МІНІСТЕРСТВО ОСВІТИ УКРАЇНИ
УКРАЇНСЬКИЙ ДЕРЖАВНИЙ МОРСЬКИЙ
ТЕХНІЧНИЙ УНІВЕРСИТЕТ

МЕТОДИЧНІ ВКАЗІВКИ

ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З КУРСУ "ОСНОВИ ПРОГРАМУВАННЯ ТА АЛГОРИТМІЧНІ МОВИ"

Частина 1

МИКОЛАЇВ - 1999

ББК 22.18

С 32

УДК 681.3.06:800.92

Приходько С.Б. Методичні вказівки до виконання лабораторних робіт з курсу "Основи програмування та алгоритмічні мови". Частина 1. - Миколаїв: УДМТУ, 1999. - 58 с.

Кафедра інформаційних технологій

Методичні вказівки призначені для студентів першого курсу спеціальностей УДМТУ напрямку "Комп'ютерні науки", а також для студентів усіх інших спеціальностей, що вивчають основи програмування мовою Turbo Pascal 6.0 (7.0).

Рецензент к.т.н., доцент С.В.Суслов

ВСТУП

Відповідно з програмою Міністерства освіти України курс "Основи програмування та алгоритмічні мови" відноситься до циклу фундаментальних дисциплін для усіх спеціальностей напрямку "Комп'ютерні науки" (7.080401 - "Комп'ютеризовані системи обробки інформації і управління", 7.080403 - "Програмне забезпечення обчислювальної техніки і автоматизованих систем" та інші). В УДМТУ цей курс викладається студентам на протязі чотирьох семестрів: у першому та другому семестрах вивчається програмування мовою Pascal, а у третьому та четвертому - C і C++.

Методичні вказівки призначені для студентів першого курсу спеціальностей УДМТУ напрямку "Комп'ютерні науки", а також для студентів усіх інших спеціальностей, що вивчають основи програмування мовою Pascal. Методичні вказівки містять в собі 8 лабораторних робіт. Кожна з лабораторних робіт розрахована на 4 години і складається з коротких теоретичних відомостей, варіантів завдань, приклада виконання роботи та контрольних запитань.

Методичні вказівки орієнтовані на вивчення основ програмування мовою Turbo Pascal 6.0 або 7.0. **Усі розбіжності версії 7.0 від 6.0 в методичних вказівках виділені напівжирним шрифтом.**

Орієнтацію на версію 7.0 зумовлено тим, що модернізація версії 7.0 від попередньої версії 6.0 харектеризується рухом у напрямку Windows, та, відповідно, у напрямку мови C++ (базової для програмування у Windows), яка вивчається студентами на другому курсі. **Найважливішими моментами цієї модернізації є апарат ASCIIZ-рядків, моделі пам'яті `near` і `far`, а також відкриті масиви та рядки, які використовуються як параметри підпрограм.**

При вивченні основ програмування мовою Turbo Pascal студенти усіх спеціальностей (окрім спеціальностей напрямку "Комп'ютерні науки") за рішенням викладача мають можливість орієнтуватися *тільки на версію 6.0. Можуть бути також вилучені завдання 4.3 та* 5.3, а завдання 5.2 замінене на відповідне завдання з додатку 1.

Починати лабораторні роботи ми рекомендуємо з вступного заняття (2-4 години), на якому студенти знайомляться з роботою у середовищі системи програмування Turbo Pascal 6.0 (додаток 2).

Робота № 1

Розробка та реалізація програми з лінійною структурою

Ціль роботи: закріплення знань алфавіту мови програмування Turbo Pascal, придбання навичок запису його констант, змінних, виразів, операторів присвоєння; оволодіння навичками складання програми з лінійною структурою та виконання її у середовищі системи програмування Turbo Pascal 6.0.

Завдання:

1. Записати мовою Pascal математичні вирази (завдання 1.1)
2. Представити математичний запис виразу на мові Pascal і показати порядок дій (завдання 1.2).
3. Скласти програму обчислення наступних величин та виконати її у середовищі системи програмування Turbo Pascal 6.0 (завдання 1.3).

Короткі теоретичні відомості

Алфавіт мови Turbo Pascal складається з сукупності символів, які наведені у додатку 3. За допомогою символів алфавіту можна складати різноманітні конструкції: константи, змінні, оператори, тощо.

Постійна величина (константа) не змінюється в процесі роботи програми. Є декілька видів констант: числові (цілі та дійсні), символьні, рядкові, логічні.

Цілі константи складаються з послідовності десяткових цифр, перед якою може стояти знак "+" або "–". Приклад: `-15`

У мові Turbo Pascal цілі константи можна зображати і в шістнадцятковій системі числення. В цьому випадку константи починаються знаком "$", за яким йдуть шістнадцяткові цифри. Перед знаком "$" може бути "+" або "–". Приклад: `$2A`

Дійсні константи можна записувати одним з двох способів: як дійсну константу з фіксованою крапкою та як дійсну константу з плаваючою крапкою. Наприклад, число 0,015 можна записати як 0.015 та 0.15E–01, або 1.5E–02.

Символьна константа - це символ в лапках. Приклад: `'f'`

Рядкова константа - це послідовність символів в лапках. Наприклад: `'Введи X'`

Логічні константи приймають тільки два значення: `True` (Істина) та `False` (Хибність).

Коментар - це текст, обмежений знаками { і } або (* і *). Коментар може знаходитися у будь-якому місці програми, його довжина не обмежена. Коментар, обмежений знаками { і }, може бути поміщений в інший коментар, обмежений знаками (* і *), і

навпаки. Інші випадки вкладеності не допускаються. Приклад: `{Коментар(*вкладений коментар*)}`

Якщо безпосередньо за знаком початку коментаря { йде знак $, то такий коментар називається директивою компілятору. Так, включення файла t1.pas здійснює директива `{$I t1.pas}`.

Змінна визначається ім'ям (ідентифікатором), типом та значенням.

Ідентифікатор є послідовністю букв, цифр і знака підкреслювання, причому перший символ не є цифрою. Усі символи при цьому значущі і їх кількість не може перевищувати 127, але транслятор розпізнає тільки перші 63 символи. Великі та малі букви в іменах не розрізнюються. Букви кирилиці в ідентифікаторах використовувати не можна. Не можна використовувати як ідентифікатори службові слова, список яких наведений у додатку 4.

Мова Pascal є суворо типізованою мовою. Це означає, що кожна змінна повинна бути описана, тобто необхідно визначити її тип.

Для цілої змінної усі типи наведені у табл. 1.1.

Таблиця 1.1

Тип	Зміст	Діапазон	Розмір в байтах
Byte	коротка ціла без знаку	0..255	1
Shortint	коротка ціла із знаком	–128..127	1
Word	ціла без знаку	0..65535	2
Integer	ціла із знаком	–32768..32767	2
Longint	довга ціла із знаком	$(-2^{31})..(2^{31}-1)$	4

Для дійсної змінної усі можливі типи наведені у табл. 1.2.

Таблиця 1.2

Тип	Зміст	Діапазон	Кількість цифр	Розмір в байтах
Real	дійсний	$\pm(2{,}9\ 10^{-39}..\ 1{,}7\ 10^{38})$	11–12	6
Single	одинарна точність	$\pm(1{,}5\ 10^{-45}..\ 3{,}4\ 10^{38})$	7–8	4
Double	подвійна точність	$\pm(5{,}0\ 10^{-324}..\ 1{,}7\ 10^{308})$	15–16	8
Extended	розширена точність	$\pm(3{,}4\ 10^{-4932}..\ 1{,}1\ 10^{4932})$	19–20	10
Comp	з нульовою мантисою	$(-9{,}2\ 10^{18})..(9{,}2\ 10^{18})$	19–20	8

Стандартний тип для дійсної змінної – Real. Інші типи рекомендується вживати, якщо в ПЕОМ є математичний сопроцесор. Якщо сопроцесора нема, то його можна емулювати за допомогою спеціальної директиви транслятора `{$E+}`. При цьому, дійсні змінні будуть оброблятися програмно, що суттєво зменшує швидкість *виконання програми*.

Обробка даних виконується в виразах. Вираз складається з операцій і операндів. Операндом може бути константа, змінна або ім'я функції. Обчислення виконуються зліва направо з урахуванням пріоритету операцій. У мові Pascal є чотири рівня пріоритету операцій (подані в порядку зменшення):

1) @, операція заперечення not, унарний мінус;

2) мультиплікативні операції *, /, div, mod, and, shl, shr;

3) адитивні операції +, –, or, xor;

4) операції відношення =, <>, <, >, <=, >=, in.

Зміст, тип операндів, тип результату операції заперечення, мультиплікативних і адитивних операцій наведені у табл. 1.3. Прийняті позначення: Int - цілий, R - дійсний, B - логічний.

Таблиця 1.3

Операція	Зміст	Тип операндів	Тип результату
not	Арифметичне НІ	Int	Int
not	Логічне НІ	B	B
and	Арифметичне І	Int	Int
and	Логічне І	B	B
shl	Арифметичний зсув вліво	Int	Int
shr	Арифметичний зсув вправо	Int	Int
*	Множення	Int, R	Int, R
/	Ділення	Int, R	R
div	Цілочислове ділення	Int	Int
mod	Остача від ділення	Int	Int
+	Плюс	Int, R	Int, R
–	Мінус	Int, R	Int, R
or	Арифметичне АБО	Int	Int
or	Логічне АБО	B	B
xor	Арифметичне роздільне АБО	Int	Int
xor	Логічне роздільне АБО	B	B

Відзначимо, що на відміну від операції div при виконанні операції ділення "/" цілого на ціле у результаті набуваємо дійсну частку. Наприклад, 5 div 2 = 2, але 5/2 = 2.5.

Арифметичні операції and, or і xor здійснюють дії над операндами відповідно з таблицею істинності (табл. 1.4). Операнди записуються у десятковій формі, але під час виконання перекладаються в двійкову систему числення. Результат подається у десятковій формі.

Приклад. Обчислити результат виконання виразу 12 or 22. Операнди займають в пам'яті по 2 байта і у двійковій системі числення $12 = 0000000000001100_{(2)}$ та $22 = 0000000000010110_{(2)}$. Згідно з

табл. 1.4 набудемо $0000000000011110_{(2)}$, що відповідає числу 30 у десятковій формі. Отже, результатом виразу `12 or 22` є 30.

Операції `shl` та `shr` (`K shl N` та `K shr N`) забезпечують зсуви відповідно ліворуч та праворуч `N` двійкових розрядів числа `K`.

Таблиця 1.4

Операція	Дія	Вираз	A	B	Результат
and	Арифметичне І	A and B	1	1	1
			1	0	0
			0	1	0
			0	0	0
or	Арифметичне АБО	A or B	1	1	1
			1	0	1
			0	1	1
			0	0	0
xor	Арифметичне роздільне АБО	A xor B	1	1	0
			1	0	1
			0	1	1
			0	0	0

Використання операції `not` до даних цілого типу викликає побітну інверсію двійкового коду числа. Наприклад, результатом виконання виразів `not 0` та `not 127` є відповідно –1 та –128.

Логічні операції `not`, `and`, `or` і `xor` здійснюють дії над операндами відповідно з таблицею істинності (табл.1.5). Наприклад, при K=1 результатом виразу `(K>0) and (K<4)` є `True`.

Таблиця 1.5

Операція	Дія	Вираз	A	B	Результат
not ¬	Логічне НІ	not A	True		False
			False		True
and ∧	Логічне І	A and B	True	True	True
			True	False	False
			False	True	False
			False	False	False
or ∨	Логічне АБО	A or B	True	True	True
			True	False	True
			False	True	True
			False	False	False
xor	Логічне роздільне АБО	A xor B	True	True	False
			True	False	True
			False	True	True
			False	False	False

Структура програми. Програма мовою Pascal складається із заголовку і блоку. В заголовку вказуються ім'я програми та способи зв'язку програми з операційною системою. Мова Turbo Pascal розглядає заголовок як коментар.

Заголовок програми починається з ключового слова program, за яким йде ідентифікатор - ім'я програми. Ім'я програми є формальним. Параметрами програми можуть бути стандартні файли введення-виведення Input і Output, проте вони можуть бути опущені за замовчуванням. В кінці заголовка ставиться крапка з комою.

Блок містить описову і виконувану частини. В кінці блоку ставиться крапка. До описової частини можуть входити наступні розділи описів: міток (label); констант (const); типів (type); змін-них (var); процедур і функцій. Розділи описової частини можуть йти один за одним у будь-якому порядку і зустрічатися довільну кількість разів. Важливо лише, щоб описи йшли до використання об'єктів, які описано. Будь-які (або всі) з розділів цієї частини можуть бути відсутніми.

Кожна змінна, що використовується в програмі, повинна бути описана в розділі опису змінних. Опис складається з ключового слова var, імен змінних (через кому), двокрапки (:), імені типу або його опису і крапки з комою (;). Наприклад: var X,Y: real;

Виконувана частина містить розділ операторів. Вона присутня завжди і йде останньою. Виконувана частина починається ключовим словом begin, після якого йде послідовність операторів; а закінчується ключовим словом end і крапкою (.). Наприклад:

```
program First_1;
begin
 Writeln('Ми вивчаємо мову Turbo Pascal')
end.
```

Результат роботи програми First_1 є такий рядок:

```
Ми вивчаємо мову Turbo Pascal
```

Ще приклад

```
program First_2;
var X,Y: integer;
begin
 X:=9; Y:=X*X; Writeln(Y)
end.
```

У результаті роботи програми First_2 буде виведено число 81

Виконувана частина блока містить оператори, кожний з яких здійснює певні дії над даними. Відокремлюються оператори один від одного крапкою з комою (;). Оператори поділяються на прості та структуровані; структуровані, в свою чергу, - на складені, розгалуження і цикли.

Оператор присвоєння має такий загальний вигляд: A:=B, де A - ідентифікатор, а B - вираз. При виконанні цього оператора спочатку обчислюється вираз, а потім результат присвоюється змінній A, точніше значення виразу B записується в область пам'яті, яку займає змінна A. Тип змінної A і тип виразу B повинні бути однакові. Існує виняток: змінній дійсного типу можна присвоювати цілочислене значення. *Присвоювати можна дані будь-яких типів*, крім файлових.

Якщо змінній цілого типу необхідно присвоїти дійсне значення, використовують функцію округлення Round або функцію визначення цілої частини числа Trunc. Наприклад, у результаті виконання операторів присвоєння `I:=Round(3.7); J:=Trunc(3.7)` змінні цілого типу I та J дорівнюють I=4, J=3.

Процедури введення-виведення дають змогу працювати з різними зовнішніми пристроями введення-виведення. У цій роботі ми розглянемо введення з клавіатури і виведення на дисплей.

Процедура Read призначена для введення необхідних значень змінних. Формат процедури: Read(A1,..., An), де A1,..., An- змінні допустимих типів. При введенні значення A1,..., An відокремлюються пропуском. Введення з клавіатури завершується натискуванням клавіші Enter. Кількість значень, що вводяться, та їх типи *повинні відповідати кількості і типам змінних в процедурі Read.* У противному разі виникає помилка введення-виведення. Так, при виконанні оператора Read(A, B) дійсні змінні A і B набудуть відповідно значення 3.2 і 4.1, якщо введення значень буде таким: `3.2 4.1`

Процедура Write служить для виведення значень змінних, виразів і рядків символів. Формат процедури: Write(A1,..., An), де A1,..., An - змінні допустимих типів, вирази або рядки символів. При виведенні рядок береться в одинарні лапки.

