УПРАВЛІННЯ ОСВІТИ І НАУКИ
ВОЛИНСЬКОЇ ОБЛАСНОЇ ДЕРЖАВНОЇ АДМІНІСТРАЦІЇ

ЛУЦЬКИЙ ПЕДАГОГІЧНИЙ КОЛЕДЖ

Циклова комісія викладачів фізико-математичних дисциплін

Навчально-методичне забезпечення

дисципліни

«Програмування з методикою навчання»

(Частина ІІ)

Навчально-методичний посібник

Луцьк – 2016

УДК 378.147:004.41
ББК 74.263.2
Н 15

Навчально-методичне забезпечення дисципліни «Програмування з методикою навчання» (Частина ІІ) : навч.-метод. посіб. / [укл. Т. Г. Четверикова] – Луцьк, 2016. – 100 с.

Автор-укладач: Четверикова Т. Г.

Рецензенти:

Гайдай С. І., кандидат фізико-математичних наук, доцент кафедри прикладної математики та інформатики Східноєвропейського національного університету імені Лесі Українки

Собчук О. М., кандидат педагогічних наук, доцент кафедри прикладної математики та інформатики Східноєвропейського національного університету імені Лесі Українки

Розглянуто на засіданні циклової комісії викладачів фізико-математичних дисциплін і рекомендовано до друку (16.11.2015, протокол № 3)

Рекомендовано до друку науково-методичною радою Луцького педагогічного коледжу (14.01.2016 р., протокол № 3)

Навчально-методичний посібник містить методичні матеріали з дисципліни «Програмування з методикою навчання»: тексти лекцій, розробки практичних занять, матеріали для організації самостійної позааудиторної роботи студентів, перелік питань для підготовки до модульних контрольних робіт та семестрового екзамену.

Матеріали, подані у посібнику, стануть у нагоді студентам педагогічних коледжів, які здобувають освіту за спеціальністю «Початкова освіта» з додатковою кваліфікацією «Вчитель інформатики в початковій школі».

УДК 378.147:004.41
ББК 74.263.2
Н 15

Зміст

Передмова

Невід’ємною складовою інформаційної культури сучасного студента – майбутнього вчителя інформатики в початковій школі є формування алгоритмічного мислення та вмінь використовувати програмне забезпечення для створення нових програмних продуктів, розв’язування задач користувача.

Метою вивчення дисципліни «Програмування з методикою навчання» є ознайомлення студентів з основними принципами технології програмування та оволодіння основами мов програмування, методами проектування та створення програм згідно сучасних технологій програмування, формування фундаментальної теоретичної бази, вироблення практичних навичок свідомого і раціонального використання комп'ютерів у повсякденній практичній діяльності, формування та розвиток просторового мислення, що є одним з важливих показників інтелектуального розвитку.

Вивчення курсу "Програмування з методикою навчання" передбачає отримання базових знань, умінь і навичок візуального та структурного програмування при підготовці учителя початкових класів з додатковою кваліфікацією «Вчитель інформатики в початковій школі».

Мета вивчення дисципліни досягається через практичне оволодіння студентами навичками роботи з основами технології розв'язування задач за допомогою комп'ютера, починаючи від їх постановки й побудови відповідних інформаційних моделей і закінчуючи практичною реалізацією в певному середовищі програмування.

Основними завданнями вивчення дисципліни "Програмування з методикою навчання" є:

- ознайомлення з ідеологією розробки програм засобами програмування;
- ознайомлення з середовищем розробки програм засобами програмування;

- сполучення традиційних навичок програмування з новими засобами;
- опанування основних засобів створення програм засобами візуального об’єктно-орієнтованого програмування.

У результаті вивчення курсу студенти повинні набути базові знання, на основі котрих у межах дисциплін педагогічного та психологічного циклів, що вивчаються у наступних семестрах, будуть формуватися такі вміння й навички роботи з інформацією за допомогою комп’ютера й інформаційно-комунікаційних технологій (ІКТ), які дозволяють у подальшому всебічно, усвідомлено й ефективно використовувати комп’ютер і засоби ІКТ у своїй професійній діяльності.

Пропонований навчально-методичний посібник містить методичні матеріали згідно програми курсу: лекції, розробки практичних занять, матеріали для організації самостійної поза аудиторної роботи студентів, перелік питань для підготовки до модульних контрольних робіт з дисципліни «Програмування з методикою навчання» (II, III семестр). Матеріали, подані у посібнику, стануть у нагоді студентам педагогічних коледжів, які здобувають освіту за спеціальністю «Початкова освіта» з додатковою кваліфікацією «Вчитель інформатики в початковій школі».

Тематичний план з дисципліни «Програмування з методикою навчання»

№ з/п	*Тема заняття*	*К-ть год.*	*Тип заняття*
Змістовий модуль 3. Програмування розв'язку математичних задач			
1, 2	Опис вказівок розгалуження мовою програмування PASCAL.	2	лекція
3, 4	Створення та реалізація програм із розгалуженням.	2	практична робота
5, 6	Створення та реалізація програм із розгалуженням. Команда вибору.	2	практична робота
срс	Оператори розгалуження та вибору.	2	
7, 8	Вказівка повторення з параметром.	2	лекція
9, 10	Створення програм з використанням циклу з параметром.	2	практична робота
срс	Цикл з параметром.	2	
11, 12	Вказівки повторення з післяумовою і передумовою.	2	лекція
13, 14	Створення програм з використанням циклу з передумовою і післяумовою.	2	практична робота
срс	Оператори циклу.	2	
15, 16	Розв'язування задач із застосуванням команд циклів і розгалужень.	2	практична робота
17, 18	Модульна контрольна робота.	2	
Змістовий модуль 4. Використання можливостей мови Паскаль для побудови різних структур			
19, 20	Рядкові величини. Операції над рядковими величинами. Алгоритми роботи з рядками.	2	лекція
21, 22	Створення програм опрацювання рядкових величин.	2	практична робота
23, 24	Створення програм опрацювання рядкових величин.	2	практична робота
срс	Рядкові величини. Стандартні функції роботи з рядками.	2	

№ з/п	*Тема заняття*	*К-ть год.*	*Тип заняття*
25, 26	Табличні величини. Алгоритми роботи з табличними величинами. Одновимірний масив.	2	лекція
27, 28	Створення програм на опрацювання одновимірних масивів.	2	практична робота
срс	Одновимірний масив.	2	
29, 30	Табличні величини. Алгоритми роботи з табличними величинами. Двовимірний масив.	2	лекція
31-34	Створення програм на опрацювання двовимірних масивів.	4	практична робота
35, 36	Алгоритми сортування і пошуку елементів масивів.	2	лекція
37, 38	Використання алгоритмів сортування.	2	практична робота
срс	Табличні величини.	2	
39, 40	Модульна контрольна робота.	2	
Змістовий модуль 5. Підпрограми. Застосування графіки в Паскаль			
1, 2	Опис мовою Паскаль вказівок звернення до процедур.	2	лекція
3, 4	Опис мовою Паскаль вказівок звернення до функцій.	2	лекція
5, 6	Підпрограми. Використання процедур і функцій.	2	практична робота
срс	Підпрограми в мові Паскаль	2	
7, 8	Стандартні графічні процедури і функції в Паскаль.	2	лекція
9, 10	Створення графічних зображень.	2	практична робота
срс	Елементи комп’ютерної графіки.	2	
11, 12	Анімація в Паскаль. Створення рухомих зображень.	2	практична робота
13, 14	Модульна контрольна робота.	2	

Лекція. Опис вказівок розгалуження мовою програмування PASCAL. Оператор вибору

Мета, завдання лекції: ознайомлення із правилами запису команд вказівок розгалуження і вибору, їх використання при розв'язуванні задач.

План і організаційна структура лекції:

1. Вказівка розгалуження, її форми.
2. Оператор вибору.
3. Приклади розв'язування задач.

Зміст лекційного матеріалу:

Крім арифметичних виразів, у Паскаль існує ще один тип виразів – логічний.

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

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

Простий логічний вираз – це два арифметичні вирази, з'єднані операцією порівняння.

Операції порівняння записують так:

> – більше,

< – менше,

>= – більше або дорівнює (не менше),

<= – менше або дорівнює (не більше),

= – дорівнює,

<> – не дорівнює.

Наприклад, простими є умови: a < 5; c >= a; a < b.

Складена умова записується з допомогою логічних операцій: *and* – і, *or* – або, *not* – не.

Наприклад, складеними є умови: (a>5) and (a<10); (a=2) or (b=3).

Результатом виразу умови завжди буде величина булевого типу *true* (істина) або *false* (хиба). *True* і *false* називаються *логічними сталими*.

Приклад. Нехай змінна x = 0, а y = 2. Які значення мають такі прості логічні вирази:

а) x < 5; б) 5 > 7; в) y = 2; г) x + y >= 2 * x; д) x <> y.

Відповідь: логічні вирази а, в, г, д мають значення true, а вираз б – false.

Визначимо, які значення виразів а, в, г, д, якщо x = 2, а y = 0?

Відповідь: а) true, б) false, в) false, г) false, д) true.

Розглянемо приклади типових задач, які ведуть до використання умов

і складання логічних виразів:

а) серед можливих значень деякої функції вибрати лише додатні або від'ємні;

б) серед учнів школи вибрати лише тих, чий зріст понад 165 см;

в) вивести на екран список учнів, які отримали «9» з інформатики;

г) для поїздки на екскурсію вибрати лише тих учнів, які мають оцінки «9» — «12» з математики.

Щоб записати логічний вираз, потрібно ввести імена для величин, наприклад, такі: а) y; б) rist; в) oc_inf; г) oc_mat. Логічні вирази для задач а) — г) матимуть такий вигляд:

а) y > 0 (або y < 0); б) rist > 165; в) oc_inf = 9; г) oc_mat >= 9.

Логічні вирази є складовими вказівки розгалуження в мові програмування Паскаль.

Вказівка розгалуження може записуватись в повній або скороченій формі.

Повна форма вказівки розгалуження

if <умова>
then <вказівка1>
else <вказівка2>;

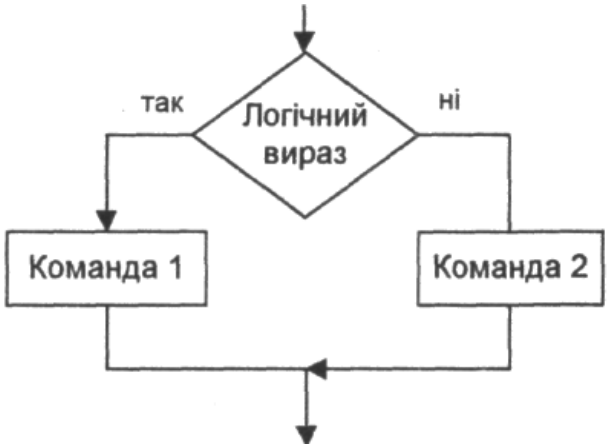


Рис. Блок-схема алгоритму повної форми вказівки розгалуження

Повна форма вказівки розгалуження виконується так:

- якщо умова істинна, то виконується вказівка 1, а потім вказівка, яка знаходиться в програмі після всієї вказівки розгалуження;
- якщо умова хибна, то виконується вказівка 2, а потім вказівка, яка знаходиться в програмі після всієї вказівки розгалуження.

Скорочена форма вказівки розгалуження

if <умова>
then <вказівка1>;

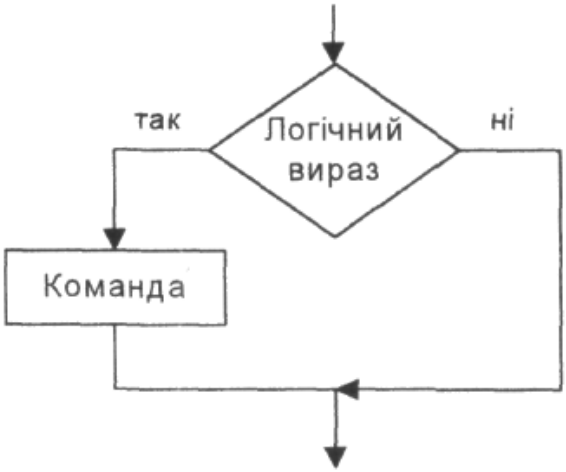


Рис. Блок-схема алгоритму скороченої форми вказівки розгалуження

Скорочена форма вказівки розгалуження виконується так:

- якщо умова істинна, то виконується вказівка 1, а потім вказівка, яка знаходиться в програмі після всієї вказівки розгалуження;
- якщо умова хибна, то виконується вказівка, яка знаходиться в програмі після вказівки розгалуження.

Задача (про два числа). Скласти програму, де у змінні *a* та *b* треба ввести два цілих числа. Третій змінній (*c*) присвоїти значення більшого з-поміж уведених. Якщо числа однакові, то змінній *c* присвоїти значення будь-якого з чисел.

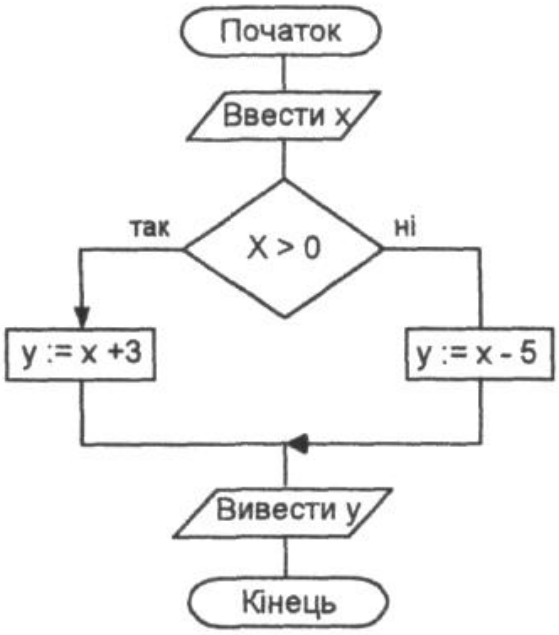
```
program
var a, b, c : integer;
begin
   write ('Введіть два числа: '); readln (a, b);
   if a > b then c : =a;
   if a < b then c := b;
   if a = b then c := a;
   writeln (‘c= ’, c);
   readln
end.
```

Виконаємо програму тричі для таких значень a і b: 1) 7 і 5; 2)5 і 8; 3) 4 і 4.

На екрані відповідно отримаємо: 1) c=7; 2) c=8; 3) c=4.

Задача. (про складену функцію). Увести будь-яке значення *x* й обчислити значення складеної функції y, яка задана формулою

$$y = \begin{cases} x+3, якщо\_x > 0, \\ x-5, якщо\_x \le 0. \end{cases}$$




```
program SF;
var x, y : real;
begin
    write ('Введіть x: '); readln (x);
    if x > 0 then y: = x+3 else y: = x-5;
    writeln (‘y = ’, y)
end.
```

Виконаємо програму для x = 2. Отримаємо y =5. Надамо x значення -2, тоді y = -7.

Задача про квадратне рівняння. Розв'язати квадратне рівняння $ax^2 + bx + c = 0$, *де* $a \neq 0$.

```
program Zad1;
var a, b, c, x1, x2, d: real;
begin
     write ('Введіть коефіцієнти: '); readln (a, b, c);
     d := b * b – 4 * a * c;
     if d < 0 then  writeln(‘Коренів немає’) else
          begin
               x1 := (-b + sqrt (d))/(2 * a);
               x2 := (-b – sqrt (d))/(2 * a);
               writeln (‘Корені є’, x1 : 6 : 2, x2 : 6 : 2)
          end
end.
```

Вказівка вибору

У випадку, коли треба використовувати декілька вказівок

розгалуження, вкладених одна в одну, використовують команду вибору CASE:

case <вираз> **of**
    <список1>:оператор1;
    <список2>:<оператор2>;
    . . .
    <список N>:<оператор N>
else <команда>
end;

Вираз, який записується між службовими словами **case** і **of**, називається *селектором*. Селектор повинен належати до одного з цілого, булевого, символьного типів або типу користувача. Дійсні і рядкові типи в селекторах використовувати не можна.

Списки варіантів – константи, типи яких повинні співпадати з типом селектора. Всі мітки варіантів повинні бути різними. Список може складатися з одного чи декількох елементів, перерахованих через кому (1, 2, 3, 3) або розділених двокрапкою (1:4).

Вказівка вибору виконується так: спочатку обчислюється значення виразу селектора, а тоді виконується та вказівка, перед якою стоїть стала вибору, яка відповідає значенню селектора. Якщо жодна з констант не дорівнює поточному значенню селектора, то виконується оператор, що стоїть після вказівки вибору (після слова end).

Приклад. Скласти програму, яка визначає пору року за порядковим номером місяця.

```
program PORA;
var MONTH: integer;
begin
    write (‘Введіть порядковий номер місяця’);
    readln (MONTH);
    if  (MONTH <1) or (MONTH >12)
    then writeln (‘Такого місяця не існує’)
    else
        case MONTH of
            3, 4, 5: writeln(‘весна’);
            6, 7, 8: writeln(‘літо’);
            9, 10, 11: writeln(‘осінь’);
            12, 1, 2: writeln(‘зима’)
        end;
end.
```

Приклад. Змоделювати роботу автомата з продажу квитків. Нехай населені пункти (k) позначені номерами 1, 2, 3, 4, 5, 6, 7, 8. Вартість одного квитка ($cina$) до конкретного пункту визначена так:

$$cina = \begin{cases} 22{,}00 \text{ грн.}, \text{якщо } k = 1, \\ 35{,}00 \text{ грн.}, \text{якщо } k = 2, 3, \\ 50{,}50 \text{ грн.}, \text{якщо } k = 4, \\ 72{,}30 \text{ грн.}, \text{якщо } k = 5, 6, \\ 97{,}00 \text{ грн. до інших пунктів} \end{cases}$$

Скільки коштуватимуть *m* квитків до деякого населеного пункту?

```
program Kvytky;
var k, m: integer;
    cina: real;
begin
    write ('Введіть номер населеного пункту'); readln (k);
    write ('Введіть кількість квитків'); readln (m);
    case k of
        1 : cina := 22.00;
        2,3 : cina := 35.00;
        4 : cina := 50.50;
        5..6: cina := 72.30
    else  cina := 97.00;
    end;
writeln (m, 'квитків до пункту', k 'коштують', m*cina)
end.
```

Практична робота. Створення та реалізація програм із розгалуженням

Мета: формування навичок складання програм розв'язування задач з використанням вказівки розгалуження.

Завдання:

- *студенти повинні знати:* структуру та принцип дії повної і неповної вказівки розгалуження;
- *студенти повинні вміти:* виконувати програму з розгалуженням в середовищі програмування.

Обладнання: зошит, середовище програмування.

Хід виконання:

Завдання 1. Скласти програму обчислення функції. Реалізувати для заданих значень змінних.

№з/п	*Функція*	*Умова*	*Значення змінних*
Варіант 1	$y=\begin{cases} x^2+1, \\ x-4 \end{cases}$	x < 3; x >= 3	x_1 = -4; x_2 = 5; x_3 = 3
Варіант 2	$y=\begin{cases} cosx+1, \\ 2-\sin x \end{cases}$	x <= 0; x > 0	x_1= 0,1; x_2 = 0; x_3 = - 3
Варіант 3	$y=\begin{cases} 2x+5{,}6, \\ \sqrt{x+4} \end{cases}$	x < 5; x >= 5	x_1 = 21, x_2 = 0; x_3 = 5
Варіант 4	$y=\begin{cases} 7{,}8-x^3, \\ 5x-3 \end{cases}$	x <= 8; x > 8	x_1 = 10; x_2 = 1; x_3 = - 3
Варіант 5	$y=\begin{cases} 25x, \\ -4+2{,}8x \end{cases}$	x < -4; x >= -4	x_1= - 7; x_2 = - 1; x_3 = 9
Варіант 6	$y=\begin{cases} 1+\sin x, \\ 1-\cos x \end{cases}$	x <= 6; x > 6	x_1 = 1; x_2 = 0; x_3 = - 1
Варіант 7	$y=\begin{cases} tgx+2, \\ ctgx-3 \end{cases}$	x < -1; x >= -1	x_1 = 1; x_2 = - 1; x_3 = 0
Варіант 8	$y=\begin{cases} \cos 2x, \\ tgx \end{cases}$	x <= 1; x > 1	x_1 = 0,5; x_2 = 0; x_3 = - 0,5

Завдання 2. Скласти програму обчислення функції. Реалізувати для заданих значень змінних.

