МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗБОРІВСЬКИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО ДЕРЖАВНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
імені ІВАНА ПУЛЮЯ

ЕЛЕКТРОННИЙ ПІДРУЧНИК

з предмету

«Основи програмування та алгоритмічні мови»

для спеціальності 5.05010301
“Розробка програмного забезпечення”

Розробила викладач: Оробчук О. Р.

Розглянуто і затверджено на засіданні ЦМК викладачів предметів циклу програмування, інформаційних та комп’ютерних технологій, інформатики, операторів комп’ютерного набору і майстрів виробничого навчання

Протокол № ***1*** від ***1 вересня 2008*** р.

Зборів 2008 р.

ЗМІСТ

1. ОСНОВНІ ПОНЯТТЯ АЛГОРИТМІЧНОЇ МОВИ

СКЛАД МОВИ. Звичайна розмовна мова складається із чотирьох основних елементів: символів, слів, словосполучень і пропозицій. Алгоритмічна мова містить подібні елементи, тільки слова називають елементарними конструкціями, словосполучення - виразами, речення - операторами. Символи, елементарні конструкції, вираження й оператори становлять ієрархічну структуру, оскільки елементарні конструкції утворяться з послідовності символів, вираження - це послідовність елементарних конструкцій і символів, а оператор - послідовність виражень, елементарних конструкцій і символів.

Опис кожного елемента мови задається його СИНТАКСИСОМ і СЕМАНТИКОЮ. Синтаксичні визначення встановлюють правила побудови елементів мови. Семантика визначає зміст і правила використання тих елементів мови, для яких були дані синтаксичні визначення.

СИМВОЛИ мови - це основні неподільні знаки, у термінах яких пишуться всі тексти мовою.

ЕЛЕМЕНТАРНІ КОНСТРУКЦІЇ - це мінімальні одиниці мови, що мають самостійний зміст. Вони утворюються з основних символів мови.

ВИРАЗИ в алгоритмічній мові складаються з елементарних конструкцій і символів, вони задають правила обчислень деяких значень.

ОПЕРАТОР задає повний опис деякої дії, яку необхідно виконати. Для опису складної дії може знадобитися група операторів. У цьому випадку оператори поєднуються в СКЛАДЕНИЙ РПЕРАТОР або БЛОК.

Дії, задані операторами, виконуються над ДАНИМИ.

Об'єднана єдиним алгоритмом сукупність описів і операторів утворює ПРОГРАМУ алгоритмічною мовою.

2. О С Н О В Н І С И М В О Л И

ТУРБО ПАСКАЛЬ включає наступний набір основних символів:

1) 26 латинських рядкових і 26 латинських прописних букв:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

a b c d e f g h i j k l m n o p q r s t u v w x y z

2) символ підкреслення ‘ _ ’

3) 10 цифр:

0 1 2 3 4 5 6 7 8 9

4) знаки операцій:

+ - * / = <> < > <= >= := @

5) обмежувачі:

. , ' ( ) [ ] (. .) { } (* *) .. : ;

6) специфікатори:

^ # $

7) службові (зарезервовані) слова:

ABSOLUTE	EXPORTS	LIBRARY	SET
ASSEMBLER	EXTERNAL	MOD	SHL
AND	FAR	NAME	SHR
ARRAY	FILE	NIL	STRING
ASM	FOR	NEAR	THEN
ASSEMBLER	FORWARD	NOT	TO
BEGIN	FUNCTION	OBJECT	TYPE
CASE	GOTO	OF	UNIT
CONST	IF	OR	UNTIL
CONSTRUCTOR	IMPLEMENTATION	PACKED	USES
DESTRUCTOR	IN	PRIVATE	VAR
DIV	INDEX	PROCEDURE	VIRTUAL

DO	INHERITED	PROGRAM	WHILE
DOWNTO	INLINE	PUBLIC	WITH
ELSE	INTERFACE	RECORD	XOR
END	INTERRUPT	REPEAT	
EXPORT	LABEL	RESIDENT	

Крім перерахованих, у набір основних символів входить пробіл. Пробіли не можна використовувати усередині здвоєних символів і зарезервованих слів.

3. Е Л Е М Е Н Т А Р Н І К О Н С Т Р У К Ц І Ї

Елементарні конструкції мови ПАСКАЛЬ містять у собі імена, числа й рядки.

Імена (*ідентифікатори*) складають елементи мови - константи, мітки, типи, змінні, процедури, функції, модулі, об'єкти. **Ім'я** - це послідовність букв і цифр, що починається з букви. В іменах може використовуватися символ _ підкреслення. Ім'я може містити довільну кількість символів, але значущими є 63 символи.

Не дозволяється в мові ПАСКАЛЬ використовувати як імена службові слова й стандартні імена, якими названі стандартні константи, типи, процедури, функції й файли.