Процедура Writeln аналогічна процедурі Write, проте на відміну від неї, процедура Writeln переводить курсор на наступний рядок.

Можна здійснювати також форматоване виведення даних. Для цього після імені змінної через двокрапку ставиться число - кількість знакомісць, які виділяються для виведення значення даної змінної. Якщо виводиться значення змінної дійсного типу, то можна також вказати і кількість знакомісць для виведення дробової частини значення змінної або виразу. Наприклад, при виконанні фрагмента програми

```
var N: integer; X,Y: real; L: boolean;
. . . . . .
N:=25; X:=51.2; Y:=2.0*X; L:=(X>1)and(X<90);
Writeln('N=',N:3,' X=',X:5:1,' Y=',Y:10,' L=',L);
```

результат матиме вигляд:

```
N= 25 X= 51.2 Y= 1.024E+02 L=TRUE
```

ЗАВДАННЯ 1.1

Дані математичні вирази записати мовою Pascal. Прийняті позначення: ∨ - об'єднання множин, ∧ - переріз множин.

№	Математичні вирази	№	Математичні вирази
1-3	а) $\frac{\ln\lvert 3z\rvert+\operatorname{arctg}2z^3}{3(z+1)^2+2{,}1\cdot10^6}$ б) $\ln\lvert x+z\rvert>0 \wedge 0<b<1$	4-6	а) $\frac{10^{-4}e^{-2f}+\ln\lvert z^3\rvert}{2(z+2)^{1,5}}$ б) $x+z<0 \vee 0<f<0{,}2$
7-9	а) $\frac{10^{-7}\ln\lvert 2z\rvert+\sin 2z^3}{3(z+3)^2+2{,}1\cdot10^7}$ б) $\lvert x+z\rvert>1 \wedge 1<b<2$	10-12	а) $\frac{10^{-7}\ln\lvert 3z\rvert+b^{0,4}}{\ln(z+1)^2+4{,}2\cdot10^4}$ б) $\lvert x\rvert>2 \vee 0<b<3$
13-15	а) $\frac{10^{-5}e^{-5f}+\sin^2\lvert z^3\rvert}{5(z+1)^5+10^6}$ б) $0<b<1 \vee 0<f<0{,}5$	16-18	а) $\frac{10^{-6}\ln\lvert 3z^3\rvert+\ln^2 z^3}{6(z+1)^6+10^6}$ б) $\cos\lvert x+z\rvert>0 \vee 0<b<6$
19-21	а) $\frac{\ln\lvert 7z\rvert+\operatorname{arctg}2z^2}{7(z+1)^{0,5}+2{,}7\cdot10^6}$ б) $\lvert x+z\rvert>0 \wedge 0<b<7$	22-24	а) $\frac{10^{-7}\sin\lvert 3z\rvert+b^{1,2}}{(z+1)^2+1{,}2\cdot10^6}$ б) $\ln\lvert x+z\rvert>0 \vee 0<b<1$
25-27	а) $\frac{10^{-7}\ln\lvert 9z^3\rvert+\cos 2z^2}{\lvert z+1\rvert^2+2\cdot10^6}$ б) $\lvert x+z\rvert>0 \wedge b>9$	28-30	а) $\frac{10^{-7}\ln\lvert 3z^3\rvert+\sin 2z^2}{(z+1)^{0,5}+10^6}$ б) $\lvert x+z\rvert>0 \vee 0<b<1$

ЗАВДАННЯ 1.2

Представити математичний запис виразу і показати порядок дій.

```
1. X+2/3/X/A+Sqr(Sin(X))/2*Sqrt(X)+1.0E-6*Exp(1/7*Ln(X))
2. (X+7)/3*X+3*ArcTan(X)/2/X+1.0E7-Sqr(1/3*Exp(5*Ln(X)))
3. X+2/3*X/A+Sqr(Cos(X))/2/Sqrt(X)+1.0E-5*Exp(7*Ln(X))
4. (X+4)/3/X+Exp(Abs(ArcTan(X)))/2*X+1.0E-6*Exp(1/3*Ln(X))
5. X+2/3/X/A+Sqr(Sin(X))/2/Ln(X)+1.0E5*Exp(2/7*Ln(X/3))
6. 1.4E-4*Exp(3*Ln(2*X))+Sqrt(Sin(X))/2+Sqr(Cos(X))/2/X
7. Sqr(Cos(X))/2/X-5/7*X/A/1.0E-6*Exp(1/3*Ln(X/2))*Abs(X)
8. X+2/3/X*A+Sqrt(Sin(X))/2/Ln(X)+1.0E-3*Exp(2/3*Ln(X/7))
9. (X+7)/3*X+3*ArcTan(X)/2/X+1.0E7-Sqr(4*Exp(B*Ln(X)))
10. Sqrt(Cos(Sqr(X)))/2/X-5/7*X/A/1.0E-6*Exp(1/8*Ln(X/2))
11. X+9/(3*X/A)+Sqr(Cos(X))/2/Sqrt(X)+1.0E-5*Exp(9*Ln(X))
12. X+4/3/X+Exp(Abs(ArcTan(X)))/2*X+1.0E-4*Exp(1/3*Ln(X))
13. Sqr(Cos(X))/2/(X-5/7)*X/A/1.0E-6*Exp(1/7*Ln(X/2))/11
14. X+2*3/X*A+Sqr(Sin(X))/(2*Ln(X)+1.0E-3*Exp(5/3*Ln(X)))
15. X+4/3/(X+Abs(ArcTan(X)))/2*X+1.0E-5*Exp(5/3*Ln(X))
```

ЗАВДАННЯ 1.3

Скласти програму обчислення наступних величин та виконати її у середовищі системи програмування Turbo Pascal 6.0. Позначення: N - це номер варіанта за списком групи.

№	Умова
1	Модуля вектора $5\,a + 10\,b$, якщо a={3; 2} і b={0; –1}
2-5	Суми усіх парних чисел від 2 до 50*N
6-9	Суми усіх двозначних цілих чисел, які кратні N
10	Кута між векторами a={1; 2} і b={1; –0,5}
11	Площі чотирикутника з вершинами $A(0;0)$, $B(-1;3)$, $C(2;4)$, $D(3;1)$
12	Суми одинадцяти перших членів арифметичної прогресії, якщо $a_3+a_9=8$
13	Периметра трикутника з вершинами $A(1; 1)$, $B(4; 1)$, $C(4; 5)$
14	Модуля вектора $-2\,a + 4\,b$, якщо a={3; 2} і b={0; –1}
15	Кутів трикутника з вершинами $A(0; 1{,}7)$, $B(2; 1{,}7)$, $C(1{,}5; 0{,}85)$
16	Шостого члена геометричної прогресії 5, –10, ...
17	Кута між векторами a={2; –4; 4} і b={–3; 2; 6}
18	Модуля вектора $a-b$, якщо $\|a\|=3$, $\|b\|=5$, та a і b утворюють кут у 120^0
19	Суми усіх двозначних цілих чисел
20	Модуля вектора $a + b$, якщо $\|a\|=11$, $\|b\|=23$, $\|a - b\|=30$
21-25	Суми усіх трьохзначних чисел, які у разі ділення на 5 дають остачу 26–N
26-30	Суми усіх непарних чисел від 3 до 5*N

Приклад виконання роботи

Завдання 1.1. Данні математичні вирази записати мовою Pascal:

а) $\dfrac{10^{-3}\cdot e^{-2f}+tg^2z}{5\cdot(z+1)^2\cdot b^{0,3}}$; б) $|b|<0 \wedge f>1$.

Розв'язання:

а) (1.0E–3*Exp(–2*F)+Sqr(Sin(Z)/Cos(Z)))/5/Sqr(Z+1)/Exp(0.3*Ln(B))

б) (Abs(B)<0) and (F>1)

Завдання 1.2. Представити математичний запис виразу 0.51E–6*Sqrt(3*X)+2*B*B і показати порядок дій.

Розв'язання:

$0{,}51\cdot10^{-6}\cdot(3x)^{0,5}+2\cdot b^2$ $\qquad$ $\overset{3}{0.51E}\overset{2}{-}\overset{1}{6}\overset{6}{*}\overset{4}{Sqrt}\overset{5}{(3*X)}+2*B*B$

Завдання 1.3. Скласти програму обчислення скалярного добутку двох векторів a={4; –2; –4} і b={6; –3; 2} та виконати її у середовищі системи програмування Turbo Pascal 6.0.

Розв'язання:

1. Постановка задачі

Скласти програму обчислення скалярного добутку двох векторів a={4; –2; –4} і b={6; –3; 2} на мові Turbo Pascal.

2. Методика розв'язання задачі

Скалярний добуток двох векторів обчислюється за формулою:

$$ab = A_1 \cdot B_1 + A_2 \cdot B_2 + A_3 \cdot B_3, \qquad (1.1)$$

де A_1, B_1, A_2, B_2, A_3, B_3 – відповідні координати векторів a і b.

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

3.1. Ввести координати векторів a і b;

3.2. Обчислити скалярний добуток за формулою (1.1);

3.3. Вивести значення скалярного добутку двох векторів.

Представимо алгоритм розв'язання задачі на мові Turbo Pascal, позначив змінні A_1, B_1, A_2, B_2, A_3, B_3 і ab відповідно як A1, B1, A2, B2, A3, B3 і AB (усі типу Real).

4. Текст програми

```
program LR1;
{Програма обчислення скалярного добутку двох векторів}
uses Crt; {Підключення стандартного модуля Crt}
var A1, A2, A3, B1, B2, B3, AB: real;
begin
  ClrScr; {Повністю очищає екран}
  Writeln(' Введіть координати векторів A і B:');
  Writeln(' A1, A2, A3, B1, B2, B3');
  Readln(A1,A2,A3,B1,B2,B3);
  AB:=A1*B1+A2*B2+A3*B3;
  Writeln(' Скалярний добуток двох векторів AB=',AB:10)
end.
```

5. Результати роботи програми

```
 Введіть координати векторів A і B:
 A1, A2, A3, B1, B2, B3
4.0 -2.0 -4.0 6.0 -3.0 2.0
 Скалярний добуток двох векторів AB= 2.200E+01
```

Контрольні питання

1. Які дані можна вживати в мові Turbo Pascal 6.0?
2. З якою метою використовують директиву {$E+}?
3. Назвіть порядок виконання операцій в виразі.
4. Яка структура програми на мові Turbo Pascal 6.0?
5. Як працює оператор присвоєння?
6. Як працюють процедури введення і виведення?

Робота № 2

Розробка та реалізація програми з розгалуженою структурою

Ціль роботи: оволодіння навичками складання програми з розгалуженою структурою за допомогою умовного оператора `if` або оператора вибору `case` та виконання її у середовищі системи програмування Turbo Pascal 6.0.

Завдання:

1. Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання (завдання 2.1).
2. Скласти програму обчислення значень функції та виконати її у середовищі системи програмування Turbo Pascal 6.0

Короткі теоретичні відомості

Умовний оператор `if` призначений для виконання або невиконання деякого оператора (простого або складеного) залежно від істинності тих чи інших умов.

Загальні вигляди умовного оператора `if`:

`if` *Вираз (умова)* `then` *оператор 1*;

`if` *Вираз (умова)* `then` *оператор 1* `else` *оператор 2*.

Виконання умовного оператора `if` полягає в обчисленні логічного *виразу (умови)*. Якщо його значення True, то виконується оператор що стоїть за словом `then`. Якщо значення логічного виразу False і умовний оператор не містить слова `else`, то його виконання завершується. Якщо слово `else` є, то виконується оператор, що стоїть після нього (*оператор 2*).

Якщо в якій-небудь вітці умовного оператора треба виконати кілька операторів, то їх слід об'єднати в складений оператор (`begin end`). Один умовний оператор може входити в інший умовний оператор. При цьому кожне слово `else` відповідає останньому перед ним `then`. Так, після виконання наступного фрагмента програми:

```
X:=3;
if (X>0)and(X<=1) then Y:=1
                  else if X>1 then Y:=10
                              else Y:=0;
```

змінна Y має значення 10.

Оператор вибору призначений для виконання одного з кількох можливих операторів. Він складається: з ключового слова `case`, за яким йде селекторний вираз; ключового слова `of`; послідовності операторів, кожному з яких передує значення виразу-селектора (або список, або деякий діапазон значень виразу-селектора), яке відокремлено від оператора двокрапкою (:); ключового слова `end`.

Як і умовний оператор, оператор вибору може містити ключове слово else, яке повинно стояти останнім перед end.

Загальний вигляд оператора вибору:

```
case вираз-селектор of
  список 1 : оператор 1;
             . . . . . .
  список N : оператор N
  else оператор
end
```

Виконання оператора вибору починається з обчислення значення *виразу-селектора*. При першому збігові цього значення із значенням *списку* (*1*,..., *N*) виконується відповідний *оператор*. Якщо жодного збігу не зафіксовано, а є слово else, то виконується *оператор*, наступний за else. У противному разі виконання оператора вибору завершується.

Вираз-селектор та значення виразу-селектора повинні бути того ж самого порядкового (ординального) типу. В списку значення виразу-селектора відокремлюються одне від одного комою. Кожне *значення в списку повинно зустрічатися лише один раз.*

Приклад. Після виконання наступного фрагмента програми:

```
var Letter: char;
. . . . . .
Letter:= 'M';
case Letter of
 'A','E','I','J','O','U','Y':Write('Англійська голосна');
 'B'..'D','F'..'H','K'..'N',
 'P'..'T','V'..'X','Z': Write('Англійська приголосна')
 else Write('Дещо інше')
end.
```

буде надруковано Англійська приголосна

ЗАВДАННЯ 2.1

Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання. Позначення: N - це номер варіанта за списком групи.

№	Фрагмент програми	№	Фрагмент програми

1-5	T:=N; X:=T;
	if(T>1)and(T<3) then X:=3;
	if(T<=1) then X:=0
		6-10	T:=N; X:=0;
			if T<0 then X:=-T
			else X:=T

11-15	A:=N; B:=13; C:=12; X:=A; if X<B then X:=B; if X<C then X:=C	16-20	A:=N; B:=17; C:=18; X:=A; if B<X then X:=B; if C<X then X:=C
21-25	X:=N; Y:=0; If X>22 then Y:=Sqr(X-22); if X<0 then Y:=X	26-30	X:=N; Y:=X/3; if(X>27)and(X<29) then Y:=9; if X<=27 then Y:=X

ЗАВДАННЯ 2.2

Скласти програму обчислення значень функції та виконати її у середовищі системи програмування Turbo Pascal 6.0

№	Умова	№	Умова	№	Умова	№	Умова
1	$y=tgx/lnx$	2	$y=lnx/tgx$	3	$y=arcsin\ x/x$	4	$y=ctg\ lnx$
5	$y=ctg\ x^{1/3}$	6	$y=tgx/(1-x)$	7	$y=tgx/ln^{2/3}x$	8	$y=x^{0,2}\cdot tgx$
9	$y=ln\ tg\ x$	10	$y=tg^{1/3}x/x$	11	$y=arcctg^{1/3}x$	12	$y=arccos\ x$
13	$y=arcsin\ x$	14	$y=tgx/(x-1)$	15	$y=lnx/(1-x^2)$	16	$y=x/(1+tgx)$
17	$y=lnx/(1-x)$	18	$y=arcctg\ x$	19	$y=tgx/lnx$	20	$y=x^{1/7}\cdot ctgx$
21	$y=arccos\ x/x$	22	$y=x/(1-x^2)$	23	$y=arcsin^{1/3}x$	24	$y=arcctg\ x/x$
25	$y=tg^2lnx$	26	$y=tg^{1/3}x$	27	$y=tgx/(1-x^2)$	28	$y=ln\ tgx^{1/3}$

Приклад виконання роботи

Завдання 2.1. Представити математичний запис фрагмента програми

```
T:= -8;
if T>0  then X:= exp(1/3*ln(T));
        else if T=0   then X:=0
                      else X:=-exp(1/3*ln(-T))
```

і обчислити значення змінної X після його виконання.