№з/п	*Функція*	*Умова*	*Значення змінних*
Варіант 1	$y=\begin{cases} x^2-\frac{7}{x^2}; \\ ax+7\sqrt{x}; \\ \frac{x+9}{x} \end{cases}$	-1 ≤ x ≤ 2 x <-1 x > 2	a=1,5 x=1
Варіант 2	$y=\begin{cases} ax^2+bx+c; \\ (a+bx)+\sqrt{x}; \\ \frac{a}{x+\sqrt{x^2+1}} \end{cases}$	x < 1,2 x = 1,2 x > 1,2	a=2,8 b=-0.,3 c=4 x=1

Варіант 3	$y = \begin{cases} x^2 + \frac{6}{x+1}; \\ \ln(x+2b) \\ bx^2 - 8\sqrt{x}; \end{cases}$	x < 1,4 x = 1,4 x > 1,4	b=0,5 x=9
Варіант 4	$y = \begin{cases} 1{,}5\cos^2 x + 5a; \\ 1{,}8ax + tgx; \\ (x-2)^2 + 6 \end{cases}$	x < 2 x = 2 x > 2	a=2,3 x=3
Варіант 5	$y = \begin{cases} x\sqrt{x-a}; \\ x\sin(ax); \\ \cos(ax) + 2{,}5 \end{cases}$	x > a x = a x < a	a=2,5 x=6,5
Варіант 6	$y = \begin{cases} bx - \ln(bx); \\ 12; \\ bx + \ln(bx) \end{cases}$	bx < 1 bx = 1 bx > 1	b=2,5 x=4
Варіант 7	$y = \begin{cases} \frac{5x^2+1}{\sqrt{x}+t}; \\ \sqrt{x} + t + \frac{1}{x}; \\ \cos(x) + tx^2 \end{cases}$	x < 0,5 x = 0,5 x > 0,5	t=2,2 x=1,2
Варіант 8	$y = \begin{cases} at^2 + b\sin t; \\ at + b; \\ at^2 - b\cos t \end{cases}$	t<0,1 t=0,1 t>0,1	a=2,5 b=0,4 t=0,1

Завдання 3. Реалізувати програму розв'язання задачі згідно завдання викладача.

1. Дано значення дійсної величини X. Визначити:

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

Примітка: Для розв'язування цієї задачі необхідно врахувати, що ділити на нуль не можна.

2. При даному значенні X обчислити: $\sqrt{X^3 - \sqrt{X-1}}$

Примітка: Для розв'язування цієї задачі необхідно пам'ятати, що не можна знайти квадратний корінь з від'ємного числа.

3. Змінній К надати номер тієї чверті площини, де знаходиться точка з координатами х та у (ху<>0).

4. Дано три дійсних числа. Піднести до квадрату ті з них, значення яких невід’ємні.

5. Якщо сума трьох попарно різних чисел X, Y, Z є меншою за одиницю, то менше з X і Y замінити півсумою Y і Z, інакше – більше з X і Y замінити на Y3.

Практична робота. Створення та реалізація програм із розгалуженням. Команда вибору Case

Мета: формування навичок складання програм розв’язування задач з використанням вказівки вибору.

Завдання:

- *студенти повинні знати:* структуру та принцип дії команди вибору;
- *студенти повинні вміти:* створювати та виконувати програми з використанням вказівки вибору.

Обладнання: зошит, середовище програмування.

Хід виконання:

Завдання1. Розглянути приклади реалізації задач за допомогою конструкції вибору.

Приклад 1. Скласти програму визначення дня тижня за його порядковим номером.

```
program NOMER;
var DEN : integer;
begin
   write (‘Введіть номер тижня’);
   readln (DEN);
   if  (DEN < 1) or (DEN > 7)
      then writeln (‘Такого дня тижня не існує’)
   else
      case DEN of
            1 : writeln (‘понеділок’);
            2 : writeln (‘вівторок’);
            3 : writeln (‘середа’);
```

```
        4 : writeln (‘четвер’);
        5 : writeln (‘п´ятниця’);
        6 : writeln (‘субота’);
        7 :  writeln (‘неділя’)
    end;
end.
```

Приклад 2. На станції є бензин кількох марок і з різною ціною. Клієнт вибирає марку бензину і називає кількість літрів. Виведіть чек обслуговування клієнта.

Нехай марки бензину позначені цифрами 1, 2, 3, 4, 5, 6, 7.

```
program BENZYN;
var MARKA: integer; K, CINA : real;
begin
write (‘Введіть марку бензину');
readln (MARKA);
write (‘Введіть кількість літрів’);
readln (K);
    case MARKA of
        1 : cina:=10.00;
        2, 3 : cina:=11.35;
        4 : cina:=8.20;
        5..7 : cina:=10.30
    end;
writeln ( ‘Бензин марки’, MARKA);
writeln ( ‘Ціна’, CINA);
writeln ( ‘Кількість літрів’, K);
writeln ( ‘Оплачено:’, K*CINA);
end.
```

Завдання2. Реалізувати програму розв’язання задачі згідно завдання викладача.

1. Максимальні неперервні інтервали роботи за комп’ютером для дітей орієнтовно такі: 6-8 років – 15 хв, 9-11 років – 30 хв, 12-15 років – 45 хв, 16-17 років – 60 хв. Сюжет алгоритму такий: комп’ютер запитує вік та час, уже проведений за комп’ютером, і повідомляє, скільки хвилин залишилося до закінчення сеансу роботи.

2. Задайте відстані до міст A, B, C, D. Нехай на 100 км потрібно 7 л бензину. Комп’ютер запитує про пункт призначення (треба буде ввести одну букву) і повідомляє про необхідну кількість бензину.

3. Введіть номер місяця. Виведіть кількість днів у ньому.

Лекція. Вказівка повторення з параметром

Мета, завдання лекції: ознайомлення із правилами запису команд та принципом дії вказівки повторення з параметром.

План і організаційна структура лекції:

1. Вказівка повторення, її форми.
2. Дія вказівки.
3. Приклади розв'язання задач.

Зміст лекційного матеріалу:

Для реалізації циклічних алгоритмів використовують вказівки *повторення з параметром* (цикл з лічильником), *вказівки повторення з передумовою та післяумовою*.

Вказівка повторення з параметром призначена для організації багатократного виконання тіла циклу для значень параметра циклу з деякої впорядкованої множини.

Вказівку повторення з параметром використовують у тих випадках, коли наперед можна визначити кількість повторень виконання вказівок циклу.

Вказівка повторення з параметром записується у вигляді;

1. **for** <параметр *i* := вираз1> **to** <вираз2> **do**
```
begin
    <вказівка 1>;
      …
    <вказівка N>;
end;
```
2. **for** <параметр *i* := вираз1> **downto** <вираз2> **do**
```
begin
    <вказівка 1>;
      …
    <вказівка N>;
end;
```

Параметр вказівки (лічильник циклу) – це змінна скалярного (цілого, символьного, логічного або перелічувального) типу, крім дійсного.

Якщо в тілі вказівки повторення з параметром необхідно виконати тільки одну вказівку, то складеної вказівки **begin…end** використовувати не потрібно:

for <параметр *i* := вираз1> **to** <вираз2> **do** <вказівка 1>;

Рядок **for … to … do** (англ. *for* – для, *to* – до, *do* – робити, виконувати) називається ***рядком заголовку команди циклу з лічильником***. Змінна в рядку заголовка команди циклу з лічильником, що стоїть перед знаком присвоювання, називається ***лічильником циклу***.

Розглянемо дію вказівки **for – to – do.**

Наприклад:

```
for i:= K to M do
  begin
      <вказівка 1>;
      …
      <вказівка N>;
  end;
```

Дія вказівки. Параметру циклу *i* присвоюється початкове значення *K*. Він порівнюється з кінцевим значенням *M*. Якщо ***K <= M,*** то виконується тіло вказівки повторення. Значення K автоматично ***збільшується*** на 1 і знову порівнюється зі значенням *M*. Якщо під час перевірки отримаємо, що *K>M,* то виконання вказівки повторення припиняється і виконується наступна після неї вказівка програми. Якщо під час першого порівняння *K* і *M* виявиться, що *K>M*, то тіло вказівки не виконується жодного разу.

Розглянемо дію вказівки **for – downto – do.**

Наприклад:

```
for i:=K downto M do
  begin
      <вказівка 1>;
        …
      <вказівка N>;
  end;
```

Дія вказівки. Параметру циклу *i* присвоюється початкове значення *K*. Він порівнюється з кінцевим значенням *M*. Якщо ***K >= M***, то виконується тіло вказівки повторення. Значення *K* автоматично ***зменшується*** на 1 і знову порівнюється зі значенням *M*. Якщо під час перевірки отримаємо, що *K<M,* то виконання вказівки повторення припиняється і виконується наступна після неї вказівка програми. Якщо під час першого порівняння *K* і *M* виявиться, що *K<M*, то тіло вказівки не виконується жодного разу.

Основні властивості вказівки повторення з параметром:

- Вказівку повторення з параметром зручно використовувати в тих випадках, коли попередньо можна визначити кількість повторень.
- Параметр циклу повинен бути описаний в розділі змінних.

– Значення параметра циклу після закінчення виконання вказівки повторення володіє кінцевим значенням, тому перед повторним його використанням йому необхідно присвоїти нове початкове значення.

Приклад. Знайти суму всіх натуральних чисел від 1 до N.

```
program SUMA3;
var i, N, S: integer;
begin
   writeln (‘Введіть кількість натуральних чисел N=’);
   readln (N);
   S:=0;
   for i:=1 to N do
      S:=S+1;
      writeln (‘S=’,S);
end.
```

Приклад. Вивести на екран таблицю квадратів і кубів чисел від 10 до 20.

```
program STEPENI;
var i: integer;
begin
   for I := 10 to 20 do
   writeln (i : 6, i * i : 7, i * i * i : 8);
end.
```

Приклад. Один долар коштує 23,5 грн. Вивести у вигляді таблиці вартість 1,2,…,10 доларів.

```
program BANK;
var d : integer; gr : real;
begin
writeln (‘Долари        Гривні’);
 for d := 1 to 10 do
     begin
        gr := 23.5 * d;
        writeln(d : 4, gr : 15 : 2);
     end;
    end.
```

Приклад. Вивести на екран монітора всі літери латинської абетки.

```
program alphabet;
var i : char;
   begin
      for i:=’a’ to ‘z’ do
```

```
    write (i, ’’);
    repeat until KeyPressed
end.
```

Оператор ***repeat until KeyPressed*** призначено для організації затримки виведення літер: він означає, що виведення буде продовжуватись до тих пір, поки ми не натиснемо клавішу ***Enter*** на клавіатурі.

Розглянемо також інші способи організації затримки виконання програми в Паскаль:

1) використання процедури ***readln*** без параметрів:

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

2) Функція ***ReadKey*** зчитує з буфера клавіатури одиницю інформації, що міститься першою в буфері, й отримує значення натиснутої клавіші. Якщо перед виконанням цієї функції буфер був порожнім, то програма переходить в режим очікування, поки в буфер буде щось занесено. У цьому випадку після виконання функції ***ReadKey*** буфер знову стає порожнім. Якщо ж під час виконання функції ***ReadKey*** буфер не був порожнім, то він спорожниться тільки тоді, коли функцію виконають стільки разів, скільки інформації є в буфері. Цією особливістю можна скористатися, щоб організувати затримку виконання програми до натискання будь-якої клавіші:

```
program my_prog;
  uses CRT; {функція працює тільки при підключеному модулі
                  екрана!}
  var k : char;
begin
    {тіло програми}
    k := ReadKey;
end.
```

Практична робота. Створення програм з використанням циклу з параметром

Мета: формування навичок складання програм для розв’язку задач з використанням циклу з параметром.

Завдання:

– *студенти повинні знати:* структуру та принцип дії вказівки циклу з параметром;
– *студенти повинні вміти:* розв'язувати задачі, використовуючи цикл з параметром, виконувати її в середовищі програмування.

Обладнання: зошит, середовище програмування.

Хід виконання:

Завдання 1. Виведіть на екран у вигляді таблиці номери і значення перших 15 елементів числової послідовності, загальний елемент якої має вигляд:

№з/п	Загальний елемент послідовності
Варіант 1	$7 - 5\ i^2$
Варіант 2	$4 - \sin 2i$
Варіант 3	$15 - 2\ i^2$
Варіант 4	$3 - \sin i^2$
Варіант 5	$3\ i^2 - 30$
Варіант 6	$4 - 2\ \sin 2i$
Варіант 7	$2\ i^2 + 3$
Варіант 8	$6 - \sin 2i^2$

Завдання 2. Знайти суму натуральних чисел:

№з/п	Початкове та кінцеве значення параметра
Варіант 1	від 50 до 100
Варіант 2	від 25 до 55
Варіант 3	від -10 до -30
Варіант 4	від 15 до 35
Варіант 5	від -25 до -50
Варіант 6	від 20 до 80
Варіант 7	від -15 до -35
Варіант 8	від 62 до 102

Завдання 3. Знайти добуток натуральних чисел від 15 до 30.

№з/п	Початкове та кінцеве значення параметра
Варіант 1	від -5 до -10
Варіант 2	від 10 до 15
Варіант 3	від 5 до 15
Варіант 4	від -10 до -15
Варіант 5	від 8 до 18
Варіант 6	від -4 до -14
Варіант 7	від 7 до 17
Варіант 8	від -6 до -16

Додаткове завдання:

1. Виведіть на екран у вигляді таблиці номери і значення з двома цифрами після крапки перших десяти елементів числової послідовності, загальний елемент якої має вигляд $7 - 5 \sin i^2$.
2. Серед перших 20 елементів $1 - 3 \sin i^2$ виведіть на екран номери та значення лише від'ємних елементів.
3. У якій з послідовностей більше додатних елементів:

   1) $2 \sin 3i$, $i = 1, .., 10$ чи 2) $2 - 3 \cos i$, $i = 1, .., 15$?
4. Виведіть на екран елементи числової послідовності $5 - 3 \cos 2i$ від 10-го до 20-го і визначте з-поміж них максимальний і мінімальний, а також їхні місця в послідовності.
5. Обчислити значення факторіала n! = 1 * 2 * 3 * … * n (якщо n = 0, то потрібно знову водити значення n)
6. Обчислити значення функції $y = x^2 + x^3 + x + 1$ при x = 1, 2, 3, …, 10.
7. Обчислити суму значень функції

   y = 1 * sin(1) + 3 * sin(3) + 5 * sin(5) + … + k * sin(k) при k = 15.
8. Побудувати таблицю множення числа 23 на числа з проміжку [15; 25].

Лекція. Вказівки повторення з післяумовою і передумовою

Мета, завдання лекції: ознайомлення із правилами запису команд та принципом дії вказівок повторення з передумовою та післяумовою.

План і організаційна структура лекції:

1. Вказівка повторення з передумовою (Цикл ***while***).

2. Вказівка повторення з післяумовою (Цикл ***repeat – until***)
3. Приклади розв'язання задач.

Зміст лекційного матеріалу:

Якщо кількість повторень команд тіла циклу до початку виконання команди циклу невідома, потрібно використовувати *команду циклу з передумовою* або *команду циклу з післяумовою*.

Цикл з передумовою

Цикл називається *циклом з передумовою*, якщо умова перевіряється до початку виконання команд тіла циклу.

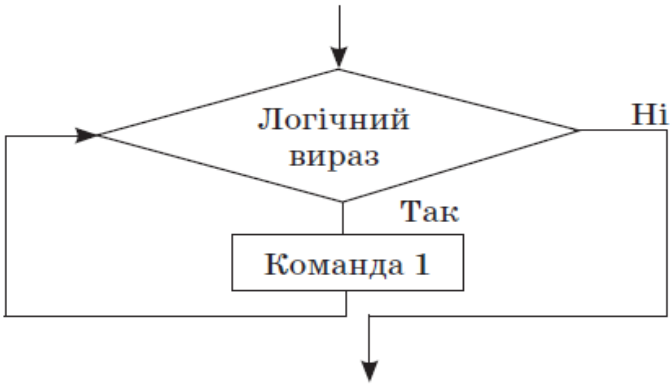


Рис. Блок-схема циклу з передумовою

Загальний вигляд ***команди циклу з передумовою*** (англ. *while* – поки) такий:

while *<логічний вираз>* ***do***
begin
<команди тіла циклу>
end;

Якщо тіло циклу складається лише з однієї команди, операторні дужки *begin* і *end* можна не ставити.

Виконується команда циклу з передумовою так:

1. Обчислити значення логічного виразу.
2. Якщо це значення *true*, то виконати команди тіла циклу і перейти до команди 1, а якщо *false*, то виконати команду, наступну за командою циклу.

Правила використання циклу ***while***:

– Якщо умова у вказівці повторення хибна вже при першій перевірці, то тіло циклу не виконується жодного разу. Умова вказівки є умовою входження в цикл.
– Умову вказівки повторення з передумовою необхідно підбирати таким чином, щоб в деякий момент вона змінила своє значення з TRUE на

FALSE, інакше виникне ситуація "зациклювання" програми.

– Програмісту необхідно самому потурбуватись про необхідність збільшення (зменшення) лічильника циклу на 1. Це можна зробити, наприклад, таким чином i := i + 1.
– Входження в тіло вказівки повторенням можливе тільки через заголовок цієї вказівки.

Цикл з післяумовою

При виконанні наведеного алгоритму команди тіла циклу обов'язково виконуватимуться хоча б один раз, оскільки команда перевірки умови виконується після виконання команд тіла циклу. Такий цикл називається ***циклом з післяумовою***.

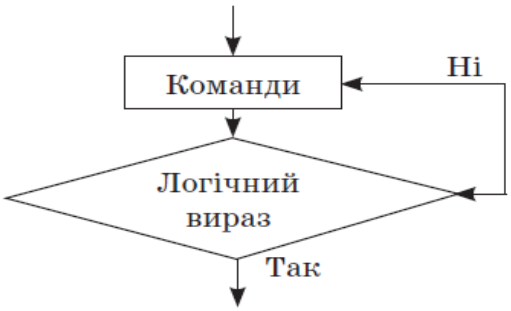


Рис. Блок-схема циклу з післяумовою

Загальний вигляд ***команди циклу з післяумовою*** (англ. *repeat* – повторити, *until* – доки, не раніше як) такий:

repeat

<команди тіла циклу>

until *<логічний вираз>;*

Виконується команда циклу з передумовою так:

1. Виконати команди тіла циклу.
2. Обчислити значення логічного виразу.
3. Якщо це значення ***false***, то виконати п. 1, а якщо ***true***, то виконати команду, наступну за командою циклу.

Правила використання циклу ***repeat-until***:

– Команду циклу з післяумовою доцільно використовувати в тих випадках, коли команди тіла циклу повинні виконатися хоча б один раз.
– Тіло циклу, яке складається з групи вказівок, не потрібно брати в операторні дужки *begin – end*), оскільки службові слова *repeat – until* відіграють роль дужок.
– Тіло циклу у вказівці повторення з післяумовою виконується щонайменше один раз.
– Умова вказівки є умовою виходу з циклу.

– Входження в тіло вказівки повторенням можливе тільки через заголовок цієї вказівки.

Задача. Знайти суму всіх натуральних чисел від 1 до N.

Приклад використання вказівки повторення з передумовою

```
program SUMA1;
var i, N, S : integer;
begin
   writeln(' Введіть кількість натуральних чисел: ');
       write(‘ N= ’);
       read(N):
       i := 1;
       S := 0;
       while i <= N do
           begin
              i := i+1
           end;
   writeln(‘ S= ’, S)
end.
```

Приклад використання вказівки повторення з післяумовою

```
program SUMA2;
var i, N, S : integer;
begin
      writeln(' Введіть кількість натуральних чисел:');
      write(‘N=’);
      read(N):
      i:=1;
      S:=0;
      repeat
          S:=S+i;
          i:=i+1
      until i>N;
      writeln(‘S=’, S)
end.
```

Практична робота. Створення програм з використанням циклу з передумовою і післяумовою

Мета: формування навичок складання програм для розв’язку задач з використанням циклу з передумовою і післяумовою.

Завдання:

- *студенти повинні знати:* структуру та принцип дії вказівки циклу з передумовою і післяумовою;
- *студенти повинні вміти:* розв'язувати задачі, використовуючи цикли з передумовою і післяумовою, виконувати їх в середовищі програмування.

Обладнання: зошит, середовище програмування.

