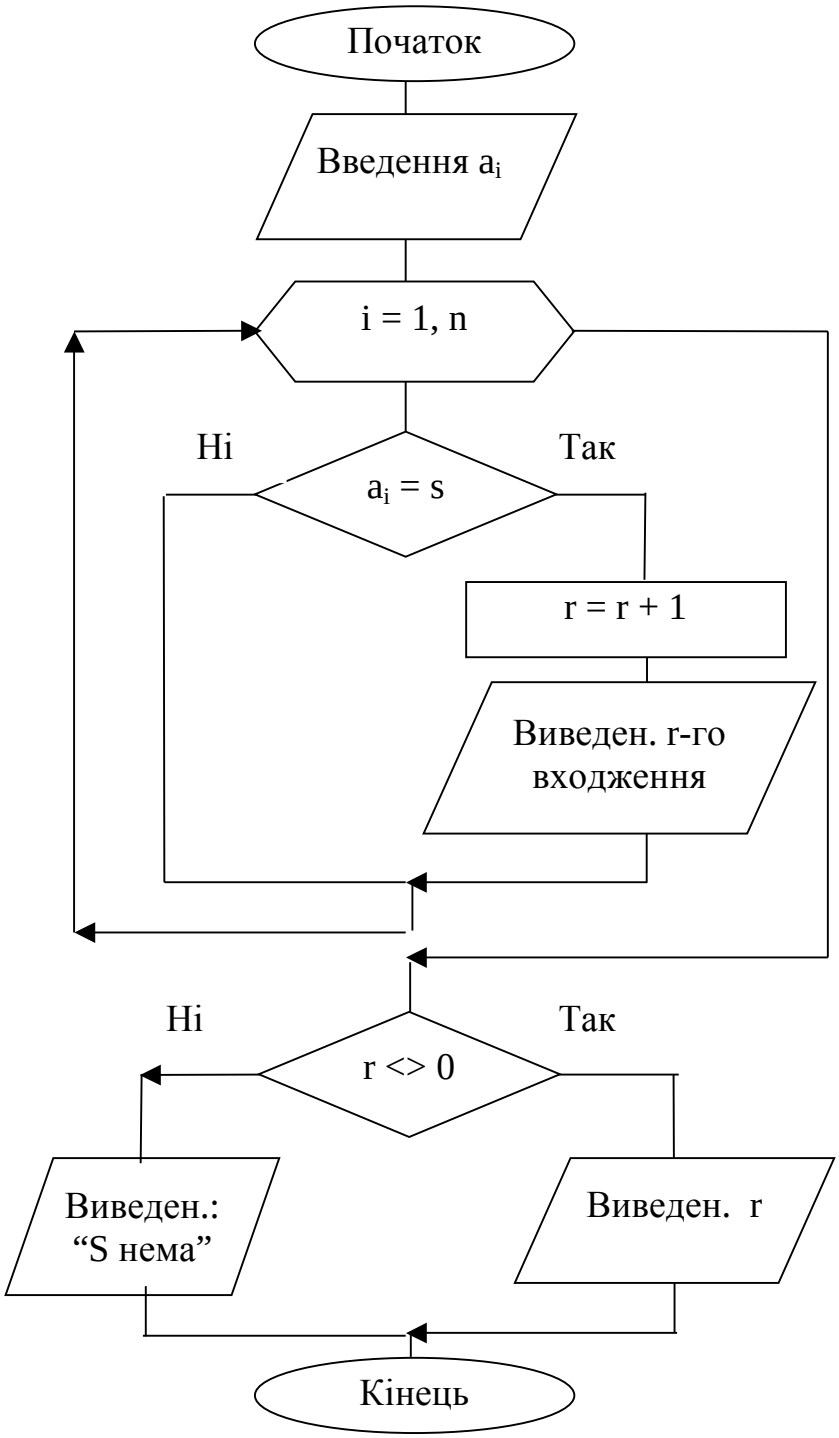


Рисунок 52 – Схема задачі 16

про те, чи міститься останнє введене число в масиві.

Деталізувати введення одного масиву і виведення заданого числа s в схемі алгоритму (рис. 52) немає необхідності.

Програма:

```
program poisk;
{Послідовний пошук в масиві}
```

```
const n =10;
var
i, s, r : integer;
a:array[1..n] of integer;
begin
for i := 1 to n do {Введення одновимірного масиву ai}
begin
   write('Введіть', i ,'–й елемент масиву ');
   readln(a[i])
end;
   write('Введіть число для пошуку');
   readln(s);
for i := 1 to n do
   begin
      if a[i] = s then   begin
      r :=r+1;
```

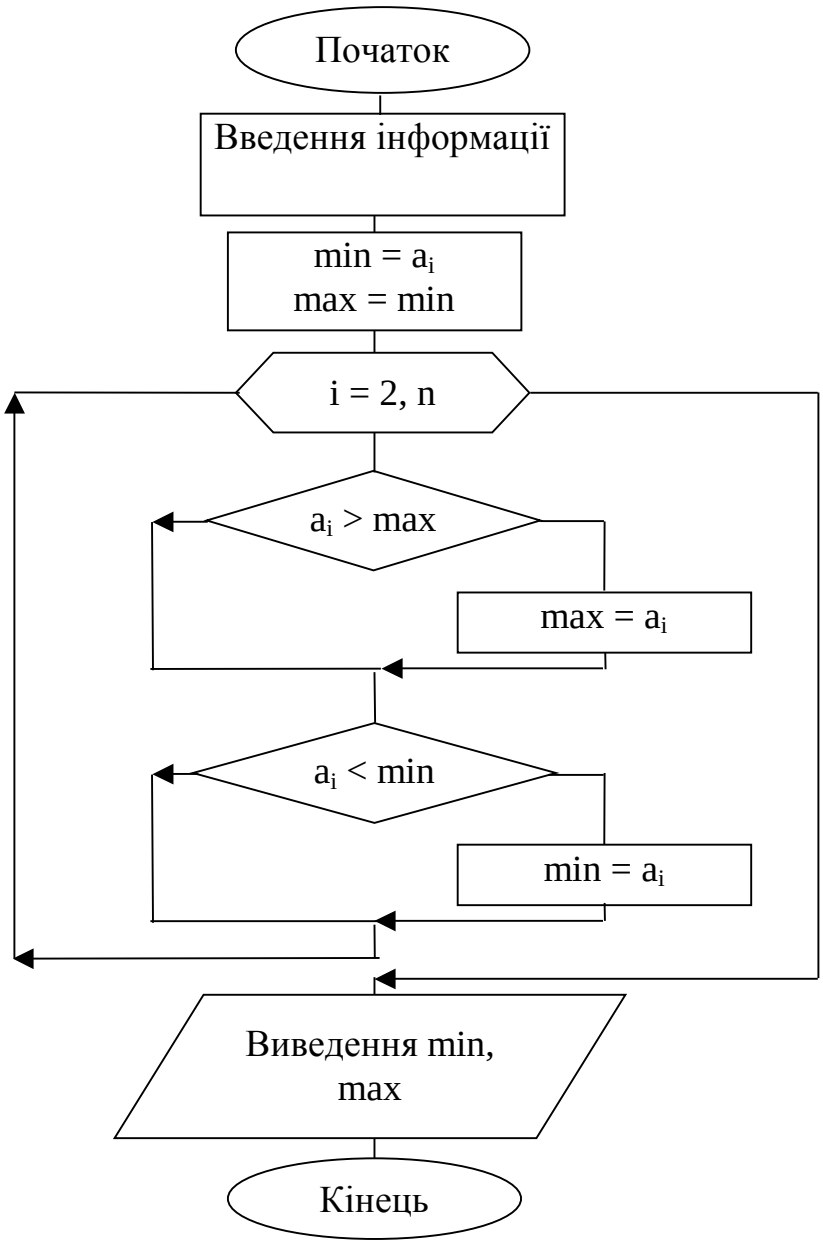


Рисунок 53 – Схема задачі 17

```
writeln('Знайдено', r ,'–входжень числа ',s,' в масив a в позиції', i);
        end;
     end;
          if r <> 0 then
writeln('Число ',s,'зустрічається в масиві a', r ,' раз')
                    else
writeln('Число ', s, ' не зустрічається в масиві a')
 end.
```

Перед оператором введення readln(a[i]) виконується оператор виведення write ('Введіть', i,'–й елемент масиву'), який представляє собою підказку для користувача при введенні вихідної інформації. При i = 1 на екрані висвітиться: введіть 1-й елемент масиву, маркер зупиниться в кінці виведе-

ного рядка, ЕОМ при-зупинить виконання програми до моменту введення користувачем числової інформації.

При i = 2 на екрані висвітиться інформація:

Введіть 2-й елемент масиву, і так далі до кінця виконання циклу.

Задача 17. Знайти мінімальний та максимальний члени послідовності a, яка складається з 10-ти елементів.

Присвоїмо змінним min і max значення першого елементу і далі будемо перебирати всі елементи послідовності від 2 до n, порівнюючи a_i з max і з min (рис.53). Якщо a_i виявиться більше ніж max, то max присвоюється значення a_i. Якщо a_i виявиться менше ніж min, то min присвоюється значення a_i. При закінченні циклу max прийме значення найбільшого елементу послідовності, а min – найменшого.

Програма:

```
program minmax;
{Пошук мінімального і максимального значення}
const n = 10;
type
  list = 1..n;
var
  i : list;
  min, max : integer;
  a ; array [list] of integer;
begin
  for i := 1 to n do
     begin
       write('Введіть',i ,'–й елемент масиву');
       readln(a[i])
     end;
  min := a[1]; max := min;
  for i := 2 to n do
      begin
        if a[i] > max then max := a[i];
        if a[i] < min then min := a[i];
      end;
    write ('Максимальний елемент =',max,
'Мінімальний елемент =', min,)
  end.
```

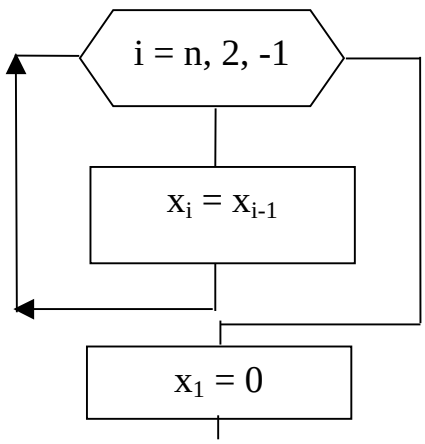


Рисунок 54

В програмі показано що задана в розділі описання типів змінна list може бути використана в якості граничної пари в описанні масиву .

Задача 18. Всі члени послідовності X з 10 елементів зсунути на одну позицію

праворуч.

Фрагмент схеми (рис.54) демонструє алгоритм зміщення елементів масиву на одну позицію вправо. Кожному попередньому елементу x_i присвоюється значення попереднього x_{i-1}. Для x_1 такого значення не існує, отже $x_i=0$.

Програма :

```
Program sdvig;
{Зсув масиву на одну позицію праворуч }
const n=10;
type
      mas=array[1..n] of integer;
{Визначення типу масиву}
var
     i :  integer;
   X : mas;
begin
        for i=1 to n do
          begin
                write('Введіть ', i, ' –й елемент масиву');
                readln(x[i]);
          end;
        for i := n to downto 2 do x[i] := x[i-1]; x[1] := 0;
          writeln('Результат');
            for i := to n do write(x[i] :4);
end.
```

В програмі визначений новий тип даних – масив, який складається з 10-ти компонентів цілого типу.

Задача 19. Скласти програму впорядкування елементів масиву x, який складається з nd елементів, розташувавши їх в порядку зростання в тому ж самому масиві.

Впорядкування елементів масиву за зростанням можна виконати використовуючи метод знаходження найменшого елементу. В початковій послідовності знайдемо найменший елемент і запам'ятаємо його порядковий номер k. Елемент x_1 поміняємо місцями з елементом $x_к$. Після цього найменший елемент опиниться на першому місці. Таку ж процедуру необхідно проробити з тими елементами, що залишились від 2 до nd. В цьому випадку найменший елемент $x_к$ потрібно поміняти місцями з другим елементом. На другому місці опиниться другий за величиною елемент. Процедуру пошуку потрібно повторити nd-1 раз. Останній елемент можна не порівнювати сам з собою. В алгоритм пошуку мінімального елементу вводимо блок для запам'ятовування порядкового номеру цього елементу. Такий метод впорядкування масивів називають **методом перестановок**.

Приведемо схему алгоритму рішення цієї задачі повністю (рис. 55)

Блок 2 – введення розміру масиву; блоки 3, 4, 5 – введення одновимірного масиву. Блок 6 – цикл для визначення порядкового номера елементу, з якого починається пошук мінімального. Блок 7 – початкове значення xmin і k. xmin – змінна для елементу, який має найменше зна-чення, k – змінна для порядкового номера цього елементу. Блоки 8, 9, 10 – алгоритм пошуку мінімального елементу послідовності. Блок 11 – перестановка мінімального і і-го (з якого почали пошук) елементів. Блоки 12,13– виведення впорядкованої послідовності.

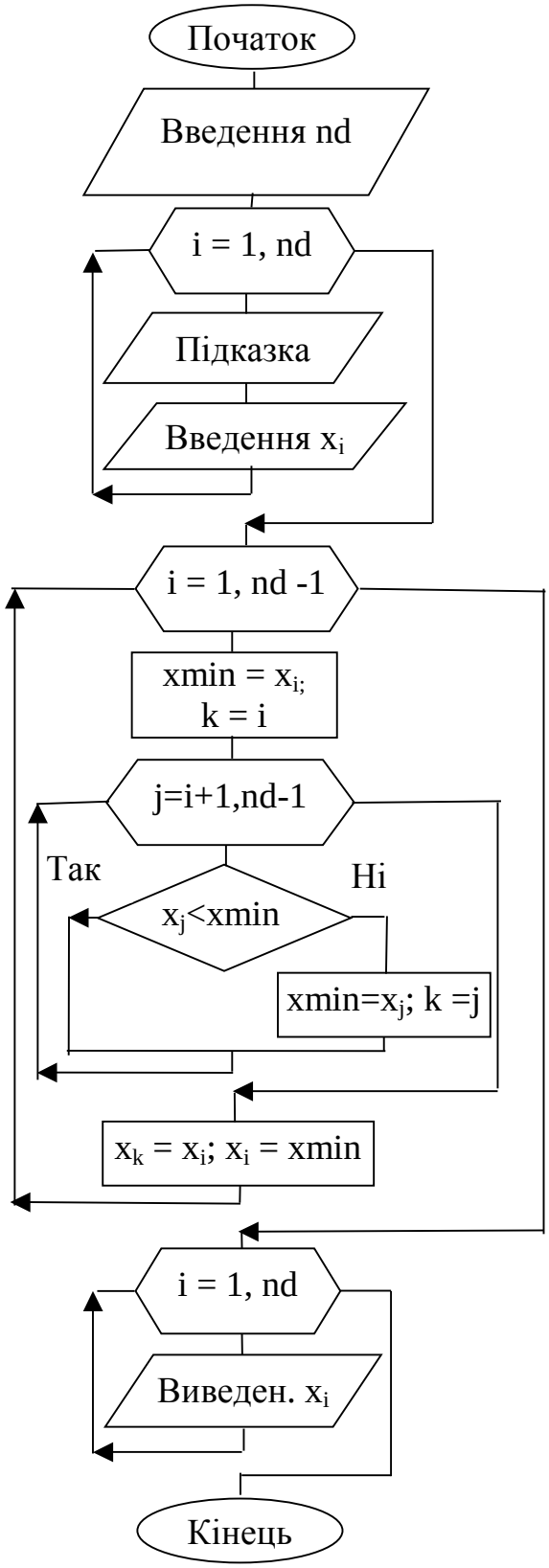


Рисунок 55 – Схема до задачі 19

Програма:

```
program perestan;
{Впорядкування елементів масиву методом перестановки}
const nmax = 100;
type
  mass = array [1.. nmax] of real;
var
  i, j, k, nd : integer;
  xmin : real;
  x : mass;
begin
  read(nd);
  for i := 1 to nd do
     begin
       write('Введіть',i,'-й елемент масиву');
       readln(x[i])
     end;
  for i := 1  to nd-1 do
     begin
       xmin := x[i]; k := i;
         for j := i+1 to nd do
           if x[j] < xmin then
             begin
               xmin := x[j];
               k := j
             end;
```

```
            x[k] := x[i];
            x[i] := xmin
          end;
       for i := 1 to nd do write(x[i]:8:2)
end.
```

В програмі під масив x відводиться 100 комірок пам’яті. Очевидно що змінна nd не може бути більше 100. Було б добре доповнити програму перевіркою nd при введенні. Якщо введене значення nd більше 100, дати відповідне повідомлення і повернутися до початку введення.

Задача 20. Скласти програму впорядкування масиву x, який складається з nd елементів, розташувавши їх в порядку зростання в тому ж масиві. Використовувати метод “бульбашки”.

Сортування множини даних є однією з центральних проблем оброблення даних. Задачу впорядкування елементів масиву можна вирішувати різними способами. **Бульбашковий метод** полягає в тому, що послідовно порівнюються між собою два сусідніх елемента і міняються місцями якщо $x_{j-1} > x_j$.

Програма:

```
program puzirok;
{Впорядкування елементів масиву методом порівняння двох сусідніх елементів}
  const nmax = 100;
  type
    mass = array [1..nmax] of real;
  var
    i, j, nd : integer;
    y ; real;
    x : mass;
  begin
    read(nd);
    for i := 1 to nd do
       begin
         write(‘Введіть ’, i, ’-й елемент масиву’);
         readln(x[i])
       end;
    for i := to 2 do
       begin
          for j := nd downto i do
             if x[j-1] > x[j] then
               begin
                  y := x[j-1];
                  x[j-1] := x[j];
                  x[j] := y
               end;
```

```
    end;
  for i := 1 to nd do write(x[i]:8:2)
end.
```

При кожному значенні і найменший елемент переміщується до і-1 позиції в масиві.

Задача 21. Представити в вигляді діаграми підсумок діяльності чотирьох агентів фірми зі збуту n-ї продукції.

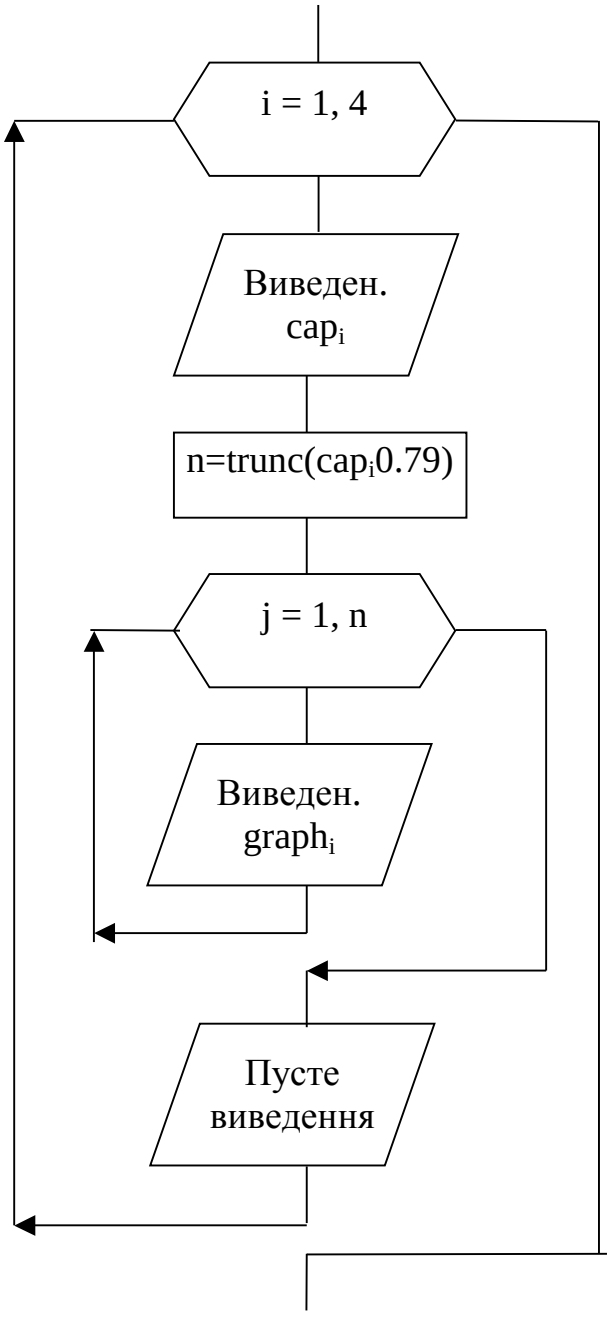


Рисунок 56– Схема до задачі 21

Просту діаграму можна побудувати використовуючи засоби псевдографіки. Для збереження об'ємів збуту продукції чотирьох агентів виділимо масив з 4-х елементів cap. Тоді cap – об'єм і-го агента. Ця вихідна інформація повинна вводитись з клавіатури. Загальний об'єм збуту $num=\sum_{i=1}^{4} cap_i$. Об'єм збуту в процентах від загальної суми $cap_{i=}\frac{cap_i}{num}\times 100$ необхідно подати у вигляді стовпцевої діаграми, зобразивши стовпці відповідні кожному агенту своїм особливим символом псевдографіки напри-клад ASCII-кодами 176, 177, 178 і 218 (це прямокутники різного ступеня яскравості). Якщо n – це процентний вклад і-го агента в загальний об'єм збуту продукції, то символ псевдографіки відповідний і-му агенту graph(i) повинен виводитись на екран n раз.

for j := 1 to n do write(graph(i)).

У вигляді схеми (рис.56) представимо фрагмент алгоритму – виведення діаграми на екран.

і – порядковий номер агента.

Виведення cap – виведення числового значення об'єму збуту і-го агента.

Функція trunc дає цілу частку аргументу. Аргументом є об’єм збуту помножений на 0.79. Кое-фіцієнт 0.79 вводиться для враху-вання тієї обставини, що в рядку екрана монітора може поміститися 79 символів. Перевищення гранич-ного значення приводить до авто-матичного переведення курсора на наступний рядок і до викривлення графіка. Така ситуація може трапитись в тому випадку, коли на одного агента припадає більше 80% об’єму збуту.

Програма:

```
program grafik;
uses crt;
const
graph : array [1..4] of char = (#176, #177, #178, #219);
{Типізовані константи символів для стовпцевих діаграм}
var
  cap : array [1..4] of real; {Об’єм збуту}
  flag : char;
  num ; real;
  i, j, n : integer;
begin
  repeat
    clrscr; num := 0;
    for i := 1 to 4 do cap[i] := 0;
    for i := 1 to 4 do
      begin
        writeln(‘Введіть дані про об’єм збуту’, i, ’-го агента’);
        readln(cap[i]);
        num ;= num + cap[i]
      end;
  for i := 1 to 4 do cap[i] := cap[i]/num*100;
clrscr;
writeln(‘Об’єм збуту агентів в процентах від загального об’єму’);
writeln(‘загальний об’єм збуту’, num:12:1,’грн’);
for i := 1 to 4 do
   begin
     writeln(‘Агент ’, i, ’   ’,cap[i]:5:2,’%’);
     n := trunc(cap[i]*0.79);
     for j := 1 to n do write(graph[i]);
     writeln;
     writeln;
   end;
write(‘Для закінчення розрахунків натисніть n’);
flag := upcase(readkey);
until (flag = ‘N’);
end.
```

Всі дії в програмі виконуються в циклі, організованому за допомогою оператора циклу з післяумовою repeat … until. Цикл виконується до тих пір, поки змінна символьного типу flag стане рівною N. Для введення значення змінної flag використовується функція **readkey**, яка зчитує один символ з буфера. Введений символ на екрані не висвічується. Функція readkey знаходиться в модулі crt, тому в описову частину програми включена uses – бібліотека. Функція upcase переводить рядкові символи в прописні.