Розв'язання:

Цей фрагмент програми реалізує обчислення функції $x=t^{1/3}$. Після виконання цього фрагмента X=–2.

Завдання 2.2. Скласти програму обчислення значень функції $y=ctgx$ і виконати її у середовищі системи програмування Turbo Pascal 6.0.

Розв'язання:

1. Постановка задачі

Скласти програму обчислення значень функції $y=ctg\ x$ на мові Turbo Pascal.

2. Методика розв'язання задачі

Функція $y=ctg\ x$ обчислюється за формулою

$$y=cos\ x/sin\ x\ , \qquad (2.1)$$

якщо $sin\ x \neq 0$ (2.2)

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

3.1. Ввести значення x.

3.2. Перевірити умову (2.2). Якщо умова істина, то обчислити значення функції за формулою (2.1) і вивести його, інакше вивести повідомлення: 'Функція не існує'.

Остаточно представимо алгоритм розв'язання задачі на мові Turbo Pascal, позначив змінні x і y відповідно як X і Y (обидві типу Real).

4. Текст програми

```
program LR2;
{програма обчислення функції Y=Cos(X)/Sin(X)}
uses Crt;
var X, Y: real;
begin
  ClrScr;
  Write(' Введіть X=');
  Readln(X);
  if Sin(X)<>0 then  begin
                       Y:=Cos(X)/Sin(X);
                       Writeln(' Y=',Y:10)
                     end
                else Writeln(' Функція не існує')
end.
```

5. Результати роботи програми

```
Введіть X=1.57
Y= 7.963E-04
```

Контрольні питання

1. Як працюють оператори if і case?
2. Коли застосовують складений оператор?
3. Як виконується умовний оператор, якщо до нього входить інший умовний оператор?

Робота № 3

Розробка та реалізація програми з циклічною структурою

Ціль роботи: оволодіння навичками складання програми з циклічною структурою за допомогою операторів циклу `while`, `repeat until` і `for` та виконання її у середовищі системи програмування Turbo Pascal 6.0.

Завдання:

1. Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання (завдання 3.1).
2. Скласти програму табулювання функції з завдання 2.2 при зміні значення x від –1 до 1 з кроком 0,2 та виконати її у середовищі системи програмування Turbo Pascal 6.0 (завдання 3.2).

Короткі теоретичні відомості

Оператор циклу з передумовою `while` складається з ключового слова `while`, за яким йдуть *вираз* логічного типу (*умова*), ключове слово `do` та виконуваний у циклі *оператор* (простий чи складений).

Загальний вигляд оператора циклу з передумовою:

`while` *Вираз (умова)* `do` *оператор (тіло циклу).*

Виконання оператора циклу з передумовою починається з обчислення значення *виразу*. Якщо це значення False, то *тіло циклу* не виконується (управління передається на оператор, який стоїть одразу за циклом). Якщо значення *виразу* True, *тіло циклу* виконується, після чого знову обчислюється *умова*. Щоб запобігти зацикленню, слід передбачити зміну значення *виразу* (*умови*) всередині *тіла циклу*. Наприклад, після виконання наступного фрагмента програми:

```
A:= True; X:= 5;
while A or (X< 9) do {цикл завершиться, як тільки}
begin                {вираз набуде значення False}
  A:=not A; X:=X+2
end
```

змінна X має значення 11, а змінна A – False.

Оператор циклу з післяумовою `repeat until` складається з ключового слова `repeat`, за яким іде виконуваний у циклі *оператор* (послідовність операторів); ключового слова `until` і *виразу* логічного типу (*умови*).

Загальний вигляд оператора циклу з післяумовою:

`repeat` *оператор (тіло циклу)* `until` *Вираз (умова).*

Виконання цього оператора циклу відбувається так. Спочатку виконується *тіло циклу*, а потім визначається значення *виразу* логічного типу (*умови*). Якщо значення *виразу* True, то виконання циклу припиняється. Якщо це значення False, то відбувається виконання *тіла циклу*, а потім знову обчислюється *вираз* (*умова*). Наприклад, після виконання наступного фрагмента програми:

```
A:= True; X:= 5;
repeat
  A:=not A; X:=X+2    {цикл завершиться, як тільки}
until not(A or(X< 9)) {вираз набуде значення True}
```

змінна X має значення 11, а змінна A – False.

Ще приклад. Оператор `repeat until KeyPressed` застосовується для організації паузи в процесі виконання програми (до натискування будь-якої клавіші). В мові Turbo Pascal логічна функція `KeyPressed` контролює натискування клавіш. Якщо жодна з клавіш не натиснута, то ця функція виробляє значення False, в противному разі - True.

Треба підкреслити, що на відміну від циклу `while`, тіло циклу з післяумовою `repeat until` завжди виконується хоча б один раз і його виконання завершується, коли вираз логічного типу набуде значення True.

Оператор циклу з параметром `for` складається з ключового слова `for`, за яким йдуть параметр циклу; знак присвоєння; початковий вираз, що визначає початкове значення параметра циклу; ключове слово `to` або `downto`; кінцевий вираз, що визначає кінцеве значення параметра циклу; ключове слово `do` і виконуваний у циклі оператор (простий або складений).

Загальний вигляд оператора циклу `for`:

`for` *параметр*:=*Вираз1* `to` {`downto`} *Вираз2* `do` *оператор*,

де *параметр* – змінна цілого, символьного, логічного або перечислюваного типу. Параметр циклу (*параметр*), початковий та кінцевий вирази (відповідно *Вираз1* і *Вираз2*) повинні бути одного типу.

При виконанні циклу *параметр* набуває послідовних значень даного типу в порядку їх збільшення (якщо є слово `to`) або зменшення (якщо є слово `downto`). Кількість виконань тіла циклу обчислюється як: *Вираз2 – Вираз1 + 1* (якщо є слово `to`) або *Вираз1 – Вираз2 + 1* (якщо є слово `downto`). Якщо в циклі з словом `to` (`downto`) початковий вираз більший (менший) від кінцевого, то тіло циклу не виконується жодного разу. Всередені тіла циклу не слід змінювати значення параметра циклу, хоча така зміна не

фіксується як помилка і не впливає на кількість виконань тіла циклу. Після закінчення роботи оператора циклу for значення параметра циклу дорівнює кінцевому значенню. Якщо тіло циклу не виконалося, то значення параметра циклу не визначено.

Приклад, після виконання наступного фрагмента програми:

```
A:=True; X:=5;
for K:=1 to 3 do {цикл завершиться, як тільки K}
begin            {набуде значення більше за 3}
  A:=not A; X:=X+2
end
```

змінна X має значення 11, а змінна A – False.

Ще приклад. В результаті виконання наступного оператора: `for Ch:='z' downto 'a' do Writeln(Ch)` будуть надруковані малі букви англійського алфавіту в зворотному порядку.

Мова Turbo Pascal 7.0 доповнена двома стандартними процедурами без параметрів break та continue, семантика яких повністю співпадає з аналогічними операторами мови С.

Процедура break припиняє виконання оператора циклу, з якого була викликана ця процедура. Управління передається на оператор, який стоїть одразу за оператором циклу.

Процедура continue припиняє виконання поточної ітерації оператора циклу, в якому була викликана ця процедура, і здійснює перехід к виконанню наступної ітерації.

ЗАВДАННЯ 3.1

Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання. Позначення: N - це номер варіанта за списком групи.

№	Фрагмент програми	№	Фрагмент програми

1-5	`X:=1;`
	`for J:=7 downto N do`
	`  X:=X*J;`
	`X:=2*X`
		6-10	`X:=0; J:=1;`
			`repeat`
			`  X:=X+J; J:=J+2`
			`until J>N`

11-15	`X:=0;`
	`for J:=1 to N do`
	`  X:=X+2;`
	`X:=2*X`
		16-20	`X:=1;`
			`while X<=N do`
			`  X:=X+1;`
			`X:=2*X`

21-25	`X:=1; J:=1; X1:=N div 5;`
	`repeat`
	`  X:=X*X1; J:=J+1;`
	`  if J=4 then break`
	`until J>5`
		26-30	`X:=N;`
			`for K:=1 to 6 do`
			`begin`
			`  if K<3 then continue;`
			`  X:=X+1`
			`end`

ЗАВДАННЯ 3.2

Скласти програму табулювання функції з завдання 2.2 при зміні значення x від –1 до 1 з кроком 0,2 та виконати її у середовищі системи програмування Turbo Pascal 6.0

Приклад виконання роботи

Завдання 3.1. Представити математичний запис фрагмента програми

```
X:=1;
for J:=1 to 5 do
  X:=X*J
```

и обчислити значення змінної X після його виконання.

Розв'язання:

Цей фрагмент програми реалізує обчислення $x!=1\cdot2\cdot\ldots\cdot5$. Після виконання цього фрагмента X=120.

Завдання 3.2. Скласти програму табулювання функції $y=ctg\,x$ при зміні значення x від $a=-1$ до $b=1$ з кроком $h=0{,}5$ і виконати її у середовищі системи програмування Turbo Pascal 6.0

Розв'язання:

1. Постановка задачі

Скласти програму табулювання функції $y=ctgx$ при зміні значення x від $a=-1$ до $b=1$ з кроком $h=0{,}5$ на мові Turbo Pascal.

2. Методика розв'язання задачі

Методика розв'язання задачі збігається з методикою з приклада завдання 2.2 (при оформленні роботи треба навести методику наново).

3. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

3.1. Ввести значення a, b, h;

3.2. Присвоїти x значення a;

3.3. Обчислити n (кількість повторень тіла циклу) для змінювання x від a до b з кроком h за формулою $n = ((b - a)/h)+1$;

3.4. Повторювати n разів наступні дії:

3.4.1. Перевірити умову (2.2). Якщо умова істинна, то обчислити значення функції y за формулою (2.1) і вивести x та y, інакше вивести повідомлення: 'не існує';

3.4.2 Присвоїти x нове значення, яке дорівнює старому значенню x плюс крок h.

Запишемо алгоритм розв'язання задачі мовою Turbo Pascal, позначив змінні x, y, a, b, h відповідно як X, Y, A, B, H (усі типу Real), а n як N (тип Integer), та, врахував те, що оскільки кількість

повторень тіла циклу заздалегідь відома, то логічніше вживати цикл for з параметром циклу I (типу Integer).

4. Текст програми

```
program LR3;
{програма табулювання функції Y=Cos(X)/Sin(X)}
uses Crt;
var X, Y, A, B, H: real;
    I, N: integer;
begin
   ClrScr;
   Writeln(' Введіть A, B, H');
   Readln(A, B, H);
   N:=Trunc((B-A)/H)+1; X:=A;
   for I:=1 to N do
   begin
      if Sin(X)<>0
         then
            begin
               Y:=Cos(X)/Sin(X);
               Writeln(' X=',X:6:2,' Y=',Y:11)
            end
         else Writeln(' X=',X:6:2,' Y= не існує');
      X:=X+H
   end
end.
```

5. Результати роботи програми

```
 Введіть A, B, H
-1.0 1.0 0.5
 X= -1.00 Y=-6.4209E-01
 X= -0.50 Y=-1.8305E+00
 X=  0.00 Y= не існує
 X=  0.50 Y= 1.8305E+00
 X=  1.00 Y= 6.4209E-01
```

Контрольні питання

1. Як працюють оператори циклу while do, repeat until, for do?
2. Чим цикл repeat until відрізняється від циклу while?
3. Коли застосовують складений оператор?
4. Дані якого типу можна вживати як параметр циклу for?
5. Як працюють процедури break та continue?

Робота № 4

Розробка та реалізація програми з масивами

Ціль роботи: оволодіння навичками складання програми з масивами та виконання її у середовищі системи програмування Turbo Pascal 6.0, придбання уміння використання алгоритму послідовного пошуку в масивах.

Завдання:

1. Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання (завдання 4.1).

2. Скласти програму обчислення наступних величин та виконати її у середовищі системи програмування Turbo Pascal (завдання 4.2).

3. Оцінити ефективність алгоритму послідовного пошуку (завдання 4.3).

Короткі теоретичні відомості

Масив - це упорядкована послідовність однойменних елементів, кожен з яких має один і той самий тип. Елементи масиву можуть мати любий стандартний тип (окрім файлового) або тип, введений користувачем. Тип елементів масиву називається базовим. Елементи масиву розміщені впорядковано, кожен має свій номер, який називається індексом. Доступ до елементів масиву відбувається шляхом вказування імені масиву та порядкового номера елемента (індексації). Індексом може бути будь-який вираз порядкового типу. Тип індексу визначає межі зміни значень індексу.

Опис масиву складається з ключового слова `array`, за яким в квадратних дужках записано тип індексу, ключового слова `of` і опису типу елементів. Загальний вигляд опису масиву:

`type` *ім'я типу* = `array` [*тип індексу*] `of` *тип елементів*;

`var` *ім'я* масиву: *ім'я типу*;

або

`var` *ім'я* масиву: `array` [*тип індексу*] `of` *тип елементів*;

Наприклад,

```
type  RGB = (red, green, blue);
      Nm = 1..MaxInt;
      Ai = array [1..20] of integer;{20 елементів}
var M,L0: Ai;
    A,B: array [1..10] of integer;
    Color: array [RGB] of byte;
    MatrC: array [1..3,1..4] of real;{3 рядки, 4 стовпця}
    A2: array [Nm] of Ai;
```

Якщо базовим типом є масив, то утворюється багатовимірний масив. У наведеному прикладі змінні A2 і MatrC є двовимірні масиви. Опис змінної MatrC еквівалентний такому опису

```
var MatrC: array [1..3] of array [1..4] of real;
```

Елементи масиву зображаються у програмі за допомогою імені елемента масиву, який складається з імені масиву та індексу в квадратних дужках. Для наведеного вище прикладу A[1] – ціла величина, Color[green] – байтова величина, MatrC[1,2] – дійсна величина. Для багатовимірних масивів допускається ще така форма доступу до елемента: MatrC[1][2] – відповідно те саме, що й MatrC[1,2].

Елементи масиву розміщуються у пам'яті послідовно. Елементи з меншими значеннями індексу зберігаються в більш низьких адресах пам'яті. Елементи багатовимірних масивів розміщуються таким чином, що самий правий індекс зростає самим першим. Так, для масиву D: array[1..2, 1..2] of real елементи масиву розміщуються у пам'яті за зростанням адресів: D[1,1], D[1,2], D[2,1], D[2,2].