Для поліпшення наочності програми в неї можуть вставлятися пробіли. Принаймні один пробіл потрібно вставити між двома послідовними іменами, числами або службовими й стандартними іменами. Пробіли не можна використовувати усередині імен і чисел.

Приклади імен мови ПАСКАЛЬ:

A b12 r1m SIGMA gamma I80_86

Числа в мові ПАСКАЛЬ звичайно записуються в десятковій системі числення. Вони можуть бути цілими й дійсними. Додатний знак числа може бути опущений. Цілі числа записуються у формі без десяткової крапки.

Дійсні числа записуються у формі з десятковою крапкою або у формі з використанням десяткового порядку, що зображується буквою Е:

28.6 0.65 -0.018 4.0 5E12 -1.72E9 73.1E-16

ПАСКАЛЬ допускає запис цілих чисел і фрагментів дійсних чисел у формі з порядком у шестнадцатиричной системі числення:

$7F $40 $ABC0

Рядки в мові ПАСКАЛЬ - це послідовність символів, записана між апострофами. Якщо в рядку як змістовний символ необхідно вжити сам апостроф, то варто записати два апострофи.

4. КОНЦЕПЦІЯ ТИПУ ДЛЯ ДАНИХ

У математиці прийнято класифікувати змінні відповідно до деяких важливих характеристик. Відбувається строге розмежування між дійсними, комплексними й логічними змінними, між змінними, що представляють окремі значення й багато значень і так далі.

При обробці даних на ЕОМ така класифікація ще більш важлива. В будь-якій алгоритмічній мові кожна константа, змінна, вираз або функція є певного типу.

У мові ПАСКАЛЬ існує правило: тип явно задається в описі змінної або функції, що передує їхньому використанню. Концепція типу мови ПАСКАЛЬ має наступні основні властивості:

- будь-який тип даних визначає багато значень, до якого належить константа, які може приймати змінна або вираз, або повертати операція або функція;

- тип значення, що задається константою, змінною або виразом, можна визначити по їхньому виді або опису;

- кожна операція або функція вимагає аргументів фіксованого типу й видає результат фіксованого типу.

Звідси витікає, що транслятор може використовувати інформацію про типи для перевірки обчислюваності й правильності різних конструкцій.

Обов'язковий опис типу приводить до надмірності в тексті програм, але така надмірність є важливим допоміжним засобом розробки програм і розглядається як необхідна властивість сучасних алгоритмічних мов високого рівня. У мові ПАСКАЛЬ існують скалярні й структуровані типи даних.

До скалярних типів відносять стандартні типи й типи, обумовлені користувачем.

Стандартні типи включають цілі, дійсні, символьний, логічні й адресні типи. Типи, обумовлені користувачем, - перераховуваний й інтервальний.

Структуровані типи мають чотири різновиди: масиви, множини, записи й файли.

Крім перерахованих, TURBO PASCAL включає ще два типи - процедурний і об'єктний.

5. СТАНДАРТНІ ТИПИ ДАНИХ

До стандартного відносять цілі, дійсні, логічні, символьний і адресний типи.

ЦІЛІ типи визначають константи, змінні й функції, значення яких реалізуються безліччю цілих чисел, припустимих у даної ЕОМ.

тип	діапазон значень	необхідна пам'ять
Shortint	-128 .. 127	1 байт
Integer	-32768 .. 32767	2 байти
Longint	-2147483648 .. 2147483647	4 байти
Byte	0 .. 255	1 байт
Word	0 .. 65535	2 байти

До аргументів цілого типу застосовні наступні стандартні функції, результат виконання яких має цілий тип:

Abs(X), Sqr(X), Succ(X), Pred(X),

і які визначають відповідно абсолютне значення X, X у квадраті, X+1, X-1.

Наступна група стандартних функцій для аргументу цілого типу дає дійсний результат:

Sin(X), Cos(X), ArcTan(X), Ln(X), Exp(X), Sqrt(X).

Ці функції обчислюють синус, косинус і арктангенс кута, заданого у радіанах, логарифм натуральний, експоненту й корінь квадратний відповідно.

Результат виконання функції перевірки цілої величини на непарність Odd(X) має значення *істина*, якщо аргумент непарний, і значення *неправда*, якщо аргумент парний:

X=5 Odd(X)=TRUE , X=4 Odd(X)=FALSE.

ДІЙСНІ типи визначає ті дані, які реалізуються підмножиною дійсних чисел, припустимих у даної ЕОМ.

Тип	Діапазон значень	Кількість цифр мантиси	Необхідна пам'ять (байт)
Real	2.9e-39 .. 1.7e+38	11	6
Single	1.5e-45 .. 3.4e+38	7	4
Double	5.0e-324 .. 1.7e+308	15	8
Extended	3.4e-4932 .. 1.1e+4932	19	10
Comp	-9.2e+18 .. 9.2e+18	19	8

Тип Real визначений у стандартному ПАСКАЛІ й математичним співпроцесором не підтримується.