Хід виконання:

В-1

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Визначити суму непарних чисел від 1 до 50.
2. Обчисліть добуток чисел від 10 до 25, кратних 5.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y=cos2x*** на проміжку [-1;1] з кроком 0,25.
2. Протабулюйте функцію ***y=2sin2x*** на проміжку [-2;2] з кроком 0,2 і визначте кількість додатних елементів.

В-2

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Вивести на екран усі двозначні числа, які діляться на 3.
2. Обчисліть добуток непарних чисел від 7 до 15.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = sin3x*** на проміжку [-1; 1] з кроком 0,2.
2. Протабулюйте функцію ***y = cos(x-1)*** на проміжку [-2 ;2] з кроком 0,2 і визначте суму від'ємних елементів.

В-3

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Вивести на екран десять рядків таблиці відповідності між двома наступними мірами, знаючи що 1 карат = 0,2 г
2. Обчисліть добуток чисел від 5 до -5.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = cos(2-x)*** на проміжку [0; 2] з кроком 0,2.

2. Протабулюйте функцію ***y = 4 – 2 sin2x*** на проміжку [-1;1] з кроком 0,25 і визначте добуток додатних елементів.

В-4

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Визначити суму парних чисел від 10 до 50.
2. Вивести на екран усі двоцифрові числа, які діляться на 10.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = 1 – cosx*** на проміжку [-1; 1] з кроком 0,25.
2. Протабулюйте функцію ***y = 3 – 3 sinx²*** на проміжку [-2; 2] з кроком 0,2 і визначте кількість від'ємних елементів.

В-5

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Вивести на екран усі двозначні числа, які діляться на 5.
2. Обчисліть добуток чисел від 10 до 40 кратних 5 .

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = cosx –3*** на проміжку [0; 2] з кроком 0,25.
2. Протабулюйте функцію ***y = 2 + |sin3x|*** на проміжку [-3; 1] з кроком 0,2 і визначте суму від'ємних елементів.

В-6

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Визначити суму чисел від 1 до 100, які діляться на 10.
2. Обчисліть добуток чисел від 10 до 20, кратних 3 .

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = cos3x*** на проміжку [-1; 1] з кроком 0,25.
2. Протабулюйте функцію ***y = x sinx*** на проміжку [1; 2] з кроком 0,1 і визначте добуток додатних елементів.

В-7

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Вивести на екран десять рядків таблиці відповідності між двома наступними мірами, знаючи що 1 гран = 0,068 г

2. Обчисліть добуток чисел від 15 до 28.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = cos(x-2)*** на проміжку [0; 2] з кроком 0,1.
2. Протабулюйте функцію ***y = 2 – sin3x*** на проміжку [-2; 2] з кроком 0,2 і визначте кількість додатних елементів.

В-8

Використовуючи оператор WHILE, скласти і реалізувати програми розв'язку наступних задач:

1. Визначити суму парних чисел від 1 до 50.
2. Вивести на екран усі двоцифрові числа, які діляться на 20.

Використовуючи оператор REPET, скласти і реалізувати програми розв'язку наступних задач:

1. Протабулюйте функцію ***y = 4 – cosx*** на проміжку [-2;0] з кроком 0,25.
2. Протабулюйте функцію ***y= 2 x – sinx*** на проміжку [-3;1] з кроком 0,2 і визначте суму від'ємних елементів.

Практична робота. Створення програм з використанням циклу з передумовою і післяумовою

Мета: формування навичок складання, введення та виконання програм для розв'язку задач з використанням циклів та структури розгалуження.

Завдання:

- *студенти повинні знати:* структуру та принцип дії циклічних алгоритмів та конструкції розгалуження;
- *студенти повинні вміти:* розв'язувати задачі, використовуючи вказівку розгалуження, цикли з передумовою, післяумовою та лічильником, виконувати їх в середовищі програмування.

Обладнання: зошит, середовище програмування.

Хід виконання:

Завдання 1. Реалізувати програму розв'язку задачі згідно варіанту, заданого викладачем:

1. Дано два числа. Зменшити перше число у два рази, якщо його модуль більше другого, інакше залишити його без змін. Реалізувати програму для чисел -8; 6.
2. Дано три числа. Знайти найбільше серед них. Реалізувати програму для чисел 20; 40; 60.

3. Дано три відрізки а, в, с. Скласти програму, яка визначає, чи існує трикутник з такими сторонами і визначити, який це трикутник (рівнобедрений, рівносторонній, різносторонній).
4. Сторони трикутника а, в, с. Визначити, який це трикутник: прямокутний, гострокутний, тупокутний. (За наслідками теореми косинусів).
5. Визначити, чи являється число а парним. Якщо так, то знайти квадрат і куб цього числа. Реалізувати програму для чисел а = 100; а = 11.
6. Визначити, чи є серед цифр заданого тризначного числа однакові.
7. Визначити, чи дорівнює квадрат заданого тризначного числа кубу суми цифр цього числа.
8. Дано три числа. Піднесіть до квадрата ті з них, значення яких недодатні. Додатні числа піднесіть до куба.
9. Визначити, чи попадає точка М(х; у) в коло радіуса R з центром в початку координат. Реалізувати програму для а) R = 5; М (3;4); б) R = 5; М (2;5).
10. Дано три додатні числа А, В, С. Визначити, чи кратне трьом число М = (А + B^2) С. Якщо воно не кратне трьом, то знайти остачу від ділення М на 3. Реалізувати програму для чисел 5; 3; 2.
11. Виявнити, чи являється трикутник з даними сторонами 3, 4, х прямокутним. Реалізувати програму для х=5; х=6.

Завдання 2. Протабулювати функцію f(x) на заданому інтервалі [a; b] з заданим кроком зміни аргумента h.

№з/п	***Функція f(x)***	***Інтервал***	***Крок***
1	$x^3 + 3{,}2\ x^2 - 5\ x \sin 2x$	[1; 3]	0,2
2	$\sin 3x \cos 3x + 2{,}4\ x^2 - ½\ x$	[0; 4]	0,4
3	$x^4 - 2{,}5\ x^2 + \sin 2x \cos 2x$	[1; 5]	0,4
4	$tg3x \sin 2x + 3{,}4\ x^3 - 2{,}3x$	[0; 3]	0,25
5	$3\ x^2 + \sin 4x \cos 3x - \lvert x - 3{,}6 \rvert$	[1; 4]	0,25

Завдання 3. Реалізувати програму розв’язку задачі згідно варіанту, заданого викладачем:

1. Визначити серед десяти натуральних числа, кратні трьом. Порахувати кількість цих чисел і надрукувати ці числа. Реалізувати програму для

чисел 2; 3; 6; 8; 9; 18; 20; 30; 112; 120.

2. Визначити серед дванадцяти натуральних чисел числа, кратні 5. Порахувати кількість цих чисел і надрукувати ці числа. Реалізувати програму для чисел 2; 5; 7; 12; 15; 17; 25; 315; 311; 402; 500; 100.
3. Дано десять чисел. Визначити, скільки серед них від'ємних.
4. Знайти середнє арифметичне число всіх натуральних чисел, менших 20.
5. Знайти суму натуральних чисел, кратних 3, які більші за 20 і менші за 100.

Лекція. Рядкові величини. Операції над рядковими величинами. Алгоритми роботи з рядками

Мета, завдання лекції: ознайомлення з процедурами і функціями роботи з рядками.

План і організаційна структура лекції:

1. Запис рядкових величин.
2. Функції та процедури для роботи з рядками.
3. Приклади розв'язування задач.

Зміст лекційного матеріалу:

Рядок – це послідовність символів кодової таблиці ЕОМ. При використанні у виразах рядок охоплюється з двох сторін апострофами. Кількість символів у рядку (максимальна довжина рядка) може змінюватись від 0 до 255.

Для опису рядкових величин використовують ідентифікатор ***string***, після якого в квадратних дужках записується максимальне значення довжини рядка для даної величини.

Загальний вигляд команди:

var <ідентифікатор>: **string** [максимальна довжина рядка];

Приклад:

```
var R1: string [10];
    R2: string [4];
```

Максимальна довжина рядка для змінної r1 = 10, r2 = 4.

Якщо довжина рядка не вказана, то вона автоматично приймає значення 255 байт.

Рядкові величини використовуються в програмі і у вигляді констант.

Наприклад,

```
Const NAME = 'інформатика';
```

Для роботи з рядковими величинами існує ряд процедур і функцій:

Функція **Concat(R1, R2, R3);**

Функція *Concat* здійснює склеювання рядків R1, R2, R3 в один рядок в такому порядку, в якому вони були записані.

Наприклад.

```
program Fconcat;
const R1 = ’Мова’; R2 = ’програмування’; R3 = ’Pascal’;
var r : string[35];
begin
    R := concat(R1,R2,R3);
    writeln(R);
end.
```

На екрані дисплея буде надруковано: *Мова програмування Pascal*.

Результатом використання функції *Concat*(‘20’, ’05’); буде значення: ‘2005’.

Даний результат можна отримати ще з допомогою операції склеювання – "+".

Доступ до і-го символу текстового даного з іменем а можна отримати з допомогою виразу а[i].

Наприклад, якщо а := ’Роман’, то а[1] – це ‘Р’, а[3] – ‘М’.

Функція **Length(R);**

Функція *Length* видає фактичну довжину рядка, який міститься в даній змінній. При підрахуванні довжини рядка враховуються всі символи, в тому числі і проміжки.

Результатом використання функції Length(‘Іван Підкова’); буде значення 12.

Функція **Copy(R, Poz, N);**

Функція *Copy* копіює фрагмент довжиною N рядка R, починаючи з позиції Poz.

Результатом використання функції Copy(‘Іван Підкова’, 6, 7); буде значення: ‘Підкова’;

Функція **Pos(WORD, R);**

Функція *Pos* знаходить номер позиції P, з якої починається перше входження слова WORD в рядку R. Якщо слово WORD в рядку R не знайдено, то буде надруковано число 0.

Наприклад, Pos(‘’,‘Іван Підкова’);

Задача 1. Скласти програму для визначення кількості букв «а» в деякому заданому тексті B.

```
program Kilkist;
var B : string; n, i, k : integer;
begin
writeln('Введіть текст');
    readln(B);
    n := 0; k := length(B);
    for i := 1 to k do
        if b[i] = 'a' then n := n + 1;
    writeln(n);
end.
```

Процедура **Insert(Word, R, Poz);**

Процедура *Insert* вставляє слово Word в рядок R, починаючи з позиції Poz.

Наприклад, Нехай L := 'Львівська політехніка'

Внаслідок застосування процедури Insert('НУ',L,1); отримаємо значення: 'НУ Львівська політехніка'.

Процедура **Delete(R, Poz, N);**

Процедура *Delete* знищує слово, яке починається з вказаної позиції Poz і має задану довжину N в рядку R.

Внаслідок застосування процедури Delete(L,1,13); отримаємо значення: 'політехніка'.

Задача 2. Змінна a містить текст з двох слів. Слова розмежовані пропуском. Вивести на екран друге слово.

```
program Slovo;
var a : string;
    begin
        writeln('Введіть текст');
        readln(a);
        delete(a, 1, pos(' ', a));
        writeln (a);
    end.
```

Приклади складання та реалізації програм опрацювання рядкових величин

1.
```
program Fconcat;
var R : string [35];
begin
  R := 'Мова ' + 'програмування' + 'Pascal';
```

```
    writeln(R);
end.
```

На екрані буде надруковано*: Мова програмування Pascal.*

2.
```
program Flength;
    const R = ’Turbo Pascal’ ;
    var N : integer;
    begin
        N := length(R);
        writeln(' n= ', N);
    end.
```

На екрані буде надруковано*: n = 12*

3.
```
program Fcopy;
    const R = ’Turbo Pascal’;
    var WORD : string [6];
    Poz, N : integer;
    begin
        Poz := 7;
        N := 6;
        WORD := Copy(R, Poz, N);
    writeln(WORD);
    end.
```

На екрані буде надруковано: *Pascal.*

4.
```
program Fpos;
    const R = 'Севастополь';
    WORD = 'сто';
    var P : integer;
    begin
        P := Pos(WORD,R);
        writeln(‘ P= ’, P);
    end.
```

На екрані буде надруковано: *P = 5*.

5.
```
program Finsert;
    var Poz : integer;
         R, WORD : string [35];
    begin
        Poz = 20;
        R :=  ‘Мова програмування Pascal';
        word := ’Turbo';
```

```
        insert(WORD, R, Poz);
        writeln(R);
    end.
```

На екрані буде надруковано: *Мова програмування Turbo Pascal.*

6.
```
program Fdelete;
    var R : string [35];
        N, Poz : integer;
    begin
        R := ’Мова програмування Turbo Pascal';
        Poz := 1;
        N := 19;
        delete(R, Poz, N);
        writeln(R);
    end.
```

На екрані буде надруковано: *Turbo Pascal.*

7.
```
program on_the contrary;
    var S, S_new : string;
         i : integer;
    begin
        write(’Введіть слово: ’);
        S_new := ’ ’;
        for i := 1 to length (S) do
        S_new := S[i] + S_new;
    writeln(’Слово навпаки: ’, S_new);
    Repeat until KeyPressed
    end.
```

На екрані буде надруковано слово навпаки. Наприклад, ввели слово ’алгоритм’ – отримали ’мтирогла’).

Практична робота. Створення програм опрацювання рядкових величин

Мета: формування навичок складання програм опрацювання рядкових величин.

Завдання:

– *студенти повинні знати:* опис рядкової величини, функції і процедури для роботи з рядками;

– *студенти повинні вміти*: рядкові величини як сталі і змінні; створювати найпростіші програми, які передбачають опрацювання рядкових величин.

Обладнання: зошит, середовище програмування.

Теоретичні відомості:

Процедури і функції, які використовують для роботи з рядками:

Concat (R1, R2 ,R3); здійснює склеювання рядків R1, R2, R3 в один рядок в такому порядку, в якому вони були записані.

Н-д, Concat (’20’, ’05’); Значення: ’2005’.

Length (R); видає фактичну довжину рядка, який міститься в даній змінній. При підрахуванні довжини рядка враховуються всі символи, в тому числі і проміжки.

Н-д, Length (’Іван Підкова’); Значення: 12.

Copy (R, Poz, N); копіює фрагмент довжиною N рядка R, починаючи з позиції Poz.

Н-д, Copy (’Іван Підкова’, 6, 7); Значення: ’Підкова’.

Pos (WORD, R); знаходить номер позиції P, з якої починається перше входження слова WORD в рядку R. Якщо слово WORD в рядку R не знайдено, то буде надруковано число 0.

Н-д, Pos (’ ’, ’Іван Підкова’);

Insert (Word, R, Poz); вставляє слово Word в рядок R, починаючи з позиції Poz.

Н-д, Нехай L := ’Львівська політехніка’.

Insert(’НУ’, L, 1); Значення: ‘НУ Львівська політехніка’

Delete (R, Poz, N); знищує слово, яке починається з вказаної позиції Poz і має задану довжину N в рядку R.

Delete (L, 1, 13) Значення: ‘політехніка’

Хід виконання:

Скласти програми для виконання наступних задач:

B-1

1. Дано рядкові величини: A1 = 'привіт', A2 = 'сестричко', A3 = 'Олю', A4 = '!', A5 = ','. Вивести на екран 3 можливі привітання, утворені із даних рядків.
2. Дано рядкові величини B1 = 'класична', B2 = 'музика'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить назву Вашого міста. Виведіть її символи у стовпчик.

В-2

1. Дано рядкові величини: В1 = 'мер', В2 = 'міста', В3 = 'Романюк'. Вивести на екран 3 можливі словосполучення, утворені із даних рядків.
2. Дано рядкові величини С1 = 'Волинський ', С2 = ' край'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить Ваше прізвище. Виведіть її символи у стовпчик.

В-3

1. Дано рядкові величини: С1 = 'друже', С2 = 'Віктор', С3 = 'привіт', С4 = '!', С5 = ','. Вивести на екран 3 можливі привітання, утворені із даних рядків.
2. Дано рядкові величини М1 = 'художній', М2 = 'колектив'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить назву Вашої додаткової кваліфікації. Виведіть її символи у стовпчик.

В-4

1. Дано рядкові величини: К1 = 'матуся', К2 = 'люба', К3 = 'з святом', А4 = '!', А5 = ','. Вивести на екран 3 можливі привітання, утворені із даних рядків.
2. Дано рядкові величини А1 = 'педагогічний', А2 = 'коледж'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить назву Вашої вулиці. Виведіть її символи у стовпчик.

В-5

1. Дано рядкові величини: А1 = 'класний', А2 = 'керівник', А3 = 'добрий', А4 = '!', А5 = ',', А6= 'день'. Вивести на екран 3 можливі привітання, утворені із даних рядків.
2. Дано рядкові величини В1 = 'студентський', В2 = 'колектив'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить Ваше ім'я. Виведіть її символи у стовпчик.

В-6

1. Дано рядкові величини: В1 = 'Луцьк', В2 = 'обласний', В3 = 'Волині', А4 = 'центр', А5 = ". Вивести на екран 3 можливі словосполучення, утворені із даних рядків.
2. Дано рядкові величини С1 = 'музичний', С2 = 'калейдоскоп'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.

3. Введіть рядкову величину, яка містить прізвище Вашого товариша (подруги). Виведіть її символи у стовпчик.

В-7

1. Дано рядкові величини: C1 = '8 Березня', C2 = 'міжнародне', C3 = 'жіноче', C4 = '-', C5 = 'свято'. Вивести на екран 3 можливі словосполучення, утворені із даних рядків.
2. Дано рядкові величини N1 = 'українська', N2 = 'мова'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить назву музичного гурту. Виведіть її символи у стовпчик.

В-8

1. Дано рядкові величини: P1 = 'до побачення', P2 = 'урок', P3 = 'учні', P4 ='!', P5 = ',', P6 = 'закінчено'. Вивести на екран 3 можливі словосполучення, утворені із даних рядків.
2. Дано рядкові величини D1 = 'математичний', D2 = 'диктант'. Вивести на екран 3-4 нових слова (іменники), утворені з даних рядків.
3. Введіть рядкову величину, яка містить слово 'Україна' . Виведіть її символи у стовпчик.

Практична робота. Створення програм опрацювання рядкових величин

Мета: формування навичок складання програм опрацювання рядкових величин.

Завдання:

- *студенти повинні знати:* опис рядкової величини, функції і процедури для роботи з рядками;
- *студенти повинні вміти*: рядкові величини як сталі і змінні; створювати найпростіші програми, які передбачають опрацювання рядкових величин.

Обладнання: зошит, середовище програмування.

Хід виконання:

Написати програму для реалізації завдання згідно заданого варіанту.

В-1

1. Ввести довільний текст. Відредагувати його, замінивши всі символи «.» на «_» і підрахувати кількість виконаних замін.
2. Ввести деяке текстове дане. Вивести його зображення в зворотному

порядку.

В-2

1. Ввести довільний текст. У даному тексті знищити всі символи «,» і підрахувати довжину утвореного тексту.
2. Скласти програму, яка б заміняла в заданому тексті букву «е» на букву «и».

В-3

1. Ввести довільний текст. Перевірити, чи в заданому тексті кількість відкритих дужок дорівнює кількості закритих дужок.
2. Ввести довільний текст. У даному тексті всі буквосполучення Sin замінити на Cos.

В-4

1. Ввести довільний текст. Поміняти місцями перший і останній символи тексту..
2. Ввести довільний текст. У даному тексті знайти і надрукувати всі символи, які записані після першого символу «:».

В-5

1. Ввести довільний текст. Поміняти місцями перший і останній символи тексту..
2. Ввести довільний текст. У даному тексті знайти і надрукувати всі символи, які записані після першого символу «:».

В-6

1. Ввести довільний текст. В даному тексті підрахувати кількість букв а і в. Якщо букв а більше, ніж букв в, то в тексті знищити всі букви в, інакше – а.
2. Ввести довільний текст. У даному тексті визначити, скільки разів зустрічається словосполучення ко.

В-7

1. Ввести довільний текст. В даному тексті замінити всі слова children на pupil.
2. Ввести довільний текст. У даному тексті є символи «:». Надрукувати ту частину тексту, яка знаходиться до першого символу «:».

В-8

1. Замінити в заданому тексті сполучення «коло» на «квадрат». Наприклад, «фігура - коло» на «фігура - квадрат».
2. Скласти програму для підрахунку сумарної кількості букв «а», «е», «о» в тексті.