Задача пошуку. Задані одновимірні масиви: масив елементів $A=\{a_1, a_2, \ldots, a_n\}$ та список ключів $K=\{k_1, k_2, \ldots, k_l\}$. Потрібно для кожного значення $k_i \in K$ знайти всі елементи з A, які дорівнюють k_i. Часто зустрічається ситуація, коли K містить один ключ k і масив A має не більш одного такого елемента.

Ефективність алгоритму пошуку S оцінюється максимальною max(S) та середньою avg(S) кількістю порівнянь, необхідних для визначення елемента k, за формулами

$$\max(S) = \max\{q_i\}, \qquad (i=1,2,\ldots,n); \qquad (4.1)$$

$$\text{avg}(S) = \sum_{i}^{n} f_i q_i , \qquad (4.2)$$

де q_i – кількість порівнянь, потрібних для пошуку елемента a_i; f_i – відносна частота використання елемента a_i, $f_i=m_i/n$; m_i – абсолютна частота використання елемента a_i.

Послідовний пошук. Усі елементи масиву переглядаються послідовно в порядку розміщення, поки не знайдеться елемент, рівний k. Якщо не відомо, чи є шуканий елемент, то потрібно стежити, щоб пошук не вийшов за межі масиву.

Очевидно, що max послідовного пошуку дорівнює n. Якщо частота f кожного елемента масиву однакова, $f=1/n$, то avg послідовного пошуку дорівнює $n/2$. За різної частоти використання можна поліпшити avg, розміщуючи елементи, що зустрічаються частіше, на початку масиву.

ЗАВДАННЯ 4.1

Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання, якщо елементи масиву визначаються за формулою A[I+1]=(37*A[I]+3) mod 64. *Значення A[1] дорівнює номеру варіанта за списком групи.*

№	Фрагмент програми	№	Фрагмент програми
1-5	`T:=2; N:=3; X:=A[1]; for J:=1 to N do X:=X*T+A[J+1];`	6-10	`N:=4; X:=A[N]; for J:=N-1 downto 1 do X:= A[J]+1/X;`
11-15	`N:=4; X:=A[1]; for J:=2 to N do if A[J]<X then X:=A[J];`	16-20	`T:=3; N:=3; X:=A[N+1]; for J:=1 to N do X:= X+A[J]*Exp((N-J-1)*Ln(T));`
21-25	`N:=4; M:=N div 2; K:=N; for J:=1 to M do begin Y:=A[J]; A[J]:=A[K]; A[K]:=Y; K:=K-1 end; X:=A[1];`	26-30	`N:=4; X:=0; K:=0; for J:=1 to N do if A[J]>0 then begin X:=X+A[J]; K:=K+1 end;if K<>0 then X:=X/K;`

ЗАВДАННЯ 4.2

Скласти програму обчислення наступних величин та виконати її у середовищі системи програмування Turbo Pascal 6.0, якщо елементи масиву визначаються за формулою a_{i+1}= (37·a_i+3) mod 64. Значення a_1 дорівнює N (номеру варіанта за списком групи); i змінюється від 1 до 19.

№	Умова
1-3	Найбільшого елемента масиву ***a*** і його порядкового номера.
4-6	Суми елементів масиву ***a***, значення яких кратні N
7-9	Суми елементів масиву ***a***, значення яких парні числа
10-12	*Середнього арифметичного додатних елементів масиву* ***a***.
13-15	Суми елементів масиву ***a***, значення яких непарні числа
16-18	Середнього геометричного додатних елементів масиву ***a***.
19-21	Суми елементів масиву ***a***, значення яких двозначні парні числа
22-24	Добутку найбільшого і найменшого елементів масиву ***a***
25-27	Суми елементів масиву ***a***, значення яких двозначні непарні числа
28-30	Модуля вектора ***a/3***

ЗАВДАННЯ 4.3

Оцінити ефективність алгоритму послідовного пошуку, для чого:

1. Створити одновимірний масив ***a*** з 20 елементів за формулою a_{i+1}=(37·a_i+3) mod 64. Значення a_1 дорівнює N (номеру варіанта за списком групи); i змінюється від 1 до 18; a_{20} дорівнює 190–G, де G - номер групи.

2. Скласти програму послідовного пошуку ключа k в масиві ***a***, якщо k=N+1.

3. Обчислити avg за формулою (4.2).

4. Якщо можливо, зменшити avg, розміщуючи елементи, що зустрічаються частіше, на початку масиву, та обчислити avg.

5. Зробити висновки щодо поліпшення ефективності алгоритму послідовного пошуку.

Приклад виконання роботи

Завдання 4.1. Представити математичний запис фрагмента програми

```
N:=4; X:=A[1];
for I:=2 to N do if A[I]<X then X:=A[I]
```

и обчислити значення змінної X після його виконання, якщо елементи визначаються за формулою A[I+1]=(37*A[I]+3) mod 64. Значення A[1] дорівнює 40.

Розв'язання:

Цей фрагмент програми находить найменший елемент одновимірного масиву (вектора) a={40; 11; 26; 5}, $x = \min a_i$, (i=1, 2, 3, 4). Після виконання цього фрагмента X=5.

Завдання 4.2. Скласти програму перестановки елементів масиву a в зворотному порядку і виконати її у середовищі системи програмування Turbo Pascal 6.0, якщо елементи масиву визначаються за формулою a_{i+1}= (37·a_i+3) mod 64. Значення a_1 дорівнює N (номеру варіанта за списком групи); i змінюється від 1 до 17.

Розв'язання:

1. Постановка задачі

Скласти програму перестановки елементів масиву a в зворотному порядку на мові Turbo Pascal, якщо елементи масиву визначаються за формулою a_{i+1}= (37·a_i+3) mod 64. Значення a_1 дорівнює 40; i змінюється від 1 до 17.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

2.1. Ввести елементи масиву a;

2.2. Визначити M - кількість перестановок елементів масиву a, як результат цілочислового ділення числа елементів масиву N на 2;

2.3. Присвоїти J - номеру поточного елемента масиву, який переставляється на місце елемента з меншим номером - значення N;

2.4. Повторювати M разів наступні дії (I= 1, 2, ..., M):

2.4.1. Присвоїти X - змінної для тимчасового зберігання елемента з меншим номером - значення I-го елемента масиву a;

2.4.2. Присвоїти I-му елементу масиву a значення J-го елемента;

2.4.3. Присвоїти J-му елементу масиву a значення X;

2.4.4. Зменшити значення J на 1;

2.5. Надрукувати елементи масиву *a* після перестановки.

Запишемо алгоритм розв'язання задачі мовою Turbo Pascal, позначив масив *a* через A, елементи якого мають тип Integer. Змінні I, J, N, M, X мають тип Integer. Врахував те, що оскільки кількість повторень тіла циклу заздалегідь відома, то логічніше вживати цикл for з параметром циклу I (типу Integer).

3. Текст програми

```
program LR4;
{програма перестановки елементів масиву A[1..N]}
uses Crt;
const N=18;
var A: array[1..N] of integer; I, J, M, X : integer;
begin
  ClrScr; Writeln(' Вводимо масив A[1..',N:2,']');
  A[1]:=40;
  for I:=1 to N-1 do A[I+1]:=(37*A[I]+3) mod 64;
  for I:=1 to N do Write(A[I]:3); Writeln;
  M:=N div 2; J:=N;
  for I:=1 to M do
  begin
    X:=A[I]; A[I]:=A[J]; A[J]:=X; J:=J-1
  end;
  Writeln(' Масив A після перестановки');
  for I:=1 to N do Write(A[I]:3); Writeln;
end.
```

4. Результати роботи програми

```
Вводимо масив A[1..18]
40 11 26  5 60 47 14  9 16 19  2 13 36 55 54 17 56 27
Масив A після перестановки
27 56 17 54 55 36 13  2 19 16  9 14 47 60  5 26 11 40
```

Завдання 4.3. Оцінити ефективність алгоритму послідовного пошуку.

Розв'язання:

1. Постановка задачі

Оцінити ефективність алгоритму послідовного пошуку, для чого:

1.1. Створити масив *a* за формулою $a_{i+1}=(37 \cdot a_i+3) \bmod 64$. Значення a_1 дорівнює 40; i змінюється від 1 до 17; $a_{18}=30$.

1.2. Скласти програму послідовного пошуку ключа k в масиві *a*, якщо $k=41$.

1.3. Обчислити avg за формулою (4.2).

1.4. Якщо можливо, зменшити avg, розміщуючи елементи, що зустрічаються частіше, на початку масиву, та обчислити avg.

1.5. Зробити висновки щодо поліпшення ефективності алгоритму послідовного пошуку.

2. Текст програми

```
program LR4_2;
{програма послідовного пошуку ключа K в масиві A[1..N]}
uses Crt;
const N=20;
var A: array[1..N] of integer; I, M, K: integer;
begin
   ClrScr; Write(' Ввести ключ K='); Readln(K);
   Writeln(' Вводимо масив A[1..',N:2,']');
   A[1]:=40; A[18]:=30; {створення масиву}
   for I:=1 to N-2 do A[I+1]:=(37*A[I]+3) mod 64;
   for I:=1 to N do Write(A[I]:3)
   Writeln; M:=0; {послідовний пошук}
   for I:=1 to N do
    if A[I]=K then
    begin Writeln(' A[',I:2,']=',A[I]:3); M:=1
    end;
   if M=0 then
    Writeln(' Ключ ',K,' в масиві не зустрічається');
end.
```

3. Результати роботи програми

```
Ввести ключ K=41
Вводимо масив A[1..18]
 40 11 26  5 60 47 14  9 16 19  2 13 36 55 54 17 56 30
Ключ 41 в масиві не зустрічається
```

4. Обчислення avg

$$\text{avg}(S) = \sum_{i=1}^{18} f_i q_i = 9$$

5. Зменшення avg

Зменшити avg не можливо тому, що частота використання кожного елемента масиву однакова, f=1/18.

6. Висновки

Поліпшити ефективність послідовного пошуку ключа k=41 в масиві не можливо тому, що частота використання кожного елемента масиву однакова.

Контрольні питання

1. **Який тип можуть мати елементи масиву?**
2. **Як розміщуються в пам'яті елементи масиву?**
3. **Як формулюється задача пошуку?**
4. **Як оцінити ефективність алгоритму пошуку?**
5. **Як виконується послідовний пошук?**

Робота № 5

Розробка та реалізація програми з вкладеними циклами

Ціль роботи: оволодіння навичками складання програми з вкладеними циклами та виконання її у середовищі системи програмування Turbo Pascal 6.0; придбання уміння використовувати найпростіші методи сортування масивів та методи пошуку для упорядкованих масивів.

Завдання:

1. Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання (завдання 5.1).
2. Скласти програму сортування масиву визначеним методом та виконати її у середовищі Turbo Pascal (завдання 5.2).
3. Оцінити ефективність алгоритму бінарного або m-блочного пошуку (завдання 5.3).

Короткі теоретичні відомості

Тіло циклу одного оператора циклу може містити інші оператори циклу. Такі цикли називаються вкладеними. При складанні програм з вкладеними циклами треба пам'ятати про наступне:

- передача керування дозволена з внутрішнього циклу в зовнішній, але не навпаки;
- область дії внутрішнього циклу не повинна виходити за межі зовнішнього циклу.

Крім того, для операторів циклу `for` діє ще одне правило: імена параметрів зовнішнього і внутрішнього циклів повинні бути відмінними. В випадку вкладених циклів `for` при кожному фіксованому значенні параметра зовнішнього циклу параметр внутрішнього циклу пробігає усі свої значення. Наприклад, у результаті виконання фрагмента програми

```
X:= 0;
for I:=1 to 2 do     {I=1;                  I=2}
begin
  P:=1;              {P=1;                  P=1}
  for J:=1 to 3 do   {J=1; J=2; J=3;        J=1; J=2; J=3}
    P:=P*J;          {P=1; P=2; P=6;        P=1; P=2; P=6}
  X:=X+P             {X=6;                  X=12}
end;
```

змінна X отримає значення 12.

При роботі з двовимірними масивами (матрицями) вкладені цикли рекомендується організовувати так, щоб у зовнішньому циклі змінювався індекс рядка (перший індекс), а у внутрішньому - індекс стовпця (другий індекс), що підвищує ефективність Pascal-

програми завдяки послідовному зчитуванню елементів масиву з пам'яті. Для виведення елементів матриці A (N рядків, M стовпців) по рядкам використовують наступний фрагмент:

```
for I:=1 to N do
begin
  for J:=1 to M do Write(A[I,J]);
  Writeln
end;
```

Задача сортування. Задається одновимірний масив елементів $A= \{a_1, a_2, \ldots, a_n\}$. Потрібно переставити елементи масиву A так, щоб дістати упорядкований масив $A' = \{a'_1, a'_2, \ldots, a'_n\}$, де для будь-якого $1 \le i \le n$ елемент $a'_i \le a'_{i+1}$.

Обмінне сортування. Масив A' будується з A систематичним обміном двох суміжних елементів, що не відповідають потрібному порядку, доки такі пари існують. Найпростіший метод систематичного обміну сусідніх елементів з неправильним порядком при перегляді всього масиву зліва направо визначає бульбашкове сортування: максимальні елементи ніби бульбашки спливають у кінці масиву. При цьому враховується, що цілій змінній m присвоюється значення 1, в противному разі їй присвоюється 0. Наприклад,

$A = \{12, 5, 10, 8\}$,
$A = \{5, 10, 8, 12\}$, $m=1$
$A = \{5, 8, 10, 12\}$, $m=1$
$A = \{5, 8, 10, 12\}$, $m=0$.

У всіх наступних прикладах вважаємо, що сортується масив в порядку зростання його елементів.

Сортування вставкою. Упорядкований масив A' будується з A таким чином. Спочатку першим його елементом вибирається a_1. Потім виконується вставка елемента a_i , $i=2,\ldots,n$ так, щоб A' залишався упорядкованим масивом довжини i. Наприклад, для $A = \{12, 5, 10, 8\}$:

$A = \{5, 10, 8\}$, $A'=\{12\}$
$A = \{10, 8\}$, $A'=\{5, 12\}$
$A = \{8\}$, $A'=\{5, 10, 12\}$
$A = \{\ \}$, $A'=\{5, 8, 10, 12\}$

При сортуванні вставкою масиву з n елементів кількість порівнянь має порядок $O(n^2)$, і додаткова пам'ять не потрібна.

Сортування за допомогою вибору. Упорядкований масив A' будується з A багаторазовим застосуванням вибору з A мінімального елемента, вилученням його з A і додаванням в кінці масиву A', який спочатку має бути порожнім. Наприклад,

$A = \{12, 5, 10, 8\}$, $A' = \{\ \}$
$A = \{12, 10, 8\}$, $A' = \{5\}$
$A = \{12, 10\}$, $A' = \{5, 8\}$
$A = \{12\}$, $A' = \{5, 8, 10\}$
$A = \{\ \}$, $A' = \{5, 8, 10, 12\}$.

При сортуванні за допомогою вибору здійснюється $O(n^2)$ порівнянь, і додаткова пам'ять не потрібна.