До дійсних аргументів застосовні функції, що дають дійсний результат:

Abs(X), Sqr(X), Sin(X), Cos(X), ArcTan(X), Ln(X), Exp(X), Sqrt(X), Frac(X), Int(X), Pi.

Функція Frac(X) повертає дробову частину X, функція Int(X) – цілу частину X.

Безаргументна функція Pi повертає значення числа Пі дійсного типу.

До аргументів дійсного типу застосовні також функції

Trunc(X) і Round(X),

що дають цілий результат. Перша з них виділяє цілу частину дійсного аргументу шляхом відсікання дробової частини, друга округляє аргумент до найближчого цілого.

ЛОГІЧНИЙ тип (Boolean) визначає ті дані, які можуть приймати логічні значення TRUE і FALSE.

До булевих операндів застосовні наступні логічні операції:

not and or xor.

СИМВОЛЬНИЙ тип (Char) визначає впорядковану сукупність символів, припустимих у даної ЕОМ. Значення символьної змінної або константи - це один символ із припустимого набору.

Символьна константа може записуватися в тексті програми трьома способами:

- як один символ, укладений в апострофи, наприклад:

  'A' 'a' 'Ю' 'ю';

- за допомогою конструкції виду #K, де K - код відповідного символа, при цьому значення K повинне перебувати в межах 0..255;

- за допомогою конструкції виду ^C, де C - код відповідного управляючого символу, при цьому значення C повинне бути на 64 більше коду керуючого символу.

До величин символьного типу застосовні всі операції відношень.

Для величин символьного типу визначені дві функції перетворення

Ord(C) Chr(K).

Перша функція визначає порядковий номер символу C у наборі символів, друга визначає по порядковому номері K символ, що стоїть на K-му місці в наборі символів. Порядковий номер має цілий тип.

До аргументів символьного типу застосовуються функції, які визначають попередній і наступний символи:

Pred(C) Succ(C). Pred('F') = 'E' ; Succ('Y') = 'Z' .

Для літер з інтервалу 'a'..'z' застосовна функція UpCase(C), яка переводить ці літери у верхній регістр 'A'..'Z'.

АДРЕСНИЙ тип (Pointer) визначає змінні, які можуть містити значення адрес даних або фрагментів програми. Для зберігання адреси потрібні два слова (4 байти), одне з них визначає сегмент, друге - зсув.

6. К О Н С Т А Н Т И

Тип констант у мові ПАСКАЛЬ визначається по їхньому виді: константи цілого типу - це цілі числа, що не містять десяткової крапки, константи дійсного типу - дійсні числа, логічні константи - логічні значення TRUE і FALSE.

7. З М І Н Н І. І Н І Ц І А Л І З А Ц І Я З М І Н Н И Х

Тип змінних визначається користувачем у розділі опису змінних.

У даний час у професійному програмуванні прийнято записувати імена змінних з використанням так званої угорської нотації.

Угорська нотація - це угода про найменування змінних і функцій. Угода широко використовується при програмуванні на мовах PASCAL, C і в середовищі WINDOWS.

Угорська нотація ґрунтується на наступних принципах:

- імена змінних і функцій повинні містити префікс, що описує їхній тип;
- імена змінних і функцій записуються повними словами або словосполученнями або їхніми скороченнями, але так, щоб по імені можна було зрозуміти призначення змінної або дію, виконувану функцією.

Префікси записуються малими буквами, перша буква кожного слова - заголовна, префікси й слова записуються або разом, або через символ _ (підкреслення).

У відкомпільованій програмі для всіх змінних відведене місце в пам'яті, і всім змінним привласнені нульові значення.

Для завдання початкових значень змінним (ініціалізації змінних) TURBO PASCAL дозволяє привласнювати початкові значення змінним одночасно з їхнім описом. Для цього використовується конструкція

ім'я змінної: тип = значення;

яка повинна бути розміщена в розділі опису констант, наприклад:

const rWeight: Real = 0.4;

8. В И Р А З И

Вирази складаються з констант, змінних, покажчиків функцій, знаків операцій і дужок. Вираз задає правило обчислення деякого значення. Порядок обчислення визначається старшинством (пріоритетом) операцій, що втримуються в ньому. У мові ПАСКАЛЬ прийнятий наступний пріоритет операцій:

1. унарна операція not, унарний мінус -, узяття адреси @
2. операції типу множення * / div mod and shl shr
3. операції типу додавання + - or xor
4. операції відносини = <> < > <= >= in

Вирази входять до складу багатьох операторів мови ПАСКАЛЬ, а також можуть бути аргументами вбудованих функцій.

9. О П Е Р А Т О Р П Р И С В О Є Н Н Я

Тип змінної й тип виразу повинні збігатися крім випадку, коли вираз відноситься до цілого типу, а змінна - до дійсного. При цьому відбувається перетворення значення виразу до дійсного типу.