В-9

1. Ввести деяке текстове дане. Вивести його зображення в зворотному порядку.
2. Задано два слова Т, С. Скласти програму для визначення, в якому із слів - Т чи С - буква «а» зустрічається частіше.

Лекція. Табличні величини. Алгоритми роботи з табличними величинами. Поняття одновимірного масиву

Мета, завдання лекції: ознайомлення з поняттям масиву, правилами опису, заповнення та виведення масивів.

План і організаційна структура лекції:

1. Поняття одновимірного масиву.
2. Опис типів масивів та оголошення змінних типу масив.
3. Доступ до і-го елементу масиву.
4. Заповнення одновимірних масивів.
5. Методи виведення елементів одновимірного масиву на екран.
6. Використання масивів при розв’язуванні задач.

Зміст лекційного матеріалу:

Поняття одновимірного масиву

Масив – це впорядкований набір даних одного типу. Основне призначення масивів – запам'ятовування великої кількості елементарних даних одного типу під час виконання роботи.

Даними одного типу можуть бути *числові* (значення температур, які були протягом одного місяця; оцінки учнів за контрольну роботу; значення функції), *текстові* (список учнів класу; назви видів продукції).

Масиви мають імена, які їм надає користувач.

Масив складається з елементів. Кожен елемент має індекси, за якими його можна знайти в масиві. Кількість індексів визначає розмірність масиву. Наприклад, одновимірні (з одним індексом) та двовимірні (з двома індексами) масиви. Одновимірні масиви ще називають *лінійними таблицями.*

Приклад 1: Значення температур протягом місяця лютого (одновимірний масив). Назвемо цей масив t. Він має 28 елементів, які позначаються t_1, t_2,…,t_{28}. Кожному елементу масиву відповідає *індекс* (порядковий номер елементу масиву):

$t_1 = -10$, $t_2 = -9.5$, $t_3 = -6$, …, $t_{26} = -5,7$, $t_{27} = -3$, $t_{28} = 1$.

Числа 1, 2, …, 28 – індекси елементів масиву.

-10	-9,5	-6	-2,2	…	-5,7	-3	1
1	2	3	4	…	26	27	28

Кожний елемент цього масиву є даним дійсного типу (real). Отже, t – це масив з 28 елементів дійсного типу.

Перевага масиву: без масиву значення температур потрібно було б описувати за допомогою 28 змінних t_1, t_2, …, t_{28}. Такі змінні не можна опрацьовувати за допомогою циклів, а змінні з індексами t_i – можна.

Будь-яка задача з масивами складається з таких етапів:

- опис масиву;
- формування масиву (заповнення масиву даними);
- виведення масиву на екран;
- обробка масиву (дії з даними);
- виведення на екран результатів обробки.

Опис типів масивів та оголошення змінних типу масив

Розв'язуючи задачу з використанням масивів, користувач повинен виконати такі дії:

1. З'ясувати, які типи масивів потрібні для розв'язування задачі і описати їх (типи та розміри масивів) у розділі опису типів.

2. Визначити кількість та імена масивів кожного типу і оголосити їх в розділі оголошення змінних.

3. Ввести конкретні значення елементів масивів і опрацювати їх відповідно до умови задачі.

Описати масив можна таким чином:

І-ий спосіб

type <назва типу> = **array** [<розмір>] **of**<назва базового типу>;

var <список імен масивів>:<назва типу масивів>;

Назву типу масиву дає користувач; розмір (кількість елементів масиву) задають у вигляді діапазону. Наприклад, діапазон 1..10 (5..15) задає 10 елементів масиву. Назва базового типу (наприклад, real, integer, …) описує тип елементів масиву.

Наприклад,

type MAS = **array** [1..5] **of** integer;

var A : MAS;

ІІ-ий спосіб

var <список імен масивів> : **array**[<розмір>] **of**<назва базового типу>;

Наприклад,

const n = 100;

var A : **array** [1..n] **of** integer;

B : **array** [1..100] **of** real;

Для задачі про температуру у лютому оголосити масив можна так:

var t : **array**[1..28] **of** real;

Доступ до і-го елементу масиву. Змінна з індексом

Доступ до кожного елементу масиву забезпечується виразом, що складається з імені масиву та індексу.

<ім'я масиву>[<індекс>]

Наприклад, t[1] = -10, t[28] = 1 (з таблиці).

Занумерувати елементи в таблиці можна з починаючи з 0 або з будь-якого іншого *цілого* числа.

Приклад: Нехай кількість учнів у школі протягом 1986-1992 років була такою:

Учні	780	820	860	830	800	820	840
Роки	1986	1987	1988	1989	1990	1991	1992

Масив можна описати так:

var PUPILS : **array**[1986 .. 1992] **of** integer;

PUPILS[1986] – перший елемент масиву PUPILS;

PUPILS[1986] = 780

Заповнення одновимірних масивів

1) за формулою:

```
for i := 1 to n do
M[i] := i * i - 10 {або будь-яка формула};
```

2) з клавіатури:

```
for i := 1 to n do
begin
write('Введіть M[', i, ']: ');
readln(M[i]);
end;
```

3) випадковим чином (генератором випадкових чисел) із проміжку [A,B]:

```
for i:=1 to n do
M[i] := random(B-A) + A;
```

Методи виведення елементів одновимірного масиву на екран

1) виведення у стовпчик:

```
for i := 1 to n do
writeln(M[i]);
```

2) виведення у рядок:

```
for i := 1 to n do
write(M[i] : 5);
```

При виведенні елементів масиву у рядок бажано зазначити формат виведення, наприклад, write (M[i] : 10 : 3) – для дійсних чисел або write (M[i] : 5) – для цілих.

Використання масивів при розв'язуванні задач

Задача 1. У розчинах А і В міститься по 10 компонентів. Кількість кожного компонента кожному розчині задано формулами: $a_i = 1 + sini$, де i = 1, 2, .., 10, $в_i = 1 + cosi$, де i= 1, 2, .., 10.

Розчини змішали і отримали суміш С. визначити маси всіх компонентів у суміші та масу суміші С.

```
program SUMISH;
const n=10;
var A, B, C : array[1..n] of real; i : integer; S : real;
begin
S := 0;
for i := 1 to n do
    begin
        a[i] := 1 + sin(i);
        b[i] := 1 + cos(i);
        c[i] := a[i] + b[i];
        S := S + c[i];
        writeln('c[', i, ']= ', c[i] : 4 : 2);
    end;
     writeln('Masa sumishi:', S : 4 : 2);
end.
```

Задача 2. Знайти суму елементів одновимірного масиву.

```
program SUMA;
var A : array[1..5] of integer; i, S : integer;
begin
for i := 1 to 5 do
    begin
        write ('a[', i, ']= ');
        readln (a[i]);
    end;
S := 0;
writeln ('масив А: ');
for i := 1 to 5 do
    begin
       write (a[i] : 5);
       S := S + a[i];
```

```
    end;
  writeln (’Сума: ’, S);
  end.
```

Задача 3. Сталий масив.

Вісім суддів виставили спортсменам такі оцінки: 9.2, 9.4, 9.6, 9.3, 9.5, 9.4, 9.1, 9.3. Вивести середню оцінку (s1) з точністю до трьох знаків після десяткової крапки. Скільки було оцінок (k), вищих від середньої?

```
program COMPETITION;
const n = 8;
A : array[1 .. n] of real = (9.2, 9.4, 9.6, 9.3, 9.5, 9.4, 9.1, 9.3);
var i, k : integer; s, s1: real;
begin
    s := 0; k := 0;
    for i := 1 to n do
    S := S + a[i];
    S1 := S/n;
    writeln (‘Середня оцінка:’, S1 : 6 : 2);
    for i := 1 to n do
    if a[i] > s1 then k := k+1;
    writeln (‘Вищих від середньої було: ’, k, ’оцінок’)
end.
```

Практична робота. Створення та реалізація програм опрацювання одновимірних масивів

Мета: формування навичок складання і виконання програм обробки одновимірних масивів.

Завдання:

- *студенти повинні знати:* поняття одновимірного масиву, його опис, алгоритми роботи з табличними величинами;
- *студенти повинні вміти:* створювати програми опрацювання одновимірних масивів.

Обладнання: зошит, середовище програмування.

Хід виконання:

Написати програму для реалізації завдання згідно заданого варіанту.

В-1

1. Сформуйте масив з 10 елементів за правилом c[i] = 2i − 8 та виведіть його на екран. Обчисліть суму та кількість від’ємних елементів масиву.

2. Сформуйте одновимірний масив з 5 цілих чисел з клавіатури та виведіть його на екран. Обчисліть суму квадратів усіх елементів масиву.

В-2

1. Сформуйте одновимірний масив з 6 цілих чисел з клавіатури та виведіть його на екран. Замініть елементи, що менші від 1, на число 10 та виведіть масив після заміни на екран.

2. Сформуйте масив з 10 елементів за правилом c [i] = 10 − 2i та виведіть його на екран. Обчисліть кількість та суму додатних елементів масиву.

В-3

1. Сформуйте масив з 9 елементів за правилом a[i] = 2i − 6 та виведіть його на екран. Обчисліть суму квадратів додатних елементів масиву.

2. Сформуйте одновимірний масив з 7 цілих чисел з клавіатури та виведіть його на екран. Обчисліть кількість та суму елементів масиву, більших від 1.

В-4

1. Сформуйте з клавіатури масив з 6 цілих чисел та виведіть його на екран. Підрахуйте кількість елементів масиву, що більші за 3.

2. Сформуйте масив з 8 елементів за правилом a[i] = 5 − i та виведіть його на екран. Обчисліть кількість та добуток додатних елементів масиву.

В-5

1. Сформуйте масив з 9 елементів за правилом m[i] = 10 − 2i. Підрахуйте кількість невід'ємних елементів масиву.

2. Сформуйте одновимірний масив з 6 цілих чисел з клавіатури та виведіть його на екран. Замініть елементи, що більші від 2, нулями і виведіть змінений масив на екран.

В-6

1. Сформуйте масив з 5 елементів за правилом m[i] = 7 − 2i. Підрахуйте кількість від'ємних елементів масиву.

2. Сформуйте одновимірний масив з 5 цілих чисел з клавіатури та виведіть його на екран. Обчисліть середнє арифметичне елементів цього масиву.

В-7

1. Сформуйте масив з 7 елементів за правилом b[i] = 2i − 5. Підрахуйте кількість додатних елементів масиву.

2. Сформуйте з клавіатури масив з 6 цілих елементів та виведіть його на екран. Замініть додатні елементи нулями та виведіть масив після

заміни на екран.

В-8

1. Сформуйте з клавіатури масив з 8 цілих елементів та замініть від'ємні елементи нулями. Виведіть на екран масив до і після заміни.
2. Сформуйте масив з 7 елементів за правилом b[i] = 2i − 5 та виведіть його на екран. Обчисліть кількість та суму додатних елементів масиву.

В-9

1. Сформуйте одновимірний масив з 8 цілих чисел з клавіатури та виведіть його на екран. Знайдіть кількість та суму елементів, що менші від 2.
2. Сформуйте масив з 6 елементів за правилом a[i] = 2i − 7 та виведіть його на екран. Знайдіть кількість та суму від'ємних елементів масиву.

Лекція. Табличні величини. Алгоритми роботи з табличними величинами. Поняття двовимірного масиву

Мета, завдання лекції: ознайомлення з поняттям двовимірного масиву, правилами опису, заповнення та виведення на екран двовимірних масивів; формування вмінь складати та реалізовувати програми опрацювання двовимірних масивів.

План і організаційна структура лекції:

1. Поняття двовимірного масиву.
2. Опис двовимірного масиву.
3. Звертання до елементу двовимірного масиву.
4. Заповнення масиву.
5. Виведення двовимірного масиву на екран.
6. Використання масивів при розв'язуванні задач.

Зміст лекційного матеріалу:

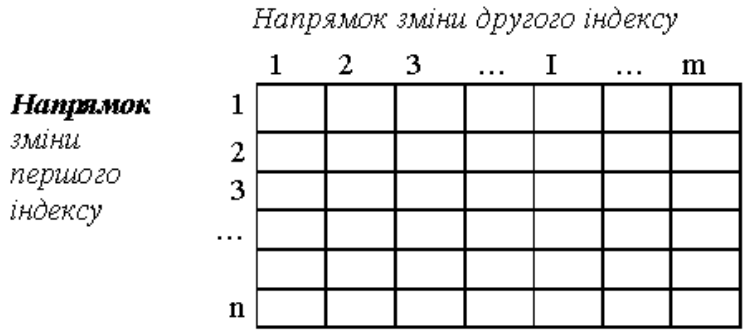
Розглянемо ***двовимірний масив*** (прямокутну таблицю) – структуру, за допомогою якої можна зберігати дані однакового типу в пам'яті комп'ютера. В прямокутну таблицю записують дані спостережень, експериментів, оцінки учня з усіх предметів за рік.

Прикладом двовимірного масиву може бути фрагмент учнівського табеля за чотири чверті навчання з п'яти предметів:

I	II	III	IV
8	9	10	9
7	8	9	8
8	8	8	9

9	9	10	10
9	9	11	10

Двовимірний масив – це масив, де кожному елементу ставиться у відповідність два індекси.



Для початку роботи з масивом готуємо місце в пам'яті у вигляді прямокутника, що має задану кількість рядків і стовпчиків. Для цього описуємо його в розділі оголошень, використовуючи зарезервоване слово **array**, після якого в квадратних дужках вказуємо розмірність масиву, причому враховуємо, що на першому місці вказуються індекси рядків, а на другому – стовпчиків, і обов'язково тип елементів.

Опис двовимірного масиву

I-й спосіб

```
const n := 100;
      m := 100;
var A : array[1 .. n, 1 .. m] of real;
    D : array[1 .. 10, 1 .. 100] of integer;
```

II-й спосіб

```
type My2DMasyv = array[1 .. n,1 .. m] of<назва базового типу>;
var b1, b2, b3 : My2DMasyv;
```

III-й спосіб (Сталий масив)

```
type MasyvTabel = array[1 .. 5, 1 .. 4] of integer;
const b: MasyvTabel = ((8,9,10,9), (7,8,9,8), (8,8,8,9), (9,9,10,10), (9,9,11,10));
```

або

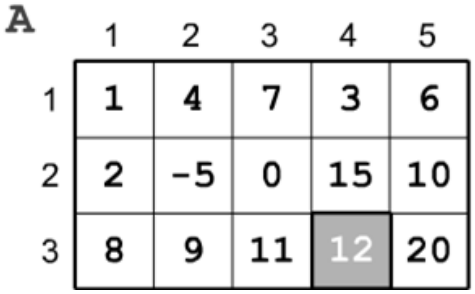
```
const b : array[1 .. 5, 1 .. 4] of integer = ((8,9,10,9), (7,8,9,8), (8,8,8,9),
(9,9,10,10), (9,9,11,10));
```

Звертання до елементу двовимірного масиву:

Ім'я_масиву [<індекс_рядка>, <інд_стовпчика>]

Наприклад, b[1, 1] – елемент, який знаходиться на перетині першого рядка і першого стовпця, b[3, 4] – елемент, який знаходиться на перетині

третього рядка і четвертого стовпця.



A[1,1]=1;
A[1,2]=4;
A[1,3]=7;
A[2,3]=0;
A[3,4]=12.

Заповнення масиву

- **з клавіатури:**

```
for i := 1 to n do
    for j :=1 to m do
        begin
            write ('введіть A[', i,' , ', j , ']: ');
            readln (A[i, j]);
        end;
```

- ***за формулою:***

```
for i :=1 to n do
for j :=1 to m do
A[i, j] := i * i − 10 {або будь-яка формула};
```

- ***випадковим чином із проміжку [K, L]:***

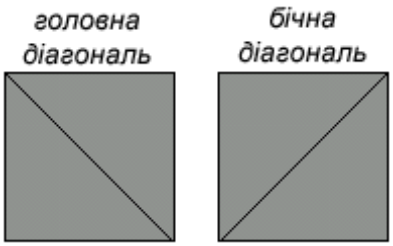
```
for i:=1 to n do
for j:=1 to m do
A[i, j] := random (L-K) + K;
```

Виведення двовимірного масиву на екран

```
    for i := 1 to n do
    begin
        for j := 1 to m do
        write (A[i, j] : 8);          {виведення в рядок}
        writeln;                      {перехід на новий рядок}
    end;
```

Для роботи з масивом потрібен будь-який оператор повторення. У двовимірному масиві необхідно використовувати їх два: один цикл, внутрішній, потрібен для переходу між елементами рядка (тобто, по стовпчиках), а другий, зовнішній, – для переміщення між рядками.

Якщо в матриці (двовимірному масиві) кількість рядків і стовпчиків однакова, то таку матрицю називають ***квадратною*** (на відміну від звичайної прямокутної таблиці). Тільки в квадратних матрицях існують головна та бічна діагоналі.



Елементи, що стоять на головній діагоналі, мають індекси (1, 1), (2, 2), (3, 3), ... , (i, i). ..., (n, n), тобто номер рядка дорівнює номеру стовпчика! Елементи, що стоять на бічній діагоналі, мають такі індекси (1, n), (2, n-1), (3, n-2), ..., (i, n+1-i), (n,1), тобто індекси елементів взаємозалежні за формулою j=n+1-i.

Задача 1. Скласти програму для формування таблиці множення, занесення її до двовимірного масиву та виведення масиву на екран.

```
program TABLICA;
const n = 910;
var P : array[1 .. n, 1 .. n] of integer;
      i , j : integer;
begin
    for i:=1 to n do
      begin
          for j :=1 to n do
              begin
                  p[i, j] := i * j;
                  write (p[i , j];
              end;
              writeln;            {перехід на новий рядок}
      end
end.
```

Задача 2. Обчислити суму елементів у рядку, стовпці чи вздовж діагоналі матриці.

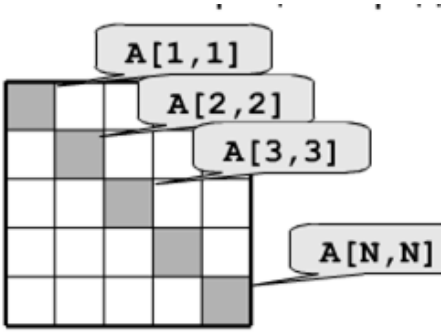
Наприклад, суму елементів другого рядка обчислюють так:

```
s := 0;
  for j : = 1 to m do
     s:=s + a[2, j];
```

Наприклад, суму елементів, які знаходяться вздовж діагоналі квадратної матриці обчислюють так:

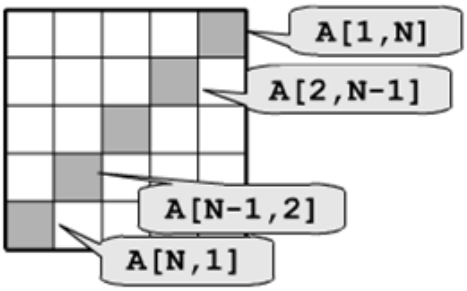
```
S1 := 0;
  for i := 1 to n do
     s1 := s + a[i ,i];
```

Наприклад, вивести на екран елементи головної діагоналі квадратної матриці можна так:




```
for i := 1 to n do
    write (a[i, i] : 5);
```

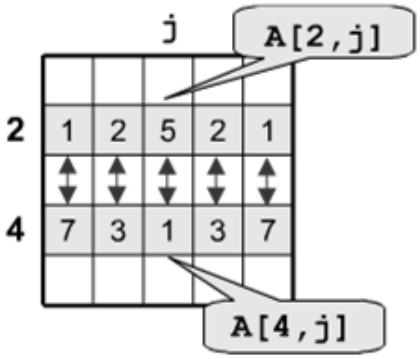
Наприклад, вивести на екран елементи допоміжної діагоналі квадратної матриці :



Сума номерів рядка і стовпця: N+1; індекси елементів взаємозалежні за формулою j = n + 1 - i.

```
for i := 1 to n do
    write (a[i, N + 1 - i] : 5);
```

Задача. Перестановка рядків і стовпців. В матриці з N рядків і M стовпців поміняти місцями 2-й і 4-й рядок.




```
for j := 1 to m do
  begin
    c := A [2, j];
    A [2, j] := A [4, j];
    A [4, j] := c;
  end;
```

Задача. До третього стовпця матриці додати шостий.

```
for i := 1 to n do
    A [i, 3] := A [i, 3] + A [i, 6];
```

Практична робота. Створення та реалізація програм опрацювання двовимірних масивів

Мета: формування навичок складання і виконання програм обробки двовимірних масивів.