Сортування квадратичним вибором. Початковий масив A з n елементів ділиться на m підмасивів $A_1, A_2, \ldots, A_m$, де $m = n^{0,5}$, приблизно рівної довжини, і в кожному A_i відшукується мінімальний елемент b_i. Найменший елемент масиву A визначається як мінімальний елемент b_i масиву $B = \{b_1, b_2, \ldots, b_m\}$. Вибраний елемент b_i в масиві B замінюється новим найменшим елементом з масиву A_i і т.д. Процес продовжується до упорядкування всього масиву. При сортуванні квадратичним вибором кількість порівнянь має порядок $O(n \cdot n^{0,5})$, але використовується додаткова пам'ять.

Методи пошуку. Для упорядкованих одновимірних масивів існують ефективніші за послідовний пошук методи.

Бінарний пошук полягає в тому, що ключ k порівнюється з середнім елементом масиву. Якщо вони рівні, то шуканий елемент знайдено, інакше пошук продовжується в одній половині масиву.

Бінарним пошуком елемент визначається досить швидко. Доведено, що max бінарного пошуку дорівнює $\log_2 n$. Якщо частота використання кожного елемента однакова, і avg бінарного пошуку дорівнює $\log_2 n$.

m-Блочний пошук полягає в тому, що початковий упорядкований масив A з n елементів ділиться на m підмасивів $A_1, A_2, \ldots, A_m$ відповідно з $n_1, n_2, \ldots, n_m$ елементів ($n_1+n_2+\ldots+n_m = n$), таких, що $A_1, A_2, \ldots, A_m = A$. При пошуку ключа k спочатку визначається перший з масивів A_i ($1 \le i \le m$), останній елемент якого більший від k, а потім до нього застосовується послідовний пошук. Якщо довжини всіх підмасивів приблизно однакові, $m = n^{0,5}$, то max m-блочного пошуку дорівнює $2 \cdot n^{0,5}$. За однакової частоти використання всіх елементів avg m-блочного пошуку дорівнює $n^{0,5}$.

Описані алгоритми пошуку ускладнюються, коли не відомо, чи є в масиві шуканий елемент k, а коли є, то чи точно один. Можливі такі варіанти: або шуканого елемента в масиві немає, або їх в масиві кілька. Якщо замість одного ключа є упорядкований масив K ключів, то послідовний або m-блочний пошуки можуть виявитися

зручнішими і швидшими від бінарного, оскільки вони не потребують повторної ініціації для кожного нового ключа з K.

ЗАВДАННЯ 5.1

Представити математичний запис фрагмента програми і обчислити значення змінної X після його виконання, якщо елементи масиву A (який описується як `var A: array [1..3, 1..3] of integer;`) визначаються за формулами A[L, M]=B[I] (L, M=1,2,3; I=1, 2,..., 9) та B[I+1]=(37*B[I]+3) mod 64 (I=1, 2,..., 8). Значення B[1] дорівнює номеру варіанта за списком групи.

№	Фрагмент програми	№	Фрагмент програми

1-5	`X:=A[1,3]; N1:=0; N:=3;`
	`for I:=1 to N do`
	`for J:=I to N do`
	`if X<A[I,J] then`
	`begin X:=A[I,J];`
	`N1:=I end;`
		6-10	`X:=A[1,3]; N1:=0; N:=3;`
			`for I:=1 to N do`
			`for J:=N-I+1 to N do`
			`if A[I,J]<X then`
			`begin X:=A[I,J];`
			`N1:=I end;`

11-15	`X:=A[1,3]; N1:=0; N:=3;`
	`for I:=1 to N do`
	`for J:=N-I+1 to N do`
	`if X<A[I,J] then`
	`begin X:=A[I,J];`
	`N1:=I end;`
		16-20	`X:=A[1,3]; N1:=0; N:=3;`
			`for I:=1 to N do`
			`for J:=I to N do`
			`if X>A[I,J] then`
			`begin X:=A[I,J];`
			`N1:=I end;`

21-25	`X:=A[1,3]; N1:=0; N:=3;`
	`for I:=1 to N do`
	`for J:=N-I+1 to N do`
	`if X>A[I,J] then`
	`X:=A[I,J];`
		26-30	`X:=A[1,3]; N1:=0; N:=3;`
			`for I:=1 to N do`
			`for J:=I to N do`
			`if X<A[I,J] then`
			`X:=A[I,J];`

ЗАВДАННЯ 5.2

Скласти програму сортування масиву a в порядку зростання його елементів визначеним методом та виконати її у середовищі системи програмування Turbo Pascal 6.0, якщо елементи масиву визначаються за формулою $a_{i+1}= (37 \cdot a_i+3) \bmod 64$. Значення a_1 дорівнює N (номеру варіанта за списком групи); i змінюється від 1 до 19. Метод сортування визначається за наступною таблицею.

№	Метод сортування	№	Метод сортування
1-8	Сортування за допомогою вибору	9-18	Сортування вставкою
19-24	Сортування квадратичним вибором	25-30	Обмінне сортування

ЗАВДАННЯ 5.3

Оцінити ефективність алгоритму бінарного пошуку (непарні варіанти) та m-блочного пошуку (парні варіанти) за завданням 4.3.

Приклад виконання роботи

Завдання 5.1. Представити математичний запис фрагмента програми

```
var  A:array[1..3, 1..3] of integer;
     R:array [1..3] of integer;
. . . . . . . . . . . . . . .
for I:=1 to 3 do
begin
  X:=0;
  for J:=1 to 3 do X:=X+A[I,J];
  R[I]:=X
end;
```

і обчислити значення змінної X після його виконання, якщо A[1,1]=40, A[1,2]=11, A[1,3]=26, A[2,1]=5, A[2,2]=60, A[2,3]=47, A[3,1]=14, A[3,2]=9, A[3,3]=16.

Розв'язання:

Цей фрагмент програми обчислює суму елементів у кожному рядку матриці A. В результаті його виконання змінна X набуває значення 39, а R[1]=78, R[2]=112 і R[3]=39.

Завдання 5.2. Скласти програму сортування масиву a в порядку зростання його елементів методом бульбашкового сортування та виконати її у середовищі системи програмування Turbo Pascal 6.0 (7.0), якщо елементи масиву визначаються за формулою $a_{i+1}= (37 \cdot a_i+3) \bmod 64$. Значення a_1 дорівнює N (номеру варіанта за списком групи); i змінюється від 1 до 17.

Розв'язання:

1. Постановка задачі

Скласти програму сортування масиву a в порядку зростання його елементів методом бульбашкового сортування, якщо елементи масиву визначаються за формулою $a_{i+1}= (37 \cdot a_i+3) \bmod 64$; $a_1=40$, $i= 1,2,\ldots, 17$.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

2.1. Ввести елементи масиву a;

2.2. Повторювати наступні дії до виконання умови M≠1:

2.2.1. Присвоїти M значення 0;

2.2.2. Повторювати N-1 разів наступні дії:

2.2.2.1. Присвоїти X - змінної для тимчасового зберігання меншого за значенням елемента - значення (I+1)-го елемента масиву a;

2.2.2.2. Присвоїти (I+1)-му елементу масиву *a* значення I-го елемента;

2.2.2.3. Присвоїти I-му елементу масиву *a* значення X;

2.2.2.4. Присвоїти M значення 1;

2.3. Надрукувати елементи масиву *a* після сортування.

Запишемо алгоритм розв'язання задачі мовою Turbo Pascal, позначив масив *a* через A, елементи якого мають тип Integer. Змінні I, N, M, X мають тип Integer. Врахував те, що оскільки кількість повторень тіла циклу (2.2.2) заздалегідь відома, то логічніше вживати цикл for з параметром циклу I (типу Integer).

3. Текст програми

```
program LR5;
{бульбашкове сортування елементів масиву A[1..N]}
uses Crt;
const N=18;
var A: array[1..N] of integer; I, M, X: integer;
begin
 ClrScr; Writeln(' Вводимо масив A[1..',N:2,']');
 A[1]:=40;
 for I:=1 to N-1 do A[I+1]:=(37*A[I]+3) mod 64;
 for I:=1 to N do Write(A[I]:3); Writeln;
 repeat
  M:=0;
  for I:=1 to N-1 do
   if A[I]>A[I+1] then
   begin X:=A[I+1]; A[I+1]:=A[I]; A[I]:=X; M:=1
   end
 until M<>1;
 Writeln(' Масив A після сортування');
 for I:=1 to N do Write(A[I]:3); Writeln
end.
```

4. Результати роботи програми

```
 Вводимо масив A[1..18]
 40 11 26  5 60 47 14  9 16 19  2 13 36 55 54 17 56 27
 Масив A після сортування
  2  5  9 11 13 14 16 17 19 26 27 36 40 47 54 55 56 60
```

Контрольні питання

1. Які цикли називаються вкладеними?
2. Назвіть правила роботи з вкладеними циклами.
3. Як рекомендується організовувати вкладені цикли при роботі з двовимірнимі масивами?
4. Які методи використовуються для сортування масивів?
5. Як виконуються бінарний та m-блочний пошуки?

Робота № 6

Розробка та реалізація програми з використанням процедур та функцій

Ціль роботи: оволодіння навичками складання програми з використанням процедур і функцій та виконання її у середовищі системи програмування Turbo Pascal 6.0

Завдання:

1. *Обчислити значення змінних* X, Y *після виконання фрагмента* програми (завдання 6.1).

2. Скласти програму обчислення величин із завдання 4.2 з використанням процедур або функцій і виконати її у середовищі системи програмування Turbo Pascal 6.0 (завдання 6.2).

Короткі теоретичні відомості

У мові Pascal передбачено засоби, завдяки яким можна оформляти послідовність операторів як підпрограму. Розрізняють два види підпрограм: процедури і функції.

Усі процедури і функції поділяють на два класи: стандартні (зарезервовані) і визначені користувачем. Стандартні процедури й функції є частиною мови, вони заздалегідь не описуються. Процедури і функції, визначені користувачем, обов'язково описуються в розділі опису підпрограм. Сам опис не передбачає жодних дій. Для виконання процедури в програмі є оператор виклику процедури. Функція буде виконана, якщо її ім'я зустрінеться у деякому виразі.

Процедура має ту саму структуру, що й програма, і складається із заголовка і блока (тіла). Заголовок процедури містить ключове слово `procedure`, *ім'я процедури* і, якщо необхідно, список *формальних параметрів* із зазначенням *типу* кожного з них. *Тип параметра* повинен бути простим або описаним за допомогою імені раніше введеного типу. Блок процедури аналогічний блоку Pascal-*програми, крім того, що після останнього* `end` *процедури ставиться* крапка з комою (;).

Загальний вигляд опису процедури:

`procedure` *ім'я процедури*(*ім'я формального параметра*: *тип*;...
`var` *ім'я формального параметра: тип*);
розділ описів
`begin`
розділ операторів
`end`;

Функція має ту ж саму структуру, що й процедура, крім того, що *ім'я функції* є вихідним параметром, тобто може повертати один результат простого типу. Заголовок функції містить ключове слово

function, *ім'я функції*, необов'язковий список *формальних параметрів* із зазначенням *типу* кожного з них та імені *типу функції*.

Загальний вигляд опису функції:

```
function ім'я функції (ім'я формального параметра: тип; ...
                        ім'я формального параметра: тип): тип функції ;
розділ описів
begin
   розділ операторів
end;
```

Параметри процедури і функції дають змогу при кожному виклику процедури чи функції працювати з об'єктами, що задаються в момент виклику через список фактичних параметрів. При використанні формальних і фактичних параметрів необхідно пам'ятати про наступні правила:

- кількість формальних і фактичних параметрів повинна бути однаковою;
- перший фактичний параметр відповідає першому формальному, другий – другому і т.д.;
- кожний фактичний параметр повинен мати той самий тип, що й відповідний йому формальний параметр.

Turbo Pascal підтримує три види параметрів: параметри-значення, параметри-змінні і нетипізовані параметри-змінні. Цім трьом видам параметрів відповідають три способи передачі параметрів: за значенням, за посиланням і передача нетипізованих параметрів за посиланням.

У версії 7.0 був доданий ще один вид: параметр-константа. Насправді параметр-константа є поодинокій випадок перших двох попередніх видів параметрів; його особливість полягає у забороні зміни параметра-константи.

Параметр-значення - це локальна змінна для підпрограми. В списку формальних параметрів він зображається як параметр, перед яким є відсутнім ключове слово var і за яким іде його тип. Фактичний параметр при цьому може бути константою, змінною або виразом. У момент виклику процедури чи функції відбувається обчислення виразу, і отримане значення присвоюється локальній змінній. У подальшому при виконанні підпрограми ця змінна зовсім не відрізняється від інших локальних змінних. Зміна формальних параметрів-значень всередині процедури чи функції не викликає зміни відповідних фактичних параметрів.

Для опису параметра-змінної у заголовку підпрограми використовується ключове слово var, яке ставиться перед відповідними змінними, та ідентифікатор типу, який іде за параметром. Якщо

підпрограма містить формальний параметр-змінну, то фактичний параметр може бути лише змінною того самого типу, а формальний параметр позначає цю змінну на весь час виконання підпрограми. Характерною ознакою параметра-змінної є той факт, що будь-яка зміна формального параметра викликає зміну фактичного параметра. Параметри-змінні використовуються тоді, коли цей параметр призначений для передачі деякого значення у місце виклику підпрограми.

Область дії імені. Опис імені (константи, типу, змінної, процедури, функції) поширюється на весь блок, в якому міститься цей опис. Область, в якій справедливий опис імені, визначається областю дії цього імені.

Ім'я, яке описане в блоці основної програми, називається глобальним. Ім'я, яке описане в блоці підпрограми, називається локальним. Ім'я є глобальним відносно блока, якщо воно описане в зовнішньому (по відношенню до заданого) блоку. Опис локального імені за межами "свого" блоку не визначений. Якщо у вкладеному блоці описане локальне ім'я, що збігається з глобальним, то областю дії цього локального імені буде "свій" блок, за його межами діє опис глобального імені. Наприклад, при виконанні програми

```
program ELR6;
var A,X,Y: real;
procedure D(var X,Y:real);
begin
  A:=2*X;{зміна значення глобальної змінної A=2*2=4}
  Y:=A*X+1 {Y=4*2+1=9}
end;
begin
  A:=2; X:=3; Y:=2;
  D(Y,X);
  Y:=A*Y+X; {Y=4*2+9=17}
  Writeln(X,Y)
end.
```

дістаємо результат: X= 9, Y= 17.

Побічним ефектом підпрограми називають зміну значення глобальної змінної всередені підпрограми. Якщо такий ефект не передбачений програмістом спеціально, то він може стати джерелом помилок, що досить важко виявити. У наведеному вище прикладі побічним ефектом процедури є зміна значення змінної A.

Якщо процедура чи функція викликає сама себе, то вона називається рекурсивною. Глибина рекурсії, тобто кількість викликів, у системі Turbo Pascal не обмежується. Реально вона залежить від ресурсів пам'яті конкретного комп'ютера.

В загальному випадку рекурсивність - це не властивість самої процедури чи функції, а властивість її опису. Рекурсивна підпрограма, як правило, коротша і наочніша, але при виконанні вимагає більше часу і пам'яті за рахунок повторних звертань до самої себе і дублювання локальних змінних. Необхідно добре розуміти, що кожний черговий рекурсивний виклик приводить до утворення нової копії локальних об'єктів підпрограми і усі ці копії, що відповідають ланцюжкові активізованих і незавершених рекурсивних викликів, існують незалежно один від одного.