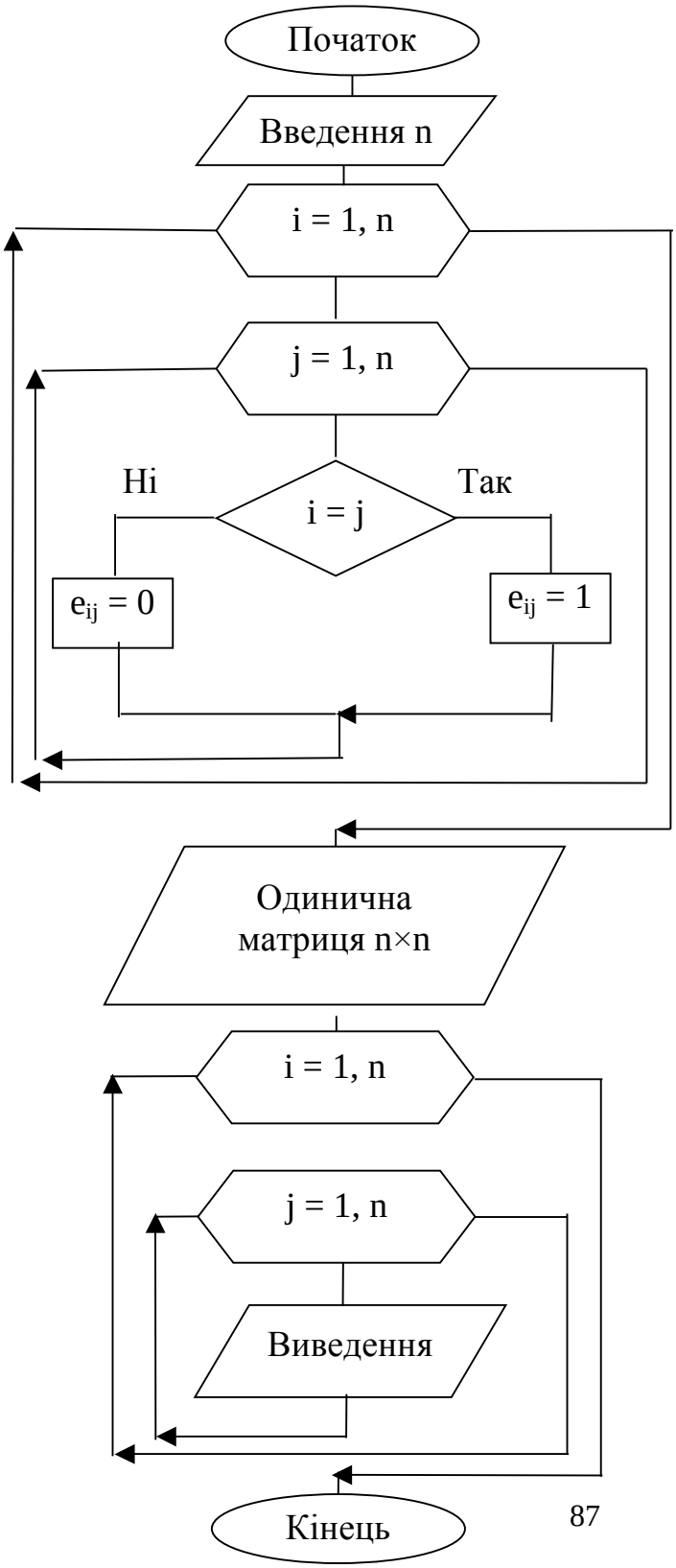


Рисунок 58 – Схема задачі 22

Оскільки розрахунки можуть повторюватися багато разів, то необхідно на початку кожного підрахунку проводити ініціалізацію змінних

num := 0; for i := 1 to 4 do
cap[i] := 0;

Нагадаємо, що оператор write при виконанні програми залишає маркер на рядку введення. Оператор writeln переводить маркер на наступний рядок.

В цій програмі оголошується типізована константа graph типу array of char, яка представляє собою масив символів псевдографіки для наглядного виведення діаграм.

Екран виведення приведено на рис.57.

Об'єм збуту агентів в процентах від загального об’єму
Загальний об’єм збуту
Агент 1 23.00%

Агент 2 15.00%

Агент 3 35.00%

Агент 4 27.00%

Для закінчення розрахунків натисніть n

Рисунок 57 – Результат виведення задачі

6.2 Робота з двовимірними масивами

Задача 22. Сформувати одиничну матрицю E розміром n×n. Одиничною називається квадратна матриця, в якій на головній діагоналі одиниці, а усі інші елементи дорівнюють нулю. $e_{ij} = 1$, якщо i = j, інакше $e_{ij} = 0$. Алгоритм цієї задачі (рис.58) включає в себе: введення матриці розміру n; форматування матриці; виведення результату розрахунку на екран монітора.

Програма:

```
program matr1;
     uses crt;
     const size = 10;
  var
      i,j,n : integer;
   e : array [1..size, 1..size] of integer;
begin
    write ('Введіть розмір матриці');
    readln(n);
    for i := 1 to n do
       for j := 1 to n do
           if i = j then e[i,j] := 0;
    clrscr;
    gotoxy(3,3);
   write ('Одинична матриця розміром', n,'на', n);
   for i := j to n do
      for j := 1 to n do
          begin gotoxy(3*i, 5+j);
                write(e[i,j]);
          end
end.
```

Індекси i та j означають номер рядка та стовпця, на перехресті яких знаходиться кожний елемент матриці, і крім цього позначають положення курсора на екрані в процедурі gotoxy(3*i, 5+j).

Задача 23. Написати програму додавання двох матриць.

Дві матриці можна додати, якщо вони мають однаковий розмір. В результаті додавання вийде матриця такого ж розміру, кожний елемент якої дорівнює сумі відповідних елементів початкових матриць:

$$c_{ij} = a_{ij} + b_{ij}.$$

Алгоритм рішення задачі включає такі етапи: введення вхідної матриці a; введення вхідної матриці b; розрахунок елементів матриці c і виведення результату на екран монітора.

Програма:

```
program clgmatr;
uses crt;
const line = 20; column = 30;
type
  t = array[1..column] of real;
var
  i,j,m,n : integer;
  a,b,c : array [1..line] of t;
begin
  clrscr;
  write('Число рядків матриці (менше 21)'); readln(n);
  write('Число стовпців матриці (менше 31)'); readln(m);
  clrscr; {Введення матриці a}
  for i:=1 to n do
     for j :=1 to m do
       begin
          write('a[', i, ',' , j, ']');   readln(a[i, j])
        end;
clrscr; {Введення матриці b}
for i := 1 to n do
   for j:=1 to m do
     begin
       write('b[', i, ',' , j, ']');  readn(b[i, j]);
     end;
   clrscr; {Розрахунок і виведення матриці c}
   for i := 1 to n do
      begin
         for j := 1 to m do
             begin
                  c[i, j] := a[i, j] + b[i, j]; write(c[i, j]:6:2);
             end;
         writeln
     end;
    repeat until keypressed
end.
```

В програмі визначений новий тип даних – масив, який складається з 30-ти елементів дійсного типу.

Для того, щоб кожен рядок вводився на екрані з нового рядка, в цикл по і включений пустий оператор виведення writeln.

Функція keypressed з модуля crt дає значення true, якщо в буфері input не залишилось неврахованих символів, і false в іншому випадку. Цикл repeat unil keypressed включений в програму для затримки зображення на екрані виводу. Цикл repeat… until буде робити доки не буде натиснута будь-яка клавіша.

7 РЯДКОВІ ТИПИ

Дані рядкового типу – це послідовність символів змінної довжини. Такий тип ще називають *типом* ***string.*** Він багато в чому схожий на одновимірний масив символів, однак, на відміну від останнього, кількість символів у рядку-змінній може змінюватися від 0 до N, де N - максимальна кількість символів у рядку.

Опис рядкового типу *складається з ключового слова* string*, після якого в квадратних дужках зазначена максимальна кількість символів рядка даного типу.* Ця кількість може виражатися за допомогою *цілої константи або імені цілої константи.* Якщо максимальний розмір рядка не зазначений, то він автоматично приймається рівним 255 – максимально можлива довжина рядка. (Існують ще ASCIIZ-рядка, довжина яких може досягати 65536 символів, але для роботи з такими рядками потрібна особлива директива компіляторові). Довжина змінної такого типу може динамічно змінюватися між одиницею і значенням константи. Символи в рядку варто сприймати як пронумеровані в інтервалі від одиниці до значення константи.

Приклад:

```
type
  cities = string [20];
  names = string [12].
```

Змінна типу cities – рядкового типу, тобто рядка символів, що складаються максимально з 20 символів. Змінні типу names максимально можуть складатися з 12 символів. Змінні можна оголосити в такий спосіб:

```
var
  ci: array [1..20] of char;
  na: array [1..12] of char.
```

У цьому випадку говорять, що ci і na – символьні вектори. **Символьні вектори** можна розглядати як рядкові змінні, що представляють послідовності постійної довжини. Завдяки такій інтерпретації імена символьних векторів і імена їхніх елементів можуть використовуватися в рядкових виразах там, де можуть використовуватися імена рядкових змінних.

Кожна величина рядкового типу займає в пам'яті на один байт більше ніж максимальна кількість символів у рядку даного типу. У додатковому байті зберігається інформація про поточну довжину рядка. Якщо n – поточна довжина рядка, то в компоненті 0 міститься номер n останнього символа відносно початку рядка.

Тип даних string призначений для обробки рядків. З даними типу string зв'язаний цілий набір операцій. Рядки можуть виводитися на екран монітора за допомогою стандартних процедур Write і Writeln і вводитися за допомогою стандартних процедур Read і Readln. Наступна програма демонструє зчитування рядка.

```
program string_test;
 var s: string[10];
 begin
   writeln('Задайте кілька символів');
   readln (s);
   writeln(s,'     довжина:' , ord(s[0]):5);
end.
```

У цьому прикладі при введенні можна записати скільки завгодно символів, але за допомогою readln(s) можна вважати максимум 10 символів, тому що за описом змінна s може містити максимум 10 символів. Інші символи ігноруються. Символів, що зчитуються, може бути і менше 10.

Операції над рядками

Рядки можна присвоювати, з'єднувати і порівнювати.

Присвоювання послідовності символів рядковим змінним здійснюється за допомогою оператора присвоювання. З правої сторони оператора присвоювання може знаходитися довільний рядковий вираз, а з лівої - ім'я рядкової змінної.

У Турбо Паскалі *існують два шляхи обробки змінних типу* string. Перший шлях допускає обробку всього рядка як єдиного цілого, тобто єдиного об'єкта. Другий шлях розглядає рядок як складений об'єкт, що складається з окремих символів, тобто елементів типу char, що при обробці доступні кожний окремо. Так, перший шлях надає можливість присвоєння рядкової змінної за одну операцію значення цілого рядка символів:

```
str_1 := ”Це рядок!'.
```

Значення рядка, що присвоюється, так само як і окремий символ типу char, заключається в апострофи. Якщо апострофи опущені, то компілятор розглядає приведений фрагмент тексту як числову величину або як ідентифікатор.

Другий підхід забезпечує доступ до окремих символів рядка за номером їхньої позиції:

```
str_1 [1]   = 'Э'; str_1 [2] := 'т';  str_[3] := 'о';  str_1 [4] := ‘ ';
str_ [5]   = 'с';    str_ [6] := 'т';    str_ [7] := 'р';  str_ [7] := 'о';
str_ [8]   = 'к';    str_ [9] := 'а';    str_ [10] := '!'
```

Змінна str_1 приймає те ж саме значення, що і при першому підході. Для доступу до окремого символу в рядку необхідно вказати ім'я рядка й у квадратних дужках номер позиції елемента (символу) у рядку. При цьому стосовно окремого символа рядка можливі всі ті ж операції, що і до змінної типу char. Необхідно нагадати, що, крім класичного запису символьної змінної в одинарних лапках, Турбо Паскаль вводить ще дві форми. Одна з них – представлення символа його кодом ASCII за допомогою спеціального префікса #, друга, для керуючих символів, – значок ^ і буква алфавіту з тим же номером (див. главу 1). При

необхідності включення в рядок керуючих кодів можна користуватися"клавіатурними" позначеннями. У цьому випадку значення складається ніби з склеєних кусків:

^G ' після сигналу натисніть '^J' клавішу пропуску '

У такому записі не повинно бути пропусків поза лапками.

Якщо довжина рядка праворуч більше максимального розміру змінної, то присвоювання відбувається зліва направо, а надлишкові символи опускаються.

Приклад.

```
type     name = string [7]
var      fname, Iname : name;
begin
     fname := 'Наталія';   Iname := 'Кучинська';
     write (fname, ‘ ‘, Iname)
end.
```

В результаті виконання програми буде виведено: Наталія Кучинсь, тому що максимальна довжина рядка 7. Відповідно до опису

var a: string [5]; b: string [10];

можна присвоїти значення b := a, хоча типи змінних різні. При присвоєнні a := b, в "a" запишуться тільки перші 5 символів рядка "b".

Об'єднання двох рядків в одну може бути здійснене за допомогою **оператора конкатенації.** Цей оператор записується за допомогою знака + (плюс). Об'єднаний рядок являє собою рядок, що складається з усіх символів першого рядка і наступних за ними всіх символів другого рядка. Наприклад:

Вираз	Результат
'adam' + 'eva'	'adameva'
'5' + '.' + '4'	'5.4'
'Це' + ' ' + 'рядок' + '!'	'Це рядок!'

Довільні пари рядків можуть порівнюватися за допомогою операторів відносин. Два рядки порівнюються посимвольно зліва направо. При виявленні першого незбіжного символа приймається рішення відповідно до таблиці кодів ASCII. Відношення 'balkon' > 'balken' дає результат true, тому що символ 'o' стоїть в таблиці кодів після символа 'e'. Відношення 'Pascal' = 'pascal' дає результат false, тому що в малої букви код більше ніж у великої. Символ '+' стоїть в таблиці кодів перед символом ' -', тому порівняння '+' < '-' дає в результаті true.

Якщо два порівнювані рядки мають різну довжину, але збігаються аж до останнього символа більш короткого рядка, останній рядок вважається меншим. Два рядки вважаються однаковими тільки тоді, коли вони мають однакову довжину і всі їхні символи збігаються.

Стандартні процедури і функції для рядків

Турбо Паскаль представляє в розпорядження користувача цілий ряд функцій і процедур, призначених для обробки рядків.

Функція Length

Вбудована функція **Length** (довжина) дозволяє визначити фактичну довжину текстового рядка, що зберігається в зазначеній змінній.

Приклад.

```
var
  word: string;
begin
      write (' Введіть, будь ласка, слово: ');
      readln (word);
      writeln ('Це слово складається з ', length(word) : 3, ' букв!')
end.
```

При підрахунку фактичної довжини рядка враховуються усі вхідні в нього символи, у тому числі і пропуски.

Функція Upcase

Функція **Upcase** дозволяє перетворювати символ будь-якої літери з рядкового в великий. Ця функція розрахована на обробку окремого символа, тому для обробки рядка символів за допомогою цієї функції приходиться організовувати цикл.

Приклад.

```
var
   word : string;
        i : byte;
begin
      word := 'Фірма Microsoft';
      for i := 1 to length (word) do word [i] := upcase (word[i]);
      writeln (word);     {Виводиться текст " Фірма MICROSOFT"}
end.
```

Російські літери не можуть оброблятися цією функцією, тому в результаті роботи цієї програми на термінал видається рядок, що містить великі англійські і маленькі російські букви.

Функція Copy

Функція **Copy** дозволяє копіювати фрагмент деякого рядка з однієї перемінної в іншу. Викликаючи функцію Copy, необхідно вказувати такі параметри:

- ім'я рядка, з якого повинен витягатися копіровний фрагмент;
- позицію в рядку, починаючи з якої буде копіюватися фрагмент;
- число копіровних символів.

Приклад

```
Var
        ws: string [79];
        w1,w2,w3 : string [20];
 begin
      ws   = 'фотографирование';
      w1   = copy (ws, 1, 4); writeln (w1);
      w2  =  copy (ws, 5, 4); writeln (w2);
      w3  = copy (ws, 10, 3); writeln (w3);
end.
```

У результаті виконання програми на екран виводиться:

```
фото
граф
ров
```

Повідомлення про помилку не буде у випадках, якщо початкова або кінцева позиції копіровного фрагмента знаходяться поза межами вихідного рядка символів. Результатом виконання операції в першому випадку буде рядок нульової довжини, у другому – фрагмент від початкової позиції копіювання до кінця вихідного рядка.

Функція Pos

За допомогою функції **Pos** здійснюється пошук визначеного фрагмента в рядку. Якщо заданий фрагмент у рядку присутній, то функція повертає номер позиції в рядку, з якого фрагмент починається. Якщо в рядку фрагмент не знайдений, то функція повертає нуль.

```
var
    ws:   string [79];
    sw:   string [20];
     p : byte;
begin
      ws   = 'Электрификация';
      sw = 'Эл';  p := Pos (sw, ws);  writeln(p);
      sw = 'три'; p := Pos (sw, ws); writeln(p);
      sw  = 'к';    p := Pos (sw, ws); writeln(p);
 end.
```

У результаті виконання програми на екран виводиться:

```
1
5
4
```

Функція Pos вимагає повного збігу шуканого фрагмента і фрагмента рядка, у якому виконується пошук. Великі і маленькі букви вважаються різними символами.

Процедури Insert і Delete

Процедура **Insert** вставляє у вихідний рядок який-небудь інший рядок, починаючи з зазначеної позиції. Оператор Insert (w1, ws, 4)**;** вставляє рядок w1 у рядок ws перед 4-ою позицією.

Процедура **Delete** видаляє у вихідному рядку фрагмент визначеної довжини, що починається з зазначеної позиції. Оператор Delete (ws, 2, 3); видаляє з рядка ws фрагмент, що складається з трьох символів і починається з другої позиції.

```
var
   ws : string [79];
   sw : string [20];
begin
      ws := 'комп‘ютеризація';  writeln (ws);
      delete (ws, 1, 7);  writeln (ws);
      delete (ws, 3, 2);  writeln (ws);
       sw := 'Г ';  insert(sw, ws, 1); writeln (ws);
       sw := 'не'; insert(sw, ws, 3); writeln (ws);
end.
```

У результаті роботи даної програми на екран будуть виведені такі рядки:

```
комп‘ютеризація
еризація
ерація
Герація
Генерація
```

Щоб уникнути помилок при застосуванні процедур Insert і Delete зазначені в списку параметрів позиції повинні обов'язково бути присутніми в оброблюваних рядках.

Процедура Str

Процедура **Str** (x [:width [idecimals]], st) перетворить число x будь-якого дійсного або цілого типів у рядок символів st. Параметри width і decimals, якщо вони присутні, задають формат перетворення. Width визначає загальну ширину поля, виділеного під відповідне символьне представлення дійсного або цілого числа x, a decimals – кількість символів у дробовій частині (має сенс тільки в тому випадку, коли x – дійсне число).

```
var
   y : integer; x : real; st : string [8];
begin
    y := 45; str(y, st); writeln (st);
    y := 45; str(y:3, st); writeln (st);
    y := -2; str(y:5, st); writeln (st);
    x := 45.678;
```

```
    str(x:7, st); writeln (st);
    str(x:10, st); writeln (st);
    str(x:6:2, st); writeln (st);
    str(x:8:3, st); writeln (st);
end.
```

Виведення на екран дисплея:

```
 45
_45
_ _ _-2
_4.6E+01
_4.568E+
_45.68
_ _45.678
```

Процедура Val

Процедура val (st, x, code) перетворить рядок символів st у внутрішнє представлення цілої або дійсної змінної x, що визначається типом цієї змінної. Параметр code містить нуль, якщо перетворення пройшло успішно, і тоді в x міститься результат перетворення, у протилежному випадку він містить номер позиції в рядку st, де виявлений помилковий символ, і в цьому випадку вміст x не змінюється. Пропуски в рядку st можуть бути присутніми лише на початку.

```
var
   x : real; y : integer; st : string;
begin
    st := '45.678';  val(st,x,y);  writeln (x:6:3, '    ', y);
    x := 0;
    st  :=  '3,1415';  val(st,x,y);    writeln  (x:6:3,  '        ',  y);        {кома –
неприпустимий
                                                                  символ}
    st :- = '   45.678';  val(st,x,y);  writeln (x:6:3,'     ', y);
    st := '45.6   ';     val(st,x,y);      writeln (x:6:3, '     ', y);
end.
```

Результат виконання програми:

```
45.678   0
  0.000   2
45.678   0
  0.000   5
```

Практичні приклади роботи з рядковими даними

Задача 24. Визначити скільки разів у тексті зустрічається задана буква. За допомогою функції Copy копіюємо в змінну a по одному послідовно всі символи з тексту t. У випадку a = x, де x вихідна буква, необхідно включити лічильник. Довжину тексту визначимо за допомогою функції length.

Програма:

```
program bukva;
var
   t, a : string;
   x : char; i,k : integer;
begin
        write('Введіть вихідний текст');
        readln(t);
        writeln('Введіть букву');
        readln(x);
        k := 0;
       for i := 1 to length (t) do
       begin
            a := copy (t, i, 1);
            if a = x then k := k + 1;
      end;
   write ('Буква ', x, ' зустрічається   ', k,      ' разів.')
end.
```

У цій задачі функцію Copy використовувати не обов'язково. Можливе і таке рішення:

```
for i := 1 to length (t) do
  if t(i) a x then k := k + 1;
```

Задача 25. Підрахувати скільки слів в тексті починається на задану букву. Слова в тексті розділені пропусками.

Перша буква кожного слова в тексті йде після пропуску. Визначивши позицію пропуску і виділивши фрагмент тексту від пропуска і до кінця, визначимо чи перша буква фрагмента задана за умовою.

Програма

```
program bukslova;
var
   t, a : string;
   x : char;
   i, k : integer;
begin
      write (' Введіть вихідний текст ');
      readln (t);
      writeln (' Введіть букву ');
      readln(x);
      i := 1; k := 0;
      while i <> 0 do
      begin
          a := copy (t, 1, 1);
```

```
            if a = x then k := k + 1;
            i := pos(' ',t);
            t := copy (t, i + 1, length(t) - i);
        end;
        write ('Кількість слів на букву ', x, ' = ', k)
    end.
```

У змінну a копіюємо першу букву фрагмента тексту; i – номер позиції пропуску; t – вихідний текст і поточний фрагмент тексту від пропуску і до кінця. Цикл продовжується доти, поки поточний фрагмент t містить пропуски.

Задача 26. У заданому тексті здійснити заміну одного слова на інше.

Програма:

```
program slovo;
var
  t,a,b,c,s : string;
          i : integer;
begin
    write(' Введіть початковий текст ');
    readln(t);
    writeln(' Введіть слово, яке потрібно замінити ');
    readln(a);
    writeln(' Введіть слово, для заміни ');
    readln(b);
    t := t + ' ';
    writeln ('Початковий текст ');
    writeln (t); i := 1; s := ‘ ‘;
    while i <> 0 do
    begin
        i := pos(' ', t);
        c := copy (t, 1, i - 1);
        if c = a then s := s + b + ' '   else s := s + c + ' ';
        t ;= copy (t, i + 1, length(t) - i)
    end;
    writeln ('Новий текст ');    writeln (s)
end.
```

Виділяємо з тексту кожне слово за допомогою функції Copy. Виділене слово "c" порівнюємо з заданим "a": якщо вони рівні, то в новий текст потрібно записати слово для заміни "b", інакше в новий текст записуємо старе слово "c". Текст повинен закінчуватися пропуском, інакше не буде можливості виділити останнє слово. Про всякий випадок запишемо в кінець вихідного тексту пропуск (t := t + ' ';).