Завдання:

- *студенти повинні знати:* поняття двовимірного масиву, його опис, алгоритми роботи з табличними величинами;
- *студенти повинні вміти:* створювати програми опрацювання двовимірних масивів.

Обладнання: зошит, середовище програмування.

Хід виконання:

1. Дати відповіді на запитання:
   1) Поняття двовимірного масиву.
   2) Опис двовимірного масиву .
   3) Звертання до елементу двовимірного масиву.
   4) Заповнення масиву.
   5) Виведення двовимірного масиву на екран.
   6) Використання масивів при розв'язуванні задач.
2. Скласти та реалізувати програми розв'язання задач:

   1) Заповнити двовимірний масив з трьох рядків і чотирьох стовпців значеннями з клавіатури, вивести масив на екран у вигляді таблиці і знайти суму елементів третього стовпця.

   2) Побудувати і вивести на екран двовимірний масив $d_{ij} = 2i \cos j$, i = 1 .. 4, j = 1 … 4. Знайти мінімальний елемент масиву і добуток від'ємних елементів масиву.

   3) Дано масив A(1 .. 5 , 2 .. 6). Скласти програму заміни всіх його елементів, що більші за число10, на нулі. Вивести масив на екран.

Практична робота. Створення та реалізація програм опрацювання двовимірних масивів

Мета: формування навичок складання і виконання програм обробки двовимірних масивів.

Завдання:

- *студенти повинні знати:* поняття двовимірного масиву, його опис, алгоритми роботи з табличними величинами;
- *студенти повинні вміти:* створювати програми на опрацювання двовимірних масивів.

Обладнання: зошит, середовище програмування.

Хід виконання:

Реалізувати завдання згідно заданого варіанту.

В-1

1. Дано прямокутну таблицю А, яка складається з 5 рядків і 4 стовпців. Елементи таблиці задано таким чином: $a_{i,j} = \sin(i + j)$. Вивести масив А на екран у вигляді таблиці і обчислити суму квадратів елементів таблиці А.

2. В масиві D[1 .. 4, 1 .. 5], елементи якого довільні цілі числа, знайти мінімальний елемент цього масиву.

В-2

1. Дано прямокутну таблицю, яка складається з 4 рядків і 5стовпців. Елементи таблиці задано таким чином: $b_{i,j} = \cos(i + j)$. Вивести масив на екран у вигляді таблиці і обчислити суму додатних елементів четвертого стовпця масиву В.

2. У двовимірному масиві Н[1 .. 5, 1 .. 6] всі від'ємні елементи замінити на їх модулі і вивести новий масив на екран дисплея.

В-3

1. Дано прямокутну таблицю А, яка складається з 5 рядків і 4 стовпців. Елементи таблиці задано таким чином: $a_{i,j} = \sin(i + j)$. Вивести масив А на екран у вигляді таблиці і обчислити середнє арифметичне додатних елементів таблиці В.

2. Дано двовимірний масив G [1 . . 4, 1. .3], який містить довільні цілі числа. Знайти кількість та суму елементів, таких що 0 < G[i, j] < 10.

В-4

1. Дано двовимірний масив К [1 . . 4, 1 . . 4], елементи якого знаходяться за формулою К [i] = 2i * cosj. Обчислити добуток від'ємних елементів у ньому. Вивести масив на екран.

2. Дано два масиви A [1 . . 3, 1 . . 4] та B [0 . . 2, 0 . . 3], у яких містяться цілі числа. У якій таблиці: A чи B, більше додатних елементів?

B-5

1. Знайти максимальний елемент в масиві K [0 . . 4, 0 . . 5], елементи якого знаходяться за формулою K [i, j] := 3j - 2i. Вивести на екран цей масив та максимальний елемент.

2. Дано масив A[1 . . M, 1 . . N], N = 4, M = 5. Скласти програму підрахунку суми елементів першого і останнього рядків цього масиву.

B-6

1. Дано масив A [1 .. 4, 1 .. 3], що містить елементи A[i, к] = 2i + 2к. Обчислити суму елементів цього масиву.

2. Двовимірний масив M [1 .. 3, 1..3], містить довільні дійсні числа. Знайти кількість елементів, яка рівні числу 5.

B-7

1. Дано масив A [1 .. 4, 1 .. 4], що містить елементи A[i, к] = 2i + 2к. Знайти кількість елементів, які більші за число 10.

2. В масиві D [0 .. 7,0 .. 5], елементи якого довільні цілі числа, обчислити добуток від'ємних елементів масиву.

Додаткові завдання:

1. В масиві Q [0 .. 5, 0.. 3] від'ємні елементи замінити нулями. Вивести на екран початковий та змінений масив.

2. Дано масив A [1 .. 4, 1 .. 3], що містить елементи A [i, к] = 2i + 2к. Обчислити суму елементів цього масиву.

3. В масиві D [0 .. 7, 0 .. 2], елементи якого довільні цілі числа, обчислити добуток від'ємних елементів масиву.

4. Двовимірний масив M [1 ..3 , 1..3], містить довільні дійсні числа. Знайти кількість елементів, яка рівні числу 5.

5. В масиві Q [0 .. 5, 0 .. 3] від'ємні елементи замінити нулями. Вивести на екран початковий та змінений масив.

6. Створити масив P з елементами, що шукаються за формулою P[i, k] = A[i, k] + B[i, k] , елементи A[i, k] та B[i, k] взято із попередньої задачі. Вивести на екран всі три масиви.

7. Дано масив A [1 .. 4, 1 .. 4], що містить елементи A [i, к] = 2i + 2к. Обчислити добуток елементів головної діагоналі.

8. В масиві D [1 .. 4, 0 .. 3], елементи якого довільні цілі числа, обчислити суму додатних елементів верхнього трикутника.

9. Двовимірний масив M [1 .. 3, 1 .. 3], містить довільні дійсні числа. Знайти у нижньому трикутнику кількість елементів, яка рівні числу 0.

10. У масиві P [0 .. 7, 0 .. 7], елементи якого знаходяться за формулою P[i, k] = 3i - 2k, знайти кількість додатних елементів у верхньому трикутнику та кількість від'ємних елементів у нижньому трикутнику.

11. В масиві Q [0 .. 5, 0 .. 5], елементи якого шукаються за формулою Q [i, k] = k, всі елементи головної діагоналі замінити нулями. Вивести на екран змінений масив.

12. Знайти максимальний елемент в масиві K [0 .. 11], елементи якого знаходяться за формулою K[i] := 10 - 2i. Вивести на екран цей масив та максимальний елемент.

13. В масиві D [1 .. 4, 1 .. 5], елементи якого довільні цілі числа, знайти мінімальний елемент цього масиву.

Лекція. Алгоритми сортування і пошуку елементів масивів

Мета, завдання лекції: ознайомлення з поняттям сортування масивів; основними способами сортування масивів; формування вмінь складати та реалізовувати програми опрацювання двовимірних масивів.

План і організаційна структура лекції:

1. Способи сортування масивів.
2. Сортування методом вставки.
3. Сортування методом вибору.
4. Сортування методом обміну.

Зміст лекційного матеріалу:

Однією з найбільш поширених операцій обробки масивів є їх упорядкування, або сортування. ***Упорядкування масиву*** – це зміна порядку розташування його елементів за певним критерієм. Наприклад, числовий масив можна упорядкувати за зростанням значень його елементів або за їх спаданням, а масив рядків можна відсортувати в алфавітному порядку. Найчастіше сортування масиву здійснюється з метою полегшення подальшого пошуку.

Відомо багато методів сортування масиву, що відрізняються швидкодією й обсягом оперативної пам'яті, яка при цьому використовується. Найбільш відомими елементарними методами

сортування масиву є:

- сортування вставкою (включенням);
- сортування вибором;
- сортування обміном (бульбашкове сортування).

Сортування методом вставки

На кожному кроці цього методу масив розділений на дві частини: ліву, вже відсортовану, та праву, ще невідсортовану. Перший елемент правої частини вставляється до лівої частини так, щоб ліва частина залишалася відсортованою. У результаті відсортована частина збільшується на один елемент, а невідсортована – на один елемент зменшується. Отже, на кожному кроці алгоритму сортування методом вставки слід виконати дві операції: пошук позиції для вставки елемента та власне його вставку із подальшим зсувом на одну позицію вправо від елементів відсортованої частини. Цей зсув «затре» перший елемент невідсортованого підмасиву останнім елементом відсортованого. Спочатку відсортованим підмасивом вважаємо перший елемент, а решту елементів масиву відносимо до невідсортованої частини.

Формалізуємо алгоритм сортування методом вставки, а також наведемо його програмну реалізацію. Підмасив, що містить елементи з другого до останнього, будемо вважати частиною невідсортованого. Поки ця частина не стане порожньою, виконуватимемо послідовність таких дій.

1. Зберегти перший елемент невідсортованого підмасиву в допоміжній змінній.
2. Визначити позицію вставки збереженого елемента у масив. Для цього дотримуватися перелічених далі вказівок:
   2.1. Вважати перший елемент масиву поточним.
   2.2. Доки елемент для вставки більше за поточний, збільшувати індекс поточного елемента.
3. Вставити збережений на кроці 1 елемент на знайдену позицію вставки, зсунувши на одну позицію вправо решту відсортованої частини.
4. Пересунути початок невідсортованої частини на одну позицію вправо.

```
program ex6_1;              {сортування вставкою}
uses crt;
var n, i, j, k : integer;   {кількість та індекси елементів}
```

```
a : array[1 .. 10] of integer;
tmp : integer;  {елемент, що вставляється у відсортовану частину масиву}
begin
clrscr;
randomize;              {ініціалізувати генератор випадкових чисел}
writeln('insertion sort');
writeln('enter number of the components (<=10)');
        readln(n);   {ввести кількість елементів масиву}
        for i := 1 to n do   {генерувати масив}
        a[i] := random(30);
writeln ('generated array');
for i := 1 to n do       {вивести згенерований масив}
write (a[i], '   ');
writeln;
writeln ('sort process');
   for i := 2 to n do           {сортувати методом вставки}
   begin                        {i - початок невідсортованого підмасиву}
      tmp := a[i];              {вибрати елемент для вставки}
      j:=1;                     {цикл пошуку позиції вставки}
      while tmp > a[j] do   {якщо елемент, що вставляється, менший або
      рівний поточному, то j фіксує позицію вставки }
      j:=j+1;
      for k := i - 1 downto j do  {зсунути вправо елементи невідсортованої
      частини }
      a[k + 1] := a[k];
      a[j] := tmp;              {вставка вибраного елемента у позицію j}
      for k := 1 to n do        {виведення проміжних результатів}
      write (a[k], ' ');
      writeln;
   end;
writeln ('sorted array');
for i:=1 to n do                {виведення відсортованого масиву}
write (a[i], '  ');
readln;
end.
```

Сортування методом вибору

Цей метод, як і метод сортування вставкою, розділяє масив на дві частини: ліву, вже впорядковану, та праву, ще не впорядковану. Вибирають найменший елемент невідсортованої частини. Цей елемент

міняють місцями з її першим елементом, збільшуючи на одиницю довжину відсортованої частини масиву. Отже, на першому кроці алгоритму невпорядкованою частиною є весь масив, з котрого вибирають мінімальний елемент. Цей елемент міняють місцями з першим елементом масиву. На другому кроці невпорядковану частину масиву складають елементи від другого до останнього. Серед цих елементів вибирають найменший, котрий міняють місцями з другим. Процес триває доти, доки у невідсортованій частині не залишиться один елемент. Формалізуємо алгоритм сортування методом вибору і реалізуємо його мовою Pascal. Весь масив вважати невідсортованою частиною. Поки ця частина містить більше одного елемента, виконувати такі дії:

1. Вибрати перший елемент невідсортованої частини масиву і вважати його мінімальним; запам'ятати індекс цього елемента.
2. Для елементів від наступного після вибраного і до останнього повторювати такі дії:
   2.1. Порівняти вибраний елемент і поточний.
   2.2. Якщо вибраний елемент більший за поточний, запам'ятати поточний елемент як мінімальний, а його індекс – як індекс мінімального елемента.
3. Поміняти місцями мінімальний і вибраний на кроці 1 елементи.
4. Пересунути початок невідсортованої частини на одну позицію вправо.

```
program ex6_2;       {сортування методом вибору}
uses crt;
var n, i, j : integer;     {кількість та індекси елементів}
a : array[1 .. 10] of integer;
min, imin : integer;    {мінімальний елемент і його індекс}
begin
clrscr;
randomize;              {ініціалізувати генератор випадкових чисел}
writeln (‘selection sort’);
writeln (‘enter number of the components (<=10)’);
readln (n);             {ввести кількість елементів масиву}
for i :=1 to n do       {генерувати масив}
        a[i] := random(30);
        writeln ('generated array');
for i := 1 to n do      {вивести згенерований масив)
write(a[i], '  ');
```

```
writeln;
writeln ('series of selection');
for i := 1 to n - 1 do
    begin
    min := a[i];            {пошук мінімального елемента в діапазоні від
    і-го до останнього елемента}
    imin := i;              {індекс мінімального елемента}
    for j := i + 1 to n do  {пошук мінімального елемента}
    if min > a[j] then
            begin
               min := a[j];
               imin := j;
            end;
    a[imin] := a[i];
    a[i] := min;        {обмін місцями мінімального та поточного
    елементів}
    for  j:= 1 to n do {виведення проміжних результатів}
    write (a[j] ,'  ');
    writeln;
    end;
writeln ('sorted array');
for i := 1 to n do      {виведення відсортованого масиву}
write(a[i], '  ');
readln;
end.
```

Сортування методом обміну

Базовою операцією в цьому методі є порівняння двох сусідніх елементів масиву. Якщо їх розташування суперечить умові впорядкування, вони міняються місцями. Послідовне застосування такої операції до всіх пар елементів масиву, від останньої пари до першої, дозволить виявити найменший елемент в першій позиції. Друга назва цього метода – бульбашкове сортування – пояснюється схожістю процесу обміну місцями сусідніх елементів зі спливанням більшої бульбашки. Під час сортування методом обміну впорядкованою буде ліва частина масиву, а щойно описаний процес повторюється для правої частини, котра на кожній ітерації методу зменшуватиметься на один елемент.

Розглянемо алгоритм сортування *методом обміну* та його реалізацію мовою Pascal:

1. Установити лічильник ітерацій рівним одиниці.
2. Для елементів масиву, від останнього до елемента з індексом, що дорівнює поточному значенню лічильника ітерацій, повторювати такі дії:
   2.1. Якщо поточний елемент більший за попередній, поміняти ці елементи місцями.
   2.2. Перейти до попереднього елемента.
3. Збільшити лічильник ітерацій. Якщо значення лічильника дорівнює кількості елементів масиву, завершити сортування.

```
program ex6_3;              {сортування обміном}
uses crt;
var n, i, j, k : integer;   {кількість та індекси елементів}
a : array[1 .. 10] of integer;
tmp : integer;              {допоміжний елемент для обміну}
begin
clrscr;
randomize;
writeln('exchange sort');
writeln('enter number of the components (<=10)');
readln(n);
   for i := 2 to n do       {генерувати масив}
a[i] := random(30);
writeln('generated array');
for i:=l to n do            {вивести згенерований масив}
write( a[i], ' ');
writeln;
writeln('sort process');
   for i := 2 to n do       {сортувати методом обміну}
for j := n downto i do
if a[j] < a[j - 1] then
    begin      {поміняти елементи місцями}
        tmp := a[j];
        a[j] := a[j - 1];
        a[j - 1] := tmp;
        {виведення проміжних результатів}
        for k := 1 to n do
        write (a[k], ' ');
        writeln;
    end;
```

```
 writeln('sorted array');
for i := 1 to n do        {вивести відсортований масив}
write (a[i], ' ');
readln;
end.
```

Практична робота. Використання алгоритмів сортування

Мета: формування навичок розв’язування задач на впорядкування масиву.

Завдання:

- *студенти повинні знати:* поняття масиву, його опис, алгоритми роботи з табличними величинами, методи впорядкування масивів;
- *студенти повинні вміти:* створювати програми на впорядкування одновимірних масивів.

Обладнання: зошит, середовище програмування.

Хід виконання:

1. Опрацювати теоретичний матеріал, розглянути приклади алгоритмів сортування та дати відповіді на запитання:
   1) Способи сортування масивів.
   2) Сортування методом вставки.
   3) Сортування методом вибору.
   4) Сортування методом обміну.

2. Виконати індивідуальне завдання згідно варіанту.

Скласти алгоритм та програму для розв’язування задачі:

Задано одномірний масив.

1. Впорядкувати його другу половину за спаданням непарних елементів.
2. Впорядкувати його першу половину за зростанням елементів з парними індексами.
3. Впорядкувати останні k елементів за спаданням значень парних елементів.
4. Впорядкувати елементи, розташовані до першого від'ємного елемента в порядку зростання елементів.
5. Впорядкувати елементи, розташовані після максимального елемента, за спаданням значень елементів.
6. Впорядкувати елементи, розташовані між першим і останнім від'ємним елементом, за зростанням значень елементів.

7. Впорядкувати елементи, розташовані між мінімальним і максимальними елементами, за спаданням значень елементів.
8. Впорядкувати елементи, розташовані до мінімального елемента, за зростанням значень елементів.
9. Впорядкувати елементи, розташовані після мінімального елементам за спаданням значень елементів.
10. Впорядкувати елементи, розташовані до максимального елементам за зростанням значень елементів.
11. Впорядкувати елементи, розташовані між першими і другими додатніми елементами за спаданням значень елементів.
12. Впорядкувати перші k парних елементів за спаданням значень елементів.
13. Впорядкувати тільки додатні елементи за зростанням.
14. Впорядкувати тільки ті елементи, що більше заданого n за зростанням.
15. Впорядкувати тільки елементи з непарними індексами за спаданням.

Лекція. Опис мовою Паскаль вказівок звернення до процедур

Мета, завдання лекції: ознайомлення з поняттям підпрограм (процедур), їх структурою та описом.

План і організаційна структура лекції:

1. Поняття процедури, її структура та опис.
2. Процедури без параметрів та процедури з параметрами.
3. Локальні та глобальні змінні у процедурах.
4. Параметри-значення і параметри-змінні у процедурах.

Зміст лекційного матеріалу:

Підпрограмою називається найменована логічно закінчена група вказівок, яку можна викликати для виконання довільну кількість разів з різних місць програми.

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. Формат запису процедури:

procedure <ім'я процедури>;
<тіло процедури>;

Процедури без параметрів

Використання процедур дає можливість підвищити продуктивність праці, зробити програму більш наочною, спростити розробку великих програм тощо.

Так, наприклад, програму знаходження об'єму конуса можна розбити на декілька процедур:

1) **procedure** InputDate;
2) **procedure** Vkazivka;
3) **procedure** OutputDate.

Процедура InputDate виконує введення даних в програму, Vkazivka – виконує вказівки для знаходження об'єму конуса, процедура OutputDate виконує виведення даних на дисплей.

Всі змінні, які використовуються в процедурах без параметрів, описуються в основній програмі.

Задача. Скласти програму знаходження об'єму конуса.

```
program Vkonus;
const P = 3.14;
var R, H, V : real;
    procedure InputDate;
    begin
      write ('r=' );
      readln (R);
      write ('b= ');
      readln (H);
    end;
        procedure Vkazivka;
        begin
            V := P * Sqr(R) * H/3;
            end;
            procedure OutputDate;
            begin
            writeln ('V= ', V);
            end;
            begin       {Основна програма}
            InputDate;
            Vkazivka;
            OutputDate;
```

end.

Процедури з параметрами

У процедурі можна оголошувати константи, змінні, інші процедури і функції. Розділ опису в процедурах має таку саму структуру, як і в основній програмі.

Оголошені всередині процедури змінні називаються ***локальними*** по відношенню до даної процедури. Локальні змінні не доступні 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.

Приклад програми, в якій використовується принцип локалізації

```
program LOKALIZACIA;
var A, B : integer;
procedure LOKAL;
var A, X : char;
    begin
      A : = '!'; X := '?'; B := B + 1;
    end;
       begin
          B := 100;
          LOKAL;
          writeln ('a= ', A,' b= ', B);
       end.
```

Змінна X – локальна для процедури LOKAL, тому головна програма не може змінити її значення або звернутись до неї.

Змінна B – глобальна для процедури LOKAL, вона відома і в програмі, і в процедурі. Оскільки в процедурі немає іншого оголошення для змінної B, то всі зміни, які відбуваються зі значенням B у процедурі, зберігаються і після виходу з неї.