10. О П Е Р А Т О Р И В В О Д У Й В И В О Д У

Розглянемо організацію вводу й виводу даних з термінального пристрою. Термінальний пристрій - це пристрій, з яким працює користувач, звичайно це екран (дисплей) і клавіатура.

Для вводу й виводу даних використовуються стандартні процедури вводу й виводу Read і Write, що оперують стандартними послідовними файлами INPUT і OUTPUT.

Ці файли розбиваються на рядки змінної довжини, відокремлювані один від одного ознакою кінця рядка. Кінець рядка задається натисканням клавіші ENTER.

Для вводу вихідних даних використовуються оператори процедур вводу:

```
Read(A1,A2,...AK);
ReadLn(A1,A2,...AK);
ReadLn;
```

При введенні вихідних даних відбувається перетворення із зовнішньої форми подання у внутрішню, обумовлене типом змінних. Змінны, утворюючи список введення, можуть належати або до цілого, або до дійсного, або до символьного типу. Читання вихідних даних логічного типу в мові ПАСКАЛЬ неприпустимо.

Оператори введення при читанні значень змінних цілого й дійсного типу пропускає пробіли, що передують числу. У той же час ці оператори не пропускають пробілів, що передують значенням символьних змінних, тому що пробіли є рівноправними символами рядків.

Значення вихідних даних можуть відділятися один від одного пробілами і натисканням клавіш табуляції й Enter.

Для виводу результатів роботи програми на екран використовуються оператори:

```
Write(A1,A2,...AK);
WriteLn(A1,A2,...AK);
WriteLn;
```

Змінні, що складають список виводу , можуть відноситись до цілого, дійсному, символьному або булевого типів. Як елемент списку виводу крім імен змінних можуть використовуватися вирази і рядки.

Оператор виводу дозволяє задати ширину поля виводу для кожного елемента списку висновку.

11. С Т Р У К Т У Р А П Р О Г Р А М И

Програма мовою ПАСКАЛЬ складається із заголовка, розділів описів і розділу операторів.

Заголовок програми містить ім'я програми, наприклад:

Program PRIM;

Описи можуть містити в собі розділ бібліотек, що підключаються, (модулів), розділ опису міток, розділ опису констант, розділ опису типів, розділ опису змінних, розділ опису процедур і функцій.

Розділ опису модулів визначається службовим словом USES і містить імена модулів, що підключаються, (бібліотек) як вхідних до складу системи TURBO PASCAL, так і написаних користувачем. Розділ опису модулів повинен бути першим серед розділів описів. Імена модулів відділяються один від одного комами:

uses CRT, Graph;

Опис констант дозволяє використовувати імена як синоніми констант, їх необхідно визначити в розділі описів констант:

const K= 1024; MAX= 16384;

У розділі опису змінних необхідно визначити тип всіх змінних, використовуваних у програмі:

```
var P,Q,R: Integer;
    A,B:   Char;
    F1,F2: Boolean;
```

Розділ операторів являє собою складений оператор, що містить між службовими словами ***begin.....end*** послідовність операторів. Оператори відділяються один від одного символом **;**.

Текст програми закінчується символом крапка.

Крім описів і операторів ПАСКАЛЬ - програма може містити коментарі, які являють собою довільну послідовність символів, розташовану між відкриваючою дужкою коментарів { і закриваючою дужкою коментарів }.

Текст ПАСКАЛЬ - програми може містити ключі компіляції, які дозволяють управляти режимом компіляції. Синтаксично ключі компіляції записуються як коментарі. Ключ компіляції містить символ **$** і букву-ключ із наступним знаком + (включити режим) або - (виключити режим). Наприклад:

{$E+} - эмулировать математичний співпроцесор;

{$F+} - формувати далекий тип виклику процедур і функцій;

{$N+} - використовувати математичний співпроцесор;

{$R+} - перевіряти вихід за межі діапазонів.

Приклад запису простої програми:

```
Program TRIANG;
 var A, B, C, S, P: Real;
 begin
  Read(A,B,C);
  WriteLn(A,B,C);
   P:=(A+B+C)/2;
   S:=Sqrt(P*(P-A)*(P-B)*(P-C));
  WriteLn('S=',S:8:3)
 end.
```

12. ЕЛЕМЕНТИ СТРУКТУРНОГО ПРОГРАМУВАННЯ

Структуризована програма (або підпрограма) - це програма, складена з фіксованої безлічі базових конструкцій.

З операцій, розвилок і злиттів будуються базові конструкції: *лінійна програма, розгалуження, цикл*. Застосовуючи тільки ці три конструкції, можна реалізувати алгоритм рішення будь-якого завдання.

Конструкція, що представляє собою послідовне виконання двох або більше операцій, називається **лінійною**.

Конструкція, що складається з розвилки, двох операцій і злиття, називається **розгалуженням**. Одна з операцій може бути відсутня.

Конструкція, що має лінії керування, що ведуть до попередніх операцій або розвилок, називається **циклом**.