Задача 27. Скласти програму, що за заданою датою відображає на екрані назву відповідного знака Зодіаку.
Астрологи поділяють рік на 12 періодів і кожному з них ставлять у відповідність знак Зодіаку. Граничні дати періодів задамо типізованою константою у вигляді одновимірного масиву дійсних чисел, а назва кожного періоду типізованою константою – одновимірним масивом рядків. При введенні дати необхідно передбачити правильне введення місяця і числа. Номер місяця повинен знаходитися в інтервалі від 1 до 12 , а число в інтервалі від 1 до 31. Щоб не ускладнювати задачу, більш точну перевірку опустимо.

Програма:

```
program zodiak;
   uses crt;
  label 1;
const
  dt : array[1..12] of real =
        (01.19,2.18,3.20,4.19,5.20,6.21,7.22,8.22,9.22,10.22,11.22,12 _21);
      zod: array[1..12] of string[10] = ('Водолій','Риби','Овен',
                                  Тілець', 'Близнюки', 'Рак', 'Лев', 'Діва ,
                                'Терези', 'Скорпіон', 'Стрілець',  'Козеріг');
var
  s:string[10];
  x, i:integer;
  r, y:real;
 sem:char;
begin
   repeat
    1: clrscr;
       write('Введіть дату народження, наприклад: 01.22 або 12.28'   );
       readln(r);
       {Перевірка правильності введення місяця народження}
       x := trunc(r);
       if (x < 1) or (x > 12) then
        begin
             write('Місяць, введений неправильно');
             delay(1000); goto 1
        end;
      {Перевірка правильності введення числа народження}
      y := frac(r);
      if (y < 0.01) or (y > 0.31) then
      begin
        write(' число введено неправильно ');
        delay(1000); goto 1
```

```
    end;
   {Визначення знака Зодіаку}
   s := zod[12];
  for i := 1 to 11 do
  if(r > dt[i]) and (r <= dt[i+1]) then
                        s := zod[i];
 {Виведення результатів}
 writeln('Ви народилися ', r:5:2);
 writeln ('під знаком Зодіаку ',s);
 writeln ('якщо  хочете перервати роботу, натисніть Y');
 sem := readkey;
 until upcase(sem) = 'Y'
end.
```

Всі основні дії в програмі виконуються в циклі (repeat until) доти, поки не будуть введені символи Y або y. При введені малої букви y функція Upcase перетворить її в велику. Процедура **delay**(t) модуля crt припиняє виконання програми на t мс.

8 ЗАПИСНІ ТИПИ

У багатьох економічних і інформаційних задачах обробляються відомості, документи, каталоги, списки. При цьому з'являється необхідність поєднувати дані різного типу в одну групу. Для роботи з групою даних у мові Паскаль уведене поняття запису.

Запис – це структура даних, що складається з фіксованого числа компонентів різного типу. Складові компоненти запису називаються **полями запису.** На відміну від масиву компоненти (поля) запису можуть бути різного типу. Щоб можна було посилатися на той або інший компонент запису, поля іменуються. Записний тип ще називають *комбінованим типом.*

Записний тип даних надає програмістові можливість об'єднати в одну зв'язану структуру різні за типом і змістом елементи. Елементами запису можуть бути і структуровані типи даних, наприклад масиви й інші підпорядковані записи. Для обробки доступний як весь запис, так і окремі її поля.

Поняття запису розглянемо на прикладі відомості списку учнів з їх оцінками.

№	Прізвище И. О	Оцінки		
1	Іванов И. И.	5	3	4
2	Андрєєва А. А.	5	5	3
3	Керенський А. Ф.	5	4	5
4	Ульянов В. И.	5	5	5

Кожен рядок у цій відомості складається з окремих елементів даних різного типу:

– порядковий номер – ціле десяткове число;
– прізвище та ініціали – масив символів;
– оцінки – масив цілих чисел.

Ці дані можна об'єднати в одну групу і вважати записом. Уведемо такі позначення: B – ім'я всього запису; n – порядковий номер; fio – прізвище, ім'я, по батькові; mark – оцінки.

Звертання до елементу запису в програмі виконується за допомогою уточненого (складеного) імені. **Уточнене ім'я** містить ім'я запису й ім'я елементу і записується в такому вигляді:

ім'я запису **.** ім'я елементу

Наприклад,

B **.** n
B **.** fio
B **.** mark

Записи, як і інші дані, з'являються в розділі описів і використовуються в розділі операторів.

Структура оголошення типу запису така:

<ім'я типу> = record <список полів> end

Тут <ім'я типу> – правильний ідентифікатор; **record, end** – зарезервовані слова (запис, кінець); <список полів> – послідовність розділів запису, між якими ставиться крапка з комою. Кожен розділ запису складається з одного або декількох ідентифікаторів полів, відокремлених один від одного комами. За ідентифікатором (ідентифікаторами) ставиться двокрапка й опис типу поля (полів).

Для представленої відомості оголошення запису має вигляд:

```
type
   list = record
      n : integer;
    fio : array [1..20] of char;
   mark : array [1..3] of integer
  end;
 var B : list;
```

Можливе оголошення запису в розділі змінних:

```
var   B : record
        n : integer;
      fio : array [1..20] of char;
     mark : array [1..3] of integer
     end;
```

Елемент запису використовується в програмі в тому самому значенні, що і звичайна змінна. Елемент запису можна вказувати як у лівій частині оператора присвоювання, так і у виразах. Над елементом запису можна

виконувати дії, допустимі для даних його типу. Якщо тип елементу запису – цілий, то виконуються всі операції, допустимі для цілих даних. Для розглянутої відомості над елементами запису можна зробити, наприклад, такі операції:
порядковому номерові n запису B присвоїти значення 2:

```
B . n := 2;
```

знайти суму трьох оцінок:

```
s := B . mark [1] + B . mark [2] + B . mark [3];
```

ввести значення порядкового номера:

```
read (B . n).
```

Звертання до запису в цілому, а не тільки до її елементів, допускається лише в операторі присвоювання. Ліворуч і праворуч від знака присвоювання при цьому повинні використовуватися імена записів однакового типу.

Оголосимо тип birthday, що містить три поля з іменами day, month і year; змінні a і b містять запис типу birthday; c – запис, що містить в якості елементу підпорядкований запис.

```
type
    birthday = record
                day, month : byte;
                     year  : word
end;
var a, b:birthday;
      c:record
         name:string;
         bd: birthday
      end;
```

В цьому випадку можливо a:=b; a.day:=27; b.year:=1939; для вкладених полів необхідно продовжувати уточнення: c.bd.year, c.bd.day, c.bd.month.

Оператор присвоювання With ... do

Звертання до конкретних полів запису пов’язане з деякими незручностями. Складові ідентифікатори виходять довгими. При сумісній обробці ряду полів одного запису довжину ідентифікаторів можна скоротити.

Щоб спростить доступ до полів запису, використовується **оператор присвоювання With :**

with<змінна>do<оператор>

Тут with, do – ключові слова(с, робити); < змінна > – ім’я змінної типу запис, за яким слідує список вставлених полів; <оператор> будь-який оператор Турбо Паскаль.

Оператор присвоювання With ... do відкриває запис. В полі його дії імена запису спрощуються. Наприклад:

```
with c.bd do month := 9;
```

Це еквівалентно:

```
with c do with bd do month : = 9;
```

або

```
c.bd.month := 9.
```

У такому записі не почувається перевага використання оператора with. Коли обробляються всі поля запису, оператор with ... do дає істотну економію в розмірі програми.

Задача 28. Скласти програму формування архіву даних про видані книги. Дані включають: автора, назву, рік видання, місто, де видане, і ціну книги.

Припустимо, що в архів необхідно занести дані про 50 книг. Дані будуть являти собою масив записів. Кожен запис складається з:

а) прізвища та ініціалів автора (author);
б) назви книги (title);
в) назви міста, у якому книга видана (city);
г) року видання (year);
д) ціни (cost).

Дані author, title, city – символьного типу. Year – може бути змінна інтервального типу. Ціна (cost) повинна бути змінною дійсного типу. Оголошення нового типу даних entry як записного буде виглядати в такий спосіб:

```
entry = record
          author, title, city : string;
          year : 1..9999;
          cost : real;
        end;
```

Далі варто оголосити масив з 50 елементів, що мають тип entry. Введення елементів запису організуємо в циклі, причому закінчення циклу зафіксуємо допоміжною змінною ch. Перед введенням вмісту чергового запису зробимо запит на продовження поповнення архіву. Для введення компонентів запису використовуємо оператор readln. Компоненти запису в програмі мають складові імена: m[i].author, m[i].title, m[i].city, m[i].year, m[i].cost.

Програма:

```
program archives;
uses crt;
 label 10;
 type
 entry = record {Вихідні дані про книги}
          author, title, city : string;
```

```
            year : 1..9999;
            cost : real;
        end;
  var
     m : array[1..50] of entry;
    {Архів не перевищує 50 книг}
    {Масив, елементами якого є записи}
    i : integer;
   ch : char;
  begin
    i := 0;
   {Блок формування архіву даних про книги}
   writeln ('Поповнюєш архів? - Так - [Y]');
  ch := readkey;
  if upcase(ch) = ' Y ' then
  repeat
      i:=i+1;
     writeln ('Автор книги ');  readln(m[i].author);
     writeln ('Назва книги ');  readln(m[i].title);
     writeln ('Місто  видання ');   readln(m[i].city);
     writeln (' Рік видання ');        readln(m[i].year);
     writeln ('Ціна книги ');          readln(m[i].cost);
     writeln ('Поповнюєш архів? - Так - [Y]');
     ch := readkey
  until upcase(ch) <> 'Y'
 else
   begin
        write ('Ви передумали? До побачення!');
        goto 10
   end;
 repeat until keypressed;
10:end.
```

Задача 29. Ускладнимо попередню задачу. Скласти програму добірки книг за зазначеним автором із сформованого списку масиву записів.

Для формування архіву використовуємо попередню програму 28. Уведемо змінну символьного типу ss – прізвище автора, список книг якого необхідно вивести. Розмір архіву запам'ятаємо в змінній k. Для більш компактного представлення використовуваних змінних, скористаємося оператором приєднання with ... do. При цьому імена полів запису можуть фігурувати як імена звичайних змінних. Для можливості виведення декількох списків організуємо зовнішній цикл з ознакою закінчення ch = 'Y'.

Програма:

```
program archives2;
uses crt;
   label 10;
  type
       entry = record {Вихідні дані про книги}
                author, title, city : string;
                year : 1..9999;
               cost : real;
           end;
var
  m : array[1..50] of entry;
 {Архів не перевищує 50 книг}
{Масив, елементами якого є записи}
 i, j, k : integer,
 ss : string; {Прізвище для пошуку}
 ch : char;
begin
      i := 0;
    {Блок формування архіву даних про книги}
    writeln ('Поповнюєш архів?    Так -  [Y] ' );
    ch := readkey;
   if upcase(ch) = 'Y'  then
   repeat
      i := i+1;
     with m[i] do
     begin
          writeln('Автор книги '); readln(author);
          writeln('Назва книги'); readln(title);
          writeln('Місто видання '); readln(city);
          writeln('Рік видання '); readln(year);
          writeln('Ціна книги '); readln(cost)
    end;
    writeln ('Поповнюєш архів? -Так – [Y] )
         ch := readkey
  until upcase(ch) <> 'Y'
 else
    begin
         write ('Ви передумали? До побачення!");      goto 10
   end;
 k := i;
{Блок добірки всіх книг  зазначеного автора}
 repeat
      writeln('Вкажіть прізвище автора'); readln(ss);
```

```
        writeln( 'Список книг, автор яких ‘, ss:15)
        for i := 1 to k do
       with m[i] do
       begin
            if ss = author then
              begin
                    writeln(title);
                    writeln(‘Місто  ', city);
                    writeln (‘Рік видання ', year);
                    writeln ('Ціна ', cost:8:2);
                    writeln;
             end;
     end;
    writeln('Потрібно ще видавати авторський список? - Так -[Y]');
    ch := readkey;
  until upcase(ch) <> 'Y';
  repeat until keypressed;
  10:end.
```

Варіантні записи

Склад даних залежить від виду об'єкта, на який складений запис. Наприклад, якщо вихідні дані книги містять її назву, рік і місце видання, назву видавництва, то для журнальної статті важливо знати назву журналу, номер і рік випуску. Виникає необхідність внесення в структуру запису деякої варіантної частини. За допомогою записів з варіантами можна одночасно зберігати різні структури даних, що мають загальну частину, однакову у всіх структурах, і невеликі частини, що відрізняються, у різних структурах.

Варіантний записний тип – це записний тип, що містить кілька варіантів структури запису. Розходження може стосуватися як числа компонентів, так і їхніх типів.

Припустимо, що є, наприклад, такий опис:

type status = (married, widowed, divorced, single),

що означає одружений, вдівець, розведений, самотній. У цьому випадку людину можна описати за допомогою даних такого типу:

```
type person = record
        {тут поля загальні для всіх person}
              case status of
                   married: ( { поля тільки для сімейних });
                   single:  ({ поля тільки для самотніх });
                   ...
              end;
```

Кожному значенню, що відноситься до типу, на підставі якого йде розрізнення варіантів (типові ознаки), повинен відповідати один з

варіантів. Це означає, що перед варіантами повинні з'являтися всі значення типу ознаки. У нашому прикладі, крім констант married і single, повинні з'явитися і константи widowed і divorced.

Звичайно деякий компонент (поле) запису сам вказує, про який варіант мова йде. У приведеному описі типу варто було б мати загальне для усіх варіантів поле ms : status. *Опис, який визначає варіант компоненти* можна включити в заголовок варіанта. Такий опис називається **полем ознаки.** Наприклад:

case ms : status of

Перш ніж почати визначати структуру запису з варіантами, що відповідає, наприклад, типові person, корисно буває виписати всю необхідну інформацію.

1. Ім'я(name) – перше,останнє(first, last).
2. Ріст (height) – ціле число.
3. Стать (sex) – чоловік, жінка (male, female).
4. Дата народження (date) – рік, місяць, день (year, month, day).
5. Число утриманців (depds) – ціле число.
6. Родиний стан (status):

   Якщо в шлюбі (married) або вдівець (widowed):

   а) дата весілля (mdate) – рік, місяць, день (year, month, day).

   Якщо розведений (divorced):

   а) дата розлучення (ddate) – рік, місяць, день (year, month, day),

   б) перше розлучення (firstd) – ні, так (false, true).

   Якщо самотній (single): інформації немає.

Визначення записного типу person можна сформулювати так:

```
type
  status = (married, widowed, divorced, single);
 date = record
    year : 1900..2100;
    mo : (jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec);
    day : 1..31;
   end;
 natural = 0..max;
 person = record
    name : record first, last : string[15] end;
    height: natural;
    sex : (male, female);
    birth : date;
    depdts : natural
    case rns : status of
    married, widowed : (mdate : date);
    divorced : (ddate : date; firstd : boolean);
```

single : ( )
end {person};

Задача 30. Скласти програму формування переліку відомостей за даними архіву книг і журналів.

Дана задача відрізняється від задачі 29 тим, що склад даних у записі розрізняється для книги і для журналу. Введемо перелічуваний тип даних entrytype, що визначає варіанти, type entrytype = (book, magazine).

Дані з кожного найменування архіву включають:

а) автора книги або статті (author) – змінна типу string;

б) найменування книги або статті (title) – string;

в) рік видання книги (журналу) – інтервальний тип 1 ... 1999;

г) якщо книга (book):

1) видавництво (publisher) – string;

2) місто (city) – string.

Якщо журнал (magazine):

а) найменування журналу (magname) – string;

б) номер журналу (vol) – integer;

в) кількість сторінок (page) – integer.

Програма:

```
program archives;
uses crt; label 10;
type
  entrytype = (book, magazine);
{Вводиться спеціальний тип даних, що визначає варіанти}
 entry - record
        author, title : string;
        year : 1..9999;
        case tag:entrytype of
        {Поле tag визначає галузь варіанта}
        book : (publisher,city : string);
        magazine : (magname : string; vol, page : integer)
          end;
  var
    m : array[1..50] of entry;
    i, j, k : integer;
    ss : string;
    ch : char;
{Блок формування архіву даних про книги}
   writeln ('Поповнюєш архів? - Так - [Y]');
   ch := readkey;
   if upcase(ch) = 'Y'  then
     repeat
       i:=i+1;
```

```
      writeln('Якщо вводите дані  книги, натисніть Y ');
      ch := readkey;
      if upcase(ch) = 'У'  then
                m[i].tag := book
           {Змінній варіанта присвоюється ознака книги - book}
                    else
                      m[i].tag := magazine;
       writeln(‘ Автор '); readln(m[i].author);
       writeln ('Назва '); readln(m[i].title);
       writeln(Рік видання '); readln(m[i].year);
  {Перевірка умови належності об'єкта до книги}
      if mfjj.tag = book then
 {Введення даних в поля, специфічні для книги}
        begin
         writeln('Місто '); readln(m[i].city);
         writeln('Видавництво'); readln(m[i].publisher);
        end
     else
{Введення даних у поля, специфічні для журналу}
        begin
         write ('Назва журналу'); readln(m[i].magname);
         writeln('Номер '); readln(m[i].vol);
         writeln('Сторінка '); readln(m[i].page);
       end;
      writeln ('Поповнюєш архів? - Так - [Y]');
      ch := readkey;
     until upcase(ch) <> 'Y'
  else
    begin   .
      write ('Ви передумали? До побачення!');
      goto 10
    end;
  k := i;
{Блок добірки списку праць зазначеного автора}
 repeat
  writeln('Вкажіть прізвище автора'); readln(ss);
  writeln('Список праць ', ss : 15);
  for i := 1 to k do
    begin
      if ss = m[i].author then
        begin
          writeln(m[i]. title);
          writeln('Рік видання ', m[i].year);
          if m[i].tag = book then
```

```
          begin
            writeln('Місто   ',m[i].city);
            writeln('Видавництво ',m[i].publisher);
            writeln
          end
        else
         begin
          writeln('Назва журналу ',m[i].magname);
          writeln('Номер ', m[i].vol);
          writeln('Сторінка ', m[i].page);
          writeln
         end
       end
    end;
   writeln ('Потрібно ще видавати авторський список? - Так -[Y]');
   ch := readkey;
  until upcase(ch) <> 'Y';
  repeat until keypressed;
10:end.
```

9 МНОЖИННІ ТИПИ

Множина – *це деякий обмежений неупорядкований набір різних елементів однакового типу*. Можна говорити, наприклад, про множину фігур на площині (прямокутник, коло, ромб, квадрат), про множину радіодеталей, транспортних засобів, верстатів і т.п.

В Турбо Паскалі існує множинний тип для введення множин і організації операцій над множинами. **Множинний тип є** деяка сукупність елементів, що є підмножиною допустимих значень визначеного порядкового типу, названий базовим. Множинний тип описує набори однотипних логічно зв'язаних один з одним об'єктів. Характер зв'язків між об'єктами розуміє лише програміст і ніяк не контролюється Турбо Паскалем. Кількість елементів, що входять у множину, може змінюватися в межах від 0 до 256. Множина, що не містить елементів, називається порожньою. *Саме непостійністю кількості своїх елементів множина відрізняється від масивів і записів.*

Множини повинні бути оголошені або в розділі змінних var, або з використанням розділу типів type. Опис типу множина має вигляд:

<ім'я типу> = set of <базовий тип>

Тут <ім'я типу> – правильний ідентифікатор; **set, of** – зарезервовані слова (множина, із); <базовий тип> – базовий тип елементів множини, у якості якого може використовуватися будь-як порядковий тип, крім word, integer, longint.

Приклад:

```
var
    year : set of 198..200;
    c : set of char;
```

Year – множина, що може складатися з елементів діапазону 198 ... 200,

c – множина усіх символів.

Або:

```
type
  digitchar = set of '0'..'9';
  digit = set of 0..9;
  color = (red, blue, green, white, black, orange);
  brush = set of color;
var
  s1, s2, s3 : digitchar;
  s4, s5, s6 : digit;
  a : brush;
```

Тип digitchar описує набір символів '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. Тип digit складається з чисел 1,2,3,4, 5, 6, 7, 8, 9. Тип brush складається з набору шести ідентифікаторів. Змінні s1,s2,s3 мають тип digitchar, тобто можуть

включати тільки перераховані елементи від '0' до '9'. Множини s4, s5, s6 можуть складатися з чисел від 1 до 9 у тому вигляді, у якому вони перераховані в типі digit. Змінна a може містити від одного до шести значень перелічуваного типу color. Нагадаємо, що елементи множини не упорядковані.

Конструктори множин

Множинне значення можна задати за допомогою **конструктора множини,** у якому зберігаються описи елементів множини, відділені один від одного комами і записуються в квадратні дужки.

Описом елементу можуть бути *константи або вирази базового типу, а також – тип-діапазон того ж базового типу* виду min..max, де значення виразів min і max являють собою нижню і верхню границі групи елементів. Якщо нижня границя більше верхньої границі (тобто min > max), то ніякий елемент не описується, тобто описується порожня множина.

Усі вирази повинні належати до одного порядкового типу, що є базовим типом для множинного типу даного конструктора. Конструктор множини [ ] позначає порожню множину для будь-якого множинного типу.

Приклади конструкторів множин:

[13]

[i+j, i-j]

['0' .. '9]

Наступні записи неправильні:

[true, 2] – різні базові типи;

[200 .. 300] – ord(300) > 255;

[4.0, 5.0] – базовий тип не є порядковим.

Операції над множинами

Якщо X – змінна-множина, а E – множинний вираз, то присвоювання:

X := E допустиме тільки в тому випадку, якщо всі елементи E відносяться до базового типу X.

До будь-яких об'єктів зі структурою множини застосовуються такі операції: *об'єднання, перетин, різниця.* Якщо припустити, що A и B – вирази одного типу, то:

A + B – множина з елементів A и B **(об'єднання);**

A * B – множина загальних для A и B елементів **(перетин);**

A – B – множина елементів A, що не входять в B **(різниця).**

До множинних операндів застосовують п'ять операцій відношення. Припустимо, A и B – множинні вирази одного типу, а e – порядковий вираз базового типу.

e in A – входження в множину; результат true, якщо e елемент A інакше – false;

A = B – рівність множин;
A <> B – нерівність множин;
A <= B – включення; результат true, якщо A підмножина B;
A >= B – включення; результат true, якщо B підмножина A.

Дві множини вважаються *еквівалентними* тоді і тільки тоді, коли всі їхні елементи однакові, причому порядок розташування елементів у множині довільний. Якщо всі елементи однієї множини входять також і в іншу, говорять про включення першої множини в другу. Порожня множина включається в будь-яку іншу.

Нехай
s1 = ['1', '2', '3'];
s2 = ['3', '2', '1'];
S3 = ['2', '3'];
s4 = [0..3, 6];
s5 = [4, 5];
s6 = [3..9];

Множини s1 і s2 еквівалентні, множина s3 включена в s2, але не еквівалентна їй.

s4 + s5 містить [0, 1, 2, 3, 4, 5, 6].
s5 + s6 містить [3, 4, 5, 6, 7, 8, 9].
s6 - s5 містить [3, 6, 7, 8, 9].
s4 - s5 містить [0, 1, 2, 3, 6].
3 in s6 повертає true.
2 * 2 in s1 повертає false.