У випадку занадто довгого ланцюжка викликів, сегмент стека, у якому динамічно розміщуються локальні змінні, може переповнитися, що приведе до аварійного закінчення програми. З другого боку, якщо заздалегідь відомо, що підпрограми не використовуються (або слабо використовуються), то значна частина сегмента стека не буде використана.

У мові Turbo Pascal є директива $M, яка, зокрема, служить для завдання розміру стека. Приклад: {$M 1024, 0, 0}

Для контролю переповнення стека перед викликом кожної підпрограми використовується директива {$S+}. Якщо задано директиву {$S–}, то стан стека не перевіряється.

У мові Turbo Pascal 7.0 підтримуються дві моделі виклику процедур і функцій - ближня `near` та дальня `far`. Процедури, які створені з використанням моделі `near`, є більш швидкими (ефективними), але застосування цієї моделі накладає деякі обмеження - процедури типу `near` можуть бути викликані тільки з модуля, де вони описані. Процедури типу `far`, можуть бути викликані з будь-якого місця програми. Недоліком процедур типу `far` є їх повільність.

Таблиця 6.1

ЗАВДАННЯ 6.1

Обчислити значення змінних X, Y після виконання фрагмента програми. Вказівка: замість N підставити номер варіанта.

№	Програма	№	Програма
1	2	3	4

1-5	`var A,X,Y:real;`
	`procedure D(var X,Y:real);`
	`begin`
	`  X:=2*X; Y:=A*X+1`
	`end;`
	`begin`
	`  A:=N; X:=3; Y:=4;`
	`  D(Y,X); Y:=A*X;`
	`  Write(X,Y)`
	`end.`
		6-10	`var A,X,Y:real;`
			`procedure D(var X,Y:real);`
			`begin`
			`  X:=2*X; Y:=A*X+1`
			`end;`
			`begin`
			`  A:=N-5; X:=3; Y:=4;`
			`  D(A,X); Y:=A*X;`
			`  Write(X,Y)`
			`end.`

1	2	3	4

11-15	var A,X,Y:real;
	procedure D(var X,Y:real);
	begin
	X:=2*X; Y:=A*X+1
	end;
	begin
	A:=N-5; X:=2; Y:=3;
	D(A,X);
	Write(X,Y)
	end.
		16-20	var A,X,Y:real;
			procedure D(var X,Y:real);
			begin
			X:=2*X; Y:=A*X+1
			end;
			begin
			A:=N-10; X:=3; Y:=2;
			D(A,Y); Y:=A*X+Y;
			Write(X,Y)
			end.

21-25	var X,Y:real;
	function F(X:real):real;
	begin
	if X=0 then F:=1
	else F:=2*X*F(X-1)
	end;
	begin
	X:=N-18; Y:=F(X);
	Write(X,Y)
	end.
		26-30	var X,Y:real;
			function F(X:real):real;
			begin
			if X=0 then F:=0
			else F:=X+F(X-1)
			end;
			begin
			X:=N-23; Y:=F(X);
			Write(X,Y)
			end.

ЗАВДАННЯ 6.2

Скласти програму обчислення величин із завдання 4.2 з використанням процедур або функцій і виконати її у середовищі системи програмування Turbo Pascal 6.0

Приклад виконання роботи

Завдання 6.1. Обчислити значення змінних X, Y після виконання такої програми:

```
var X,Y: real;
function F(X:real):real;
begin
  if X=1 then F:=1
         else F:=X*F(X-1)
end;
begin
  X:=4; Y:=F(X); Write(X,Y)
end.
```

Розв'язання:

Ця програма обчислює F!=4*3*2*1 з використанням рекурсивної функції. У результаті її виконання X=4, а Y=24.

Завдання 6.2. Скласти програму перестановки елементів масиву *a*={40; 11; 26; 5; 60; 47; 14; 9; 16; 19; 2; 13; 36; 55; 54; 17; 56; 27} в зворотному порядку з використанням процедури або функції і виконати її у середовищі системи програмування Turbo Pascal 6.0

Розв'язання:

1. Постановка задачі

Скласти програму перестановки елементів масиву a={40; 11; 26; 5; 60; 47; 14; 9; 16; 19; 2; 13; 36; 55; 54; 17; 56; 27} в зворотному порядку на мові Turbo Pascal з використанням процедури.

2. Алгоритм розв'язання задачі

Алгоритм наведений у прикладі розв'язання завдання 4.2 (при оформленні роботи алгоритм необхідно навести наново).

3. Текст програми

```
program LR6;
{програма перестановки елементів масиву A[1..N]}
uses Crt;
const N=18;
type Vect= array [1..N] of integer;
var A: Vect; I: integer;
procedure Rev(N:integer; var A:Vect);
var X,I,J,M: integer;
begin
  M:=N div 2; J:=N;
  for I:=1 to M do
  begin
    X:=A[I]; A[I]:=A[J]; A[J]:=X; J:=J-1
  end
end;
begin
  ClrScr; Writeln(' Введіть масив A[1..',N:1,']');
  for I:=1 to N do Read (A[I]); Readln;
  Rev(N, A); Writeln(' Масив A після перестановки');
  for I:=1 to N do Writeln(A[I]:5:1); Writeln;
end.
```

4. Результати роботи програми

```
 Введіть масив A[1..18]
40 11 26 5 60 47 14 9 16 19 2 13 36 55 54 17 56 27
 Масив A після перестановки
 27 56 17 54 55 36 13  2 19 16  9 14 47 60  5 26 11 40
```

Контрольні питання

1. Яка різниця між фактичними і формальними параметрами?

2. Що таке глобальна змінна?

3. Коли використовують параметри-значення, а коли параметри-змінні?

4. Чим функція відрізняється від процедури?

5. Які особливості використання процедур та функцій є у мові Turbo Pascal 7.0?

Робота № 7

Розробка та реалізація програми з використанням рядкового типу даних

Ціль роботи: оволодіння навичками складання програми з використанням *рядкового типу даних та виконання її у середовищі системи* програмування Turbo Pascal 6.0

Завдання:

1. Визначити дію фрагмента програми і обчислити значення змінних X та Y після його виконання (завдання 7.1).

2. Скласти програму для виконання наступних дій і виконати її у середовищі системи програмування Turbo Pascal (завдання 7.2).

Короткі теоретичні відомості

Рядок - це послідовність довільних символів, які при використанні беруться в апострофи. В рядку може міститися від 0 до 255 символів. Опис рядкового типу складається з ключового слова `string`, за яким в квадратних дужках може бути записано максимальну кількість символів, яку може мати змінна описуваного типу. Наприклад, `var S1: string[30];`

Символи рядка мають номери від 0 до 255, причому ASCII код символу з номером 0 дорівнює поточній довжині даного рядка. До окремого символу рядка можна звернутися за допомогою індексу, так само, як при звертанні до елемента масиву.

Для контролю правильності діапазону значень номера символу призначена директива компілятора {$R+}. Якщо встановлена така директива, то при спробі доступу до елемента, наприклад, S[76] масиву з попереднього приклада, буде зафіксована помилка. За замовчуванням встановлюється директива компілятора {$R–}, тобто перевірка не проводиться.

Над даними рядкового типу визначено операції додавання (конкатенації), відношення і оператор присвоєння.

Зчеплення двох чи більше рядків в один реалізується за допомогою операції конкатенації, яка позначається знаком "+". Додаватися можуть рядки довільної довжини, але необхідно, щоб довжина рядка-результату не перевищувала 255. Так, у результаті конкатенації 'Turbo '+'Pascal '+''+'v.6.0' маємо 'Turbo Pascal v.6.0'. Відзначемо: пара апострофів, що йдуть підряд, зображає порожній символ. Він не відображається на екрані і не має порядкового номера.

Довільні два рядки можна порівнювати один з одним за допомогою операцій віднощення (=, <>, <, >, <=, >=). Порівняння рядків відбувається зліва направо, до першого відмінного символу. Два рядки рівні, якщо вони мають однакову довжину і складаються з однакових символів, що йдуть в одному порядку. Меншим буде той

рядок, в якому символ, що не збігається, має менший порядковий (ординальний) номер. Якщо рядки мають різну довжину, але в спільній частині збігаються, то меншим є коротший рядок.

Вираз	Результат
'PASCAL'>'Pascal'	False
'Turbo '>='Turbo'	True

Оператор присвоєння використовується для присвоєння рядковій змінній результату виразу рядкового типу. Якщо вираз рядкового типу містить більше символів, ніж може містити рядкова змінна, то зайві праві символи відкидаються. В операторі присвоєння допускається змішування символьного і рядкового типів. Якщо при цьому символьній величині присвоюється рядок, поточна довжина якої більше 1, то виникає помилка при виконанні.

Для обробки рядків є стандартні процедури і функції При подальшому описі цих процедур і функцій під типом anystring будемо розуміти будь-який рядковий тип. Для обробки рядків є такі стандартні процедури: Delete, Insert, Str, Val.

Delete(var S: anystring; N,M: integer) - вилучення M символів з рядка S, починаючи з символу N. Якщо N більше довжини рядка, то не відбувається ніяких дій. Якщо N+M більше довжини рядка, то вилучаються всі символи до кінця рядка. Значення N повинно лежати в межах 1..255. Так, після виконання наступного фрагмента програми: S:='Turbo'; Delete(S,2,3); змінна S має значення 'To'.

Insert(S: anystring; var R: anystring; N: integer) - вставлення рядка S в рядок R перед символом з номером N. Якщо N більше довжини рядка, то відбувається конкатенація R і S. Якщо довжина результату перевищує максимальну довжину рядка R, то в R залишаються тільки ліві символи. Значення N повинно лежати в межах 1..255. Так, після виконання наступного фрагмента програми: R:='Turbo'; Insert('.com',R,6); змінна R має значення 'Turbo.com'.

Str(N: integer{*або* real}; var S: anystring) - перетворення цілого або дійсного значення N у рядок S. Після N може бути вказаний формат, як у процедурі Write. Якщо у форматі вказано кількість позицій, яка більша за довжину рядка S, то зайві праві символи будуть відкинуті. Наприклад, після виконання оператора Str(13:3,S); змінна S має значення ' 13'.

Val(S: anystring; var N: integer{*або* real}; var Ier: integer) - перетворення рядка S у число N. Рядок S повинен бути зображенням деякого числа і не повинен містити початкових і кінцевих пропусків. Якщо перетворення відбулось успішно, то Ier=0, в противному разі значення N не визначене, а змінна Ier дорівнює номеру першого помилкового символу в рядку S. Так, після виконання

оператора Val('3.14',N, Ier) змінна N має значення 3.14 (для N дійсного типу, Ier=0) і не визначена, якщо N має цілий тип (Ier=2).

Для обробки рядків є такі стандартні функції: Length, Pos, Copy, Concat.

Length(S: anystring): byte - результатом є поточна довжина рядка S, тобто кількість в ньому символів. Наприклад, результатом виразу Length('Turbo') є 5.

Pos(S,T:anystring): byte - результатом є номер символу, починаючи з якого рядок S входить в рядок T. Якщо входження немає, то результатом буде 0. Так, результатом виразу Pos('ur','Turbo') є 2.

Copy(S: anystring; N,M: integer): anystring - результатом є рядок з M символів рядка S, починаючи з символу з номером N. Якщо N більше довжини рядка S, то результатом функції буде порожній символ. Якщо N+M більше довжини рядка S, то результатом функції будуть кінцеві символи, починаючи із символу з номером N. Значення N повинно лежати в межах 1..255. Наприклад, результатом виразу Copy('Turbo',3,2) є 'rb'.

Concat(S1,S2,...,Sn: anystring): anystring - конкатенація рядків S1,S2,...,Sn в один у вказаному порядку. Якщо довжина рядка-результату перевищить 255, то виникне помилка при виконанні. Так, результатом виразу Concat('Tur','bo') є 'Turbo'.

ЗАВДАННЯ 7.1

Визначити дію фрагмента програми і обчислити значення змінних X та Y після його виконання, якщо `var X, S: string[18]; A:string[1]; Y:integer; X:='N'+'a1b2c3d4'; S:='24'+'N'; Y:=512.` Вказівка: замість N підставити номер варіанта.

№	Фрагмент програми	№	Фрагмент програми
1-3	`X:= Concat(Copy(X,1,3), Copy(X,7,Length(X)-6)); Y:= Length(X);`	4-6	`X:= Concat(Copy(X,1,2),S, Copy(X,8,Length(X)-7)); Y:= Length(X);`
7-9	`Y:= 0; X:=S; for I:= 1 to Length(S) do Y:=10*Y+Ord(S[I])-Ord('0')`	10-12	`Y:= Length(X) div 2; for I:= 1 to Y do Delete(X,2*I,1);`
13-15	`Y:= Length(X) div 2; for I:= 1 to Y do Delete(X,2*I-1,1);`	16-18	`Y:= Length(X) div 2-1; for I:= 1 to Y do Insert('+',X,3*I-1);`
19-21	`X:= Concat(Copy(X,2,2),S, Copy(X,7,Length(X)-6)); Y:=Length(X);`	22-24	`Y:=0; Delete(S,2,1);X:=S; for I:= 1 to Length(S) do Y:=10*Y+Ord(S[I])-Ord('0')`
25-27	`J:=Length(X); Y:=J div 2; for I:= 1 to Y do begin A:=X[I]; X[I]:= X[J]; X[J]:=A[1];J:=J-1 end;`	28-30	`J:=Length(S); Y:=J div 2; for I:= 1 to Y do begin A:=S[I]; S[I]:= S[J]; S[J]:=A[1];J:=J-1 end;`

ЗАВДАННЯ 7.2

Скласти програму для виконання наступних дій і виконати її у середовищі системи програмування Turbo Pascal 6.0

№	Умова
1-3	Визначення кількості слів в рядку.
4-6	Вилучення усіх цифр в рядку.
7-9	Інвертування символів в рядку.
10-12	Визначення кількості цифр в рядку.
13-15	Визначення слова з найменшою кількістю літер в рядку.
16-18	Визначення кількості чисел в рядку.
19-21	Визначення слова з найбільшою кількістю літер в рядку.
22-24	Заміна усіх великих букв в рядку на малі.
25-27	Вилучення зайвих символів "пробіл" в рядку.
28-30	Заміна усіх малих букв в рядку на великі.

Приклад виконання роботи

Завдання 7.1. Визначити дію фрагмента програми

```
Y:=0; Minus:=false;
if (X[1] = '+') or (X[1] = '-')
   then  begin Minus:=(X[1] = '-'); Delete(X,1,1)
         end;
for I:=1 to Length(X) do Y:=10*Y+Ord(X[I])-Ord('0');
if Minus then Y:=-Y;
```

і обчислити значення змінних X і Y після його виконання, якщо

```
var X:string[18]; A:string[1]; Y:integer; X:='-1024'; Y:=512
```

Розв'язання:

Цей фрагмент програми перетворює значення рядкової змінної X у цілочислене і присвоює результат змінній Y. Після його виконання X='1024', а Y=-1024.