Лінійні конструкції, розгалуження й цикл можна представити як операції, тому що вони мають єдиний вхід і єдиний вихід.

Довільну послідовність операцій можна представити як одну операцію.

13. РЯДКИ

Особливе місце в мові ПАСКАЛЬ займають масиви символів. Стандартний ПАСКАЛЬ допускає два способи зберігання символьних масивів у пам'яті ЕОМ: розпакований і запакований. Розпаковані масиви символів зберігаються в пам'яті ЕОМ по одному символі в машинному слові, упаковані - по одному символі в байті. При описі запакованого масиву символів використовують службове слово PACKED, наприклад:

var MAS: Packed Array[1..20] of Char;

Опис розпакованого масиву символів має вигляд:

var M: Array[1..20] of char;

Для перетворення символьного масиву з розпакованої форми в упаковану й навпаки, з упакованої в розпаковану, у мову ПАСКАЛЬ уведені дві стандартні функції Pack, UnPack.

Упакований масив символів утворить символьний рядок. Символьний рядок може бути або строковою константою, або строковою змінною. Строкова константа, або рядок, являє собою сукупність символів, укладену в апострофи. Рядок - це елементарна конструкція мови ПАСКАЛЬ.

Строкові змінні - це одномірні впаковані масиви символів, для опису яких в TURBO PASCAL уведений тип String.

Наприклад, якщо рядок містить до 30 символів, його тип буде визначений як

type s= String[30];

Довжина рядка не може містити більш, ніж 255 символів.

Тип String без вказівки довжини сумісний з усіма типами рядків.

Особливістю строкових змінних є те, що до них можна звертатися як до скалярним змінних, так і до масивів.

У пам'яті ЕОМ рядок займає кількість байтів, на одиницю більше її довжини. Нульовий байт рядка містить його довжину.

Для рядків визначені операції присвоювання, злиття (конкатенації) і порівняння.

Для порівняння рядків застосовуються всі операції відносин. Порівняння рядків відбувається посимвольно, починаючи з першого символу. Рядки рівні, якщо мають однакову довжину й посимвольно еквівалентні.

Рядки можуть бути елементами списку вводу - виводу, при цьому записується ім'я рядка без індексу.

14. П Р О Ц Е Д У Р И І Ф У Н К Ц І Ї

Алгоритм рішення завдання проектується шляхом декомпозиції всього завдання в окремі підзадачі. Звичайно підзадачі реалізуються у вигляді підпрограм.

Підпрограма - це послідовність операторів, які визначені й записані тільки в одному місці програми, однак їх можна викликати для виконання з однієї або декількох точокк програми. Кожна підпрограма визначається унікальним ім'ям. У мові ПАСКАЛЬ існують два типи підпрограм - процедури й функції.

Процедура й функція - це іменована послідовність описів і операторів. При використанні процедур або функцій ПАСКАЛЬ - програма повинна містити текст процедури або функції й звертання до процедури або функції. Тексти процедур і функцій містяться в розділ описів процедур і функцій.

Процедура може містити такі-ж розділи описів, що й ПАСКАЛЬ - програма, а саме: розділи опису модулів, міток, констант, типів, змінних, процедур і функцій.

ПЕРЕДАЧА ІМЕН ПРОЦЕДУР І ФУНКЦІЙ В ЯКОСТІ ПАРАМЕТРІВ

У багатьох завданнях, особливо в завданнях обчислювальної математики, необхідно передавати імена процедур і функцій як параметри. Для цього в TURBO PASCAL введений новий тип даних - процедурний або функціональний, залежно від того, що описується.

Опис процедурних і функціональних типів здіфйснюється в розділі опису типів:

```
type
    FuncType = Function(z: Real): Real;
    ProcType = Procedure (a,b: Real; var x,y: Real);
```

Функціональний і процедурний тип визначається як заголовок процедури й функції зі списком формальних параметрів, але без ім'я. Можна визначити функціональний або процедурний тип без параметрів, наприклад:

```
type
    Proc = Procedure;
```

Після оголошення процедурного або функціонального типу його можна використовувати для опису формальних параметрів - імен процедур і функцій.

Приклад. Скласти програму для обчислення визначеного інтегралу $I = \int_{tm}^{tk} \frac{2t}{\sqrt{1-\sin 2t}} dt$ по методу Сімпсона. Обчислення підінтегральної функції реалізувати за допомогою функції, ім'я якої передається як параметр. Значення визначеного інтеграла по формулі Сімпсона обчислюється по формулі:

```
ISimps=2*h/3*(0.5*F(A)+2*F(A+h)+F(A+2*h)+2*F(A+3*h)+...
                                    +2*F(B-h)+0.5*F(B))
```