Змінна A в основній програмі належить до цілого типу, але в процедурі LOKAL є своя власна змінна A, яка належить до типу char. Змінна A з головної програми недоступна в процедурі LOKAL. Всі зміни, які відбуваються зі значенням A в процедурі, втрачають свій зміст при виході з неї. Отже, в результаті виконання програми LOKAL на екран будуть виведені два числа: a = 0 і b = 101.

У Паскалі введено розширення мови, завдяки чому в процедурі

можуть бути доступними як локальні відносно неї змінні, так і змінні з такими ж іменами, які описані в основній програмі. Для того, щоб реалізувати таку можливість, звернення до змінних в процедурі виконується таким чином: до локальних змінних звернення реалізується просто по їх імені (тобто як в стандартній мові). Для звернення до змінних, які однойменні з локальними, але введені в основній програмі, використовується така форма:

ім'я програми. ім'я змінної

ім'я програми – ім'я, яке вказано в заголовку програми **program**.

ім'я змінної – ідентифікатор змінної, вказаний в розділі **var**.

З урахуванням цього програму LOKALIZACIA можна перетворити так, щоб у процедурі LOKAL були доступні однойменні змінні, описані як у самій процедурі, так і в основній програмі.

Звернення до процедур і функцій

```
program LOKALIZACIA1;
var A, B : integer;
procedure LOKAL;
var A, X : char;
      begin
        A := '!'; X := '?'; B := B + 1;
        LOKALIZACIA1.A := LOKALIZACIA1.A + 12;
      end;
       begin
          A := 0; B := 100;
          LOKAL;
          writeln (’a=’, A, ’b= ’, B);
       end.
```

У результаті виконання програми Lokalizacia1 на екрані появляться такі значення змінних: a = 12 і b = 101.

Параметри-значення і параметри-змінні у процедурах

1. Параметри-значення

Наведений вище формат опису процедури можна доповнити таким чином: після заголовка процедури в круглих дужках можуть вказуватись змінні (з допомогою яких у процедуру передаються дані) і їх типи, які називаються ***параметрами-значеннями.*** Перед ними відсутнє службове слово **var**.

Формат запису процедури:

procedure <ім'я процедури (ім'я змінної : тип змінної)>;
<тіло процедури>;

Змінні, які описані в заголовку процедури, називаються

формальними параметрами. Змінні або константи, які описані у вказівці процедури при її виклику, називаються ***фактичними параметрами***.

При зверненні до процедури з параметрами всі формальні параметри замінюються фактичними в порядку їх перелічення.

```
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-ий спосіб }
         param (10, 100)        { 2-ий спосіб }
      end.
```

Змінні A і B – це формальні параметри. Змінні C і D – фактичні параметри. Значення фактичних параметрів C = 10 і D = 100 передаються формальним параметрам A і B.

Зверніть увагу на те, як двома способами можна викликати процедуру і передати значення змінним.

Такий спосіб передачі параметрів 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.

2. Параметри-змінні

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

При використанні цього способу заголовок процедури змінюється таким чином: перед ідентифікатором формального параметра в заголовку процедури вказується службове слово **var**.

Змінні, перед якими записане службове слово var, називаються ***параметрами-змінними.***

При виконанні процедури формальні параметри-змінні замінюються

фактичними параметрами. Будь-які зміни значення формального параметру-змінної приводять до зміни значення фактичного параметру-змінної. За допомогою параметрів-змінних в основну програму передаються результати дії вказівок над даними.

Приклад. Дано дві трійки чисел: A1, B1, CI і A2, B2, C2. Знайти значення сум: S1 = min(Al, B1, CI) + min(A2, B2, C2); S2 = max(Al, B1, CI) + max(A2, B2, C2).

Розв'язування:

Для знаходження min і max з трьох чисел використаємо процедуру MinMax.

```
program PRIKLAD;
var A1, B1, C1, A2, B2, C2, MIN1, MIN2, MAX1, MAX2, S1, S2 : real;
procedure MinMax (A, B, C : real; var MIN, MAX : real);
    begin
      MAX := A;
      if MAX < B then MAX := B;
      if MAX < C then MAX := C;
      MIN := A;
      if MIN > B then min :=B;
      if MIN > C then MIN := C;
    end;
          begin
              write ('A1= ');
              readln (A1);
              write ('B1= ');
              readln (B1);
              write ('C1= ');
              readln (C1);
              write ('A2= ');
              readln (A2);
              write (’B2= ’);
              readln (B2); write (’C2= ’); readln (C2);
              MinMax (A1, B1, C1, MIN1, MAX1);
              MinMax (A2, B2, C2, MIN2, MAX2);
              S1 := MIN1 + MIN2; S2 := MAX1 + MAX2;
              writeln('S1= ', S1);
              writeln('S2= ', S2);
          end.
```

Лекція. Опис мовою Паскаль вказівок звернення до функцій

Мета, завдання лекції: ознайомлення з поняттям функцій, їх структурою та описом; формування вмінь розв'язувати задачі з використанням підпрограм.

План і організаційна структура лекції:

1. Поняття функції.
2. Приклади розв’язання задач з використанням процедур і функцій.

Зміст лекційного матеріалу:

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

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

Звернення до функції (обов'язково повинно бути включене у вираз як операнд) має такий вигляд: <ім'я функції> (список фактичних параметрів).

Приклад. Знайти значення числа комбінацій n!: $C_n^m = \frac{n!}{m!(n-m)!}$

Знаходження значення факторіалу числа оформимо у вигляді функції. Тоді програма розв'язання даної задачі матиме вигляд:

```
program KOMBINACIJA;
var N, M, C : integer;
function FACT (K : integer) : integer;
var i, F : integer;
    begin
    F := 1;
    for i := l to K do F := F*i;
    FACT := F;
    end;
        begin
        write ('n= '); readln (N);
        write ('m= '); readln (M);
        C := FACT(N) div (FACT(M) * FACT(N-M));
        writeln ('Кількість комбінацій з ', n,' по ', m,' = ', C);
        end.
```

Зверніть увагу на те, що в самому тілі функції FACT необхідно змінній, ім'я якої співпадає з ім'ям самої функції, присвоїти значення результату виконання функції: FACT := F.

Приклади завдань з використанням процедур і функцій

Процедури без параметрів

1. Обчислити довжини п’яти кіл і площі відповідних п’яти кругів, якщо їхні радіуси набувають значень 2, 4, 6, 8, 10. Використайте:

а) процедуру для обчислення довжини кола і процедуру для обчислення площі круга;

б) одну процедуру.

```
program one;
var r : integer;
    l, S : real;
procedure krug (r : integer; var A, B : real);
  begin
    A := 2 * pi * r;
    B := pi * sqr(r);
    writeln ('Довжина l = ', A : 5 : 2);
    writeln ('Площа S = ', B : 5 : 2);
  end;
begin
  write (Введіть r');
  readln (r);
  krug (r, l, S);
end.
```

2. Обчисліть площі поверхонь і об'єми трьох повітряних куль $S = 4\pi R^2$, $V = (4/3)\pi R^3$ за відомими радіусами, використавши процедуру для обчислення площі і об'єму. Радіуси куль відповідно дорівнюють 5, 10, 15 см.

```
program two;
var r : integer; v, S : real;
procedure kula;
  begin
    S :=4 * pi * sqr(r);
    v := (4/3) * pi * sqr(r) * r;
  end;
begin
  write (Введіть r'); readln (r);
  kula;
  writeln ('S = ', S : 5 : 2);
  writeln ('V = ', V : 5 : 2);
end.
```

Процедури з параметрами

3. Складіть процедуру, яка отримує три цілі числа, а повертає їхню суму, добуток і середнє арифметичне.

```
program three;
```

```
var A, B, C : integer;
procedure chusla (L, M, N : integer);
var s, d, ser : real;
begin
    S := L + M + N; writeln ( 'suma = ', S : 5 : 0);
    D := L * M * N; writeln ( 'dobutok = ', D : 5 : 0);
    ser := S/3; writeln ( 'ser = ', ser : 5 : 2);
end;
    begin
        write (Введіть a'); readln (A);
        write (Введіть b'); readln (B);
        write (Введіть c'); readln (C);
        chusla (A, B, C);
    end.
```

Приклад: Знайти довжину сторін трикутника АВС, якщо відомі координати його вершин $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$. Необхідно скористатись формулою:

$$d(AB) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

І спосіб (функції)

```
program zad1;
var x1, x2, x3, y1, y2, y3 : integer;
    S1, S2, S3 : real;
function kvadrat(a, b : integer) :
integer;
var k : integer;
begin
   k := sqr (a - b); kvadrat := k;
end;
begin
write ('x1='); readln (x1);
write ('x2='); readln (x2);
write ('x3='); readln (x3);
write ('y1='); readln (y1);
write ('y2='); readln (y2);
write ('y3='); readln (y3);
S1 := sqrt (kvadrat(x1, x2) +
kvadrat(y1, y2));
S2 := sqrt (kvadrat(x1, x3) +
kvadrat(y1, y3));
S3 := sqrt (kvadrat(x3, x2) +
```

ІІ спосіб (процедури)

```
program zad2;
var x1, x2, x3, y1, y2, y3 : integer;
procedure Dovz(A, B, C, D : integer);
var k : real;
begin
    k := sqrt(sqr(a - b) +sqr(c - d);
    write (k : 7 : 2);
    readln;
end;
begin
write ('x1='); readln (x1);
write ('x2='); readln (x2);
write ('x3='); readln (x3);
write ('y1='); readln (y1);
write ('y2='); readln (y2);
write ('y3='); readln (y3);
writeln (Довжина першої сторони:',
Dovz(x1, x2, y1, y2);
writeln (Довжина другої сторони:',
Dovz(x1, x3, y1, y3);
writeln (Довжина третьої сторони:',
```

```
kvadrat(y3, y2));
writeln ('S1=', S1 : 7 : 2);
writeln ('S2=', S2 : 7 : 2);
writeln ('S3=', S3 : 7 : 2);
end.
```

```
Dovz(x3, x2, y3, y2);
end.
```

Практична робота. Підпрограми. Використання процедур і функцій

Мета: формування навичок складання та реалізації програм з використанням процедур та функцій.

Завдання:

- *студенти повинні знати:* поняття підпрограми, процедури з параметрами, процедури без параметрів, функції, локальної та глобальної змінної;
- *студенти повинні вміти:* створювати програми з використанням процедур і функцій.

Обладнання: зошит, середовище програмування.

Хід виконання:

1. Вивчити можливості мови програмування Паскаль по використанню процедур і функцій:

а) процедури без параметрів;
б) локальні та глобальні змінні в процедурах;
в) фактичні та формальні параметри;
г) поняття та формат опису функції;
д) використання функцій в програмах.

2. Ознайомитись з прикладом реалізації розв'язку задачі:

Створити функцію для обчислення tg(x) та обчислити значення виразу tg(x)+ctg(x)+tg2(x).

```
program Myfunc;
uses Crt;
var x,y : real;
    function tg(x : real) : real;
      begin
        tg(x) := sin(x)/cos(x)
      end;
begin clrscr;
   writeln ('Введіть x');
   readln (x);
```

```
    y := tg(x) + 1/tg(x) + sqr(tg(x));
    writeln ('y=', y : 5 : 2);
    readln
end.
```

3. Реалізувати програми розв'язування задач згідно заданого варіанту:

B-1

1) Скласти програму для обчислення значення виразу:

$y = \frac{a!}{1+\sqrt{b}} + \frac{b!}{1+\sqrt{a}}$, обчислюючи факторіал у підпрограмі.

2) Обчислити довжину ламаної ABC, використовуючи підпрограму для обчислення відстані між двома точками. Скласти програму і перевірити для довільних точок A, B, C, заданих координатами на площині.

B-2

1) Скласти програму для обчислення виразу:

$y = \frac{(5a)!}{(4a-b+3)!}$, обчислюючи факторіал у підпрограмі.

2) Записати програму знаходження периметра чотирикутника ABCS, якщо відомі координати його вершин. Використати підпрограму для знаходження відстані між двома точками.

Запитання для контролю:

1. Для чого призначені процедури ?
2. Як описується заголовок процедури ?
3. Чим відрізняються формальні і фактичні параметри ?
4. Чим відрізняються локальні і глобальні змінні ?
5. Для чого призначені функції ?
6. Як описується заголовок функції ?
7. Яка різниця між процедурою і функцією ?
8. Як викликаються процедури і функції ?

Лекція. Стандартні графічні процедури і функції в Паскаль

Мета, завдання лекції: ознайомлення з можливостями побудови графічних зображень, основними процедурами для створення графічних примітивів.

План і організаційна структура лекції:

1. Графічний режим Паскаль.
2. Основні функції для роботи з графікою.

3. Створення анімованих зображень засобами Pascal.

Зміст лекційного матеріалу:

Будь-який монітор ПЕОМ може працювати в одному з двох режимів: текстовому та графічному.

Перший з них дозволяє виводити на екран будь-який символ ASCII-таблиці у визначене місце екрану, що знаходиться на перетині рядка та стовпчика. Кількість знакомісць залежить від текстового режиму, але найчастіше дорівнює 25 рядкам по 80 стовпців у кожному.

У графічному режимі будь-яке зображення отримується як сукупність різнокольорових точок – пікселів. Кількість елементів зображення теж задається відповідним режимом, але стандартно дорівнює 640 на 480 відповідно по горизонталі та вертикалі.

Екран у графічному режимі може адресуватися за допомогою системи координат, причому значення *X* (номера стовпчика) збільшується зліва праворуч, а значення *Y* (номера рядка) збільшується зверху до низу. За замовчанням координати екрана мають такий вигляд:

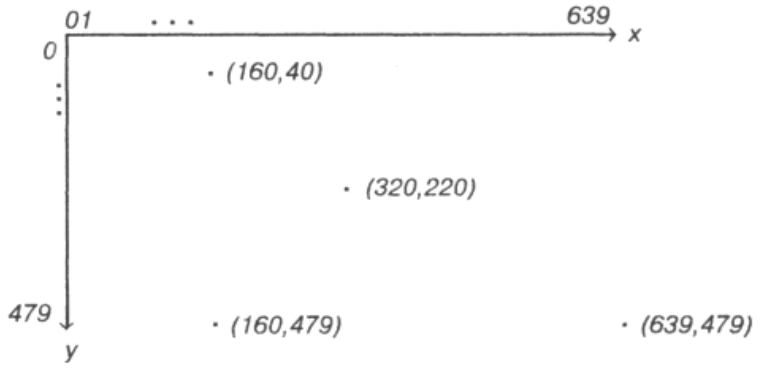
(0, 0) – лівий верхній кут;

(639, 0) – правий верхній кут;

(319, 239) – центр;

(0, 479) – лівий нижній кут екрана;

(639, 479) – правий нижній кут.



Для роботи у графічному режимі в Паскалі використовується модуль *Graph*, який складається з більш ніж 90 графічних процедур і функцій широкого профілю. Усі стандартні засоби модуля *Graph* стають доступними після його підключення до програми в розділі *Uses*: **Uses** *Graph***;**

Робота програми починається з ініціалізації (встановлення) графічного режиму процедурою ***InitGraph*** і завершується процедурою ***CloseGraph***.

program MyGrafika;

uses Graph;

```
  . . . {розділи описів програми}
var driver, mode : integer; {службові змінні}
begin
    driver := detect;  {визначення характеристик}
    initgraph (driver, mode, '<шлях до драйверу>');
    <текст програми з графічними командами>
end.
```

Стандартна стала **detect** призначена для автоматичного визначення типу адаптера монітора. Для його роботи потрібна спеціальна програма – драйвер. Її файл має назву **egavga.bgi**. Він повинен бути на диску, шлях до нього зазначають у процедурі **initgraph** у лапках. Якщо файл є в тому ж каталозі, що і виконуваний файл, то шлях зазначати не потрібно, тому пишуть так: ' '.

Щоб задати потрібний колір для деякого зображення, використовують спеціальні команди, що містять параметр, значенням якого може бути число або англійська назва кольору:

Колір:

0 – чорний Black
1 – синій Blue
2 – зелений Green
3 – блакитний Cyan
4 – червоний Red
5 – фіолетовий Magenta
6 – коричневий Brown
7 – світло-сірий LightGray
8 – темно-сірий DarkGray
9 – яскраво-синій LightBlue
10 – яскраво-зелений LightGreen
11 – яскраво-блакитний LightCyan
12 – яскраво-червоний LightRed
13 – яскраво-фіолетовий LightMagenta
14 – жовтий Yellow
15 – білий White

У графічному режимі можна виводити текст, причому є можливість масштабування і вибору типу шрифту, виконання вирівнювання виведеного тексту тощо. Наявні програми підтримують різні засоби малювання і заповнення фігур, зокрема, точки, лінії, кола, еліпси, прямокутники, багатокутники.

При виконанні графічної операції може виникнути помилка, код якої можна одержати за допомогою функції ***GraphResult***. Код помилки може приймати одне з наступних значень:

0:	Помилки немає
-1:	Режим BGI не встановлений
-2:	Графічні апаратні засоби не виявлені
-3:	Файл драйвера пристрою не знайдений
-4:	Неправильно визначений файл драйвера пристрою

-5:	Не вистачає пам'яті для завантаження драйвера
-6:	Вихід за межі пам'яті при заповненні
-7:	Вихід за межі пам'яті при заливанні
-8:	Файл із шрифтом не знайдений
-9:	Не вистачає пам'яті для завантаження шрифту
-10:	Неправильний графічний режим для обраного драйвера.

Для забезпечення переходу екрану монітора в графічний режим програма повинна починатися викликом процедури *InitGraph*, що автоматично виявляє апаратні засоби і завантажує відповідний графічний драйвер. Стандартний драйвер *EGAVGA.BGI* розміщується у каталозі BP\BGI на відповідному диску. Якщо апаратні засоби не виявлені або в процесі ініціалізації відбулася помилка, то на екран виводиться повідомлення про помилку і програма зупиняється.

Основні функції для роботи з графікою:

Arc (X, Y, поч_кут, кін_кут, радіус) – програмна процедура, результатом роботи якої є дуга кола з центром в точці (*X, Y*) і радіусом "радіус". Дуга креслиться від початкового кута ("поч_кут") до кінцевого кута ("кін_кут"). При цьому використовується поточний колір малювання. Кути задаються в градусах.

Bar (X1, Y1, X2, Y2) – процедура малює зафарбований прямокутник, використовуючи колір зафарбування, що встановлюється процедурою ***SetFillStyle.*** Контур прямокутника креслиться кольором і типом лінії, що встановлені процедурами ***SetLineStyle*** і ***SetColor***. Точки з координатами (*X1*, *Y1*) та (*X2*, *Y2*) задають дві діагонально протилежні вершини прямокутника.

Bar3D (Xl, Yl, X2, Y2; глибина, вершина : boolean) – процедура малює зафарбований тривимірний паралелепіпед. Контур паралелепіпеда креслиться кольором і типом лінії, що встановлені процедурами ***SetLineStyle*** і ***SetColor***, тип і колір зафарбування встановлюється за допомогою процедур ***SetFillStyle***. "Глибина" являє собою число елементів зображення, що задають третій вимір тривимірного контуру. Якщо змінна, зазначена як параметр "вершина", приймає істинне значення (*True*), то для паралелепіпеда малюється тривимірна вершина, у протилежному випадку вершина не малюється (що дозволяє малювати кілька паралелепіпедів, розташованих один на одному).

Circle (X, Y, радіус) – процедура малює коло поточним кольором. Точка (*X*, *Y*) – центр кола, а "радіус" – радіус кола.

ClearDevise – процедура очищує поточний графічний екран і готує його для виведення даних.

ClearViewport – процедура очищує поточне вікно.

CloseGraph – процедура припиняє роботу графічної системи (закриття графіки) і повертає монітор до текстового режиму.

Ellipse (X, Y, поч_кут, кін_кут, радX, радY) – процедура малює еліптичну дугу, використовуючи (*X*, *Y*), як точку центра і "*радX*", "*радY*" – як радіуси на горизонтальній і вертикальній осях. Дуга еліпса малюється від початкового кута (параметр "*поч_кут*") до кінцевого кута (параметр "*кін_кут*") поточним кольором.

FillEllipse (X, Y, Храдіус, Урадіус) – процедура малює зафарбований еліпс, використовуючи точку з координатами (*X*, *Y*) як центр, а "*Храдіус*" і "*Урадіус*" – у якості радіусів на горизонтальній та вертикальній осях. Контур еліпса креслиться кольором і типом лінії, що встановлені процедурами ***SetLineStyle*** і ***SetColor***, тип і колір зафарбування встановлюється за допомогою процедури ***SetFillStyle***.