Дії з множинами відносно швидкі, і їх можна використовувати для виключення більш складних перевірок. Наприклад, замість перевірки

if (ch = 'a') or (ch = 'e') or (ch = 'i') or (ch = 'o') or (ch = 'u') then . . .
можна написати більш просту
if ch in ['a', 'e', 'i', 'o', 'u'] then ...
Значення типізованої константи-множини задається у вигляді правильного конструктора множини, наприклад:

```
type
  days = set of  1..31;
  digc = set of  '0' .. '9';
  error = set of  1..24;
const
  workDays : days = [1..5, 8..12, 15..19, 22 ..26, 29,30];
  evendigits : digc = ['0', '2', '4', '6', '8'];
```

```
err : error = [ ];
```

Механізм роботи з множинами Турбо Паскаля відповідає базовим математичним діям з кінцевими множинами. Множини добре працюють там, де потрібно проводити аналіз однотипних вибірок значень або накопичувати значення, які надходять довільно. Значення типу "множина" компактно кодуються: один елемент витрачає 1 біт. Множина з 256 елементів займе всього 32 байти. Недоліком є неможливість виведення множин. Не існує механізму вилучення елементу з множини, але й у математиці така дія не визначена.

Приклади програмування задач з використанням множини

Задача 31. Організувати введення елементів множини і визначити, чи існують загальні елементи у введеної множини з множиною, що задана константою.

Дана програма продемонструє як можна задати множину. Змінні типу "множина" можуть бути типізованими константами.

Змінну v задамо початковими значеннями при описі констант. Змінна s є підмножиною множини символів коду ASCII.

Програма:

```
program wod;
const v : set of char = ['a', 'd', 'f, 'n', ‘r’];
var
  s : set of char;
  c : char;
   begin
  s := [];
  c := #0;
  while c <> '.' do
    begin
      readln(c);
      s := s + [c]
    end;
s;= s - ['.'];
if v * s   <> [] then writeln (‘Існують загальні елементи !')
 end.
```

У циклі множина s поповнюється введеним з клавіатури значенням змінної c. Сигналом до закінчення введення служить точка. Цикл продовжується доти, поки c <> **‘.’**. Після закінчення циклу точку віднімаємо з множини. Якщо в двох множинах немає загальних елементів, то результатом їхнього перетину буде порожня множина.

Задача 32. Задано дві множини. Вивести на екран результат об'єднання, різниці і перетину цих множин.

В задачі 31 показано, як ввести множину. Тепер продемонструємо як записуються операції над множинами і як організувати виведення елементів множин. При виведенні елементів множини необхідно перевіряти наявність кожного можливого елементу на предмет входження даного елементу у множину, яка виводиться. Якщо множина визначена як c: set of 1..10, те це означає, що елементами множини можуть бути числа 1, 2, 3, 4, 5, 6, 7, 8, 9, 10. У циклі для всіх і від 1 до 10 необхідно провести перевірку входження числа і у виведену множину c.

```
for i := 1 to 10 do   if i in c then write(i:3);
```

Програма:

```
program operation;
type
  menge = set of 1..10; {Заданий новий тип – множина}
var
    a, b, c : menge; {Описані змінні множинного типу menge}
    i, n : integer;
begin
  writeln ('Введіть елементи множини a. Кінець введення – 0 ‘);
  a := [];
  repeat                                {Цикл для введення множини a}
    readln(i);
    if i in [1..10] then a := a + [i];
  until i = 0;
  writeln ('Введіть елементи множини b. Кінець введення - 0');
  b := [];
  repeat                              {Цикл для введення множини b }
    readln(i);
    if i in [1..10] then b := b + [i];
  until i = 0;
  c := a * b;                          {Операція – перетинання множин}
  writeln (' Перетин множин ‘);
  for i := 1 to 10 do      if i in c then write(i:3);
  writeln;
  c := a + b;                          {Операція – об'єднання множин}
  writeln (' Об'єднання множин ');
  for i := 1 to 10 do     if i in c then write(i:3);
  writeln;
  c := a - b;                          {Операція – різниця множин}
  writeln (' Різниця множин’);
  for i := 1 to 10 do      if i in c then write( i:3);
  writeln;
{Перевірка на підмножину}
  if a <= b then writeln(' a міститься в b')
           else writeln(' a не міститься в b')
```

end.

Задача 33. Написати програму генерування чисел спортлото "6 з 49", так, щоб всі шість чисел, що випали, були різними. Використовуємо для цього тип даних "множина".

Програма:

```
program lotto_1;
uses crt;
const   n = 6;
type    lotto = set of 1..49;
var     L : lotto;    k : integer;
          i : 1..n;
          z : 1..49;
         ok : boolean;
begin
  randomize;
  L := [];            {Ініціалізація множини}
  for i := 1 to n do
{Цикл виконується  6 раз – кількість цифр у результаті}
    begin
      repeat
        z := random(49) + 1;        {Генерація випадкових чисел}
        if z in L then ok := false   {Визначення входження   згенерованого
                                      числа в множину}
                else
                     begin              {Приєднання нового числа}
                        L := L + [z];
                        ok := true
                      end;
       until ok;       {Цикл repeat повторюється доти, поки ok має
значення   false}
     end;
      clrscr;
{Виведення шести різних випадкових чисел з 49}
   for k := 1 to 49 do
             if    k in L
                     then     write ( k:4 );
end.
```

У програмі проводиться перевірка згенерованого числа на предмет входження у формовану множину. Якщо нове число z вже існує в множині L, то воно ігнорується і відбувається генерування наступного випадкового числа.

10 ВИКОРИСТАННЯ ПІДПРОГРАМ В МОВІ ПАСКАЛЬ

Програми, які не поділяються на окремі структурні елементи називають монолітними. Великі монолітні програми складні для розробки, налагодження і супроводження.

У великих програмах часто зустрічаються фрагменти, однакові за діями, які вони виконують, і відрізняються тільки в значеннях вхідних даних. При такій побудові програми треба задавати фактично одну і ту ж групу операторів для кожного повторення фрагментів одного призначення. Для ефективного програмування подібних повторень фрагментів в мовах програмування введено поняття підпрограми.

Підпрограма – це група операторів, сформованих як самостійна програмна одиниця. Підпрограма записується одноразово у визначеній частині програми, а потім в потрібних місцях програми забезпечується тільки звертання до неї. Таким чином, підпрограма – це ефективний засіб економії пам'яті . При звертанні до підпрограми в неї передаються вхідні дані, а після виконання операторів підпрограми, в основну програму передаються результати розрахунків.

Використання апарату підпрограми дозволяє скоротити об'єм і покращити загальну структуру програми з точки зору наочності і читаємості, зменшити ймовірність помилок і полегшити процес налагодження програми. Розклад монолітної програми на підпрограми дає змогу виконувати розробку окремих підпрограм різними програмістами і багато в чому незалежно один від одного.

Крім того, підпрограма може бути розглянута як самостійний блок, що дозволяє використовувати її в загальному випадку при конструюванні алгоритму і програми за принципами спадкового і неспадкового проектування.

В мові Паскаль підпрограми реалізуються у вигляді процедур і функцій. Процедури і функції, які входять в програму, можуть містити свої підпрограми та викликати процедури і функції більш низького рівня і таке інше.

Послідовно структурування програми продовжують до того часу, поки підпрограми-алгоритми, що реалізуються, не стануть настільки простими, щоб
їх можна було легко запрограмувати. Таким чином програма отримує ієрархічну структуру. Саме такі програми в літературі називають **оболочними або модульними.** Програмісти з великим практичним досвідом віддають перевагу таким програмам, оскільки вони прості для розробки та розуміння і легко модифікуються.

Тобто можна сказати, що **підпрограмою** називається найменована логічно закінчена група вказівок, яку можна викликати для виконання довільну кількість раз з різних місць програми.

В мові Паскаль існують два види підпрограм.

1. **Процедура** – це незалежна найменована частина програми, призначена для виконання конкретних дій.

Процедура складається із заголовка і тіла. Це нібито програма в мініатюрі. Коли процедура виконає своє завдання, програма продовжить виконуватися з вказівки, яка слідує безпосередньо за вказівкою виклику процедури. Використання імені процедури в програмі називається **вказівкою процедури або викликом процедури.**

Загальна структура процедури має вигляд:

Procedure < ім'я процедури > (< список параметрів >) – заголовок;

- label мітки;
- const розділ констант;
- type розділ об'яв типів;
- var розділ об'яв змінних;
- процедури, які входять в дану;
- begin
  тіло головної процедури;
- end;

Заголовок процедури на відміну від заголовка програми Program має бути обов'язково присутнім. Procedure – службове слово, ім'я – визначуване загальними правилами складання ідентифікаторів, список параметрів (формальних) – перелік імен для позначення вхідних даних і результатів роботи процедури з вказівкою їх типів.

Наприклад:

Процедура frame формує на екрані рамку розміром 8×8 із знаків '*'. Лівий верхній кут рамки має координати (x , y). Програма будує дві рамки. Функція GotoXY (x, y: byte) зміщує курсор в позицію на екрані з координатами x і y, x – стовпчик, y – рядок.

```
Program demoprocedure;
  uses crt;
....Procedure frame (x, y: byte);
   var i: byte;
   begin
        GotoXY (x, y); write('*********');
         for i:=l to 8 do
           begin
                GotoXY (x, y+1);
                write(‘*********');
           end;
              GotoXY (x, y+9);
               write (‘*********’);
   end;
  begin
```

```
        clrscr;
        frame (31, 2);
        frame (41, 12);
end;
```

Допускається опис процедури, яка не має списку параметрів. В цьому випадку параметри в процедуру і з неї передаються через систему глобальних параметрів. Оголошені всередині процедури змінні називаються **локальними** по відношенню до даної процедури. Локальні змінні недоступні поза межами даної процедури. Зміни, які відбуваються з локальними змінними всередині процедури, не впливають на значення змінних з такими самими іменами, які описані поза даною процедурою.

Змінні, які використовуються в процедурі, але описані поза нею, називаються **глобальними** по відношенню до даної процедури. Будь-які зміни значень глобальних змінних всередині процедури змінюють значення цих змінних поза процедурою.

2. **Функція.**

Якщо результатом виконання деякої процедури є одне скалярне значення, то цю процедуру бажано оформити як функцію.

Заголовок функції має вигляд:

function < ім'я > (список формальних параметрів): < тип >;

де function – службове слово, ім'я функції, створюєм за правилами складання ідентифікаторів, список формальних параметрів – перелік імен для позначення вхідних даних і результатів роботи функції з вказівкою їх типів.

Допускається опис функції без переліку формальних параметрів:

function < ім'я >; < тип >;

В цьому випадку параметри в функцію передаються через систему глобальних параметрів.

У цієї підпрограми дві основні відмінності від процедури. Перша – заголовок. Він складається із слова function, за яким йде ім'я функції, а потім в круглих дужках – список формальних параметрів, потім через двокрапку записується тип функції, тобто тип поверненого параметра. Функція може повертати типи дійсні, рядкові і будь-якого покажчика.

Друга відмінність в тому, що процедура може мати декілька вихідних параметрів-результатів, а функція – тільки одне значення, яке передається через її ім'я. Саме цим пояснюється те, що в тілі функції хоча б один раз повинно присвоюватись обчислене значення. Структура функції така ж, як і в процедурі.

Наприклад: Обчислити значення

$Y = tg(x) / (1+ tg^2x);$

```
function tan (x: real): real;
 var  tangens: real;
      begin
```

```
    tangens:=sin(x)/cos(x);
    tan:=tangens;
  end;
```

В основній програмі оператор присвоювання буде:

```
Y:=tan(x) / (1+ sqr(tan(x))).
```

При опису підпрограми і виклику їх, використовують поняття формальних і фактичних параметрів, які вказуються після заголовка процедури в круглих дужках.

Формальні параметри – це змінні, фіктивно (формально) присутні в процедурі і визначаючі тип і місце підстановки фактичних параметрів.

Фактичні параметри – це реальні об'єкти програми, які замінюють в тілі процедури при її виклику формальні параметри.

Формальні і фактичні параметри збігаються за трьома ознаками:

– **за** типом змінних, які вони використовують;

– **за** кількістю змінних;

– формальні параметри замінюються фактичними суворо в порядку їх перерахунку.

Розрізняють **параметри-значення**, це коли після заголовка процедури в круглих дужках вказуються змінні, за допомогою яких в процедурі передаються данні і їх типи. Слово var перед ними відсутнє. Такий спосіб передачі параметрів в процедурі називається **передачею за значенням**. При цьому значення фактичного параметра робиться доступним для процедури. Його можна використовувати в роботі, змінювати довільним чином. Але ці зміни проявляються тільки в межах процедури, тобто є локальними. Вони не впливають на фактичні параметри поза процедурою.

```
Program parametr;
var C, D : integer;
 procedure param (A, B : integer);
    var S : integer;
   begin S:=0; S:=A+B
         writeln('S=', S);
   end;
  begin
       C:=10; D:=100;
        param (C, D);          {1-й спосіб}
        param (10, 100);       {2-й спосіб}
  end.
```

Змінні A і B – формальні параметри, а C і D – фактичні параметри. Передачу параметрів можна здійснити двома способами.

Для того, щоб процедура змогла змінювати значення фактичних параметрів, потрібно змінити спосіб передачі параметрів в процедуру. Новий спосіб називається **передачею за іменем.**

При використанні цього способу заголовок процедури змінюється таким чином: перед ідентифікатором формального параметра в заголовку процедури вказується службове слово var. Змінні, перед якими записане слово var , називаються **параметрами-змінними.**

Приклад: сума і різниця квадратів

```
Program kvadro;
Var A,B: real; SumAB, SubAB: real;
Procedure SumSub (x, y : real; var Sum, Sub : real);
 begin
        Sum:=x*x + y*y;
        Sub:=x*x - y*y;
 end;
 begin
        A:=1.7;  B:=8.09;
        SumSub (A, B, SumAB,  SubAB);
        writeln ('Сума =', SumAB, 'різниця =', SubAB);
end.
```

Всі формальні параметри поділяються на 4 категорії:

– параметри-значення (ці параметри підпрограма може змінити в основній програмі)
– параметри-змінні (ці параметри підпрограма може змінити в основній програмі);
– параметри-константи (тільки в версії 7.0);
– параметри-процедури і параметри-функції (тобто процедурного типу).

Приклад використання параметрів-змінних.

Використати функцію для обчислення максимального елементу в масиві. В головній програмі визначається тип-масив.

```
Type aArr = Array [1..50] of integer;
Var Massiv: aArr; Maxim: integer;
```

Функція в цьому випадку має вигляд:

```
Function Max (var Mas: aArr; N: byte): Integer;
var Ma: Integer; i: byte;
begin Ma:=Mas [1];
   for i:= 2 to N do
        if Ma < mas[i] then Ma:= Mas[i];
         Max:= Ma
  end;
```

Нехай треба визначити максимальне число з перших 35 чисел масиву. Це можна зробити за допомогою оператора:

Maxim:= Max (Massiv, 35);

Іноді в якості параметра в підпрограму треба передати ту чи іншу змінну, але змінювати її підпрограма не повинна. В цьому випадку не бажано передавати цей параметр як параметр-змінну. Можна його передати, як

параметр-значення, але, якщо змінна має великий розмір (масив, запис), то копія цього параметра займе велику частину стека і навіть зможе його перетворити. Це також приведе до зменшення швидкодії програми. В цій ситуації параметр краще передавати як **параметр-константу**. Такий параметр, якщо він структурованого типу, передається своєю адресою, але передбачається захист від його зміни. Використовувати параметр-константу можна тільки в версії Турбо Паскаля 7.0.

Параметр-константа вказується в заголовку підпрограми аналогічно параметру-значенню, але перед іменем параметра записується службове слово const. Дія слова const розповсюджується до найближчої " ; " , тобто в межах однієї групи.

Приклад. Function NewString (Const S: string); string. Тип параметра-значення може бути будь-яким, крім файлового. Тип виклику підпрограми на місці параметра-змінної в якості фактичного параметра можна використовувати будь-який вираз сумісний для присвоювання типу, яка не має файлових компонентів.

Параметр-константу не можна передавати в іншу підпрограму в якості фактичного параметра.

Приклад. Type aArr = Array [1..50] of integer;

Var Massiv: aArr;

Функція обчислення максимального елементу в масиві в якості першого параметра – параметр-константа.

```
Function Max (Const Mas : aArr; N : Byte): integer;
Var Ma: integer; i: Byte;
begin
        Ma :=Mas[l];
        for i:= 2 to N
        do
      if Ma < Mas[i]  then  Ma := Mas[i];
       Max := Ma;
      end;
```

Параметри без типу. В Турбо Паскалі можна використовувати параметри-змінні і параметри-константи без вказування типу. В цьому випадку фактичний параметр може бути змінною будь-якого типу, а відповідальність за правильне використання того чи іншого параметра накладається на програміста.

Приклад. Function Tqual (Var Paraml, Param2; Len: Word): Boolean.

Тут Paraml, Param2 – параметри-змінні без типу (замість них можна використовувати, наприклад, будь-які змінні простого типу, типу-масив, типу-запис і т. і.). Len – параметр значення.

Слід мати на увазі, що параметр „без типу" в середині програми типу не має і перед його використанням потрібно перетворити до конкретного типу, використовуючи ідентифікатор відповідного типу так, як раніше вказувалось, при цьому отриманий результат може бути будь-якого розміру.

Приклад. Функція обчислення максимального елементу в масиві. При цьому в якості першого параметра використовується параметр-змінна без типу:

```
function Max (Var Masr; N: Byte): integer;
Type aArray = array [ Maxint ] of integer; {тип масиву   максимального
                                                розміру }
Var Ma: integer; i: byte;
   begin      Ma:=      aArray
        (Mas)[l]; for i:=2 to
        Ndo
   if Ma < aArray (Mas)[ i ] then Ma:= aArray( Mas )[i];
    Max:= Ma
    end;
```

В цьому випадку в якості першого параметра, що передається, можна використовувати будь-який масив (і не тільки масив), так що програма стане більш універсальною. Але все ж тут необхідно передавати в якості другого параметра фактичний розмір інформації, що не дуже зручно.

В версії 7.0 можна в якості параметрів-змінних використовувати **масиви і рядки відкритого типу,** в яких не задаються розміри. В якості фактичного параметра в цьому випадку можна використовувати масив або рядок будь-якого розміру, але масив повинен складатися з тих же компонентів, що і компоненти відкритого масиву. Такі параметри введені для того, щоб підпрограма змогла обробити масив або рядок будь-якого розміру. Фактичний розмір масиву в цьому випадку може бути визначений за допомогою функції High.

Приклад. Функція обчислення максимального елементу в масиві. В якості параметра, що передається, використовується масив відкритого типу.

```
Function Max (Var Mas : array of integer): integer;
Var Ma : integer;  i: byte;
begin Ma : = Mas[ 0 ];
          for i:= 1 to High( Mas) do
        if Ma<Mas[i] then  Mas:=Mas[i];
    Max:=Ma
        end;
```

В цьому прикладі в підпрограму передається тільки один параметр і вона може працювати з будь-яким одновимірним масивом цілих чисел. Але

потрібно мати на увазі, що при роботі підпрограми для відкритого масиву в стеку
створюється його копія і може його переповнити і, відповідно, виникнуть перебої. Перетворення стека понижує швидкодію.

Різновид відкритого масиву – відкритий рядок, який може задаватися або за допомогою стандартного типу OpenString, або за допомогою типу String і використання ключа компілятора {$P+}. Наприклад, рядок, заповнений певним символом , може мати вигляд:

```
Procedure FullChar (Var Str: OpenString; Ch: Char);
                        або
  {$P+} Procedure FullChar (Var Str: String; Ch: Char);.
```

Параметром, що передається може бути **параметр-процедура** або **параметр-функція,** тобто параметр процедурного типу. Фактично цей параметр є параметром-значенням, тому що записується без зарезервованого слова Var.

В якості фактичного параметра в цьому випадку використовується відповідна процедура або функція, яка має необхідну кількість параметрів типів, що потребуються.

Для параметрів-процедур і параметрів-функцій існують ті ж правила, що і для інших змінних процедурного типу: підпрограми повинні компілюватись з ключем {$F+} і мати директиву far, не повинні бути стандартними підпрограмами, не повинні об'являтись всередині других програм, не повинні мати директиви inline або interrupt.

Приклад:

Програма, що друкує таблицю додавання і добутку двох цілих чисел в заданому діапазоні.

```
Program Example_2_5;
    type Func = function (X, Y: integer):
      integer; {$F+}
    function Add (X, Y: integer): integer;
          begin Add:= X + Y End;
           function Multiply (X, Y: integer):
       integer;
                 begin
                       Multiply:=X*Y
                  End;
 {$F-} {Процедура друку таблиці}
Procedure PrintTable (A, B: integer; Operation: Func);
Var i, j: integer;
                   begin
                 for i:=l to A
                   do begin
                       for j:= 1 to B do
```

```
                    write (Operation (i, j ) : 5); end;
                writeln
end;
begin
  PrintTable  (10,5,  Add);  Writeln;
  PrintTable (10,5, Multiply)
   end.
```

Процедура EXIT. Відомо, що оператор Goto не можна використовувати для дострокового виходу з підпрограми. В Турбо Паскалі для цього використовується процедура EXIT. Процедура EXIT завершує роботу підпрограми з поверненням в блок, який її викликав.

Приклад. Функція, що визначає перше від'ємне число в масиві.

```
Function Minus (var Massiv; N: integer): real;
   begin Minus:= 0;              {Тут Massiv – параметр без типу}
       for i:=l to N do
          if T (Massiv) [i] <0 then          {перетворення типу}
     begin
        Minus := T( Massiv)[ i ];
        Exit                    { достроковий вихід із  функції }
     end;
   end;
```

Поняття рекурсії. В математиці рекурсією називають спосіб опису функцій або процесів через самих себе.

Якщо процедура або функція в процесі виконання викликає саму себе, то ми маємо діло з **рекурсією.** Такий виклик процедури або функції може виникнути або внаслідок рекурсивного опису, або внаслідок рекурсивного звертання. Рекурсивний опис передбачає, що у виконавчій частині блока процедури або функції присутнє звертання до неї самої. Прикладом рекурсивного опису може стати функція обчислення факторіала:

```
function Factorial (N: integer): integer;
 begin
       if N=1 then Factorials(1)
              else Factorial( N)*Factorial (N-l);
  end;
```

Тут Factorial(N) визначається через значення Factorial(N-l), яке визначається через Factorial(N-2) і до доведення до значення Factorial (0), яке визначається **явно** і дорівнює 1. Будь-яке рекурсивне звертання повинно містити явне визначення для деяких значень аргументу, тому що інакше процес доведення був би нескінченим. Таким чином при рекурсивному описі необхідна наявність базової частини опису, яка забезпечувала б завершення рекурсивних викликів функції або процедури.

Рекурсивне звертання можна розглядати на прикладі обчислення подвійного інтеграла за формулою трапеції. **Точність** цього обчислення тим вища, чим більша кількість частин **розподілу n.** Збільшуючи число n, можна досягти потрібної **точності.** Якщо функція TRAP обчислює інтеграл за методом трапецій при заданій кількості інтервалів N і A, B – межі інтегрування , а FN – функція обчислення підінтегрального виразу, обчислення подвійного інтеграла можна виконати за допомогою рекурсивного звертання до функції TRAP:

I:= TRAP (N1, Al, Bl, TRAP (N2, A2, B2, FN));

Нижче приведена рекурсивна функція призначена для обчислення найбільшого спільного дільника двох цілих чисел N1 і N2.

Приклад.

```
Function HightFactor (Nl, N2: integer): integer;
Var P: integer;
begin
        if N1 > N2 then P := HightFactor( N1, N2 )
                   else
                     if N2 <=0 then P:= N1          {нерекурсивне
                     рішення}
                else P := HightFactor( N2, N1, Mod N2 );
    HightFactor := P
  end;
```

Для того щоб виконання рекурсивної програми закінчилось, необхідно щоб було і нерекурсивне рішення. В протилежному випадку можливе зациклення.

Деякі алгоритми простіше описати, використовуючи рекурсію, ніж ітерацію. Це відноситься в першу чергу до алгоритмів, що працюють з різними спусковими структурами.