де A і B - нижня й верхня границі інтервалу інтегрування,
N - число розбивок інтервалу інтегрування,
h=(B-A)/N, причому N повинне бути парним.

```
Program INTEGRAL;
 type
   Func= function(x: Real): Real;
 var
   I,TN,TK:Real;
   N:Integer;
{$F+}
 Function Q(t: Real): Real;
  begin
    Q:=2*t/Sqrt(1-Sin(2*t));
  end;
{$F-}
 Procedure Simps(F:Func; a,b:Real; N:Integer; var INT:Real);
  var
     sum, h: Real;
     j:Integer;
  begin
    if Odd(N) then N:=N+1;
    h:=(b-a)/N;
    sum:=0.5*(F(a)+F(b));
    for j:=1 to N-1 do
      sum:=sum+(j mod 2+1)*F(a+j*h);
      INT:=2*h*sum/3
  end;
 begin
  WriteLn(' УВЕДИ TN,TK,N');
  Read(TN,TK,N);
  Simps(Q,TN,TK,N,I);
  WriteLn('I=',I:8:3)
 end.
```

15. М Н О Ж И Н И

Поняття множини в мові ПАСКАЛЬ ґрунтується на математичному представленні про множини: це обмежена сукупність різних елементів. Для побудови конкретного множинного типу використовується перераховуваний або інтервальний тип даних. Тип елементів, що становлять множину, називається базовим типом.

Множинний тип описується за допомогою службових слів **Set of**, наприклад:

```
type  M= Set of B;
```

Тут М - множинний тип, В - базовий тип.

Приклад опису змінної множинного типу:

```
type
   M= Set of 'A'..'D';
var
  MS: M;
```

Приналежність змінних до множинного типу може бути визначена безпосередньо в розділі опису змінних:

```
var
  C: Set of 0..7;
```

Проілюструємо застосування даних множинного типу на прикладі.

Приклад. Скласти програму, що вибирає й виводить на екран дисплея набори випадкових чисел для гри в "Спортлото 5 з 36". Для заповнення кожної картки спортлото необхідно одержати набір з п'яти селестичних чисел. До цих чисел пред'являються дві вимоги:

-числа повинні перебувати в діапазоні 1..36;

-числа не повинні повторюватися.

```
Program Lotto;
  var
    nb, k: Set of 1..36;
    kol, l, i, n: Integer;
  begin
    Randomize;
    WriteLn('ВВЕДИ kol');
    ReadLn(kol);
    nb:=[1..36];
    for i:=1 to kol do
     begin
      k:=[];
      for l:=1 to 5 do
        begin
          repeat
            n:=Random(36)
          until (n in nb) and not (n in k);
          k:=k+[n];
          Write(n:4)
        end;
      WriteLn
     end
    end.
```

16. З А П И С И

Запис являє собою сукупність обмеженого числа логічно зв'язаних компонентів, що належать до різних типів. Компоненти запису називаються полями, кожне з яких визначається ім'ям. Поле запису містить ім'я поля, слідом за яким через двокрапку вказується тип цього поля. Поля записів можуть відноситись до будь-якого типу, припустимому в мові Паскаль, за винятком файлового типу.

Опис запису в мові ПАСКАЛЬ здійснюється за допомогою службового слова **RECORD**, слідом за яким описуються компоненти запису. Завершується опис запису службовим словом END.

Наприклад, записна книжка містить прізвища, ініціали й номери телефону, тому окремий рядок у записній книжці зручно представити у вигляді наступного запису:

```
type   Row=Record
         FIO: String[20];
         TEL: String[7]
       end;
var    str: Row;
```

Опис записів можливий й без використання ім'я типу, наприклад:

```
var  str: Record
         FIO: String[20];
         TEL: String[7]
       end;
```

Звертання до запису в цілому допускається тільки в операторах присвоювання, де ліворуч і праворуч від знака присвоювання використовуються імена записів однакового типу. У всіх інших випадках оперують окремими полями записів. Щоб звернутися до окремого компонента

запису, необхідно задати ім'я запису й через крапку вказати ім'я потрібного поля, наприклад:

str.FIO, str.TEL

Таке ім'я називається складеним. Компонентом запису може бути також запис, у такому випадку складене ім'я буде містити не два, а більшу кількість імен.

Звертання до компонентів записів можна спростити, якщо скористатися оператором приєднання with.

Він дозволяє замінити складені імена, що характеризують кожне поле, просто на імена полів, а ім'я запису визначити в операторі приєднання:

with M do OP;

Тут M - ім'я запису, OP - оператор, простий або складений.

Оператор OP являє собою область дії оператора приєднання, у межах якої можна не використовувати складені імена.

17. Ф А Й Л И

Введення файлового типу в мову ПАСКАЛЬ викликано необхідністю забезпечити можливість роботи з периферійними (зовнішніми) пристроями ЕОМ, призначеними для уведення, виводу й зберігання даних.

Файловий тип даних або файл визначає впорядковану сукупність довільного числа однотипних компонентів.

Загальна властивість масиву, множини й записів полягає в тому, що кількість їхніх компонентів визначена на етапі написання програми, тоді як кількість компонент файлу в тексті програми не визначається і може бути довільна.