FloodFill (X, Y, колір межі) – процедура заповнює замкнену область, використовуючи поточний заповнювач, заданий процедурою ***SetFillStyle***. Точка (*X*, *Y*) є внутрішньою точкою області, що зафарбовується. Якщо точка (*X*, *Y*) знаходиться усередині замкненої області, то заповнюється внутрішня область. Якщо ця точка знаходиться поза замкненої області, то заповнюється її зовнішня частина (поле екрана, що не належить області).

GetBkColor – функція повертає поточне значення кольору тла (у діапазоні 0 – 15), встановлене процедурою ***SetBkColor***.

GetColor – функція повертає поточне значення основного кольору малювання (у діапазоні 0 – 15), встановлене раніше процедурою ***SetColor***.

GetMaxX – функція повертає максимальний для поточного графічного режиму номер правого стовпчика (дозвіл по *X*).

GetMaxY – функція повертає максимальний для поточного графічного режиму номер нижнього рядка екрана (дозвіл по *Y*).

GetX – функція повертає *X*-координату поточного покажчика СР.

GetY — функція повертає *Y*-координату поточного покажчика СР.

GraphResult – функція повертає код помилки (у діапазоні 0 – 10) для останньої графічної операції.

InitGraph (driver, mode, шлях до драйвера); – процедура ініціалізує графічну систему і переводить апаратну частину в графічний режим.

Line(X1, Y1, X2, Y2); – процедура вичерчує пряму лінію (товщина і тип якої встановлений процедурою *SetLineStyle*, колір – процедурою ***SetColor***) із точки (*X1*, *Y1*) у точку (*X2*, *Y2*).

LineRel(Dx, Dy); – процедура малює пряму лінію з точки поточного покажчика в точку, задану відносною відстанню (*Dx*, *Dy*) від поточного покажчика.

LineTo(X, Y); – процедура малює пряму лінію з точки, у якій знаходиться поточний покажчик, у точку з координатами (*X*, *Y*).

MoveRel (Dx, Dy); – процедура переміщує покажчик з поточної точки у точку, задану відносною відстанню (*Dx*, *Dy*).

MoveTo (X, Y : integer); – процедура переміщує поточний покажчик СР у точку з координатами (*X*, *Y*).

OutText (рядок); – процедура виводить текст "рядка" на монітор, починаючи з точки розташування покажчика.

OutTextXY (X, Y, текст_рядок); – процедура виводить текст, що міститься у "текст_рядок", починаючи з точки, заданої координатами (*X*, *Y*). Якщо рядок занадто довгий і виходить за межі екрана чи поточної області перегляду, то він усікається.

PieSlice (X, Y, поч_кут, кін_кут, R); – процедура малює і заповнює поточним кольором сектор кола радіусом *R*. Точка (*X*, *Y*) – центр кола, а сектор малюється від початкового до кінцевого кута.

PutPixel (X, Y, колір); – процедура зафарбовує точку з координатами (*X*, *Y*) у певний колір.

Rectangle(Xl, Yl, X2, Y2); – процедура малює прямокутник, використовуючи поточний колір і тип лінії. (*X1*, *Y1*) та (*X2*, *Y2*) – координати діагонально протилежних вершин прямокутника.

Sector(X, У, поч__кут, кін_кут, X_Радіус, Y_Радіус : word); – процедура малює і заповнює еліптичний сектор. (*X*, *Y*) – центр кола, "*X_Радіус*", "*Y_Радіус*" – горизонтальний і вертикальний радіуси. Сектор креслиться від початкового кута "*поч_кут*" до кінцевого кута "*кін_кут*". Сектор малюється поточним кольором і зафарбовується з використанням зразка зафарбування і кольорів, заданих за допомогою процедури ***SetFillStyle***.

SetBkColor (колір); – процедура встановлює поточний колір для тла.

SelColor (колір); – процедура встановлює поточний колір малювання.

SetFillStyle (заповнення, колір) – задання способу заповнення замкнутої області.

Стиль заповнення:

0 – кольором фону EmptyFill	6 – похилими зворотними лініями

	LtbkslashFill
1 – суцільне кольором зображення SolidFill	7 – прямокутною горизонтальною штриховкою HatchFill
2 – горизонтальними лініями LineFill	8 – косою штриховкою XhatchFill
3 – похилими лініями LtslashFill	9 – косою перекриваючою штриховкою InterleaveFill
4 – похилими товстими лініями SlashFill	10 – рідкими крапками WideDotFill
5 – похилими зворотними товстими лініями BkslashFill	11 – щільне заповнення крапками CloseDotFill

SetLineStyle (тип, зразок, товщина); – процедура встановлює поточну товщину і тип лінії: тип лінії: 0 –суцільна; 1 – крапчаста; 3 – штрихова; зразок: 0 – стандартна; товщина: 1 – тонка, 3 – товста.

SetTextStyle (шрифт, напрям, розмір) – процедура встановлює стиль тексту6 шрифт символів: 0 – звичайний, 1 – жирний,…, 4 – готика при наявності файлів шрифтів у бібліотеці; напрямок виведення: 0 – горизонтально чи 1 – вертикально, а також розміри символів: 1, 2, …, 10.

Приклад. Намалювати різними кольорами 10 концентричних кіл, які мають спільний центр у центрі екрана, тобто в точці з графічними координатами (320; 240) і описати навколо кіл червоний трикутник.

```
program Circle10;
uses crt, graph;
var driver, mode, r : integer;
begin clrscr;
    driver := detect;
    initgraph (driver, mode,'');
    r := 10;
     while r <= 10 do
      begin
          setcolor(r div 10);
          circle(320, 240, r);
       end;
end.
```

Створення анімованих зображень засобами Pascal

Для імітації руху зображення на екрані використовують чергування засвічень і гасінь цього зображення. Перед наступним засвічуванням об'єкт треба змістити в напрямку переміщення. Рух зображення на екрані

називають анімацією. Для імітації руху зображень на екрані потрібно виконати такий алгоритм:

1. Нарисувати об’єкт у потрібній точці і зробити паузу.
2. Знищити об’єкт, замалювавши його кольором тла.
3. Змінити координати об’єкта.
4. Перейти до пункту 1

Приклад. Зобразити рух Сонця на блакитному небі в горизонтальному напрямку.

```
program Sun;
uses crt, graph;
var driver, mode, i : integer;
begin clrscr;
    driver := detect;
    initgraph (driver, mode, ’’);
    i := 0;
    while i <= 750 do
    begin
        setcolor (14);
        setbkcolor (3);
        setfillstyle (1, 14);
        circle (i, 100, 50);
        floodfill(I, 100, 14);
        delay (200);                          {пауза 0,2с}
        setfillstyle (1, 3);
        setcolor (3);
        circle (i, 100, 50);
        floodfill (i, 100, 3);
        i := i + 5
     end;
end.
```

Практична робота. Створення графічних зображень

Мета: формування навичок складання та реалізації програм створення графічних зображень з використанням графічних примітивів.

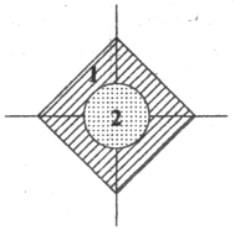
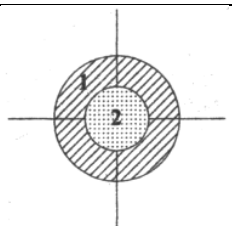
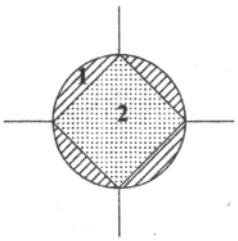
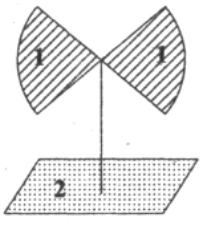
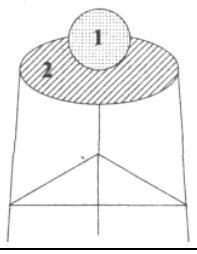
Завдання:

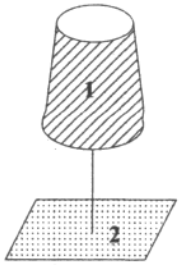
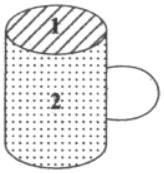
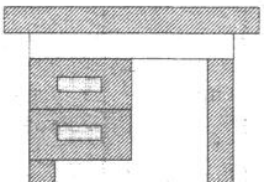
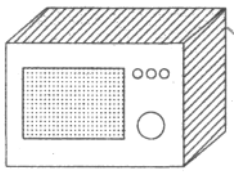
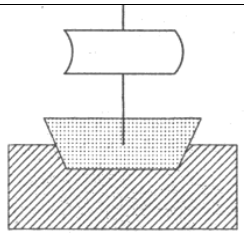
- *студенти повинні знати:* основні процедури та функції для створення графічних примітивів;
- *студенти повинні вміти:* створювати програми з використанням можливостей побудови графічних примітивів.

Обладнання: зошит, середовище програмування.

Хід виконання:

1. Вивчити теоретичний матеріал з даної теми:
   - перехід до графічного режиму роботи монітора;
   - вигляд координатної сітки» екрану монітора;
   - запис процедур і функцій зображення ліній та фігур;
   - запис процедур та функцій встановлення кольорів фону та штрихування фігур.
2. Скласти програму створення графічних примітивів (згідно свого порядкового номера).
3. Виконати програму і вивести зображення на екран монітора.

№ з/п	***Вид фігури***	***Колір фону***	***Колір ліній***	***Колір штриховки***
1.		фіолетовий	чорний	1. жовтий 2. синій
2.		блакитний	червоний	1. світло-сірий 2. білий
3.		зелений	синій	1. червоний 2. жовтий
4.		яскраво-блакитний	синій	1. червоний 2. жовтий
5.		білий	чорний	1. червоний 2. коричневий

№ з/п	*Вид фігури*	*Колір фону*	*Колір ліній*	*Колір штриховки*
6.		світло-сірий	темно-сірий	1. жовтий 2. коричневий
7.		жовтий	чорний	1. білий 2. синій
8.		білий	чорний	1. коричневий 2. сірий
9.		білий	чорний	1. світло-сірий 2. коричневий
10.		білий	темно-сірий	1. коричневий 2. блакитний

Практична робота. Анімація в Паскаль. Створення рухомих зображень

Мета: формування навичок складати та реалізовувати програми з використанням графічних примітивів.

Завдання:

- *студенти повинні знати:* основні процедури та функції для створення графічних примітивів; особливості створення рухомих зображень;
- *студенти повинні вміти:* створювати програми з використанням можливостей побудови графічних статичних та динамічних зображень.

Обладнання: зошит, середовище програмування.

Хід виконання:

1. Вивчити теоретичні відомості та дати відповіді на запитання:

а) ініціалізація графічного режиму в Pascal;
б) процедури та функції для виконання графічних побудов;
в) алгоритм побудови анімованого зображення.

2. Ознайомитися із зразком програми створення статичного зображення:

У верхній лівій частині графічного екрану на чорному фоні нарисувати блакитний квадрат, а в ньому – коло (колір ліній – чорний), зафарбоване жовтим кольором. У центрі емблеми чорними літерами написати слово «Lviv».


```
program emblema;
 uses crt, graph;
 var driver, mode : integer;
 begin
      clrscr;
      driver := detect;
      initgraph (driver, mode,' ');
      setbkcolor (0); setcolor (3);
      rectangle (100, 0, 300, 200);
      setfillstyle (1, 3);
      floodfill (200, 100, 3);
      setcolor (14);
      circle (200, 100, 100);
      setfillstyle (1, 14);
      floodfill (200, 100, 14);
      setcolor(0);
      settextstyle (0,0,4);
      outtextxy (135, 95, ' Lutsk');
      readln;
      closegraph;
 end.
```

3. Виконати практичні завдання згідно заданого варіанту:

В-1

1) Складіть програму побудови емблеми: у нижній лівій чверті розташуйте квадрат, в ньому – коло, посередині якого виведіть своє ім’я. Замкнені ділянки зафарбуйте різними кольорами.

2) Створити програму побудови зображення, яке імітує рух зафарбованого довільним кольором кола по вертикалі.

В-2

1) Складіть програму побудови емблеми: у верхній правій чверті розташуйте коло, в ньому – прямокутник, посередині якого виведіть своє прізвище. Замкнені ділянки зафарбуйте різними кольорами.

2) Створити програму побудови зображення, яке імітує рух зафарбованого довільним кольором прямокутника по горизонталі.

Матеріали для організації самостійної позааудиторної роботи студентів

Тема СРС: Оператори розгалуження і вибору

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: ознайомлення зі структурою програм з розгалуженням, практична реалізація при розв'язуванні задач.

Завдання для СРС:

1. **Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:**
   1) Опис вказівок розгалуження мовою програмування PASCAL.
   2) Формат умовного оператора. Принцип роботи.
   3) Оператор безумовного переходу.
   4) Формат оператора вибору. Принцип роботи. Приклад програм.
2. **Виконати завдання:**

Дайте відповіді на запитання:

1. Чи істинний простий логічний вираз x > 10, якщо:
а) x = 0; б) x = 2; в) x = 10; г) x = 5; д) x = 15?

2. Чи буде хибним вираз x >= 10, якщо:
а) x = 1; б) x = 3; в) x = 10; г) x = 12; д) x = 25?

3. Який результат (m = ?) виконання програми Ex1, якщо ввести значення k так:
а) 3; б) 4; в) 0; г) 2; д) -2?

```
program Ex1;
var k, m : integer;
begin
   write ( 'Введіть k: ');
   readln (k);
   m := 7;
   if k > 2 then m := k*k;
   if k<2 then m := k+5;
   writeln (m)
end.
```

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

1) **not** (x = y),	а) $x \in [0; 1]$,

2) (x < y) **or** (x = y),
3) (x < 0) **or** (x > 1),
4) (x >=0) **and** (x <= 1),

б) $x \neq y$,
в) $x \leq y$,
г) $x \notin [0; 1]$.

Завдання:

1. Увести значення аргументу й обчислити значення складеної функції

$$y = \begin{cases} x^2, & якщо \quad x < 5, \\ x - 5, & якщо \quad x \geq 5. \end{cases}$$

2. Увести число. Вивести повідомлення: число додатне, від'ємне чи нуль.

3. Увести число. Вивести повідомлення, чи число кратне 9.

4. Увести два числа – значення двох кутів трикутника. Вивести усі можливі повідомлення про властивості трикутника: трикутник прямокутний, гострокутний чи тупокутний, різносторонній, рівносторонній чи рівнобічний.

5. Дано трикутник зі сторонами а, в, с. Перевірити, чи виконується умова існування трикутника.

6. Скласти програму визначення дня тижня за його порядковим номером.

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

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

8. Введемо позначення для визначення родів в українській мові:

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

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

«жіночий рід» – «-а, -я».

9. Дано дійсне значення x і ціле значення n ($1 \leq n \leq 3$). За введеним значенням n знайти значення відповідної функції:

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

Тема СРС: Цикл з параметром

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: ознайомлення зі структурою циклу з параметром, практична реалізація при розв'язуванні задач.

Завдання для СРС:

1. **Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:**
   1. Опис вказівки повторення мовою програмування.
   2. Оператор циклу з параметром (FOR): формат і принципи роботи.
   3. Приклади розв'язання задач.
2. **Виконати завдання:**

1) Нехай M = 128. Чому дорівнюватиме значення змінної M у результаті виконання послідовності операторів:

```
i := -2;
while i <= 5 do
  begin
    M := M/2;
    i := i + 2;
  end;
```

2) Якого значення набуде змінна *m* у результаті виконання послідовності операторів:

```
for i := 1 to 5 do
  begin
    read (x);
    if x >= 0 then m := 2*i
  end;
```

де змінна x послідовно набуватиме значення:

1) 1; 4; -2; 5; 0; 2) 1; 3; 5; 2; 4?

3) Якого значення набуде змінна *s* в результаті виконання послідовності дій:

1)
```
s := 0;
i := 0;
while i < 5 do
  i := i + 1;
  s := s + 1/i;
```

2)
```
s := 0;
i := 0;
while i < 5 do
  begin
  i := i + 1;
  s := s + 1/i;
```

3)
```
s := 0;
i := 1;
repeat
   s := s + 1/i;
   i := i - 1
until i <= 1;
```

4)
```
s := 1;
n := 1;
for i := 2 to n do
    s := s + 1/i;
end;
```

3. Ознайомитись із прикладом реалізації циклу з параметром:

Для чисел від 1 до 10 обчислити квадратні корені та корені четвертого степеня. Результати вивести у вигляді таблиці.

```
program BANK;
var i : integer; rez1, rez2 : real;
begin
      writeln ('Число      Корінь квадр.      Корінь четвертого степеня');
      for i:= 1 to 10 do
         begin
            rez1 := sqrt (i);
            rez2 := sqrt (sqrt (i));
             writeln (i : 4, rez1 : 15 : 2, rez2 : 15 : 2);
         end;
end.
```

4. Написати програми реалізації задач:

1) Вивести на екран монітора своє прізвище задану кількість разів.

2) Тіло вільно падає з висоти *h*. Виведіть таблицю значень висоти протягом перших k = 8 секунд падіння.($h = \frac{gt^2}{2}$)

3) Виведіть на екран у вигляді таблиці номери і значення перших 10 елементів числової послідовності, загальних елемент якої має вигляд *12i-cosi.*

4) Серед перших 20 елементів *1-3sini²* виведіть на екран номери і значення лише від'ємних елементів.

Тема СРС: Оператори циклу

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: ознайомлення з структурою циклів з передумовою, післяумовою, практична реалізація при розв'язуванні задач.

Завдання для СРС:

1. **Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:**
    1. Вказівка повторення з передумовою (Цикл while).
    2. Вказівка повторення з післяумовою (Цикл repeat-until)
    3. Приклади розв'язання задач.
2. **Виконати завдання:**

1. Для чисел від 1 до 10 обчислити квадратні корені та корені четвертого степеня. Результати вивести у вигляді таблиці.

2. Обчислити значення функції $y = x^2 + x^3 + x + 1$ при x = 1, 2, 3, …, 10.

3. Обчислити суму значень функції

y = 1 * sin(1) + 3 * sin(3) + 5 * sin(5) + … + k * sin(k) при k = 15.

4. Протабулювати функцію $y = 3 \cos 2x$ на проміжку [-2; 2] з кроком 0,3.

5. Протабулювати функцію $y = 3 - 3 \sin x^2$ на проміжку [-2; 2] з кроком 0,2 і визначити кількість від'ємних елементів.

6. Вивести на екран усі двозначні числа, які діляться на 5.

7. Обчислити добуток непарних чисел від 7 до 15.

Тема СРС: Рядкові величини. Стандартні функції роботи з рядками

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: ознайомлення з основними функціями для роботи з рядками, практична реалізація при розв'язуванні задач.

Завдання для СРС:

1. Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:
1) Рядкові величини. Операції над рядковими величинами.
2) Алгоритми роботи з рядками.
3) Опис найпростіших алгоритмів роботи з рядками мовою програмування PASCAL.

2. Виконати завдання:

1) Дано:

```
var L: integer;
```

Вкажіть, яким вказівкам присвоєння з лівого стовпчика відповідають значення L з правого:

1) L := Length ('Pascal');	а) 4;
2) L := Length ('Фірма-IBM');	б) 11;

3) L := Length (‘ ’);	в) 0;
4) L := Length (‘Turbo’);	г) 8;
5) L := Length (‘Програма’);	д) 6;
6) L := Length(‘Microsoft’);	е) 5;
	ж) 3;
	з) 9.

2) Дано:

const WORD = ’форма’;

var R, R1: string [20];

Вкажіть, яке значення прийме змінна R з правого стовпчика після виконання вказівок з лівого:

1) R := ’інтика’;	а) ‘інтиформака’;
R1 := insert (Word, R, 3);	б) ‘інформатика’;
2) R := ’перетування’;	в) ‘інтформаика’;
R1 := insert (Word, R, 4);	г) ‘переформатування’;
	д) ‘перформаетування’;
	е) ‘перетуформавання’.