Використання рекурсивних процедур і функцій робить програму в цілому більш гнучкою і наглядною, але не завжди ефективною, тому що працює така програма, як правило, повільно і використовує більше пам'яті. Річ в тому, що при кожному виклику рекурсивній процедурі або функції відводиться пам'ять під локальні змінні.

При порівнянні ітераційних методів рішення і методів, які використовують рекурсію, частіше більш ефективні перші. Таким чином, рекурсію слід використовувати тільки там, де немає очевидного ітераційного рішення. Рівень вкладеності рекурсій може бути обмежений в конкретних реалізаціях мови.

Розрізняють пряму і непряму рекурсію. Функція HighFactor є характерним прикладом прямої рекурсії. Непряма рекурсія виникає тоді, коли один блок викликає другий, а другий, в свою чергу – перший.

11 РОБОТА З ФАЙЛАМИ

Дані різної форми зберігаються у файлах. Це програми і документи, зображення і звуки. До файлів відносяться прикладні і сервісні програми, а також дані, які отримані за допомогою програм.

Файл – це пойменована частина оперативної пам'яті на диску або другому запам'ятовувальному пристрої. Назва файла складається з двох частин: саме ім'я та розширення. За типом розширення ми пізнаємо з яким файлом маємо справу: pas, cpp, asm – тексти програм на мовах програмування, txt – текстові файли, doc документи текстового процесора Word, bmp – файли рисунків і т.д. На відміну від масивів кількість даних у файлі при його описуванні не вказується, елементи файла не мають індексів.

Файли в DOS розділяють на дві категорії – **текстові** і **двійкові**. Текстові файли призначені для читання людиною. В них зберігаються тексти програм, командних фалів DOS та інших. Всі інші файли вважаються двійковими, тобто їх не можна прочитати.

Все, що є файлом в DOS, є фізичним файлом і на Паскалі. Введення даних в програму із файла – це один із засобів введення даних. Він необхідний як правило, в великих професійних програмах, тому що програма з розміщеними всередині неї даними не дуже зручна і якщо дані змінюють, то треба змінювати і програму.

Зовнішні файли даних збільшують універсальність і гнучкість програм обробки даних. Операція введення даних означає заповнення комірок пам'яті даними, отриманими із файла, а операція виведення – пересилку даних з робочої пам'яті (ОЗП) в файл. Ці операції виконуються через область пам'яті, яка називається буфером.

Текстові файли в Паскалі – це файли послідовного доступу. Вони складаються із елементів різної довжини, до яких можна звертатись тільки послідовно – від початку до кінця. Для багатьох випадків такий порядок природний, але іноді викликає великі затрати часу.

Двійкові файли в Паскалі – це файли прямого або довільного доступу. Вони складаються з однотипних елементів, як і масиви. Знайти будь-який елемент в такому файлі можна за його порядковим номером. В загальному випадку система обробки даних з файлами прямого доступу більш ефективна ніж система з послідовними файлами, але потребує більшого часу на розробку.

Паскаль підтримує три файлових типа:

- текстові файли (типу Text);
- типізовані файли (типу File of <тип>);
- безтипові файли (типу File).

Будь-яка програма, яка використовує існуючий файл для введення або його створення при виведенні, обов'язково містить такі кроки:

– відкриття файла (у відповідному режимі);
– обробка файла (читання або записування);

– закриття файла.

Існує декілька загальних процедур для файлів всіх типів, а також додаткові процедури для кожного типу окремо.

Загальні процедури для роботи з файлами

Таблиця 1 – Обробка помилок при роботі з файлами

Процедури і функції	Для чого призначені
Assign (var f: string);	Зв’язують файлову змінну f з ім’ям фізичного файла, заданого в рядку
Reset (var f; [: File; RecSize: Word])	Відкриває існуючий файл даних на диску, ім’я якого було перед цим пов’язане з процедурою Assign з деякою файловою змінною, для його зчитування або запису в нього даних
Rewrite (var f: File; [; RecSize: Word])	Створює новий файл і відкриває його. Якщо файл з таким іменем уже існує, то його вміст знищується, а сам файл записується знову
Close (var f)	Закривається відкритий канал введення-виведення з логічним ім’ям f
Eof (var f): Boolean	Функція Eof набуває значення true, якщо досягнуто кінець файла, інакше набуває значення false, якщо файл пустий
Erase (var f)	Вилучає закритий фізичний файл, зв’язаний з файловою змінною f
Rename (var f; NewName: string)	Переміщує закритий фізичний файл, який зв’язаний з фізичною змінною f

Текстові файли

Текстовий файл розглядається як послідовність, розділена на рядки. В кінці кожного рядка є два спеціальних символи: #13 і #10. Вони означають повернення рядка (CR) і повернення каретки (LF). Кінець файла позначається явно символом #26(Ctrl+Z).

Введення і виведення даних відбувається за допомогою стандартних процедур Read і Write, спеціальні розширення яких дозволяють працювати із значеннями несимвольного типу. Послідовність символів автоматично перетворюється до значення того типу змінної, яка використовується в файлових операціях. Наприклад, виклик Read (f, n), де n – змінна типу Byte, автоматично перетворює, зчитану з файла f, послідовність цифр в число, яке і буде присвоєно n. Якщо в цьому випадку крім цифр, зустрінуться і інші символи, то буде помилка.

Для створення текстового файла визначається файлова змінна типу Text:

Var ім’я: Text;

Потім файлова змінна зв’язується з фізичним файлом процедурою Assign, після чого відкривається файл. Рядковий вираз в процедурі Assign повинен містити повне ім’я файла, яке задовольняє всім вимогам операційної системи. Таким чином, конкретному файлу на зовнішньому пристрої ставиться у відповідність логічна файлова змінна, до якої потім звертаються всі інші файлові процедури.

Наприклад, найпростіше введення в текстовий файл out.dat на диску d можна зробити так:

```
Var f: Text;
Begin
        Assign (t, ‘d:\out.dat’); {зв’язок з фізичним файлом}
        Rewrite (t); {створення і відкриття файла для запису}
        Writeln (t, ‘виведення в файл’); {запис рядка в файл}
        Close (t);
  end.
```

Крім дисплею і дисків текст можна виводити в комутаційний порт, на принтер, на пустий пристрій. Ці пристрої складні і різноманітні, тому DOS моделює їх за допомогою текстових файлів, а програміст працює з логічними моделями пристроїв.

Допускаються такі імена:

CON – консоль (введення з CON – читання з клавіатури, виведення в CON – виведення на екран);

LPT1, LPT2, LPT3 – паралельні порти, через ці імена відбувається виведення на принтер;

PRN – принтер – синонім LPT1;

COM1, COM2 – послідовні порти.

В системній бібліотечці Паскаля є дві текст-файлові змінні: INPUT і OUTPUT. Вони зв’язані з пристроєм CON автоматично і є стандартними файлами введення і виведення.

Стандартні файли введення-виведення можуть бути „переназначені” за допомогою процедури Assign. В прикладах приведені виведення на принтер:

```
Begin {‘LPT’}
  Assign (Output, ‘PRN’);
  Rewrite (Output);
  Writeln (Output, ‘*принтер*’);
  Close (Output);
end.
Var t:Text;
begin
        Assign (t, ‘’);
```

```
        Rewrite (t);
        Writeln (t, '*екран*');
        Close (t);
end.
```

Крім загальних процедур є декілька, які працюють тільки з текстовими файлами.

Таблиця 2 – Процедури для роботи з текстовими файлами

SetTextBuf (var t: Text; Var Buf[; Size: Word])	Встановлює розмір, буфера файла f до його відкриття рівним size байт
Flush (var t: Text)	Примусово записує дані з буферу файла f в фізичний файл
Append (var t: Text)	Відкриває існуючий текстовий файл f для додавання його записів в кінець файла
EoLn[(var t:Text)]: Boolean	Набуває значення true, якщо досягнуто кінець рядка або файла (#13 або #26)
SeekEoLn[(var t:Text)]: Boolean	Повертає значення true, якщо досягнуто кінець рядка або файла, або перед ними є лише проміжки і символи табуляції (#32 або #26)
SeekEof[(var f:Text)]:Boolean	Повертає значення true, якщо досягнуто кінець файла (#26), або перед ним є лише пропуски, ознака кінця рядка і символи табуляції(#32, #13, #9)

Приклад.

Дано текстовий файл test.dat. Записати в нього деякі дані, після чого прочитати всі дані, тобто вивести на екран:

```
Uses Crt;
var t: Text; z: String;
    i: Byte;
begin
  ClrScr;
  Assign (t, 'd:\test.dat');
  Append (t); {відкриття файла для поповнення}
  for i:=50 to 55 do
        Writeln (t, i:10, Sqr(i):30); {запис в файл}
        Close (t);
  Reset (t); {відкриття файла для читання}
  While Not Eof (t) do {поки не досягли кінця}
```

```
  begin
        readln (t, z); {читання з файла}
        writeln (z); {виведення на екран}
  end;
Close (t);
Readln;
end.
```

Процедура Append відкриває зовнішній існуючий текстовий файл для добавлення записів. Файл не стирається, як при виклику Rewrite, а підготовлюється для запису нових даних в кінець файла. В цьому прикладі в файл test.dat будуть добавлятися одні і ті ж шість рядків після кожного виконання програми. Конструкція While Not E of do є дуже зручною при зчитуванні даних з файла. Важливо тільки пам'ятати, що текстовий файл може бути відкритим тільки для запису або тільки для читання.

Типізовані файли.

Типізований (компонентний) файл розглядається як послідовність записів (компонент одного типу). Формат файлової змінної для нього має вигляд:

```
Var ім'я: File of базовий тип;
```

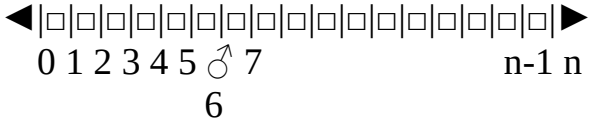
На відмінність від файлів типу Text типізований файл має сувору внутрішню структуру. При запису в нього записується машинне уявлення компоненти, будь то число, символ, рядок, запис, масив або інша структура даних (крім файлів). Файл заповнюється послідовно від початку. Його структура лінійна: запис йде за записом (в випадку типізованих файлів запис в файлі – це компонента, а в іншому випадку без типових – блок) і компоненти нічим не відокремлюються один від одного.

Число компонент файлу може змінюватися, тобто заздалегідь не фіксується як число елементів масиву. Додаються записи лише в кінець, а заміщуються – будь-які існуючі в файлі записи.

Такого терміну як кінець рядка в типізованому файлі не існує, тому введення і виведення даних виконується тільки процедурами Read і Write. Для типізованого файла Reset і Rewrite встановлюються режими читання і запису, тобто допускається чергування цих операцій незалежно від вибору процедури відкриття.

Оскільки компоненти одного типу, а відповідно і одного розміру, то стає можливим довільний або прямий доступ до них за номерами компонент. Прямий доступ означає можливість встановити всередині файла поточний покажчик на заданий рядок. Термін поточний покажчик існує у кожного файла. Це неявно описана змінна, яка вказує на умовну межу між елементами файла. Нумеруються позиції поточного покажчика цілими числами, починаючи з нуля. Реальний номер рядка завжди на

одиницю більше номера позиції. Після кожного кроку читання або запису покажчик зміщується на наступну позицію. Отже, в операції приймає участь той елемент, на який зміщується покажчик. Зобразимо записи графічно; візьмемо такі позначення: ◄► – початок і кінець файла;
| – умовна нумерована межа між записами; □ – запис; ♂ – покажчик.



На рисунку у файлі n записів, поточний покажчик позиційований в позицію 6. Тобто можна прочитати або замінити сьомий запис у файлі. В таблиці приведені функції і процедури, які реалізують прямий доступ в типізованих файлах.

Таблиця 3 – Процедури для роботи з типізованими файлами

Процедури і функції	Призначення
FileSize (var f): Longint;	Підраховує реальне число записів в відкритому файлі f
FilePos (var f): Longint;	Повертає номер запису, на який встановлений поточний покажчик в файлі f
Seek (var f); n: Longint	Встановлює поточну позицію в файлі f
Truncate (var f)	Вилучає всі записи в файлі f від початку поточного покажчика до кінця файла

Функція Eof в типізованому файлі отримує значення true тільки в випадку, коли позиція поточного покажчик збігається з кінцевою межею.

Підсумовуючи, можна сказати, що основна перевага текстових файлів – це можливість зберігати в них різноманітні дані; перевага типізованих – прямий доступ і можливість чергування операцій і запису незалежно від вибору процедури відкриття. Ці операції можливі без перетворення даних, що збільшує їх роботу. Але обмін даними можливий тільки між дисками і пам’яттю програми. Вивести дані напряму на екран не можна.

Приклад . Є символьний файл f. Записати в файл q компоненти файла f в зворотному порядку

```
Var f, g: File of Char;
        Ch: Char;
        n: word;
 Begin
        Assign (f, ‘d:\Ch2.dat’);
        Reset (f);
        Assign (g, ‘d:\Ch2.rev.dat’);
        Rewrite (g);
        N:=FileSize (f);
        Repeat
```

```
            Dec (n);
            Seek (t, n);
            Read (f, ch);
            Write (g, ch);        {Write (ch);}
        Until n=0;
        Close (f); close (g);
end.
```

Елементи файла f читаються з кінця і записуються в файл g. Для позиціювання вказівника процедура Seek використовується перед читанням кожного символа.

Розглянемо ще один типовий приклад для роботи з файлами.

Приклад . Написати процедуру для створення двовимірного масиву і запису його в файл.

```
 procedure Wr: array;
var i, j: Integer;
begin
        Assign (f, 'd:\array.dat');
        {$I-}
        Reset (f);
        {$I+}
        if IoResult=0 then Writeln ('Файл уже існує');
        else
                begin
                        Rewrite (f); {відкриття файла для запису}
                        for i:=1 to n do
                                for j:=1 to n do A[i, j]:=i*j;
                        write (f, A);
                        Close (f);
                        Writeln ('Створимо новий файл');
                end;
        end;
```

Процедура демонструє запис в файл array.dat двовимірного числового масиву. Якщо файл вже існує на диску використовуємо стандартну процедуру IOResult, яка не має параметрів. Функція має сенс тільки при включенні в програму режима компіляції {$I-} і повертає 0, якщо операція введення-виведення відбулась без помилок.

Безтипові файли. Безтиповий або інакше нетипизований файл розглядається як сукупність символів або байтів. Формат запису для нього такий:

Var Ім'я: File;

Безтиповий файл можна вважати узагальненим файловим типом. Подання Char або Byte важливо лише з точки зору, об'єму даних. Будь-який файл, створений як текстовий або типізований, можна відкрити і почати роботу з ним, як з безтиповим набором даних. Внутрішня підтримка такого файла виглядає найбільш близькою до апаратної підтримки роботи з зовнішніми носіями. За рахунок цього досягається максимально можлива швидкість доступу до наборів даних.

Безтиповий файл – це файл прямого доступу, що говорить про можливість одночасного використання операцій читання і запису. Найважливіший параметр для такого файла – це довжина запису в байтах. За замовчуванням вона складає 128 байтів, але в кожному випадку можна підібрати найбільш підходящий розмір. З цією метою розширюється синтаксис процедури введення-виведення:

Reset (var f: File; RecSize: Word) і
Rewrite (var f: File; RecSize: Word),

другий параметр в яких RecSize задає довжину запису в байтах. Щоб гарантовано забезпечити повне читання і вірний запис всього файла, величину RecSize часто вибирають рівною 1.

Для створення доступу до записів безтипового файла допустимо використання процедур Seek, FilePos і FileSize.

В таблиці приведені процедури, які замінюють стандартні процедури введення - виведення Read і Write в безтипових файлах.

Таблиця 4 – Процедури для роботи з безтиповими файлами

Процедури	Призначення
BlockRead (var f: File; var Buf; Count: Word[;var Result: Word])	Виконує читання з безтипового файла
blockWrite(var f: File; Var Buf; Count: Word[;Var Result: Word])	Виконує запис в безтиповий файл

Обидві процедури виконують операції введення-виведення блоками. Смисл параметрів однаковий:

Count – кількість записів, оброблених за один виклик;

Buf – зміна-буфер (безтиповий параметр процедури), розмір якого повинен бути рівний об'єму блоку в байтах, тобто Count*Recsize;

Result – необов'язковий параметр, який містить кількість фактично прочитаних (записаних) записів.

Приклад . Є файл f. Створити копію цього файла

```
Var f, g: File;
    Buf: Array[1…1024] of Byte;
    NumRead, NumWritten: Word;
Begin
    Assign (f, ‘’d:\ch1.dat’); Reset (f, 1);
    Assign (g, ‘d:\ch1.bak’); Rewrite (g, 1);
```

```
        Write (‘Копіюється’, FileSize (f), ‘байт…’);
        Repeat
              BlockRead (f, Buf, SizeOf(Buf), NumRead);
              BlockWrite (g, Buf, NumRead, NumWriteln);
        Until (NumRead=0) or (NumWriteln<>NumRead);
        Close (f); Close (g);
        Write (‘Готово!’); Readln;
  End.
```

Приклад демонструє роботу процедур BlockRead і BlockWrite. Змінюючи розмір буфера, порівняйте швидкість копіювання. Контроль помилок не виконується – початковий файл повинен бути на диску.

12 РОБОТА З ГРАФІЧНИМ МОДУЛЕМ

Для апаратної підтримки графіки випускається декілька типів дисплеїв (і відповідних електронних плат – адаптерів), які відрізняються роздільною здатністю, тобто кількістю точок, які можуть бути відтворені по горизонталі та по вертикалі. Найбільш поширені дисплеї типу EGA, VGA, SVGA. Ми будемо розглядати дисплеї з роздільною здатністю 640× 480 пікселів.

Для формування графічних зображень в алгоритмічній мові TURBO Pascal є бібліотека Graph. Вона підключається стандартним способом за допомогою Uses. У більшості випадків при роботі з графікою не обійтись без засобів модуля CRT, тому підключають і його:

Uses Crt, Graph;

Установка графічного режиму

З моменту підключення модуля Graph програмістові стають доступними всі програми, які в ньому знаходяться. В першу чергу викликається процедура InitGraph, яка встановлює один із можливих графічних режимів, формат процедури;

InitGraph (DriverVar, ModeVar, “D:\TP7\BGI”);

Цілочисельні змінні DriverVar і ModeVar задають драйвер і режим у відповідності зі значеннями, які притаманні даному моніторові. Наприклад: DriverVar:=VGA; ModeVar:=VGA LO;

Перший параметр може бути задано як за ім’ям, так і цифрою (у таблиці вона вказана в дужках поруч з ім’ям драйвера). Наприклад, інструкція:

DriverVar:=VGA; DriverVar:=9; еквівалентні .

Для новачків, котрі можуть не знати типу дисплею свого комп’ютера, є стандартна константа DETECT. Якщо це значення присвоєно параметру DriverVar:

DriverVar:=DETECT;

Процедура InitGraph автоматично ініціює потрібний драйвер і встановить найбільш підходящий для дисплея режим.

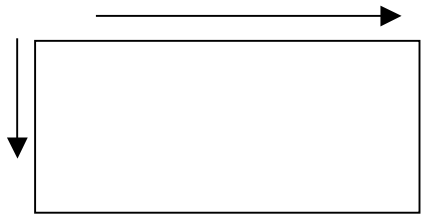
Третій параметр – маршрут до модуля Graph, якщо він розташований у активній директорії, то замість маршруту ставлять 2 апострофи.

Підсумовуючи сказане, запишемо первісну групу інструкцій, що дозволить уникнути будь-яких неприємностей на початковому етапі роботи з графікою.

```
Uses Crt, Graph;
Var DriverVar,  ModeVar:integer;
Begin
  DriverVar:=DETECT;
  InitGraph (DriverVar, ModeVar, ‘‘);
  ErrorCode:=GraphResult;
        if ErrorCode<>grOK then
        begin
              writeln(‘помилка ініціалізації графіки:’,
                    GraphError MSQ(ErrorCode));
              writeln(‘робота програми перервана’);
              Halt(1);
        End;
End.
```

В Турбо-Паскалі прийнята така система координат графічного режиму:

(0,0) коорд. X (GetMaxX, 0) коорд.Y



(0,GetMaxY) (GetMaxX, GetMaxY)

GetMaxX і GetMaxY – стандартні функції модуля Graph, які повертають відповідно максимальні координати по осях X і Y в залежності від поточного відеорежиму.

Графічних операції виконуються чисельною кількістю процедур і функцій бібліотеки Graph. Розглянемо лише групу базових процедур: PutPixel, Line, Setcolor, OutTextXY, Circle, ClearDevice, Rectangle, CloseGraph.

Які б зображення не виводились на екран, всі вони побудовані з точок, групи яких у свою чергу утворюють відрізки і криві.

Отже, маючи засіб побудови точки певного кольору в потрібному місці екрана, теоретично можна створити будь-яке зображення аж до самої картинки.

Таблиця 5 – Шкала кольорів

Темні кольори	Світлі кольори
0 (Black) – чорний	8 (DarkGrey) – темно-сірий
1(Blue) – синій	9(LightBlue) – світло-синій
2(Green) – зелений	10(LightGreen) – світло-зелений
3(Cyan) – голубий	11(LightCyan) – світло-голубий
4(Red) – червоний	12(LightRed) – світло-червоний
5(Magenta) – фіолетовий	13(LightMagenta) – світло-фіолетовий
6(Brown) – коричневий	14(Yellow) – жовтий
7(LightGrey) – сірий	15(White) – білий

У бібліотеці Graph виведення точки виконується процедурою

PutPixel (X,Y,Color), де X і Y – координати екрана, де буде розташована точка, Color – її колір. Можливі значення Color наводяться в таблиці.

Наприклад, оператор

```
For i:=1  to  600 do  PutPpixel(1, 1, 3);
```

виведе в перший рядок екрана неперервну лінію з точок блакитного кольору. Такий самий результат можна одержати за допомогою процедури креслення лінії (відрізку).

Line(X1,Y1,X2,Y2), де X1,Y1 – координати початку, X2,Y2 – координати кінця лінії.

Наприклад: Line(1,1,600,1);

Не важко помітити, що в Line нема параметра для установки кольору. В цьому і інших аналогічних випадках колір задається процедурою

SetColor(3); (або SetColor(Cyan);)

Line(1,1,600,1);

Зображення, які виводяться на екран, звичайно супроводжуються пояснювальним текстом. У графічних режимах для цього використовується, зокрема процедура OutTextXY(X,Y,Text), де X,Y – координати точки початку введення тексту, Text – константа або змінні типу String. Наприклад, щоб вивести повідомлення “Для продовження натисніть будь-яку клавішу”, починаючи з точки 60, 320, треба записати:

OutTextXY(60, 320, ”Для продовження натисніть будь-яку клавішу”);

Проблемою для початківців є виведення чисельних даних, бо у Graph немає призначених для цього процедур. Вихід простий: спочатку перетворити число в рядок за допомогою процедури Str, а потім за

допомогою “+” підключити її до рядка, який виводить OutTextXY. Наприклад:

```
Max:=34.56;
Str(Max:6:2, Smax);       {результат перетворення в Smax}
OutTextXY(400,40,’Максимум’+Smax);
```

У процесі роботи програми одні зображення постійно змінюються іншими, і , щоб уникнути хаосу, екран, доводиться періодично очищати.

Для цього використовується процедура ClearDevice. Ще одна процедура, яку необхідно знати, CloseGraph. Вона не має параметрів, і викликається, коли графічна програма завершує свою роботу або потрібно переходити з графічного режиму в текстовий і навпаки.

Більшість решти процедур бібліотеки Graph служить для креслення різних геометричних та довільних фігур (прямокутників, кіл, еліпсів, кубів тощо.) і їх оформлення для більш ефективного сприймання (‘’заливка обмежених площин різними кольорами і змістом, утворенні на екрані ілюзії тривимірного простору та багато іншого).