Завдання 7.2. Скласти програму змінювання порядку слів в рядку на протилежний та виконати її у середовищі системи програмування Turbo Pascal 6.0

Розв'язання:

1. Постановка задачі

Скласти програму змінювання порядку слів в рядку на протилежний на мові Turbo Pascal.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі можна представити у вигляді такої послідовності дій:

2.1. Ввести рядок (OldLine);

2.2. Новому рядку (NewLine) присвоїти порожній рядок (рядок, який не містить символів, тобто '');

2.3. Додати пропуск до старого рядка (OldLine);

2.4. Повторювати наступні дії поки в рядку OldLine не будуть вичерпані усі слова або поки цей рядок не стане порожнім:

2.4.1. Визначити номер (Numb) позиції межи першого слова в старому рядку, тобто номер самої лівої позиції зразка ' ';

2.4.2. *Зкопіювати з старого рядка перше слово (Word1);*

2.4.3. Утворити новий рядок (NewLine) шляхом конкатенації (сцеплення) рядків: Word1 і NewLine;

2.4.4. Вилучити з старого рядка слово Word1 з урахуванням пропуску;

2.5. Надрукувати рядок NewLine після зміни порядку слів.

Остаточно представимо алгоритм на мові Turbo Pascal.

3. Текст програми

```
program LR7;
{змінювання порядку слів в рядку на протилежний}
uses Crt;
var  NewLine, OldLine, Word1: string;
     Numb: integer;
begin
   ClrScr;
   Writeln(' Введіть рядок'); Readln(OldLine);
   NewLine:=''; OldLine:= Concat(OldLine,' ');
   while OldLine <> '' do
   begin Numb:=Pos(' ', OldLine);
         Word1:=Copy(OldLine, 1, Numb);
         NewLine:=Concat(Word1, NewLine);
         Delete(OldLine, 1, Numb)
   end;
   Writeln(' Рядок після зміни порядку слів');
   Writeln(NewLine)
end.
```

4. Результати роботи програми

```
 Введіть рядок
Pascal Turbo
 Рядок після зміни порядку слів
Turbo Pascal
```

Контрольні питання

1. Що таке конкатенація рядків?

2. *Що робе кожна з наступних стандартних процедур:* Delete, Insert, Str, Val?

3. Що робе кожна з наступних стандартних функцій: Length, Pos, Copy, Concat?

4. Як відбувається порівняння рядків?

Робота № 8

Розробка та реалізація програми з використанням файлового типу даних

Ціль роботи: оволодіння навичками складання програми з використанням файлового типу даних та виконання її у середовищі системи програмування Turbo Pascal 6.0

Завдання:

1. Визначити значення змінних X, Y, Z після виконання фрагмента програми (завдання 8.1).
2. Скласти програму обчислення величин із завдання 4.2 з використанням файлів для введення-виведення даних і виконати її у середовищі системи програмування Turbo Pascal 6.0 (завдання 8.2).

Короткі теоретичні відомості

У мові Pascal термін "файл" стосується об'єктів, утворених послідовністю компонент одного типу, причому вважають, що файл міститься зовні програми. Компоненти файла називаються його елементами. На відміну від типу масив, кількість елементів файла не визначається при описуванні файлового типу.

Об'єкт, який називається файлом, є абстрактною моделлю фізичного набору даних, що існує поза програмою. Фізичний набір даних може міститися в зовнішній чи внутрішній пам'яті комп'ютера, а також бути пов'язаним з потоками вхідних чи вихідних даних, які проходять через зовнішні пристрої ПК. Абстрагування від фізичної природи даних є суттєвою особливістю мови Pascal. Це дає змогу зосередити увагу користувача на алгоритмі, не вдаючись у деталі організації файлів в тій або іншій операційній системі чи в спосіб зображення даних у конкретному комп'ютері. З цієї точки зору робота з файлами зводиться до відкриття файла, виконання операцій введення-виведення і закриття файла.

Опис файлового типу складається з ключового слова `file`, за яким ідуть ключове слово `of` та опис типу елементів файла. Елементами файла можуть бути об'єкти довільного типу, крім файлового. Змінна файлового типу є досить специфічною. Її не можна використовувати у виразах та в операторах присвоєння. Файлова змінна ототожнюється з деяким фізичним файлом і потім у програмі лише визначає файл, над яким виконуються операції. Наприклад, в описі файлового типу `var F_In: file of real;` файлова змінна F_In зображає файл, елементи якого мають тип real.

Послідовність елементів файла встановлює їхній природний порядок, причому в кожний момент безпосередньо доступним є один елемент файла. Для доступу до елемента файла існує абстрактний

вказівник файла. Виконувати певні операції можна лише з тим елементом, на якому встановлено вказівник файла. Інші елементи стають доступними в міру пересування вказівника файла. У кожному файлі є мітка його кінця. Вказівник файла не може бути переміщений за цей кінець. Якщо вказівник файла міститься на мітці кінця файла, то при записуванні у файл деякої величини мітка його кінця разом із вказівником переміщуються далі, а при зчитуванні з файла фіксується помилка введення-виведення. Над одним і тим самим файлом можна одночасно проводити операції зчитування і записування елементів.

Для обробки файлів є стандартні процедури і функції. У подальшому anyfile означає будь-який файловий тип, anystring - будь-який рядковий тип, filetype - тип, що збігається з типом елементів файла. Для обробки файлів є такі стандартні процедури: Assign, Close, Erase, Read, Rename, Reset, Rewrite, Seek, Truncate, Write. Процедури Flush, Read, Seek, Write застосовуються лише до відкритих файлів, а процедури Erase і Rename - до закритих файлів. Порушення цього правила призводить до помилки введення-виведення.

Assign(var F: anyfile; S: anystring) - файлова змінна F ототожнюється з фізичним файлом з іменем S. Будь-яка робота з файлом повинна починатися з цієї процедури.

Close(var F: anyfile) - закриття файла F. При цьому у файл записується мітка кінця файла. Ця процедура необхідна для збереження результатів при створені чи поновленні файлів.

Erase(var F: anyfile) - вилучення файла F.

Read(var F: anyfile; var A1,...,An: filetype) - зчитування елементів з файла F і присвоєння їх значень змінним із списку. Список повинен складатися хоча б з однієї змінної, інакше не виконуватимуться ніякі дії. Зчитування починається з поточного місцезнаходження вказівніка файла, а після закінчення процедури він встановлюється на першому непрочитаному елементі. Якщо тип деякої змінної зі списку не збігається з типом елементів файла, то фіксується помилки введення-виведення.

Rename(var F: anyfile; S: anystring) - перейменування файла F у файл з іменем, заданим рядком S. Нове ім'я не повинно збігатися з іменами вже існуючих файлів.

Reset(var F: anyfile) - відкриває для читання вже існуючий файла F. Вказівник файла становиться на перший елемент файла. Якщо такого файла не існує, то виникає помилка введення-виведення.

Rewrite(var F: anyfile) - утворює новий файла F і відкриває його для запису. Якщо такий файл вже існував, то він очищається. Вказівник файла міститься на початку файла.

Seek(var F: anyfile; N: integer) - встановлює вказівник файла на N-й елемент. Перший елемент має номер 0, другий - 1 і т.д.

Truncate(var F: anyfile) - відсікає від файла його хвостову частину, починаючи з поточної позиції вказівника включно.

Write(var F: anyfile; var A1,...,An: filetype) - запис у файл F значень змінних із списку. Список повинен складатися хоча б з однієї змінної, інакше не виконуватимуться ніякі дії. Після виконання процедури вказівник файла міститься після останнього записаного елемента. Якщо тип деякої змінної зі списку не збігається з типом елементів файла, то фіксується помилки введення-виведення.

Для обробки файлів є такі стандартні функції: Eof, FilePos, FileSize. *Ці функції повинні застосовуватись до відкритих файлів.*

Eof(var F: anyfile): boolean - має значення true, якщо вказівник файла міститься на мітці кінця файла F, і false - у противному разі.

FilePos(var F: anyfile): integer - визначає номер елемента, на якому міститься вказівник файла F.

FileSize(var F: anyfile): integer - визначає кількість елементів у файлі F.

Окреме місце у мові Pascal займають текстові файли. На відміну від інших файлів, текстові файли не складаються з послідовності *однакових елементів. Компонентами їх є рядки символів, що* можуть мати довільну довжину. У кінця кожного рядка записується знак кінця рядка, що складається з пари знаків CR/LF (Carridge Return/Line Feed - повернення каретки/подання рядка) з ASCII-кодами $0D/$0A. У кінці файла записується знак кінця файла, що є символом ^Z з ASCII-кодом $1A. Текстовий файл описується за допомогою стандартного ідентифікатора `text`.

Приклад: `var T: text;`

Оскільки текстові файли не мають регулярної структури, то до них не можна застосовувати стандартні підпрограми FilePos, FileSize і Seek. Забороняється одночасно читати і писати в один і той самий текстовий файл.

Для обробки текстових файлів є такі стандартні процедури: *Append, Assign, Close, Read, Reset, Rewrite, Write, SetTextBuf.* Процедури Assign, Close, Reset і Rewrite мають ту саму дію, як для типізованих файлів. Процедури Read і Write модифіковані.

Read(var F: text; var A1,...,An: anytype) - зчитування значень з файла F і присвоєння їх змінним із списку. Тип anytype може бути символьним, рядковим, цілим або дійсним. Дія і результат процедури Read залежать від типу зчитаних величин.

Для параметра-змінної символьного типу з файла буде зчитано символ і присвоєно відповідній змінній. Якщо до виконання процедури Read функція Eof набула значення True, то символьній змінній буде присвоєно символ Chr($1A) (символ ^Z - кінець файла), а якщо функція Eoln набула значення True (при цьому функція Eof набула значення False), то змінній буде присвоєно значення Chr($0D) (символ CR - повернення каретки).

Для параметра-змінної рядкового типу процедура Read буде читати всі символи з одного рядка файла до символу кінця рядка, але не включаючи його. Якщо довжина рядка перевищує довжину рядка-змінної, то зайві праві символи буде відкинуто.

Для параметра-змінної цілого типу процедура Read буде чикати надходження послідовності символів, які утворюють цілочисловий літерал. Пропуски, символи табуляції чи мітки кінця рядка, яки передують числовій послідовності, пропускаються. Зчитування припиняється при зустрічі першого пропуску, символу табуляції чи мітки кінця рядка, що йдуть за числовою послідовністю, або при зустрічі мітки кінця файла. Якщо числова послідовність не відповідає очікуваному формату цілого числа, то фіксується помилка введення-виведення.

Для параметра-змінної дійсного типу процедура Read буде чикати надходження послідовності символів, що утворюють літерал дійсного типу. Пропуски, символи табуляції чи мітки кінця рядка, що передують числовій послідовності, пропускаються. Зчитування припиняється при зустрічі першого пропуску, символу табуляції чи мітки кінця рядка, що йдуть за числовою послідовністю, або при зустрічі мітки кінця файла. Якщо числова послідовність не відповідає очікуваному формату дійсного числа, то фіксується помилка введення-виведення.

В усіх випадках наступна процедура Read буде починати зчитування з того місця, де закінчилась попередня процедура Read.

Readln(var F: text; var A1,...,An: anytype) - процедура, аналогічна процедурі Read, проте з тією відміною, що наступна процедура Read (Readln) буде починати зчитування з початку нового рядка, а символи, залишені в попередньому рядку, ігноруються. Виклик процедури Readln без списку параметрів-змінних приводить просто до переміщення вказівника файлу на початок наступного рядка. Таким чином оператор Readln(F,A,B) аналогічний оператору: begin Read(F,A); Read(F,B); Readln(F) end

Write(var F: text; var A1,...,An: anytype) - записує у файл F одне або більше значень із списку змінних. Тип anytype може бути символьним, рядковим, логічним, цілим або дійсним. У процедурі Write допускається форматоване виведення, аналогічно форматова-

ному виведенню на екран. При цьому вільні знакомісця записуються у файл як пропуски.

При записі числових величин у текстовий файл слід враховувати такий факт. В результаті роботи, наприклад, програми

```
var A,B: integer; F: text;
begin
 Assign(F,'data6.txt'); Rewrite(F);
 A:=19; B:=96; Write(F,A,B); Close(F)
end.
```

у текстовий файл data6.txt будуть записані значення змінних A і B, причому вміст файла матиме вигляд: 1996. Якщо до цього файла застосувати операцію зчитування даних процедурою Read(F,A,B), то змінній A буде присвоєно значення 1996, а змінна B не набуває ніякого значення. Це відбувається тому, що числовий літерал 19 не відокремлено від літерала 96. При цьому слід користуватися оператором Write(F,A,' ',B).

Writeln(var F: text; var A1,...,An: anytype) - процедура, аналогічна процедурі Write, але на відміну від неї після закінчення запису значень змінних із списку у файл записується мітка кінця рядка і наступний оператор виведення здійснюватиме запис даних з нового рядка. Виклик процедури Writeln без списку параметрів-змінних приводить до запису у файл порожнього рядка.

Слід зазначити, що стандартні файли системою Turbo Pascal інтерпретуються як текстові і до них можна застосовувати процедури Readln і Writeln з тим самим змістом. Так, процедура Writeln(Lst) виводить на друкуючй пристрій порожній рядок.

Append(var F: text) - аналогічна процедурі Rewrite, але на відміну від неї не очищає файл, а тільки встановлює поточний вказівник у його кінець. Процедуру Append зручно вживати, коли необхідно додати нові рядки у кінець вже існуючого файлу.

SetTextBuf(var F: text; var Buf; {Size: word}) - формує буфер (Buf) в Pascal-програмі для обміну з файлом (F) і при необхідності задає його розмір (Size) у байтах (за замовченням 128 байтів). При використанні процедури SetTextBuf файл не повинен бути відкритим. Ця процедура, як правило, застосовується для прискорення обміну з файлом.

Для обробки текстових файлів є такі стандартні функції: Eof, Eoln, SeekEof, SeekEoln. Функція Eof діє так само, як для типізованих файлів.

Eoln(var F: text): boolean - має значення True, якщо вказівник файла міститься на мітці кінця рядка або на мітці кінця файла, і False - у противному разі.

SeekEof(var F: text): boolean - аналогічна функції Eof, але на відміну від неї ігнорує всі пропуски, символи табуляції і кінця рядка.

SeekEoln(var F: text): boolean - аналогічна функції Eoln, але на відміну від неї ігнорує всі пропуски і символи табуляції.

Система Turbo Pascal передбачає операції з файлами, які не мають типу. Ці файли сумісні з усіма типами файлів і призначені, як правило до прямого доступу до файла.

Опис безтипового файла складається з ключового слова `file` (наприклад: `var F:file;`). Файли без типу відкриваються процедурами Reset та Rewrite, але другим параметром повинен бути завданий розмір запису (елемент файла) у байтах. Для того, щоб гарантовано забезпечити повне читання усього файла, необхідно встановити розмір запису рівний 1 байту. До файлів без типу можна застосовувати всі стандартні підпрограми, крім Flush, Read і Write.

Для організації операцій введення-виведення з безтиповими файлами призначені процедури BlockRead і BlockWrite.

BlockRead(var F:file; var A:anytype; N:word; var M:word) - зчитує N записів з безтипового файла F. Результат вміщується в область оперативної пам'яті, яку займає змінна A. Параметр M показує, скільки записів було зчитано. Якщо M<N, то це означає, що в процесі зчитування файл закінчився.

BlockWrite(var F:file; var A:anytype; N:word; var M:word) - записує N записів у безтиповий файл F. Дані, які записуються, беруться з області оперативної пам'яті, яку займає змінна A. Параметр M показує, скільки блоків було записано. Якщо M<N, то це означає, що з деякої причини не всі записи були записані.