Поняття файлу досить широке. Це може бути звичайний файл на диску, комунікаційний порт ЕОМ, пристрій друку, клавіатура або інші пристрої.

При роботі з файлами виконуються операції уведення - виводу. Операція уведення означає перепис даних із зовнішнього пристрою (із вхідного файлу) в основну пам'ять ЕОМ, операція виводу - це пересилання даних з основної пам'яті на зовнішній пристрій (у вихідний файл).

Файли на зовнішніх пристроях часто називають фізичними файлами. Їхні імена визначаються операційною системою. У програмах мовою Паскаль імена файлів задаються за допомогою рядків. Наприклад, ім'я файлу на диску може мати вигляд:

'A:LAB1.DAT'

'c:\ABC150\pr.pas'

'lab3.pas'.

Операційна система MS-DOS не робить особливого розходження між файлами на дисках і пристроями ЕОМ і портами комунікацій. В TURBO PASCAL можуть використовуватися імена пристроїв і портів, певні в MS-DOS, наприклад:

'CON', 'LPT1', 'PRN', 'COM1', 'AUX', 'NUL'.

З файловою системою TURBO PASCAL зв'язане поняття буфера вводу - виводу. Уведення й вивід даних здійснюється через буфер. **Буфер** – це область у пам'яті, що виділяється для кожного файлу. При записі в файл вся інформація спочатку направляється в буфер і там накопичується доти, поки весь обсяг буфера не буде заповнений. Тільки після цього або після спеціальної команди скидання відбувається передача даних на зовнішній пристрій. При читанні з файлу дані спочатку зчитуються у буфер, причому даних зчитується не стільки, скільки запитується, а скільки поміститься в буфер.

Механізм буферизації дозволяє більш швидко й ефективно обмінюватися інформацією із зовнішніми пристроями.

Для роботи з файлами в програмі необхідно визначити файлову змінну. TURBO PASCAL підтримує три файлових типи: текстові файли, компонентні файли, безтипові файли.

Опис файлових змінних текстового типу здійснюється з допомогою службового слова Text, наприклад:

var tStory: Text;

Опис компонентних файлів має вигляд:

var fComp: File of T;

де T - тип компонента файлу. Приклади опису файлової змінної компонентного типу:

type M= array[1..500] of Longint;

var f1: File of Real;

f2: File of Integer;

fLi: File of M;

Безтипові файли описуються за допомогою службового слова File:

var f: File;

Файлові змінні, які описані в програмі, називають логічними файлами. Всі основні процедури й функції, що забезпечують уведення - вивід даних, працюють тільки з логічними файлами. Фізичний файл повинен бути пов'язаний з логічним до виконання процедур відкриття файлів.

TURBO PASCAL вводить ряд процедур і функцій, застосовних для будь-яких типів файлів:

Assign, Reset, Rewrite, Close, Rename, Erase, Eof, IOResult.

Процедура **Assign**(var f; FileName: String) зв'язує логічний файл f з фізичним файлом, повне ім'я якого задано в рядку FileName.

Процедура **Reset**(var f) відкриває логічний файл f для наступного читання даних або, як говорять, відкриває вхідний файл. Після успішного виконання процедури Reset файл готовий до читання з нього першого елемента.

Процедура **Rewrite**(var f) відкриває логічний файл f для наступного запису даних (відкриває вихідний файл). Після успішного виконання цієї процедури файл готовий до запису в нього першого елемента.

Процедура **Close**(var f) закриває відкритий до цього логічний файл. Виклик процедури Close необхідний при завершенні роботи з файлом. Якщо з якоїсь причини процедура Close не буде виконана, файл усе-ж буде створений на зовнішньому пристрої, але вміст останнього буфера в нього не буде перенесено. Для вхідних файлів використання оператора закриття файлу необов'язково.

Логічна функція **EOF**(var f): Boolean повертає значення TRUE, коли при читанні досягнуть кінець файлу. Це означає, що вже прочитано останній елемент у файлі або файл після відкриття виявився порожній.

Процедура **Rename**(var f; NewName: String) дозволяє перейменувати фізичний файл на диску, пов'язаний з логічним файлом f. Перейменування можливо після закриття файлу.

Процедура **Erase**(var f) знищує фізичний файл на диску, що був пов'язаний з файловою змінною f. Файл до моменту виклику процедури Erase повинен бути закритий.

Функція **IOResult**: Integer повертає ціле число, що відповідає коду останньої помилки уведення - виводу. При нормальному завершенні операції функція поверне значення 0. Значення функції IOResult необхідно привласнювати якій - небудь змінній, тому що при кожному виклику функція обнуляє своє значення. Функція IOResult працює тільки при виключеному режимі перевірок помилок уведення - виводу або із ключем компіляції {$I-}.