Деяке уявлення про бібліотеку Graph дає ця таблиця геометричних образів:

Таблиця 6 – Процедури геометричних образів модуля Graph

Arc	(x, y, stangle, endangle, radius)	Креслення геометричного образу дуги
Bar	(left, top, right, bottom)	Креслення геометричного образу суцільного бруса
Circle	(x, y, radius)	Креслення геометричного образу кола
Ellipse	(x, y, stangle, endangle, xradius, yradius);	Креслення геометричного образу еліпса
FillEllipse	(x, y, xradius, yradius);	Креслення геометричного образу заповненого еліпса
Rectangle	(left, top, right, bottom)	Креслення геометричного образу прямокутника
Lineto	(x, y)	Креслення лінії (від покажчика до точки)

Робота з покажчиками та координатами

GetMaxX		Видача спроможності по X
GetMaxY		Видача спроможності по Y
GetPixel	(x,y)	Видача кольору піксела
Move To	(x,y)	Пересування курсора в задану точку

Робота з кольоровою гамою

Floodfill	(x, y, Border)	Заповнення області

SetBKColor	(color)	Установка біжучого фонового кольору
SetColor	(color)	Установка біжучого кольору
SetFillStylle	(pattern, color)	Установка шаблона заповнення

Штриховка та колір

SOLIDFILL	1	суцільне штрихування поточним кольором
EMPTYFILL	0	немає штрихування
LINEFILL	2	штрихування лініями
LTSLASHFILL	3	штрихування косими лініями
SLASHFILL	4	штрихування товстими лініями
HATCHFILL	7	штрихування клітинками +++
XHATCHFILL	8	штрихування косими клітинками х х
INTERLEAVEFILL	9	густе штрихування в клітинку
WIDEDOTFILL	10	штрихування точками
CLOSEDOTFILL	11	густе штрихування точками

Ефект руху в графічному режимі

Для реалізації простих рухів фігур в графічному режимі використовують процедури GetImage, PutImage, які працюють з константами CopyPut, XorPut, OrPut і т.і.

Приклад демонструє використання операцій зберігання і відображення області екрана з різними режимами виведення графічних зображень, які мають форму прямокутника.

```
Uses Graph;
Var     Driver, Mode; integer;
        Size:Word;              {розмір області}
        X1,Y1:Word;             {координати початкової точки}
        P: pointer;             {покажчик на область}
Begin
        Driver:=DETECT;
        InitGraph(Driver, Mode,'');
        if graph result<>0 then Halt(1);
              {побудова початкової фігури}
        setfillstyle(solidfill,lightred);       {суцільна штрихова рожева}
        pieslice(getmaxY div 2, getmaxY div 2, 0, 360, 100);
{заповнений сектор зі штрихуванням кола з радіусом 100, координати центра Getmaxy (Getmaxy div 2 від 0° до 360°)}
          Readln;
{визначення розміру області}
     X1:=GetmaxX div 2 -100;
     Y1:=GetmaxY div 2 -100;
```

```
    Size:=ImageSize(X1, Y1, Y1+200, Y1+200); {визначає і виділяє
                        кількість байтів пам'яті, необхідних для зберігання
                        прямокутної області екрана}
    GetMem(p,Size); {зберігання вказаної області в пам'яті}
    GetImage(X1,Y1,X1+200,Y1+200,p^); {зберігає в буфері  двійковий
                                        образ області екрана}
  ClearDevice; {очищення екрана}
  SetBKColor(blue);
  PutImage(X1,Y1,p^,CopyPut); {виведення зображення з буферу на екран}
  Readln;
{відображення області в режимі , зображення вилучається з екрана}
        PutImage(X1,Y1,p^,X OR PUT);
        Readln;
        CloseGraph;
    End.
```

При роботі з буфером важливо знати розмір пам'яті, необхідної для зберігання образу фрагмента. Для цього використовують функцію: ImageSize(X1,Y1,X2,Y2: integer):word,

де (X1,Y1) та (X2,Y2) – координати верхнього лівого і правого нижнього кута прямокутника. Значення, яке повертає функція, є розміром частки пам'яті, яка потрібна для зберігання образу даного прямокутника (в байтах).

Оскільки для зберігання фрагмента зручніше всього використовувати динамічну область пам'яті, то результат роботи ImageSize використовують в якості вхідної інформації для процедури GetMem, яка виділяє вказаний об'єм пам'яті в динамічній області.

Зберігання образу фрагмента в пам'яті виконується процедурою GetImage(X1,Y1,X2,Y2: integer; var BitMap),

де X1,Y1,X2,Y2 – координати фрагмента, BitMap – нетипизований параметр, який повинен бути більшим або рівним плюс шість розміру пам'яті, відведеної для області екрана. Розмір збережуваного зображення не повинен перевищувати 64 Кбайти.

Процедура PutImage(X,Y: integer; var BitMap; Mode: word)відновлює збережений в буфері BitMap прямокутник, лівий верхній кут якого задається координатами (x,y). На відмінність процедури GetImage тут достатньо вказати координати тільки одного кута. Щоб вказати режим виведення зображення використовують параметр Mode, який задає оператор, котрий буде виводити двійковий образ на екран.

Допустимі значення цього параметра такі:

```
const
        CopyPut=0; {MOV}
        XORPUT=1; {XOR}
```

```
ORPUT=2; {OR}
ANDPUT=3; {AND}
NOTPUT=4; {NOT}
```

У фігурних дужках вказані відповідні оператори Асемблера.

Необхідно відмітити, що виведення фрагмента в режимі XORPUT забезпечує його видимість на будь-якому фоні.

Режим ANDPUT зручний для „вирізання” з фону фрагмента довільної форми чорного кольору.

Робота з клавіатурою

Цикл repeat until Keypressed приводить до очікування натискання любої клавіші, яка виробляє код, при умові що буфер клавіатури пустий, якщо ж буфер клавіатури містить хоча б один код (тобто Keypressed = TRUE), то цей цикл не приведе ні до яких дій і керування буде передано наступному за ним оператору.

Таким чином, щоб можна було коректно використовувати цикли очікування натискання клавіш, необхідно попередньо очищати буфер клавіатури від кодів випадково натиснутих користувачем клавіш: для цього застосовується другий цикл такого виду:

```
var     ch:char;
  begin ……..
        while KeyPressed do Ch; {очищення буферу}
  end;
```

Звук

Для створення звукових ефектів використовують процедури Sound, Nosound і Delay із модуля Crt.

```
procedure Sound(HZ:integer);
```

Параметр HZ процедури Sound задає частоту звука в Гц. Створений звук триває до подання команди Nosound, яка зупиняє дію попередньої процедури.

```
procedure Delay(MSes:integer)
```

затримує роботу програми, а відповідно і звучання на число мілісекунд, вказане параметром MSes.

Наведемо 2 приклади, які демонструють найпростіші звуки, які можна вилучити із динаміка комп’ютера за допомогою процедур Sound, Nosound, Delay.

Перший приклад демонструє звук зайнятого телефону.

```
program  PhoneIsBusy;
uses crt;              {константи підібрані для процесора Celeron 600Mhz}
var i:integer;      {константи для процесора 486DX2 треба поділити на 10}
begin
   repeat  {проводити сигнал до натискання любої клавіші}
     for i:=1 to 100 do        {цей цикл створює ефект деренчання звука }
```

```
        begin
           Sound(400);
           Delay(40);
           Nosound;
       end;  Delay(6000);
      Until Keypressed;
End.
```

В другому прикладі послідовно викликаються звуки частотою від 100 до 3000 Гц і назад. В результаті отримуємо звук сирени.

```
  program test_sound;
      uses crt;
      var i:integer;
  begin
    repeat
        for i:=100 to 300 do
           begin
              sound(i);
              delay(100);
           end;
        for i:=300 downto 100 do
           begin
              sound(i);
              delay(100);
           end;
    until keypressed;
    nosound;
  end.
```

Приклад графічної програми рисування горщика з кактусом

```
program F1;
uses graph;
var
  a,grDriver,grMode,grCode:integer;
begin
    initgraph(grDriver, grMode, 'd:\work\bpascal\bgi');
    setfillstyle(5,1);
    fillellipse(320,340,130,20);
    line(190,340,210,400);
    line(450,340,430,400);
    ellipse(320,400,180,360,110,20);
    setfillstyle(1,4);
    floodfill(320,400,15);
    setfillstyle(11,4);
```

```
    fillellipse(360,260,60,15);
    ellipse(320,260,315,225,30,120);
    floodfill(320,240,15);
    ellipse(320,260,315,225,20,120);
    fillellipse(410,237,8,30);
    readln;
end.
```

Приклад рисування зображення сонця

```
program sun;
uses crt,graph;
var x,y,cd,cm,r,i,x1,x2,y1,y2:integer;
   l,dl:real;
begin
    cd:=vga; clrscr;
    cm:=vgaHI;
    initgraph(cd,cm,'d:\work\bpascal\bgi\');
    if graphresult <> grok then halt(1);
    setcolor(1);
    setbkcolor(1);
    floodfill(160,60,1);
    setcolor(0);
    r:=50;
    l:=0;
    dl:=pi/10;
    for i:=0 to 9 do
       begin
          l:=l+dl;
          x1:=round(r*cos(i)+60);
          y1:=round(r*sin(i)+60);
          x2:=round(r*cos(i+pi)+60);
          y2:=round(r*sin(i+pi)+60);
          line(x1,y1,x2,y2);
       end;
    readln;
    closegraph;
end.
```

Приклад програми, в ході якої **зелений квадрат** з’явившись в лівому верхньому куті переміщується вправо вниз по діагоналі.

```
program kas;
   uses crt,graph;
   var gd,gm:integer;
```

```
   x,y:integer;
   i:integer;
begin
    gd:=DETECT;
    initgraph(gd,gm,'d:\work\bpascal\bgi');
    if graphresult <> grok then
    Halt(1);
    Rectangle(10,10,100,100);
    for i:=1 to 300 do
     begin
       setcolor(green);
       rectangle(10+i,10+i,100+i,100+i);
       setcolor(black);
       line(9+i,9+i,9+i,200+i);
       line(9+i,9+i,200+i,9+i);
       delay(1000);
     end;
    readln;
    closegraph;
end.
```

13 ПИТАННЯ ДЛЯ САМОКОНТРОЛЮ

1. Що таке алгоритм? Які основні властивості алгоритмів?
2. Для чого необхідна алгоритмічна мова? Як вона створювалась?
3. Як записується і виконується вказівка повторення?
4. Як записується і виконується вказівка розгалуження?
5. Що таке величина? Які існують види величин?
6. Як записується і виконується вказівка присвоювання?
7. Для чого призначена мова програмування Паскаль?
8. Що таке транслятор? Що таке інтерпретатор?
9. Як ввійти в середовище програмування Turbo Pascal?
10. Як ввійти і вийти з головного меню інтегрованого середовища програмування Паскаль?
11. Що означає термін “виконати команду”?
12. Який порядок створення програми і запису її в файл?
13. Як відредагувати програму яка записана в файл на диск?
14. Як вийти з середовища програмування ?
15. Які основні групи символів складають алфавіт мови Паскаль?
16. Які правила запису ідентифікаторів?
17. Які величини називаються змінними?
18. Як описуються в програмі змінні величини?
19. Які скалярні типи даних використовуються в Паскалі?
20. Як описуються дані цілого типу?

21. Як описуються дані дійсного типу?
22. Як описуються дані літерного типу?
23. Як описуються дані булевого типу?
24. Як описується інтервальний тип?
25. Як описуються дані рядкового типу?
26. Як записується і виконується вказівка введення даних мовою Паскаль?
27. Яка різниця між вказівками Read і Readln?
28. Які форми записів чисел використовуються в мові Паскаль?
29. Як визначається порядок дій в арифметичних виразах, писаних мовою Паскаль?
30. Які знаки арифметичних операцій використовуються для запису виразів мовою Паскаль?
31. Чи можна виконувати операцію ділення над даними цілого типу?
32. Якого типу буде результат ділення 14 на 4?
33. Які службові слова використовуються для запису лінійних алгоритмів?
34. Яка структура програми мовою Паскаль?
35. Як відділяється запис однієї вказівки від іншої?
36. Яким символом закінчується запис кінця програми?
37. Як в програмі необхідно описати підключення модуля для встановлення графічного режиму?
38. Як здійснюється ініціалізація графічного режиму?
39. Який вигляд має екранна система координат дисплея в графічному режимі?
40. З допомогою якої вказівки можна визначити колір точок ліній?
41. З допомогою якої вказівки можна змінювати колір фону?
42. З допомогою якої вказівки можна встановлювати вид штрих кодування?
43. Як здійснюється очищення екрана?
44. Яка різниця між вказівками повторення з передумовою і післяумовою?
45. Як записується і виконується вказівка повторення з параметром?
46. Як організується вихід з вказівок повторення?
47. Як записується і виконується вказівка виведення даних мовою Паскаль?
48. Яка різниця між вказівками Write і Writeln?
49. Як записується і виконується вказівка присвоювання мовою Паскаль?
50. Чи можна в середині тіла вказівки повторення з параметром змінювати параметр циклу?
51. Як записується і виконується вказівка розгалуження в повній формі?
52. Чи можна у вказівці розгалуження використовувати складені вказівки?

53. Які особливості використання символу “;” у вказівці розгалуження?
54. Як описується і для чого призначені мітки?
55. Як записується і виконується вказівка варіанта? Що таке селектор?
56. Для чого використовується вказівка безумовного переходу GO TO?
57. Як відбувається звертання, введення і виведення в одновимірних масивах?
58. Як здійснюється введення і виведення елементів двовимірного масиву?
59. Який принцип використовується під час знаходження мінімального елемента масиву?
60. Який принцип використовується під час сортування елементів масиву “бульбашковим” методом ?
61. Який принцип використовується під час сортування елементів масиву методом знаходження мінімального елементу?
62. Для чого призначенні процедури? Як описується заголовок процедури?
63. Чим відрізняються формальні і фактичні параметри?
64. Чим відрізняються локальні і глобальні змінні?
65. Для чого призначенні функції? Як описується заголовок функції?
66. Яка різниця між процедурою і функцією?
67. Як викликаються процедури і функції?
68. Як описуються рядкові величини? Як описується функція зчеплення?
69. Яка функція визначає довжину рядка?
70. Яка функція здійснює копіювання фрагментів рядка?
71. Яка функція визначає місцезнаходження фрагмента в рядку?
72. Яка процедура здійснює встановлення фрагмента в рядок?
73. Яка процедура здійснює знищення фрагмента рядка?

14 ТИПОВІ ЗАВДАННЯ ДЛЯ ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ

14.1 Задачі з теми “Використання оператора з розгалуженням”

1. Нехай на площині задані два концентричних кола радіусами R і r з центром у точці O(0,0) і точка M(x,y). Визначити до якого кола вона ближча.

2. На площині задано квадрат зі стороною a і вершинами в точках O(0, 0), A(0,a), B(a,a), C(a,0). Усередині квадрата розміщена точка M з координатами x, y. Визначити найменшу відстань від точки до сторони квадрата.

3. Паралелепіпед має розміри a, b, c. Надрукувати площу найбільшої грані.

4. На площині задані чотири точки $M1(X_1,Y_1)$, $M_2(X_2,Y_2)$, $M_3(X_3,Y_3)$, $M_4(X_4,Y_4)$. Вони розташовані так, що кожні три з них можуть бути вершинами трикутника. Знайти площу найменшого з трикутників.

5. На площині задані два концентричних кола радіусами R і r і центром у точці O(x0,y0), яка не обов'язково є початком координат. Визначити, чи попадає точка M(x,y), у середину кільця.

6. На площині розміщені чотири точки $M_1(X_1,Y_1)$, $M_2(X_2,Y_2)$, $M_3(X_3,Y_3)$ і $M_4(X_4,Y_4)$. Вони вершини опуклого чотирикутника. Вивести на друк довжину найменшої діагоналі.

7. Координати вершин чотирикутника $M_1(X_1,Y_1)$, $M_2(X_2,Y_2)$, $M_3(X_3,Y_3)$ і $M_4(X_4,Y_4)$. Визначити чи є чотирикутник паралелограмом.

8. На площині задано коло радіусом R з центром у точці $M_0(x0,y0)$ і пряма ax+by+c=0. Визначити, скільки спільних точок мають пряма і коло:

а) жодної; б) одну; в) дві.

9. На площині задані два кола радіусами R_1 та R_2. їх центри – в точках $M_1(X_1,Y_1)$ і $M_2(X_2,Y_2)$. Визначити, дотикаються вони чи перетинаються, чи не мають жодної спільної точки.

10. Три точки $M_1(X_1,Y_1)$, $M_2(X_2,Y_2)$, і $M_3(X_3,Y_3)$ вершини трикутника. Визначити, розміщена точка $M_4(X_4,Y_4)$ усередині цього трикутника чи ні.

11. Визначити найменшу відстань від кола радіусом R з центром у початку координат до прямої ax+by+c=0. Пряма не має з колом жодної спільної точки.

12. Три прямі взаємно перетинаються: $a_1x+b_1y+c_1=0$, $a_2x+b_2y+c_2=0$, $a_3x+b_3y+c_3=0$, утворюючи на площині трикутник. Визначити координати його вершин і мінімальний кут між сторонами.

13. На площині задано коло радіусом R з центром у точці $M_0(x_0,y_0)$ і три точки – вершини трикутника. Визначити, чи лежить центр кола всередині трикутника.

14. Від точки $M_1(X_1,Y_1)$ до точки $M_2(X_2,Y_2)$, розташованих на площині, можна провести ламані лінії через точку $M_3(X_3,Y_3)$ або $M_4(X_4,Y_4)$. Визначити, яка з цих ламаних ліній має найменшу довжину.

15. На площині задано чотирикутник координатами своїх вершин. Визначити найбільшу з його сторін і надрукувати її довжину.

16. Трикутник на площині задано координатами своїх вершин $A(X_1,Y_1)$, $B(X_2,Y_2)$, $C(X_3,Y_3)$. Визначити, чи трикутник гострокутний.

17. Дано чотири числа a, b, c, d. Визначити, чи є серед них від'ємні.

18. На прямій задані три точки $M_1(X_1,Y_1)$, $M_2(X_2,Y_2)$ і $M_3(X_3,Y_3)$. Визначити, яка з них виявиться розміщеною між тими, що залишилися.

19. На площині задані коло $(x-x_0)^2+(y-y_0)^2=R^2$ і пряма ax+by+c=0, що перетинає коло в двох точках. Визначити, чи розміщені ці точки в одній чверті, чи ні.

20. З'ясувати, чи належить точка (x,y) кільцю радіусами r і R і центром у початку координат.

21. Визначити, в який квадрант координатної площини потрапить точка (x, y).

14.2 Задачі з теми “ Використання циклічних операторів”

1. Протабулювати функцію y=f(x) на відрізку [a,b] з кроком h і знайти найбільше значення.

2. Надрукувати таблицю значень функції y= f(x) на відрізку [a,b] з кроком h до першого від'ємного значення включно. Якщо f(a)<0**,** то таблиця повинна мати один рядок x=a і y=f(a).

3. Задані функція і відрізок змінювання xє[a,b]. Видати на друк тільки два значення x_1 і x_2, для яких x2=x_1+h, $y_1=f(x_1)$ і $y_2=f(x_2)$ мають різні знаки. Якщо таких значень немає, то надрукувати повідомлення про те, що функція на відрізку [a,b] не змінює знак.

4. Задана функція y=f(x). Відомо, що, коли x->∞, то f(x)->0. Надрукувати таблицю значень функції з кроком h, починаючи з x=0 і закінчуючи, коли виконається умова f(x)<=ε, де ε>0 – наперед задана точність.

5. Для функції y=f(x) надрукувати в таблиці тільки ті значення, які задовольняють умовам m<=y<=M (M та m – задані числа). Аргумент змінюється від a до b з кроком h.

6. Відомо, що значення функції y=f(x) у точці x=a від'ємне. Надрукувати таблицю значень функції на відрізку [a,b] з кроком h до того значення аргументу, для якого f(x)>0.

7. Нехай функція y=f(x) наближається до нуля, коли x->∞. Протабулювати функцію з кроком h від a до такого значення x, для якого f(x) потрапляє в ε-оточення нуля.

8. Задані функція y=f(x) і відрізок змінювання аргументу [a,b]. Видати на друк усі пари значень аргументу, в яких функція має різні знаки. Якщо таких значень немає, то надрукувати повідомлення про те, що функція y=f(x) на відрізку [a,b] не змінює знак.

9. Нехай функція y=f(x) періодична. Підрахувати, скільки разів вона перетинає вісь Ox на відрізку [a,b].

10. Нехай функція y=f(x) унімодальна (має один екстремум) на відрізку [a,b]. Методом повного перебору знайти з точністю ε таке значення x, в якому функція досягає екстремуму.

11. Задана функція y=f(x) – неперервна на відрізку [a,b]. Знайти всі локальні екстремуми з точністю ε

12. Для функції y=f(x) визначити на відрізку [a,b] ділянки монотонності.

13. Задана функція y=f(x) неперервна на відрізку [a,b]. Визначити ділянки зростання.

14. Задані дві функції $y_1=f(x_1)$ і $y_2=f(x_2)$. Визначити спільні ділянки зростання цих функцій.

15. Нехай функції $y_1=f(x_1)$ і $y_2=f(x_2)$ неперервні й обмежені на відрізку [a,b]. Задаючись кроком і обчислюючи значення функції у точках визначити найменшу відстань між ними.

16. Відомо, що функції $y_1=f(x_1)$ і $y_2=f(x_2)$ неперервні на відрізку [a,b] і мають одну спільну точку. Визначити методом перебору значення x*, для якого виконується умова $| f_1(x^*)-f_2(x^*) <= \varepsilon$, де є – задана точність.

17. Є дві функції $y_1=f(x_1)$ і $y_2=f(x_2)$, неперервні й обмежені на відрізку [a,b]. Визначити, скільки разів вони перетинаються.

18. Нехай задані функція y=f(x), неперервна на відрізку [a,b], і прямі $y_1=c$ і $y_2=d$. Визначити максимальне відхилення функції y=f(x) від цих точок.

19. Нехай задані дві функції y1=f1(x) і y2=f2(x) неперервні на відрізку [a,b]. Знайти найбільшу відстань між ними.

14.3 Задачі з теми “ Опрацювання табличних величин”

1. Знайти модуль вектора (середнє квадратичне значення елементів одновимірного масиву, що імітує вектор), заданого елементами масиву та середнє арифметичне додатних чисел цього масиву.

2. Знайти суму елементів масиву $y(y_1, y_2,..y_n)$, які більші від заданого числа A.

3. Результати екзамену з "Програмування" записані у масиві $S(S_1,S_2,...,S_n)$. Визначити кількість оцінок "добре".

4. Результати вимірювання n деталей записані у масив $a(a_1,a_2,..,a_n)$. Визначити кількість бракованих, якщо номінальний розмір R, а відхилення не перевищуватиме +5%.

5. Масив $b(b_1,b_2,...,b_n)$ містить у собі вік n чоловік. Визначити кількість людей у віці від 20 до 29 років.

6. Задано масив $y(y_1, y_2….y_n)$. Скласти всі елементи, починаючи з першого, доти, доки сума не перевищить a.

7. Задано масив $a(a_1,a_2,...,a_n)$. У першій половині знайти суму додатних елементів, а в другій – суму від'ємних.

8. У масиві $x(x_1, x_2,...,x_n)$ знайти мінімальний та максимальний елементи і переставити місцями перший з x_n, а другий – з x_1.

9. Знайти скалярний добуток двох n-вимірних векторів $x(x_1, x_2,...,x_n)$ і $y(y_1, y_2,….y_n)$.

10. Задано масив $y(y_1, y_2…y_n)$. Знайти суму всіх елементів масиву, причому елементи з непарними індексами складати з коєфіцієнтом 4, а з парними – з коєфіцієнтом 2.