При виклику процедур BlockRead і BlockWrite треба, щоб розмір змінної A дорівнював N*RecSize (де RecSize - розмір запису, який *вказаний у процедурах Reset або Rewrite), тобто повинна виконуватись* умова SizeOf(A)≥ N*RecSize.

Приклад. Програма FE_8 копіює, перейменовує або вилучає *файл, який вибирає користувач, використовуючи* при цьому безтипові файли. Копіювання відбувається порціями по 1К. Цю величину можна змінити, надавши нового значення константі RecSize.

```
program FE_8;
uses Crt;
const RecSize=1024;
var A:array [1.. RecSize] of byte;
    M:word; C:char; S,T:string[14]; F,G:file;
begin
```

```
repeat
ClrScr; GotoXY(30,5); Write('Виберіть режим');
GotoXY(27,7); Write('1 - Копіювання файлів');
GotoXY(27,8); Write('2 - Перейменування файлів');
GotoXY(27,9); Write('3 - Вилучення файлів');
GotoXY(27,10); Write('4 - Завершення роботи');
 repeat C := ReadKey until C in ['1','2','3','4'];
 if C<>'4' then begin GotoXY(30,13); Write('Ім''я
                      файла');Readln(S);Assign(F,S);
                      Reset(F,1) end;
 case C of
 '1':begin
      GotoXY(30,16);Write('В який файл копіювати:');
      Read(T); Assign(G,T); Rewrite(G,1);
      repeat
       BlockRead(F,A,SizeOf(A),M); BlockWrite(G,A,M)
      until M<SizeOf(A); Close(F); Close(G)
     end;
 '2':begin
      GotoXY(30,16); Write('Нове ім''я файла: ');
      Readln(T); Rename(F,T)
     end;
 '3':Erase(F);
 '4':Exit
 end; GotoXY(27,15); DelLine; DelLine
until false
end.
```

ЗАВДАННЯ 8.1

Визначити значення змінних X, Y, Z після виконання фрагмента програми, якщо `var F:file of real; X, Y, Z: real; Assign(F,'A:\F1.txt'); Reset(F);` **а файл F1.txt містить такі дані: 10 24 5 12 7.3 6.2 5.7 25 11 3.4 7 15, які подані як рядок**

Д Е @Г Д @ГЫЩЩіГfffffFГfffff6Е НД ОВЫЩЩУ 'Д Р

Вказівка: замість N підставити номер варіанта.

№	Фрагмент програми	№	Фрагмент програми
1-5	`Read(F,X); Seek(F,N); Read(F,Y,Z);`	6-10	`Seek(F,FilePos(F)+11-N); Read(F,X,Y,Z);`
11-15	`Seek(F,FilePos(F)+19-N); Read(F,X,Y,Z);`	16-20	`Read(F,X); Seek(F,20-N); Read(F,Y,Z);`
21-25	`Seek(F,N-19); Read(F,X); Seek(F,9); Read(F,Y,Z);`	26-30	`Read(F,Y,Z); Seek(F,N-24); Read(F,X);`

ЗАВДАННЯ 8.2

Скласти програму обчислення величин із завдання 4.2 з використанням файлів для введення-виведення даних і виконати її у середовищі системи програмування Turbo Pascal 6.0.

Приклад виконання роботи

Завдання 8.1. Визначити значення змінних X, Y, Z після виконання фрагмента програми: `Seek(F,FilePos(F)+2); Read(F,X,Y,Z);` якщо `var F: file of real; X, Y, Z: real; Assign(F,'A:\F1.txt'); Reset(F);` а файл **F1.txt** містить такі дані: **10 24 5 12 7.3 6.2 5.7 25 11 3.4**, які подані наступним рядком

```
Д    Е    @Г    Д    @ГЬЩЩЩіГfffffГГfffff6Е    НД    ОВЬЩЩЩУ
```

Розв'язання:

Після виконання фрагмента змінні X, Y, Z набудуть наступні значення: X=5; Y=12; Z=7,3.

Завдання 8.2. Скласти програму перестановки елементів масиву a={40; 11; 26; 5; 60; 47; 14; 9; 16; 19; 2; 13; 36; 55; 54; 17; 56; 27} в зворотному порядку з використанням файлів для введення-виведення даних і виконати її у середовищі системи програмування Turbo Pascal 6.0

Розв'язання:

1. Постановка задачі

Скласти програму перестановки елементів масиву a={40; 11; 26; 5; 60; 47; 14; 9; 16; 19; 2; 13; 36; 55; 54; 17; 56; 27} в зворотному порядку на мові Turbo Pascal з використанням файлів для *введення-виведення даних*.

2. Алгоритм розв'язання задачі

Алгоритм розв'язання задачі наведений у прикладі розв'язання завдання 4.2 (при оформленні роботи алгоритм необхідно навести наново).

3. Текст програми

```
program LR8;
{програма перестановки елементів масиву A[1..N]}
uses Crt;
const N=18;
var A: array[1..N] of integer; I, J, M, X : integer;
    FIn,FOut: text;
begin
  ClrScr; Assign(FIn,'A:\F1.txt'); Reset(FIn);
  Assign(FOut,'A:\F2.txt'); Rewrite(FOut);
  for I:=1 to N do Read (FIn, A[I]); Readln(FIn);
  for I:=1 to N do Write(A[I]:3); Writeln;
  for I:=1 to N do Write(FOut, A[I]:3); Writeln(FOut);
  M:=N div 2; J:=N;
  for I:=1 to M do
  begin
    X:=A[I]; A[I]:=A[J]; A[J]:=X; J:=J-1
  end;
  Writeln(' Масив А після перестановки');
```

```
  for I:=1 to N do Write(A[I]:3);
  Writeln; Writeln(FOut,' Масив А після перестановки');
  for I:=1 to N do Write(FOut, A[I]:3);
  Writeln(FOut); Close(FIn); Close(FOut)
end.
```

4. Результати роботи програми

Перед виконанням програми у кореневому каталозі диска А необхідно створити файл F1.txt, в якому повинні знаходитися наступні дані: 40 11 26 5 60 47 14 9 16 19 2 13 36 55 54 17 56 27

У результаті роботи програми на екран монітора була виведена наступна інформація:

```
40 11 26  5 60 47 14  9 16 19  2 13 36 55 54 17 56 27
Масив А після перестановки
27 56 17 54 55 36 13  2 19 16  9 14 47 60  5 26 11 40
```

Крім того, у кореневому каталозі диска А був створений файл F2.txt наступного змісту:

```
40 11 26 5 60 47 14 9 16 19 2 13 36 55 54 17 56 27
Масив А після перестановки
27 56 17 54 55 36 13  2 19 16  9 14 47 60  5 26 11 40
```

Контрольні питання

1. Що таке файл, чим він відрізняється від масиву?
2. Які дані можуть бути елементами файлу?
3. Навіщо використовують модель файлу?
4. Як зв'язати файлову змінну з іменем файлу на диску?
5. В чому різниця між операторами Reset та Rewrite?
6. З якою метою використовують процедуру Seek?
7. Як добавити новий елемент у кінець файлу?
8. Чим відрізняються процедури Read і Readln при вживанні текстового файлу?

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Зуев Е.А. Язык программирования Turbo Pascal 6.0, 7.0. - М.: Веста, Радио и связь, 1993. - 384 с.
2. Приходько С.Б. Методичні вказівки до виконання лабораторних робіт з курсу "Обчислювальна техніка та програмування" (Частина 1. "Основи програмування"). - Миколаїв: УДМТУ, 1996. - 55 с.
3. Сердюченко В.Я. Розробка алгоритмів та програмування на мові Turbo Pascal: Навчальний посібник для техн.вузів / Укр. мовою - Х.: ВКП "Парітет" ЛТД, 1995. - 352 с.
4. Turbo Pascal: Алгоритми і програми: Чисельні методи в фізиці та математиці: Навч. посібник / А.Б.Бартків, Я.Т.Гринчишин, А.М.Ломакович, Ю.С.Рамський. - К.: Вища шк., 1992. - 247 с.
5. Turbo Pascal 7.0 - К.: Издательская группа BHV, 1996. - 448 с.

ЗАВДАННЯ 5.2

Скласти програму обчислення наступних величин та виконати її у середовищі системи програмування Turbo Pascal 6.0, якщо елементи масиву A (який описується як `var A: array [1..3, 1..3] of integer;`) визначаються за формулами A[L,M]=B[I] (L, M=1,2,3; I=1, 2,..., 9); B[I+1]=(37*B[I]+3) mod 64 (I=1, 2,..., 8). Значення A[1] дорівнює номеру варіанта за списком групи.

№	Умова
1-3	Найменшого елемента матриці A над головною діагоналлю
4-6	Найбільшого елемента масиву A і його індексів
7-9	Найбільшого значення з сум елементів кожного стовпця матриці A
10-12	Середнього арифметичного додатних елементів масиву A
13-15	Найменшого елемента масиву A і його індексів
16-18	Добутку найбільшого і найменшого елементів масиву A
19-21	Найменшого елемента матриці A зліва від побічної діагоналі
22-24	Середнього арифметичного елементів масиву A.
25-27	Найбільшого елемента матриці A справа від побічної діагоналі
28-30	Середнього геометричного додатних елементів масиву A

Примітка. Завдання 5.2 із додатку 1 за рішенням викладача може бути виконано студентами усіх спеціальностей (окрім спеціальностей напрямку "Комп'ютерні науки") замість відповідного завдання 5.2 (с. 31).

Робота у середовищі системи програмування Turbo Pascal 6.0

Система програмування Turbo Pascal 6.0 має зручне середовище, яке об'єднує у собі текстовий редактор, компілятор, компоновник, відлагоджувач та систему підказки.

Щоб ввійти в середовище, треба виконати файл turbo.exe. На екрані з'явиться головне меню системи

File Edit Search Run Compile Debug Options Window Help
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

У верхньому рядку подані всі режими головного меню, в нижньому - рядок статусу, в якому перелічені імена функціональних клавіш, призначених для виконання деяких операцій, які найбільш часто використовуються.

Для виходу в головне меню і роботи з його пунктами (режимами) передбачається декілька способів:

- натиснути клавішу F10 і пересуватися за пунктам головного меню за допомогою клавіш управління курсором;
- поставити курсор маніпулятора "миша" на пункт "F10 Menu" в рядку статусу і зафіксувати вибір натискуванням його лівої кнопки;
- одночасно натиснути комбінацію клавіш Alt і букви, яка виділена у назві режиму головного меню;
- поставити курсор маніпулятора "миша" на потрібний пункт головного меню і зафіксувати вибір.

Створення нового файлу починається з активізації спочатку режиму File, а потім підрежиму New. На екрані з'явиться пусте вікно текстового редактору з умовною назвою файлу Noname.pas та відповідним порядковим номером (в правому верхньому кутку).

В процесі підготовки тексту використовують режим Edit і команди текстового редактора. В режим Edit входять декілька підрежимів, деякі з яких наведені нижче:

Cut (Shift-Del) - вилучає виділений текст і поміщає його у Clipboard. Для виділення тексту використовують одночасне натискування клавіші Shift і однієї з клавіш управління курсором.

Copy (Ctrl-Ins) - поміщає копію виділеного фрагмента в Clipboard.

Paste (Shift-Ins) - вставляє виділений текст із Clipboard в визначене курсором місце. Текст остається у Clipboard, і його можна вживати необмежену кількість разів до зміни вмісту буфера обміну.

Команди редактору активізують за допомогою однієї або комбінації декількох натиснутих клавіш. Наведемо деякі з них:

Вилучити рядок	Ctrl-Y
Включити/викл. індикацію блоку	Ctrl-K-H
Позначити початок блоку	Ctrl-K-B
Позначити кінець блоку	Ctrl-K-K
Копіювати блок	Ctrl-K-C
Перемістити блок	Ctrl-K-V
Вилучити блок	Ctrl-K-Y

Для зберігання набраного тексту необхідно натиснути клавішу F2 (це відповідає підрежиму Save режиму File). На екрані з'явиться вікно з пропозицією ввести ім'я файлу, куди буде записана підготовлена у редакторі програма. Після введення імені, наприклад, lab, натискують клавішу Enter. За результатами цих дій у поточному каталозі буде записан файл lab.pas.

Після створення файлу виконують його компіляцію (гарячі клавіші підрежиму Run режиму Run - Ctrl-F9). В процесі компіляції виявляється перша синтаксична помилка, на екрані з'явиться текст програми, курсор показує місце помилки, а у верхній частині екрану - коротке повідомлення про помилку, наприклад: ; Expected (чекаємо ;).

Після виправлення помилки знову натискують F2 (щоб записати новий варіант), а потім - Ctrl-F9.

Для з'ясування причин більш складних помилок застосовується відлагоджувач програм (режим Debug).

Для перегляду результатів виконання програми, необхідно закрити вікно редактору (комбінація клавіш Alt-F5). Для повернення у вікно редагування подану комбінацію клавіш необхідно натиснути ще раз.

Коли усі помилки будуть виправлені, то програму після компіляції буде відкомпоновано і виконано. Такий режим називають "виконання програми з пам'яті", розуміючи під цим те, що на диск не записується результат компоновки. Цей режим використовується у тих випадках, якщо нема упевненості у тому, що програма правильна, або якщо вона більше не буде виконуватися.

Якщо програма відлагоджена і буде експлуатуватися, треба створити відповідний exe-файл. Для цього замість Ctrl - F9 налагоджують режим Compile/Destination так, щоб праворуч від слова Destination стояло слово Disk. Якщо там записано Memory, виставляють бар на цей рядок і натискують Enter - з'являється слово Disk. Тепер достатньо натиснути Alt-F9 (режим Compile/Compile), програма буде відкомпільована, в поточний каталог диску запишеться exe-файл (його ім'я співпадає з ім'ям відповідного pas-файлу).

Додаток 3

Алфавіт мови Turbo Pascal 6.0 (7.0)

Алфавіт мови Turbo Pascal складають:

- букви латинського алфавіту: A, B, ..., Z, a, b, ..., z та символ підкреслення _ (ASCII-код 95);
- арабські цифри: 0, 1, ..., 9;
- спеціальні символи: + - * / = < > . , : ; @ ' ( ) [ ] { } # $ ^ ;
- символи-розділителі: символ пропуску (ASCII-код 32) та управляючи символи (мають ASCII-коди від 0 до 31);
- службові (зарезервовані) слова (наведені у додатку 4).

Додаток 4

Службові слова мови Turbo Pascal 6.0 та 7.0

Службові (зарезервовані) слова - це обмежена група слів, сенс яких зафіксовано у мові. Службові слова не можна вживати як ідентифікатори змінних, констант і т.п. Усі 55 службових слів мови Turbo Pascal 6.0 наведені далі.

absolute	and	array	asm	assembler
begin	case	const	constructor	destructor
div	do	downto	else	end
external	file	for	forward	function
goto	if	implementation	in	inline
interface	interrupt	label	mod	nil
not	object	of	or	packed
private	procedure	program	record	repeat
set	shl	shr	string	then
to	type	unit	until	uses
var	virtual	while	with	xor

У мові Turbo Pascal 7.0 з'явилися ще додаткові службові слова: far, inherited, near, public.

ЗМІСТ