3) Дано:

const R = ’Константинополь’;

var R1 : string [15];

Вкажіть, яке значення прийме змінна R1 з правого стовпчика після виконання вказівок з лівого:

1) R1 := Delete (R, 1, 5);	а) ‘Конст’;
2) R1 := Delete (R, 5, 8);	б) ‘поль’;
3) R1 := Delete (R, 8, 3);	в) ’Константинополь’;
4) R1 := Delete (R, 6, 6);	г) ’Кополь’;
5) R1 := Delete (R, 6, 10);	д) ’Констполь’;
6) R1 := Delete (R, 1, 11);	е) ’Константин’;
7) R1 := Delete (R, 3, 1);	ж) ’Конполь’;
8) R1 := Delete (R, 2, 9);	з) ’Консоль’.

4) Дано:

const R = ’Константинополь’;

var S : string [20];

Вкажіть, яким вказівкам присвоєння з лівого стовпчика відповідають значення S з правого стовпчика:

1) S := Copy (R, 4, 4);	а) ‘тин’;
2) S := Copy (R, 8, 3);	б) ‘поль’;
3) S := Copy (R, 6, 4);	в) ‘пол’;
4) S := Copy (R, 12, 3);	г) ‘Конст’;

5) S := Copy (R, 12, 4);
6) S := Copy (R, 1, 5);
7) S := Copy(R, 1, 10);

д) ‘анти’;
е) ‘стан’;
ж) ‘ник’;
з) ’Константин’;
и) ’Кон’.

5) Дано:

const R = ’Без верби й калини нема України’;

var WORD : string [50]; Poz : integer;

Виконується вказівка присвоювання Poz := Pos (WORD, R). Встановіть відповідність між даними стовпців:

Значення WORD:

1) ‘без’;
2) ‘верби’;
3) ‘і’;
4) ‘калини’;
5) ‘України’;
6) ‘ ’

Значення Poz:

а) 5;
б) 13;
в) 20;
г) 32;
д) 1;
е) 4.

Тема СРС: Одновимірний масив

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: формування теоретичних знань і практичних навичок розв’язування задач з теми «Одновимірний масив».

Завдання для СРС:

1. Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:

1) Поняття одновимірного масиву.
2) Які бувають масиви?
3) Що таке індекс елемента масиву?
4) Скільки індексів у елемента лінійного масиву?
5) Яким чином проводиться введення всіх елементів масиву?
6) Яким чином проводиться виведення всіх елементів масиву?
7) Якого типу можуть бути елементи масиву?
8) Якого типу можуть бути індекси елементів масиву?
9) Якими способами може бути заповнений масив?
10) Запишіть фрагмент програми, що вводить початкові значення елементів одновимірного масиву?

11) Запишіть фрагмент програми, що виводить на екран початкові значення елементів одновимірного масиву?
12) Якими способами можна відділити значення елементів масиву при виведенні на екран монітора?
13) Які дії над елементами масивів найчастіше зустрічаються в задачах?
14) Як використовується розділ констант для задання початкових даних елементам масиву?

2. Запишіть програми розв'язку задач::

1. Сформуйте одновимірний масив з 9 цілих чисел з клавіатури та виведіть його на екран. Знайдіть його найбільший елемент, замініть цей елемент на 100 і виведіть на екран масив після заміни.
2. Сформуйте масив з 5 елементів за правилом a [i] = i - 3. Підрахуйте кількість від'ємних елементів масиву.
3. Сформуйте одновимірний масив з 8 цілих чисел з клавіатури та виведіть його на екран. Обчисліть середнє арифметичне додатних елементів цього масиву.

Тема СРС: Табличні величини

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: формування практичних навичок опрацювання табличних величин; розв'язування задач.

Завдання для СРС:

1. Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:
1) Табличні величини. Алгоритми роботи з табличними величинами.
2) Поняття масиву. Оператор опису масивів.
3) Одновимірний масив.
4) Двовимірний масив.
5) Методи впорядкування та пошуку. Алгоритми сортування масивів.

2. Виконати завдання:

1. У масиві Q [0 .. 5, 0 .. 3] від'ємні елементи замінити нулями. Вивести на екран початковий та змінений масив.
2. Дано масив A [1 .. 4, 1 .. 3], що містить елементи A [i, к] = 2 i + 2 к. Обчислити суму елементів цього масиву.

3. У масиві D [0 .. 7, 0 .. 2], елементи якого довільні цілі числа, обчислити добуток від'ємних елементів масиву.
4. Двовимірний масив M [1 .. 3, 1 .. 3] містить довільні дійсні числа. Знайти кількість елементів, яка рівні числу 5.
5. У масиві Q [0 .. 5, 0 .. 3] від'ємні елементи замінити нулями. Вивести на екран початковий та змінений масив.
6. Створити масив P з елементами, що шукаються за формулою P [i, k] = A [i, k] + B [i, k] , елементи A [i, k] та B [i, k] взято із попередньої задачі. Вивести на екран всі три масиви.
7. Дано масив A[1.. 4, 1 .. 4], що містить елементи A [i, к] = 2 і + 2 к. Обчислити добуток елементів головної діагоналі.
8. У масиві D [1 .. 4, 0 .. 3], елементи якого довільні цілі числа, обчислити суму додатніх елементів верхнього трикутника.
9. Двовимірний масив M [1 .. 3, 1 .. 3] містить довільні дійсні числа. Знайти у нижньому трикутнику матриці кількість елементів, яка рівні числу 0.
10. У масиві P [0 .. 7, 0 .. 7], елементи якого знаходяться за формулою P [i, k] = 3 i – 2 k, знайти кількість додатних елементів у верхньому трикутнику та кількість від’ємних елементів у нижньому трикутнику матриці.
11. У масиві Q [0 .. 5, 0 .. 5], елементи якого шукаються за формулою Q [i, k] = k, всі елементи головної діагоналі замінити нулями. Вивести на екран змінений масив.
12. Знайти максимальний елемент у масиві K [0 .. 11], елементи якого знаходяться за формулою K [i] := 10 - 2i. Вивести на екран цей масив та максимальний елемент.
13. У масиві D [1 .. 4, 1 .. 5], елементи якого довільні цілі числа, знайти мінімальний елемент.

Тема СРС: Підпрограми в мові Паскаль

Кількість навчальних годин: 2

Мета, завдання самостійної позааудиторної роботи: вивчення теоретичного матеріалу та вироблення практичних навичок складання програм з використанням підпрограм.

Завдання для СРС:

1. Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:

1. Опис мовою Паскаль вказівок звернення до функцій і процедур.
2. Підпрограма PROCEDURE: формат і принцип роботи. Приклади програм.
3. Фактичні і формальні параметри.
4. Функції: формат і принципи роботи.

2. Виконати завдання:

Записати результат виконання таких програм:

```
program Rysunok;
  procedure Stars (n : integer);
  var i: integer;
  begin
    for i := 1 to n do
    write ('*');
    writeln
  end;
begin
     Stars (5);
     Stars (10);
     Stars (15);
end.
```

```
program Sumy;
var k, s : integer;
    procedure Suma (n: integer;
    var s: integer);
    var i : integer;
    begin
     S:=0;
     for i: = 1 to n do
     S := s + i
  end;
begin
     suma (5, s);
     writeln (s);
     read (k);
     suma (k, s);
     writeln (s);
end.
```

Тема СРС: Елементи комп'ютерної графіки.

Кількість навчальних годин: 2

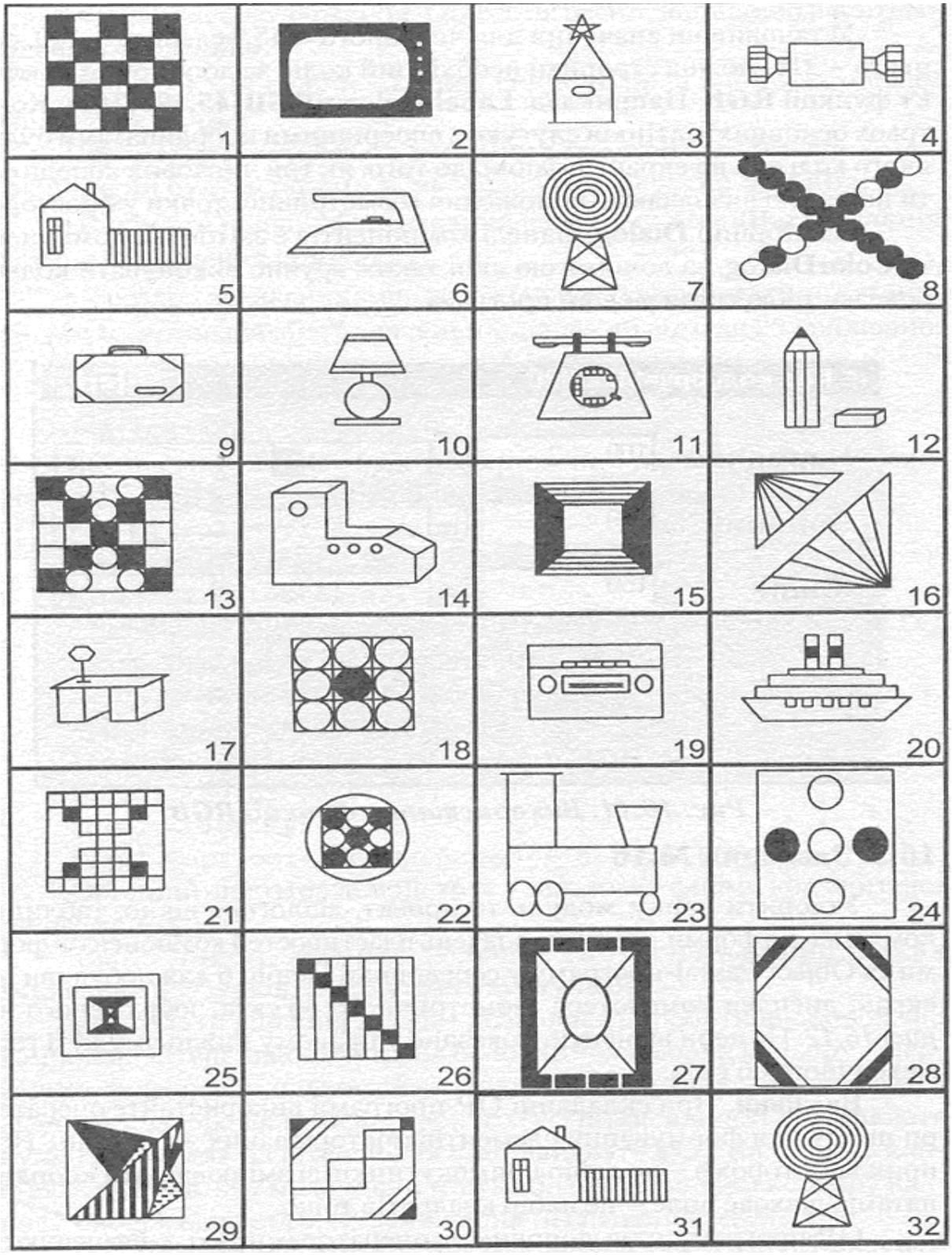
Мета, завдання самостійної позааудиторної роботи: вивчення теоретичного матеріалу та вироблення практичних навичок створення зображень в середовищі програмування.

Завдання для СРС:

1. Опрацювати теоретичний матеріал з теми і дати відповіді на запитання:
   - перехід до графічного режиму роботи монітора;

– вигляд координатної сітки екрану дисплея;
– запис процедур і функцій зображення ліній та фігур;
– запис процедур та функцій встановлення кольорів фону та штрихування фігур.

2. Скласти та виконати програму створення графічного примітиву (Номери варіантів показано в правому нижньому куті геометричного об'єкта).



Перелік питань для підготовки до Модульних контрольних робіт

Змістовий модуль 3. Програмування розв'язку математичних задач

1. Вказівка розгалуження, її синтаксис. Принцип дії вказівки розгалуження.
2. Оператор вибору *case*, принцип дії, синтаксис.
3. Цикл з параметром, принцип дії та загальний вигляд.
4. Вказівка повторення з передумовою, принцип дії та загальний вигляд.
5. Вказівка повторення з післяумовою, принцип дії та загальний вигляд.

Практичні завдання:

1. Знайти добуток цілих від'ємних чисел, кратних 2 і більших за -50.
2. Знайти суму цілих додатних парних чисел, менших за 100.
3. Скласти програму знаходження значення функції:

$$Y = \cos 1 + \cos 3 + \cos 5 + \ldots + \cos 11$$

4. Знайти середнє арифметичне всіх цілих чисел, більших за -10 і менших за 10.
5. Скласти програму знаходження значення функції:

$S = 1^3 + 2^3 + 3^3 + \ldots + n^3$, якщо n = 10.

6. Обчисліть

$Y = \frac{x^2}{2} + \frac{x^4}{4} + \ldots + \frac{x^{20}}{20}$, де $x = 0{,}2$

7. Виведіть таблицю квадратів чисел від 1 до 20.

Змістовий модуль 4. Використання можливостей мови Паскаль для побудови різних структур

1. Запис рядкових величин.
2. Функції та процедури для роботи з рядками.
3. Поняття одновимірного масиву.
4. Опис типів масивів та оголошення змінних типу масив.
5. Доступ до і-го елементу масиву.
6. Заповнення одновимірних масивів.
7. Методи виведення елементів одновимірного масиву на екран.
8. Поняття двовимірного масиву.
9. Опис двовимірного масиву.
10. Звертання до елементу двовимірного масиву.
11. Заповнення масиву.
12. Виведення двовимірного масиву на екран.
13. Використання масивів при розв'язуванні задач.
14. Способи сортування масивів.
15. Сортування методом вставки.
16. Сортування методом вибору.
17. Сортування методом обміну.

Практичні завдання:

1. Дано одновимірний масив, який складається з 10 чисел, який заповнено за формулою: *a [i] = 2 * sini.* Підрахувати кількість додатних елементів масиву і замінити їх на одиниці. Вивести отриманий масив на екран у вигляді стовпця.
2. Дано одновимірний масив з 15 цілих чисел, який заповнено з клавіатури. Скласти програму підрахунку кількості додатних елементів масиву; обчислити їх суму. Вивести масив на екран у вигляді рядка.
3. Знайти максимальний елемент та його номер в масиві B [0 .. 5, 0 .. 4], елементи якого знаходяться за формулою *B [i, j]:= -3 i + 4 j.* Вивести на екран цей масив, мінімальний елемент та його номер.
4. У масиві A[1 .. 4, 1 .. 5], елементи якого довільні цілі числа. Знайти максимальний елемент цього масиву та його номер. Вивести масив на екран.

Змістовий модуль 5. Підпрограми. Застосування графіки в Паскаль

1. Поняття процедури, її структура та опис.
2. Процедури без параметрів
3. Процедури з параметрами
4. Локальні і глобальні змінні
5. Параметри-значення і параметри-змінні у процедурах
6. Функції, формат опису, використання
7. Яка різниця між процедурою і функцією ?
8. Графічний режим в Pascal.
9. Основні функції для роботи з графікою.
10. Створення анімованих зображень засобами Pascal.
11. Як в програмі необхідно описати підключення модуля для встановлення графічного режиму?
12. Як здійснюється ініціалізація графічного режиму?
13. Який вигляд має екранна система координат дисплея в графічному режимі?
14. Як зобразити на екрані точку, відрізок прямої, прямокутник, коло, еліпс, ламану лінію?
15. З допомогою якої вказівки можна визначити колір точок та ліній?
16. З допомогою якої вказівки можна змінювати колір фону?
17. З допомогою якої вказівки можна встановити вид штрихування?
18. Як здійснюється очищення екрану?
19. З допомогою якої вказівки здійснюється закриття графічного режиму?

Практичні завдання:

1. Підрахувати середнє арифметичне додатних елементів масивів A [1 .. N_1], B [1 .. N_2], C [1 .. N_3], якщо N_1 = 6, N_2 = 8, N_3 = 10 (використовуючи процедуру).
2. Написати програму обчислення значення функції $y = \frac{ctg^2x}{1+ctgx} - ctgx$, використовуючи можливість обчислення ctg(x) як функції (в підпрограмі).

Перелік питань для підготовки до екзамену

1. Системи числення. Позиційні та непозиційні системи числення. Переведення чисел $(\ )_p \rightarrow (\ )_{10}$.
2. Системи числення. Позиційні та непозиційні системи числення. Переведення чисел $(\ )_{10} \rightarrow (\ )_p$.
3. Методи опосередковування.
4. Двійкова система числення: алфавіт, використання. Дії над двійковими числами.
5. Математична логіка. Логічні величини та логічні операції. Побудова таблиць істинності.
6. Логічні елементи. Теорія автоматів.
7. Поняття алгоритму. Властивості алгоритмів. Способи запису алгоритмів. Базові структури алгоритмів.
8. Основні етапи розв'язування задачі з використанням ЕОМ.
9. Поняття мови програмування, програми. Класифікація мов програмування. Система програмування, її складові.
10. Мова програмування Паскаль: алфавіт, синтаксис. Структура програми. Типи змінних. Правила запису арифметичних виразів.
11. Поняття оператора: оператори введення-виведення, присвоєння.
12. Вказівка розгалуження, її синтаксис. Принцип дії вказівки розгалуження. Блок-схема.
13. Оператор вибору case, принцип дії, синтаксис.
14. Цикл з параметром, блок-схема, принцип дії та загальний вигляд.
15. Вказівка повторення з передумовою, блок-схема, принцип дії та загальний вигляд.
16. Вказівка повторення з післяумовою, блок-схема, принцип дії та загальний вигляд.
17. Запис рядкових величин. Функції та процедури для роботи з рядками.
18. Поняття одновимірного масиву. Опис, заповнення одновимірних масивів. Доступ до і-го елементу масиву. Виведення елементів одновимірного масиву на екран.
19. Поняття двовимірного масиву. Опис, заповнення масиву, виведення двовимірного масиву на екран. Звертання до елементу двовимірного масиву.
20. Методи сортування масивів.

Рекомендована література

1. Абрамов В. Г. Введение в язык Паскаль : учебное пособие / В. Г. Абрамов, Н. П. Трифонов, Г. Н. Трифонова. – М. : КНОРУС, 2011. – 384 с.
2. Глинський Я. М. Паскаль. Turbo Pascal і Delphi. 2-е доп. вид. / Я. М. Глинський, В. Є. Анохін, В. А. Ряжська. – Львів : «Деол», 2001. – 144 с.
3. Гусева А. И. Учимся программировать: PASCAL 7.0 [Текст] : Задачи и методы их решения: Учеб. пособие для студ. вузов / А. И. Гусева. – М. : Диалог-Мифи, 2003. – 254 с.
4. Зуев Е. А. Turbo Pascal. Практическое программирование [Текст] / Е. А. Зуев. – М. : Стрикс, 1997. – 336 с.
5. Йенсен К., Вирт Н. Паскаль. Руководство для пользователя и описание языка / К. Йенсен, Н. Вирт. – М. : Финансы и статистика, 1982. – 151 с.
6. Караванова Т. П. Інформатика. Основи алгоритмізації та програмування (процедурне програмування): навч. посібник / Т. П. Караванова. – К. : Аспект, 2005. – 289 с.
7. Культин Н. Б. Программирование в Turbo Pascal 7.0 и Delphi / Н. Б. Культин – С.-Пб. : BHV, 1998. – 240 с.
8. Ковалюк Т. В. Основи програмування / Т. В. Ковалюк. – К. : Видавнича група BHV, 2005. – 384 с.
9. Марченко А. И. Программирование в среде Turbo Pascal 7.0 [Текст] / А. И. Марченко, Л. А. Марченко. - К. : ТОО "ВЕК+", 2000. – 464 с.
10. Програмування в середовищі DELPHI. Методичні рекомендації до практичних робіт / [укл. Т. Г. Четверикова] – Луцьк: Терен, 2012. – 110 с.
11. Сердюченко В. Я. Розробка алгоритмів та програмування мовою Turbo Pascal [Текст] : навч. посіб. для техн. вузів / В. Я. Сердюченко. - Х. : Паритет, 1995. – 350с.
12. Черняхівський В. В. Збірник задач з основ алгоритмізації. Навч. посібник / В. В. Черняхівський. – Львів : ВНТЛ, 1997. – 195 с.
13. Шост Д. М. Основи інформатики та обчислювальної техніки. Turbo pascal. 10 - 11 класи [Текст] : Зошит-конспект / Д. М. Шост. - Т. : Навчальна книга - Богдан, 2000. – 192 с.
14. Юрченко І. В. Інформатика та програмування. Частина 1. Навчальний посібник / І. В. Юрченко. – Чернівці : Книги–ХХІ, 2011. – 203 с.