11. У масиві $a(a_1, a_2…a_n)$ парна кількість елементів. У першій половині знайти кількість елементів, більших за p, а в другій – менших.

12. Задано масив $y(y_1, y_2…y_n)$. Знайти суму всіх елементів масиву, квадрат яких не перевищує числа A.

13. Нехай $y(y_1, y_2…y_n)$ масив значень функції. Знайти суму додатних різниць y_i - A.

14. Задано масив $S(S_1,S_2,...,S_n)$. Визначити суму модулів відхилення S_i від середнього значення.

15. Задані два масива $a(a_1, a_2…a_n)$. і $b(b_1, b_2…b_n)$. Знайти максимальну за модулем різницю між відповідними елементами масивів.

16. Нехай задані масив $y(y_1, y_2 \ldots y_n)$ і число С. Визначити, який з елементів масиву відрізняється від С на мінімальну величину.

17. Задано масив $y(y_1, y_2 \ldots y_n)$, де y – натуральні числа. Знайти найбільше з парних чисел і вказати його місце в масиві.

18. Для масивів $y(y_1, y_2 \ldots y_n)$ і $x(x_1, x_2 \ldots x_n)$ і знайти найбільше значення $x_i + y_i$.

19. У масиві $y(y_1, y_2 \ldots y_n)$ знайти мінімальний та максимальний елементи і переставити місцями перший з x_n, другий – з x_1.

20. У масиві $x(x_1, x_2 \ldots x_n)$ обчислити різницю між x_1 і рештою елементів. Вказати номер того елемента, для якого ця різниці найбільша.

21. Нехай масив $y(y_1, y_2 \ldots y_n)$ не має нульових елементів. Знайти найменше за модулем відношення між сусідніми елементами.

22. У масиві $y(y_1, y_2 \ldots y_n)$ знайти найбільший елемент і вказати його номер. Надрукувати також номери тих елементів, які дорівнюють найбільшому, якщо вони є в масиві.

23. Нехай функції y=f(x) і z=q(x) задані масивами значень при одному й тому ж самому наборі аргументу в n точках $y(y_1, y_2 \ldots y_n)$ і $y(z_1, z_2 \ldots z_n)$. Визначити відстань між функціями ($\min|y_i - z_i|$).

24. У масиві $y(y_1, y_2 \ldots y_n)$ усі числа цілі. Знайти найбільше серед непарних від'ємних.

25. У масиві $y(y_1, y_2 \ldots y_n)$ знайти найменший елемент, і якщо його номер більший за n/2, то в першій половині масиву також відшукати найменше значення, першу половину масиву обнулити.

14. 3. 1 Двовимірні масиви

1. У матриці A(m×n) знайти мінімальний елемент і надрукувати рядок, в якому він розміщений.

2. Дана квадратна матриця B(n×n). Знайти кількість нульових елементів, для яких сума індексів парна.

3. Дана матриця C(m×n). Побудувати масив $y(y_1, y_2 \ldots y_n)$, де y_j – максимальний елемент у j-му стовпці.

4. У квадратній матриці A(n×n) знайти суму додатних елементів над головною діагоналлю.

5. Дана матриця B(m×n) і масив $y(y_1, y_2 \ldots y_n)$. Побудувати масив $x(x_1, x_2 \ldots x_n)$, де x_i дорівнює сумі трьох елементів і-го рядка матриці, які більші за y_i

6. У матриці A(m×n) знайти кількість нульових елементів у кожному стовпці окремо. Результати запам'ятати в масиві $y(y_1, y_2 \ldots y_n)$.

7. Дана матриця B(m×n). Підсумувати елементи першого рядка, починаючи з першого елементу, другий – з другого і т.д. Результати запам'ятати в масиві $y(y_1, y_2 \ldots y_n)$.

8. Дана матриця A(m×n). Знайти окремо суму елементів кожного стовпця і запам'ятати у масиві $y(y_1, y_2 \ldots y_n)$. Причому додавати елементи в стовпці треба тільки до першого від'ємного.

9. У квадратній матриці B(n×n) підрахувати кількість ненульових елементів під головною діагоналлю матриці.

10. Дана матриця A(m×n) і масив $x(x_1, x_2 \ldots x_n)$, де $x_i>0$. Знайти окремо для кожного стовпця суму додатних елементів, але менших за відповідне значення x_i. Результати запам'ятати в масиві $y(y_1, y_2 \ldots y_n)$.

11. Нехай елементи матриці R(m×n) – результат екзаменів з n предметів у групі з m студентів. Обчислити і запам'ятати середній бал для кожного студента.

12. Дана матриця A(m×n)**.** Знайти мінімальний елемент у кожному рядку окремо і серед них вибрати найбільший.

13. У квадратній матриці A(n×n) знайти максимальний елемент над головною діагоналлю та під нею окремо. Надрукувати найменший з них з його індексами.

14. Дана квадратна матриця A(n×n). Визначити, чи вона симетрична. Якщо так, то надрукувати відповідне повідомлення. (Симетричною називається матриця, для елементів якої виконується умова $a_{ij} = a_{ji}$

15. Дана матриця B(m×n). Необхідно додавати всі елементи матриці спочатку по рядках, потім по стовпцях. І в тому, і в іншому випадку припинити цей процес, як тільки накопичена сума перевищить задане число P. Надрукувати обидва результати та кількість доданків у кожній сумі.

16. Нехай матриця A(m×n) складається тільки з цілих чисел. Визначити кількість парних чисел у кожному рядку окремо і запам'ятати в масиві $b(b_1, b_2 \ldots b_n)$.

17. У матриці A(m×n) знайти елемент, який відрізняється на найменшу величину і не перевищує числа P.

18. Дана квадратна матриця A(n×n). Визначити, для скількох елементів не виконується умова $a_{ij}=a_{ji}$.

19. У матриці B(m×n) знайти суму тих елементів, які більші за P/2, де P – максимальний елемент матриці B.

20. Нехай дана дійсна матриця розмірністю A(6×9). Знайдіть середнє арифметичне найбільшого і найменшого значень її елементів, розташованих нижче за головну діагональ.

21. Нехай дана дійсна матриця розмірністю A(18×n). Знайдіть значення найбільшого за модулем елементу матриці і вкажіть його місцеположення в матриці.

22. У даній дійсній квадратній матриці порядку n знайдіть суму елементів рядка, в якій розташований елемент з якнайменшим значенням. Передбачається, що такий елемент єдиний.

23. У даній дійсній матриці розмірністю 6×9 поміняйте місцями рядок, що містить елемент з найбільшим значенням, з рядком, що містить елемент з найменшим значенням. Передбачається, що ці елементи єдині.

24. У даній квадратній цілочисельній матриці порядку 17 вкажіть індекси всіх елементів з найбільшим значенням, що не належать головній і побічній діагоналям.

25. Нехай дана дійсна матриця розмірністю n × m, всі елементи якої різні. У кожному рядку виберіть елемент з найменшим значенням, потім серед цих чисел виберіть найбільше. Вкажіть індекси знайденого елементу.

14.4 Задачі з теми " Використання процедур і функцій"

1. Знайти довжину сторін трикутника ABC, якщо відомі координати його вершин A(xl, yl, zl), B(x2, y2, z2), C(x3, y3,c3). Необхідно скористатись формулою:

$$d = \sqrt{(x1-x2)^2 + (y1-y2)^2 + (z1-z2)^2}\ .$$

2. У масиві a_1, a_2, a_3, ..., a_n є додатні і від'ємні числа. Знайти окремо суму додатних і від'ємних чисел.

3. Скласти програму знаходження більшого з трьох чисел.

4. Скласти програму знаходження площі чотирикутника ABCD, якщо відомо довжини сторін AD, AB, AC, BC, CD. При розв'язуванні задачі використати формулу Герона.

$$P = (a + b + c)/2; \qquad S = \sqrt{P(P-a)(P-b)(P-c)}\ .$$

5. Підрахувати середнє арифметичне додатних елементів масивів A[1..N1], B[1..N2], C[1..N3], якщо N1=6, N2=8, N3 =10.

6. Підрахувати суми додатних елементів кожного рядка для матриць A[1..4, 1..5], B[1..4, 1..5].

7. Підрахувати значення виразу

$$X_{ms} = \frac{X_{m1} + X_{m2}}{2}\ ,$$

де x_{m1} і x_{m2} — найменші елементи масивів X1 [1..5], X2[1..8].

8. Підрахувати кількість нульових елементів для таблиць A[1..4, 1..6], B[1..4, 1..8].

9. Знайти суми елементів головних діагоналей таблиць A[1..5, 1..5], B[1..5, 1..5].

10. Знайти найбільший спільний дільник трьох натуральних чисел a, b, c.

11. Знайти середнє арифметичне значення двох масивів A[1..5], B[1..5] і порівняти їх між собою. Надрукувати більше з них.

12. Знайти в масивах X і Y найбільші елементи і їх порядкові номери X[1..6], Y[1..8].

13. Підрахувати кількість від'ємних елементів кожного стовпця матриць A[1..4, 1..5], B[1..4, 1..5].

14. Дано координати вершин двох трикутників. Визначити, який із них має більшу площу.

15. Знайти найменше спільне кратне двох натуральних чисел M і N з використанням підпрограми знаходження найбільшого спільного дільника двох натуральних чисел.

16. Є відрізки a, в, c, d. Для кожної трійки цих відрізків, з яких потрібно побудувати трикутник, надрукувати площу даного трикутника

(визначити процедуру ПЕЧ_ПЛОЩ, яка друкує площу трикутника зі сторонами x, y, z, якщо такий трикутник існує).

17. Дано дійсні числа x1,y1,x2,y2,… x10,y10. Знайти периметр десятикутника, вершини якого мають відповідно координати (x1,y1), (x2,y2),… (x10,y10). (Визначити процедуру обчислення відстані між двома точками, які задані своїми координатами).

18. За заданими 20-елементними дійсними масивами a, b, c, обчислити:

$$t = \begin{cases} \dfrac{\min(b_i)}{\max(a_i)} + \dfrac{\max(c_i)}{\min(b_i + c_i)} & \textit{при } \min(a_i) < \max(b_i) \\ \min(b_i + c_i) + \min(c_i) & \textit{інакше} \end{cases}$$

19. За заданими дійсними числам c и d(c<d) обчислити:

$$\int_c^d arctg^2 x dx + \int_0 \sin e^{10x} dx .$$

Інтеграли обчислювати наближено за формулою трапеції при n=20 для першого інтеграла і при n=100 для другого :

$$\int_a^b f(x)dx \approx h\left[f(a)/2 + \sum_{i=1}^{n-1} f(a+ih) + f(b)/2 \right],$$

де h=(b-a)/n.

20. В підпрограмі – функції підрахувати значення функції f(x), визначеній за формулою:

$$f(x) = \sum_{n=0}^{\infty} (-1) \frac{x^n}{(n+3)!},$$

при x=3 для всіх значень n від 1 до 15. Для кожного випадку в підпрограмі надрукувати n і відповідне f(x).

21. Скласти програму визначення найбільшого спільного дільника двох чисел, використовуючи рекурсивну процедуру.

22. Дано парне число n>2; перевірити для цього числа гіпотезу Гольдбаха. Ця гіпотеза (по сьогоднішній день не заперечену і повністю не доведену) полягає в тому, що кожне парне n, більше 2, представляється у вигляді суми двох простих чисел. (Визначити процедуру, що дозволяє розпізнати прості числа).

23. Для заданого цілого N підрахувати значення суми

$$\sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \dots \sum_{i_N=1}^{n} \left(\frac{1}{i_1 + i_2 + \dots + i_N} \right)$$

за допомогою рекурсивної функції.

24. У заданій матриці A(n×n) визначити кількість рядків, які впорядковані за збільшенням. Використовуйте підпрограму перевірки впорядкованості рядка.

25. У матриці A(n×n) визначити кількість рядків, елементи яких утворюють арифметичну прогресію. Використовуйте підпрограму перевірки рядка.

26. У заданій матриці A(n×n) знайдіть максимум зі всіх мінімальних елементів матриці по стовпцях.

14.5 Задачі з теми "Робота з рядковими величинами"

1. Дано натуральне n, символи $s_1,s_2,\ldots s_n$. Визначити, скільки раз зустрічається в данній послідовності символів група із трьох точок, які стоять поряд.

2. Відстань між двома словами рівної довжини – це кількість позицій, у яких розрізняються ці слова. У заданому реченні знайти пару найбільш далеко віддалених слів заданої довжини.

3. У реченні всі слова починаються з різних літер. Надрукувати (якщо можна) слова речення в такому порядку, щоб остання літера кожного слова збігалася з першою літерою наступного слова.

4. Надрукувати подане речення таким чином, щоб кожне його слово цілком знаходилося в одному і тому ж рядку роздруківки (тобто позбутися переносів).

5. Відредагувати подане речення, видаляючи з нього слова, що зустрічаються в реченні задану кількість разів.

6. Відредагувати задане речення, видаляючи з нього всі слова з непарними номерами і перевертаючи слова з парними номерами.

Приклад: HOW DO YOU DO → OD OD

7. Дано два речення. Знайти найкоротше зі слів першого речення, якого немає в другому реченні.

8. Знайти найдовше спільне слово двох заданих речень.

9. Задано два тексти; кожен текст складений з попарно різних слів. Визначити, чи можна одержати другий текст із першого видаленням деяких його символів.

10. Знайти множину усіх слів, що зустрічаються в кожному із двох поданих речень.

11. Із заданого тексту вибрати і надрукувати ті символи, що зустрічаються в ньому тільки один раз (у тім порядку, як вони зустрічаються в тексті).

12. Знайти найдовше симетричне слово заданого речення.

13. Для кожного зі слів заданого речення вказати, скільки разів воно зустрічається в реченні.

14. Знайти максимум довжини таких початкових відрізків поданого слова, що мають вигляд U × U, де U – симетричне слово.

15. У рядку символів знайти максимальну послідовність символів, що повторюються.

16. Характеристикою слова назвемо довжину максимальної серії, що міститься в ньому. Упорядкувати слова поданого речення відповідно до росту їхньої характеристики.

17. У поданому реченні знайти пару слів, з яких одне є оберненим до іншого.

18. Перевірити, чи вірно, що в поданому реченні будь-яке несиметричне слово має парну довжину.

19. У поданому реченні вказати слово, в якому частка голосних (A, E, I, O, U) максимальна.

20. Переставити і роздрукувати слова поданого речення відповідно до росту частки приголосних (B, C, D, F, G, H, K, L, M, N, P, Q, R, S, T, V, W, X, Z) у цих словах.

21. Для кожного символа поданого тексту вказати, скільки разів він зустрічається в тексті. Повідомлення про один символ повинне друкуватися не більше одного разу.

22. Замінити закінчення ING кожного слова, що зустрічається в поданому реченні, на ED.Знайти слова, у яких частка літер A і B максимальна.

23. Відредагувати подане речення, видаляючи з нього всі слова, цілком складені з входжень не більше ніж двох літер.

Приклад: АККА KNOBIKISE → KNOBIKISE.

24. Дано текст із 60 літер. Надрукувати цей текст, підкреслюючи (ставлячи мінуси у відповідних позиціях наступного рядка) усі заголовні і рядкові російські літери, що входять в нього.

25. Надрукувати заданий непорожній текст: видаливши з нього всі цифри і подвоївши знаки “+” і “-”; видаливши з нього всі літери *b*, безпосередньо перед якими знаходиться літера *c*.

26. Надрукувати за абеткою всі різні англійські літери, що входять у текст із 10 літер.

27. Дано текст із 80 літер. Надрукувати спочатку всі цифри, що входять у нього, а потім всі інші літери, зберігаючи при цьому взаємне розташування літер у кожній з цих двох груп.

28. Надрукувати заданий непорожній текст:

- видаливши з нього всі знаки “+”, безпосередньо за якими йде цифра;
- замінивши в ньому усі пари *ph* на літеру *f*.

29. Дано ціле число. Потрібно знайти в ньому всі цифри, що не повторюються, і вивести їх у зростаючому порядку.

30. Замінити в слові кожну літеру на нову, віддалену від даної на *m* позицій “униз” за алфавітом.

14.6 Задачі з теми “ Використання записів”

1. Записати в текстовий файл baza_l.pas відомості про автомобілі: номер; тип машини; прізвище, ім'я, по батькові й адресу власника.

Програма у файл Rezyltat.pas повинна вивести відомості про автомобілі, номери яких містять три задані цифри в будь-якому порядку.

2. Підсумки сесії групи студентів запишіть у файл baza_2.pas. Програма у файл rezyltat.pas повинна включити вміст вихідного файлу і список студентів за ознаками: «відмінники», «хорошисти», «невстигаючі», з відповідними коментарями.

3. У файл baza_3.pas запишіть зведення про працівників заводу: прізвище, ім'я, по батькові; дату народження; посаду; число місяців невиплаченої зарплати за минулий рік. Програма у файл rezyltat.pas повинна занести список працівників-пенсіонерів, у яких число неоплачених місяців роботи за минулий рік перевищує 6 місяців.

4. Перерахуйте якості, що характеризують програміста і запропонуйте одиницю вимірювання цих якостей. Опишіть запис, призначений для збереження характеристики програміста.

5. Опишіть масив записів, що містять прізвище абонента і номер його телефону. Заповніть кілька елементів цього масиву прізвищами і телефонами своїх знайомих.

6. Запрограмуйте пошук з порогом у телефонному довіднику.

7. Індексом називається таблиця, що містить значення деяких ключів і їхнє місце розташування в масиві записів. Індексом користуються для прискорення пошуку в масиві. Запрограмуйте процедури:

- складання індексу;
- послідовного пошуку за допомогою індексу.

8. Існує група студентів: A..F.

Male = [A..D]; Female = [E, F];

Aerob = [C, E]; Karate = [C..E].

- хто займається аеробікою і карате ?
- хто з хлопчиків не займається аеробікою ?
- чи є дівчата, що не займаються карате ?

9. Дано непорожню послідовність слів з рядкових українських букв; між сусідніми словами кома, після останнього слова — крапка. Надрукувати за абеткою:

- усі голосні букви, що входять у кожне слово,
- усі приголосні букви, що не входять у жодне слово;
- усі дзвінкі приголосні букви, що входять хоча б в одне слово;

10. Напишіть процедуру, що визначає, чи є рядок:

- іменем, допустимим у Паскалі;
- цілим числом;
- дійсним числом у показовій формі.

11. Організуйте масив записів, який містить інформацію про результати складання іспитів з математики та фізики студентами вашої групи. Видайте на друк прізвища тих, у кого середній бал з цих дисциплін більше 10.

12. Опишіть запис і помістіть у нього такі анкетні дані: прізвище, ім'я, по-батькові, стать, освіту (середня, технічна, вища), адресу (вулиця, номер будинку, номер квартири). Видайте на друк прізвища тих мешканців, які мають вищу освіту і проживають на вулиці Пушкіна, буд. 88.

13. Опишіть запис, який містить інформацію про місце роботи і посаду ваших батьків. Видайте на друк місце роботи в червні 2000 року.

14. Дано 100 цілих чисел від 1 до 50. Вважаючи ці числа множиною, визначити, скільки серед них чисел Фібоначчі і скільки чисел, перша значуща цифра в десятковому записі котрої – 1 чи 2. Числа Фібоначчі – це $a_1=1$, $a_2=1$, $a_3=a_1+a_2$, $a_4=a_2+a_3$ і т. д.

15. Описати логічну функцію path(G, N, K, D), яка визначає, чи є в орієнтованому графі G шлях з вершини N у вершину K, і якщо є, присвоїти параметру D довжину (число дуг) коротшого шляху з N в K. Використати таке подання графу:

type вершина = (b1, b2, b3, b4, b5, b6, b7, b8);
сусіди = set of вершина ;
граф = array[вершина] of сусіди;
G[n] – множина вершин, в які йдуть дуги з вершин X.

16. Описати запис і помістити в нього такі дані мешканців: прізвище, місто, адреса (вулиця, будинок, квартира). Створити масив записів та процедуру „Іронія долі", котра друкує прізвища двох мешканців зі списку C, які живуть у різних містах за однаковою адресою.

17. Описати функцію Sum (A, S1, S2), яка обчислює суму тих елементів матриці A, номери рядків і стовпців котрих належать відповідно непустим множинам S1 і S2 типу *ном*.

```
const n=10;
type номер = 1..n;
матриця = array[номер,ном] of real;
ном = set of номер;
```

18. Дано текст із малих латинських букв, після котрого йде крапка. Надрукувати усі букви, які входять в текст не менше двох раз; усі букви, які входять в текст один раз. Використати множинний тип.

19. Є опис

```
type продукт = (хліб, масло, молоко, м'ясо, риба, сіль, сир, цукор...)
асортимент = set of продукт;
магазини = array[1..20] of асорт.;
```

Описати процедуру наявність (Маг, A, B, C), котра з інформації з масива Маг типу магазин (Маг[i] – множина продуктів в і-му магазині) присвоює параметрам A, B, C, типу асортимент такі значення:

A – множина продуктів, котрі є у всіх магазинах;
B – множина продуктів, кожен з яких є хоча б в одному магазині;
C – множина продуктів, котрих нема ні в одному магазині.

20. Є опис

type ім’я = (Денис, Максим, Юля, Віка, Олена, Женя, Оля...)
гості = set of ім’я;
група = array[ім’я] of гості;

Описати логічну функцію *всюди* (Гр), яка визначає, є в групі Гр хоч би одна людина, яка побувала в гостях у всіх інших з групи (Гр[х] – множина людей, які були в гостях у людини на ім’я х; х є Гр [х]).

21.Задана непуста послідовність символів. Необхідно побудувати і надрукувати множину, елементами якої є літери від “A” до “Z” і цифри від “0” до “5”, які зустрічаються в послідовності.

22.Є набір продуктів – хліб, масло, сир, молоко, які є в асортименті магазину. В 3 магазини привезли різні види цих продуктів. Необхідно побудувати множини A,B,C, які містять відповідно продукти , які є одночасно у всіх магазинах.

23. Є набір продуктів – хліб, масло, сир, молоко, які є в асортименті магазину. В 3 магазини привезли різні види цих продуктів. Необхідно побудувати множини A,B,C, які містять відповідно продукти, які є в крайньому випадку в одному з магазинів.

24. Є набір продуктів – хліб, масло, сир, молоко, які є в асортименті магазину. В 3 магазини привезли різні види цих продуктів. Необхідно побудувати множини A,B,C, які містять відповідно продукти, яких нема в жодному магазині.

25. Є множина різних типів комп’ютерів, які можуть бути в університеті. Відомий набір машин, які є в кожному університеті. Необхідно побудувати і роздрукувати множину, яка містить в собі всі комп’ютери, якими забезпечені всі університети (N=10).

26. Є множина різних типів комп’ютерів, які можуть бути в університеті. Відомий набір машин, які є в кожному університеті. Необхідно побудувати і роздрукувати множину, яка містить в собі всі комп’ютери, які має хоча б один університет (N=10).

14.7 Задачі з теми “Робота з файлами”

1. Нехай є зовнішній файл *КУРС1*, типу *курс*, що містить відомості про студентів першого курсу.

```
Type рядок = packed array [1..12] of char,
   екзамен = (офеом, матем., програм..),
   студент = record ФНО: record прізвище, ім'я, побатькові: рядок
end;
оцінки: array[іспит] of 2..5;
          група: 101..116
end;
курс = file of студент;
```

Написати програму, що залишає у файлі *КУРС 1* відомості тільки про тих студентів, що успішно здали всі іспити, і виводить на друк відомості про

студентів, що мають хоча б одну заборгованість: друкує їхні прізвища і ініціали, номери їхніх груп і кількість незданих іспитів.

2. Є зовнішній текстовий файл ***T*.** Надрукувати перший із найкоротших його рядків.

3. Є зовнішній текстовий файл ***BOOK***. Написати програму, що, ігноруючи вихідний розподіл цього файлу на рядки, переформує його, розбиваючи на рядки так, щоб кожен рядок закінчувався крапкою або містив рівно 60 літер, якщо серед них немає крапки.

4. Надрукувати картинку, яка зображає множення “стовпчиком" двох заданних натуральних чисел. Можливий варіант:

```
      39624
  × 8503
    118872
   198120
  316992
  336922872
```

5. Надрукувати перші 10 рядків “*трикутника Паскаля*” у такому вигляді:

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
.....................
1  9... 126   126 ...  9  1
```

(У цьому “*трикутнику*” крайні числа дорінюють 1, а кожне внутрішнє – додаток двох чисел над ним).

6. Надрукувати таблицю значень функцій ***sin x*** і ***tg x*** на відрізку [0,3] із кроком 0.1. Значення х друкувати з однією цифрою в дробовій частині, значення сінуса – з п'ятьма, а значення тангенса – у експоненційній формі.

7. Описати процедуру ***Lines(t)***, яка порядково друкує вміст непорожнього текстового файлу ***t***, вставляючи в початок кожного рядка, що друкується, її порядковий номер (він повинний займати чотири позиції) і пропуск.

8. У текстовому файлі ***t1*** записана послідовність цілих чисел, розділених пропусками. Описати процедуру positive ***(t1,t2)***, яка записує в текстовий файл ***t2*** усі позитивні числа з ***t1***.

9. Вважаючи, що непорожній текстовий файл ***f*** розбитий на рядки, довжина кожного з яких не перевищує 80 символів, описати процедуру ***преобр(f,f80)***, що доповнюючи короткі рядки файлу ***f*** пропусками праворуч, формує текстовий файл ***f80***, усі рядки в якому мають довжину 80 символів.

10. Описати процедуру ***присв (t1, t2)***, що переписує в текстовий файл ***t1*** вміст текстового файлу ***t2***, але без порожніх рядків.

11. У текстовому файлі записана непорожня послідовність дійсних чисел, розділених проміжками. Описати функцію ***max(t)*** для перебудування найбільшого з цих чисел.

12. Описати процедуру ***присв (t1, t2)***, що переписує в текстовий файл ***t1*** вміст текстового файлу ***t2*** (зі збереженням розподілу на рядки).

13. Описати процедуру ***line 40(t)***, що зчитує з вхідного файлу літери до першої крапки і записує їх (без крапки) у текстовий файл ***t***, формуючи в ньому рядки по 40 літер (в останньому рядку літер може бути і менше).

14. type FI = file of integer;

Нехай у кожному з файлів ***b*** і ***q*** елементи упорядковані за неспаданням. Потрібно злити ці файли в один файл ***h***, також упорядкований за неспаданням. Рішення цієї задачі подати у вигляді процедури ***merge(f, q, h)*** з параметрами типу ***FI***.

15. Описати процедуру ***triangle (t)***, що формує текстовий файл ***t*** з 9 рядків, у першому з якого – одна літера “1”, у другому – дві літери “2”, ..., у дев'ятому – дев'ять літер “9”.

16. Описати функцію, що підраховує кількість порожніх рядків у текстовому файлі ***t***.

17. Описати функцію, що знаходить максимальну довжину рядків текстового файлу ***t***.

18. type FR = file of real;

Описати логічну функцію ***mid (f, m),*** що визначає, чи має файл ***f*** типу ***FR*** непарну довжину, і, якщо має, присвоїти параметру m середній елемент цього файлу

19. Є опис type людина = record
ім'я: packed array [1..9] of char;
вік: 1..99
end;
група = file of людина;

Описати процедуру *НАЙМОЛОДШІ (ГР)*, що друкує імена всіх людей з непорожньої групи *ГР*, що мають найменший вік.

20. Дано текстовий файл ***f***, який містить програму мовою *ПАСКАЛЬ*. Перевірити цю програму на невідповідність кількості круглих дужок, що відкриваються і закриваються. Вважати, що кожен оператор програми займає не більше одного рядка файлу ***f***.

21. Дано текстовий файл ***f***, який містить програму мовою *ПАСКАЛЬ*. Перевірити цю програму на невідповідність числа круглих дужок, що відкриваються і закриваються. Вважати, що кожен оператор програми може займати довільне число рядків файлу ***f***.

22. Дано файл ***f1***, що містить номера телефонів співробітників закладу; де вказується прізвище співробітника, його ініціали і номер телефону. Знайти телефон співробітника за його прізвищем і ініціалами.

23. Відомості про автомобіль складаються з його марки, номера і прізвища власника. Дано файл ***f***, що містить відомості про декілька автомобілів. Знайти:

– прізвища власників автомобілів даної марки;

– кількість автомобілів кожної марки.

24. Дано файл f, що містить різні дати. Кожна дата – це число, місяць і рік. Знайти:

– рік з найменшим номером;

– усі весняні дати.;

– саму пізню дату.

25. Відомості про автомобіль складаються з його марки, номера і прізвища власника. Дано файл f, що містить відомості про декілька автомобілів. Знайти:

– прізвища власників автомобілів даної марки;

– кількість автомобілів кожної марки.

26. Дано файл ***f***, що містить різні дати. Кожна дата – це число, місяць і рік. Знайти:

– рік з найменшим номером;

– усі весняні дати.;

– саму пізню дату.

14.8 Задачі з теми "Елементарна машинна графіка"

1. (6б.) **Поштовий індекс**. Задане шестирозрядне десяткове натуральне число зобразити цифрами за 9-сегментним шаблоном, який використовується при поштовій індексації.

2. (6б.) Зобразити довільний будиночок (дерев'яний або цегляний), відобразити матеріал стін (деревина, цегла) і даху (шифер, черепиця і т. п.). Розміри будиночка, цегли, вікон і т.д. задаються введенням.

3. (5б.) Розподіл швидкості вітру по кожному з восьми напрямів заданий масивом з восьми чисел. Побудувати «розу вітрів» з вказівкою напрямів.

4. (6б.) Для деякої функції $y = f(x,a)$ побудувати сімейство графіків для різних значень параметра a.

5. (6б.) У декартових координатах побудувати сімейство кривих за параметричними рівняннями вигляду $x = \varphi(t,a)$; $y = \psi(t,a)$ при різних значеннях параметра a.

6. (6б.) Побудувати сімейство кривих, заданих таким рівнянням в полярних координатах: $\eta = f(\varphi, a)$ при різних значеннях параметра a.

7. (9б.) Рівняння кривої F(x,y)= 0 не вдалося представити в явному вигляді. Побудувати таку криву.

8. (5б.) Зобразити на екрані шахівницю (разом з буквено-цифровим позначенням горизонталей і вертикалей) і будь-як розставлені на ній шашки.

9. (7б.) Розробити і одержати на екрані рисунок обкладинки якого-

небудь підручника разом з назвою, прізвищами авторів, рисунками, що відображають суть предмета і так далі.

10. (7б.) **Паркет**. Заповнити екран рисунком паркету «ялиночка» з прямокутних дощечок заданого розміру.

11. (6б.) Побудувати фігуру Ліссажу: x = αsin((ωυt + φx); y=βsin(ωυt + φy). Параметри α, β,ωυ, ωy, φx, φy задаються введенням.

12. (6б.) За правилами креслення зобразити проекцію якого-небудь тіла і проставити розміри.

13. (8б.) Для заданого n побудувати (розмістити на екрані) всі n-кінцеві зірки. Наприклад, для n = 13 налічується 5 різних зірок.

14. (6б.) Функція y = f(x) задана таблично у вигляді масивів X(n) і У(n). Крок по x непостійний; масив X впорядкований за збільшенням. Побудувати графік функції, використовуючи лінійну інтерполяцію між точками.

15. (7б.) Заданий масив К(n) розподілу населення за професіями (назва професії і відповідно їй кількість населення). Зобразити розподіл у вигляді кругової (секторної) діаграми з необхідними написами.

16. (8б.) **Кубик Рубіка**. Одержати в аксонометрії або діметрії кубик Рубіка в будь-якому розібраному вигляді. Кольори елементарних кубиків можна зобразити різним штрихуванням або півтонами.

17. (8б.) **Перспектива**. Зобразити вулицю, яка йде в далину, що складається з двох сторін однотипних будинків. Врахувати невидимі частини будівель.

18. (7б.) Заповнити екран колами заданого радіуса, розташувавши їх якомога щільніше, симетрично щодо меж екрана.

19. (8б.) **Алгебра і гармонія**. Розробити орнамент на основі яких-небудь математичних кривих і заповнити ними екран.

20. (7б.) **Павутина**. Одержати на екрані рисунок павутини з центром в довільній (заданій) точці, з довільним числом проміння. Павутина утворена промінням і багатокутниками.

21. (8б.) Множина точок на площині задана своїми координатами. Побудувати в декартових координатах ці крапки і опуклу оболонку множини, тобто багатокутник мінімальної площі, який охоплює всі крапки.

22. (10б.) **Графічний редактор**. Використовуючи курсор-перехрестя, реалізувати можливість ручної побудови фігур за допомогою операцій: задати колір (якщо він є), пересунутися, накреслити відрізок, «гумовий прямокутник», стерти відрізок, заштрихувати (залити область). Код операції задається з клавіатури.

23. (6б.) **Графічна інтеграція**. Для заданого диференціального рівняння вигляду y' = F(x,y) і початкової умови y(0)= y_0 побудувати кусково-лінійний графік рішення y =f(x), обчислюючи y із заданим постійним кроком по x: $y_i = y_{i-1} + h \cdot F(x_{i-1}, y_{i-1})$.

24. (4б.) **Повільне друкування**. Заданий текст друкувати великими буквами з деякими паузами між буквами, супроводжуючи кожну букву

звуковим клацанням. Перенесення здійснювати тільки цілими словами.

25. (7б.) Побудувати мозаїку з правильних шестикутників заданого розміру, зафарбувавши їх різними кольорами або застосувавши різні типи штрихувань (півтони).

26. (8б.) **Рахівниця.** Задане число (не обов'язково ціле) відкласти на бухгалтерських рахівницях, зображених на екрані.

27. (8б.) **Мікрокалькулятор.** Задане число зобразити як на індикаторі мікрокалькулятора, використовуючи для цифр 7-сегментний шаблон.

28. (7б.) Зобразити «ріг достатку», що є стилізацією закрученого баранячого рогу. При побудові корисно використовувати математичні криві (спіралі, еліпси і т. д.).

29. (7б.) **Соняшник.** Рисунок на капелюшку соняшника є сімейством логарифмічних спіралей, закручених в різні боки. Одержати такий рисунок. Між іншим, кількість «правих» і «лівих» спіралей є два сусідні числа Фібоначчі.

30. (8б.) Зобразити на екрані достатньо складну квітку (жоржина, ромашка з будь-яким числом пелюсток, калина, волошка і так далі).

31. (7б.) **Дерево**. Для заданого n побудувати двійкове дерево, що містить n рівнів (наприклад, генеалогічне).

32. (6б) **Луска.** Заповнити екран рисунком риб'ячої луски із заданим розміром елементарних лусочок.

33. (7б.) **Піраміда**. Однакові труби в кількості n штук укладені можливо компактніше — пірамідою. Одержати на екрані вигляд піраміди з торця для довільного числа n.

34. (7б.) **Сходи**. Зобразити на екрані сходи якої-небудь рослини, посадженої квадратно-гніздовим способом на грядці або на полі. Врахувати перспективу.

35. (8б.) Зобразити в зацепленні дві шестерні (зубчатих колеса) якого-небудь механізму; діаметри шестерень і кількість зубів задаються. Нехай n шестерень зацеплені.

14.9 Задачі з теми “Елементи комп'ютерної мультиплікації ”

1. (9б.) **Круги на воді.** Екран зображає басейн з водою, в який кинули камінь (у заданих координатах). Від каменя пішли круги, які, дійшовши до стінок басейну, відбиваються від них. Реалізувати цю динамічну картину.

(13б.) **Розвиток задачі**. Ефект «млинчиків» — відскоків каменя від поверхні води з подальшими падіннями.

2. (8б.) **Листівка**. Розробити і реалізувати на екрані вітальну листівку до свята (Новий рік, день народження і так далі) з динамічними елементами (миготлива гірлянда, салют, свічки, що горять, миготливі написи і так далі).

3. (8б.) **Плакат**. Розробити і реалізувати на екрані політичний або екологічний плакат з динамічними (рухомими або миготливими) ефектами.

4. (8б.) **Телезаставка**. Розробити і реалізувати на екрані заставку до однієї з популярних телепрограм («Поле чудес», «КВК» і так далі).

5. (8б.) **Вітряний млин.** Зобразити на екрані працюючий вітряний млин. Площина обертання крил необов'язково паралельно площині екрана.

6. (8б.) **Електронний годинник.** Зобразити на екрані працюючий електронний годинник з цифровим індикатором (кожна цифра зображається на 7-сегментному шаблоні). При недоступності вбудованого таймера ЕОМ реалізувати його за допомогою набудованих циклів, задаючи стартовий час при запуску програми. Ввести індикацію, дату, дня тижня.

7. (7б.) **Стрілочний годинник**. Зобразити на екрані працюючий годинник із стрілочним індикатором (3 стрілки). За відсутності таймера — імітувати його. Індикація у вікні дати і дня тижня.

8. (8б.) **Пісочний годинник.** Зобразити на екрані діючий пісочний годинник. Врахувати закони фізики: кількість піску, витікаючого з верхньої колби в одиницю часу, постійна і рівна кількості піску, що притікає в нижню колбу. Установка часу перетікання піску виробляється при запуску програми.

9. (8б.) **Більярд**. На екрані — зображення більярдного столу з однією кулею. Задається напрям і початкова швидкість кулі після удару києм. Реалізувати рух кулі після удару до попадання його в лузу або до переривання з клавіатури. Врахувати силу тертя. Можна висвічувати слід (траєкторію) кулі.

10. (8б.) **Футбольний м'яч**. Зобразити рух футбольного м'яча після удару (задається початкове положення м'яча і вектор швидкості). В процесі польоту м'яч ударяється об підлогу, стелю і стіни, втрачаючи при кожному ударі частину енергії. Врахувати опір повітря. Можна висвічувати траєкторію м'яча.

11. (6б.) **Броунівський рух**. Частинка (від заданої початкової точки) хаотично рухається у будь-якому напрямі на будь-яку відстань (в межах екрана). Одержати на екрані траєкторію руху частинки до переривання з клавіатури.

(10б.) **Розвиток задачі**. n частинок хаотично рухаються в просторі, обмеженому розмірами екрана (на кожному кроці — у будь-якому напрямі на будь-яку відстань). Удари частинок одна об одну (при перетині траєкторій) і об стінки екрана вважати абсолютно пружними. Побудувати траєкторії руху частинок (для кожної частинки — свій колір).

12. (6б.) **П'яниця**. У будь-яких точках місцевості розташовані декілька стовпів, деякі з них з'єднані парканами. П'яниця з рівною вірогідністю робить крок вперед, назад, вперед — управо або вперед — вліво (під 45^0). Натрапивши на стовп або паркан, він падає (на деякий час). Зобразити траєкторію його руху.

13. (7б.) Зобразити у дії кривошипно-шатунний механізм парового двигуна або двигуна внутрішнього згорання.

14. (7б.) **Орнамент з квадратів**. Побудувати квадрат заданого

розміру. Кожну сторону розділити в заданому відношенні m : n; одержані точки суть вершини нового квадрата. І так далі до заповнення середини квадрата. Заповнити такими квадратами весь екран.

15. (7б.) **Орнамент з трикутників.** Як в попередній задачі заповнити екран орнаментом з правильних трикутників.

16. (7б.) **Наперстовик**. Комп'ютер в ролі ведучого, користувач — «лоха». На екрані — 3 наперстки, під одним з них — кулька (на старті її видно). Ведучий в заданому темпі міняє місцями два наугад вибраних наперстка; після закінчення процесу користувач повинен вгадати, де кулька.

17. (6б.) **Конвейєр**. Зобразити діючий конвейєр, що транспортує які-небудь однотипні предмети.

18. (8б.) **Кипляча рідина**. Екран — судина з киплячою рідиною. На дні у будь-якій точці утворюється бульбашка; при русі вгору вона росте, а дійшовши до поверхні — лопається. Якщо дві бульбашки стикаються, вони зливаються в одну. Реалізувати цей процес.

19. (8б.) **Кинута палиця**. Відома кутова швидкість обертання і вектор почат-кової лінійної швидкості кинутої палиці. Зобразити її в русі до падіння. Врахувати опір повітря і відскоки від меж екрана.

20. (12б.) На екрані — мішень (спортивна або армійська), що містить 10 концентричних кілець з відповідними окулярами. У довільний момент або за натисненням клавіші відбувається «постріл», в результаті якого на мішені з'являється пробоїна. Процес триває до вичерпання запасу патронів або до натиснення клавіші; підраховується число окулярів. Користувач може задати параметри випадкового розподілу. Точка прицілу (перехрестя прицілу) задається з клавіатури або мишею, приціл спочатку «збитий»; можна дати декілька пристрілювальних патронів.

21. (12б.) **Кубик, що обертається**. Зобразити в русі кубик заданого розміру, що рівномірно обертається навколо вертикальної осі і, обертаючись, віддаляється в нескінченність.

22. (10б.) **Затемнення місяця.** Зобразити на екрані зоряне небо, повний місяць і тінь Землі, що поволі насувається на неї; потім — повільне відкриття диска Місяця. На зоряному небі — Чумацький шлях, падаючі зірки, штучні супутники.

23. (10 б.) **Сутінки**. Зобразити на екрані довільний пейзаж, натюрморт або інтер'єр. Потім будь-якими точками або прямими заповнювати екран до повного зникнення картини (зручніше реалізувати ефект управлінням палітрою). Зворотний процес: світанок або проява фотозображення.

24. (8б.) **Калейдоскоп.** Побудувати в центрі екрана трикутник заданого розміру і заповнити його довільним (жорстко заданим, будь-яким або задається з клавіатури) зображенням. Виробити багатократне дзеркальне віддзеркалення зображення від кожної сторони трикутника до заповнення всього екрана.

25. (12б.) **Паровоз**. Одержати на екрані картину, яку бачить машиніст

рухомого потягу: рейки, шпали, стовпи, придорожні будови і так далі. Врахувати повороти, стрілки, зміну швидкості потягу, зустрічні склади і так далі.

26. (7б.) **Завіса.** Зобразити фінальну сцену якого-небудь театрального дійства на екрані: довільне зображення; справа і зліва на нього насувається завіса. На завісі — напис: «КІНЕЦЬ».

27. (13б.) **Атом.** Зобразити модель атома довільного хімічного елементу: ядро і електрони, що обертаються по своїх орбітах. Розподіл електронів по орбітах задається.. У підготовленому файлі зберігається розподіл електронів по орбітах для всієї системи Менделєєва; користувач задає тільки номер або позначення хімічного елементу.

28. (9б.) **Маятник.** Одержати зображення рухомого математичного маятника. Довжина маятника і початкове положення задаються. Врахувати опір повітря; замість маятника зобразити гойдалки.

29. (7б.) **Прапор**. Зобразити прапор, що розвівається на вітрі (наприклад український двокольоровий).

30. (7б.) **Реклама**. Розробити і реалізувати динамічну рекламу якого-небудь товару або послуги.

31. (9б.) **Рахівниці**. Зобразити бухгалтерські рахівниці і реалізувати на них тренажер, що демонструє операції додавання і віднімання. Числа і знак операції вводяться з клавіатури. Передбачити затримку для наочності порозрядних операцій.

32. (9б.) **Салют.** Реалізувати на екрані картину святкового салюту: зліт, розриви, падіння піротехнічних ракет і їх осколків (з декількох стовбурів). Світлові ефекти бажано супроводжувати звуковими.

33. (8б.) **Місяць-Місяцевич.** Цикл руху Місяця по небосхилу складає 28 діб; орбіту руху можна вважати постійною. Реалізувати в прискореному темпі процеси руху, народження, зростання і «старіння» Місяця на фоні зоряного неба.

Література

1. Абрамов З. Л., Гнезділова Р. Р., Капустіна Е. Н., Селюн М. І. Задачі з програмування. — М.: Наука, 1988. – 224с.
2 Абрамов З. А, Зима Е. В. Начала інформатики. М.: Наука, 1989.- 256 с.
3. Брудно А. Л., Каплан Л. І. Олімпіади з програмування для школярів/Під ред. Б.І. Наумова, — М.: Наука, 1985. –96 с.
4. Бухтіяров А. М., Фролов Г. Д. Збірник задач з програмування на алгоритмічних мовах. — М.: Наука, 1974. –240с.
5. Ван Тассел Д. Стиль, розробка, ефективність і випробування програм:/Пер. з англ. — М.: Мир, 1981. –320 с.
6. Гудман З., Хидетниеми С. Введение в разработку и анализ алгоритмов: Пер. з англ. — М.: Мир, 1981.-368 с.
7. Джонс Ж., Харроу К. Решение задач в системе Турбо Паскаль. — М.: Финанси и статистика, 1991. –720 с.
8. Джонстон Г. Учитесь программировать. — М.: Финанси и статистика, 1989. -368 с.
9. Дробушевіч Г. А, Збірка задач з програмування: Навч. посібн. — Мн.: Висш. школа, 1983. –324 с.
10. Дьяконов В. П. Справочник по алгоритмам и програмам на языке Бейсик для персональних ЭВМ. — М.: Наука, 1987. – 240 с.
11. Касьянов В. Н., Сабельфельд В. К. Збірник вправ по практикуму на ЕОМ; Навчальний посібник для вузів. — М.: Наука, 1986. –272 с.
12. Кнут Д. Мистецтво програмування на ЕОМ. Т1. Основні алгоритми. — М.: Мир, 1976. –736 с.
13. Кнут Д. Мистецтво програмування на ЕОМ. Т. 3. Сортування і пошук. — М,: Мир, 1978. –846 с.
14. Котов Ю. В. Як рисує машина, — М.: Наука, 1988.224 с.
15. Кушніренко А. Р., Лебедев Р. В. Программування для математиків. — М.: Наука, 1988. – 384 с.
16. Основи інформатики і обчислювальної техніки, Ч. 2.
Під ред. Л. П. Ершова, В. М. Монахова. М.: Просвіта, 1986. –143с
17. Пильщиків В. Я. Збірник вправ мовою Паскаль. –.М.: Наука, 1989. – 160с.
18. Збірка задач з базової комп'ютерної підготовки: Навчальний посібник. В. С. Зубов, І. Н. Котарова, О. Г. Архипов і ін.; Під редакцією І. Н. Котарової. — М.: Видавництво МЕІ, 1998. –178 с.
19. Светозарова Г. І., Сигитов Е. В., Козловській А. В. Практикум з програмування на алгоритмічних мовах. - М.: Наука, 1980. –320 с.
20. Ставровській А. Б. Турбо Паскаль 7.0: Підручник. — К.: Видавнича група BHV, 2000. –400 с.

Навчальне видання

Людмила Михайлівна Круподьорова

Алгоритмізація і основи програмування

Навчальний посібник

Оригінал-макет підготовлено Круподьоровою Л.М.

Редактор О.Д.Скалоцька

Навчально-методичний відділ ВНТУ
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м. Вінниця, Хмельницьке шосе, 95, ВНТУ

Підписано до друку
Формат 29,7х42 ¼
Друк різографічний
Тираж прим.
Зам. №

Гарнітура Times New Roman
Папір офсетний
Ум. друк.арк.

Віддруковано в комп'ютерному інформаційно-видавничому центрі
Вінницького національного технічного університету
Свідоцтво Держкомінформу України
серія ДК № 746 від 25.12.2001
21021, м.Вінниця, Хмельницьке шосе, 95, ВНТУ