18. Т Е К С Т О В І Ф А Й Л И

Особливе місце в мові ПАСКАЛЬ займають текстові файли, компоненти яких мають символьний тип. Для опису текстових файлів у мові визначено стандартний тип Text:

```
var   TF1, TF2: Text;
```

Текстові файли являють собою послідовність рядків, а рядки - послідовність символів. Рядки мають змінну довжину, кожний рядок завершується ознакою кінця рядка.

З ознакою кінця рядка зв'язана функція **EOLn**(var T:Text):Boolean, де T - ім'я текстового файлу. Ця функція приймає значення TRUE, якщо досягнуть кінець рядка, і значення FALSE, якщо кінець рядка не досягнуть.

Для операцій над текстовими файлами, крім перерахованих, визначені також оператори звертання до процедур:

ReadLn(T) - пропускає рядок до початку наступного;

WriteLn(T) - завершує рядок файлу, у який здійснюється запис, ознакою кінця рядка й переходить до початку наступного.

Для роботи з текстовими файлами уведена розширена форма операторів уведення й виводу. Оператор

Read(T,X1,X2,...XK) еквівалентний групі операторів:

```
begin
  Read(T,X1);
  Read(T,X2);
  .....
  Read(T,XK)
end;
```

Тут T - текстовий файл, а змінні X1, X2,...XK можуть бути або змінними цілого, дійсного або символьного типу, або рядком. При читанні значень змінних з файлу вони перетворяться з текстового подання в машинне.

Оператор

Write(T,X1,X2,...XK) еквівалентний групі операторів:

```
begin
  Write(T,X1);
  Write(T,X2);
  .....
  Write(T,XK)
end;
```

Тут T - також текстовий файл, але змінні X1,X2,...XK можуть бути цілого, дійсного, символьного, логічного типу або рядком. При записі значень змінних у файл вони перетворяться із внутрішнього подання в текстове.

До текстових файлів відносять стандартні файли INPUT, OUTPUT.

Розглянуті раніше оператори уведення - виводу є частковим випадком операторів обміну з текстовими файлами, коли використовуються стандартні файли уведення – виводу INPUT, OUTPUT.

Робота із цими файлами має особливості:

- імена цих файлів у списках уведення – виводу не вказуються;

- застосування процедур Reset, Rewrite і Close до стандартних файлів уведення – виводу заборонено;

- для роботи з файлами INPUT, OUTPUT уведений різновид функції EOLn без параметрів.

TURBO PASCAL уводить додаткові процедури й функції, застосовні тільки до текстових файлів, це *SetTextBuf, Append, Flush, SeekEOLn, SeekEOF*.

Процедура **SetTextBuf**(var f: Text; var Buf; BufSize: Word) служить для збільшення або зменшення буфера уведення – виводу текстового файлу f. Значення розміру буфера для текстових файлів за замовчуванням дорівнює 128 байтам. Збільшення розміру буфера скорочує кількість звертань до диска. Рекомендується змінювати розмір буфера до відкриття файлу. Буфер файлу

почнеться з першого байта змінної Buf. Розмір буфера задається в необов'язковому параметрі BufSize, а якщо цей параметр відсутній, розмір буфера визначається довжиною змінної Buf.

Процедура **Append**(var f: Text) служить для спеціального відкриття вихідних файлів. Вона застосовна до вже існуючих фізичних файлів і відкриває із для дозаписи в кінець файлу.

Процедура **Flush**(var f: Text) застосовується до відкритих вихідних файлів. Вона примусово записує дані з буфера у файл незалежно від ступеня його заповнення.

Функція **SeekEOLn**(var f: Text): Boolean повертає значення True, якщо до кінця рядка залишилися тільки пробіли.

Функція **SeekEOF**(var f: Text): Boolean повертає значення True, якщо до кінця файлу залишилися рядки, заповнені пробілами.

19. КОМПОНЕНТНІ ФАЙЛИ

Компонентний або **типізований** файл - це файл із оголошеним типом його компонентів. Компонентні файли складаються з машинних подань значень змінних, вони зберігають дані в тім же виді, що й пам'ять ЕОМ.

Опис величин файлового типу має вигляд:

```
type M= File Of T;
```

де M - ім'я файлового типу, T - тип компонента. Наприклад:

```
type
  FIO= String[20];
  SPISOK=File of FIO;
var
  STUD, PREP: SPISOK;
```

Тут STUD, PREP - імена файлів, компонентами яких є рядки.

Опис файлів можна задавати в розділі опису змінних:

var

fsimv: File of Char;

fr: File of Real;

Компонентами файлу можуть бути всі скалярні типи, а зі структурованих - масиви, множини , записи. Практично у всіх конкретних реалізаціях мови ПАСКАЛЬ конструкція "файл файлів" неприпустима.

Всі операції над компонентними файлами здійснюються за допомогою стандартних процедур:

Reset, Rewrite, Read, Write, Close.

Для уведення – виводу використовуються процедури:

Read(f,X);

Write(f,X);

де f - ім'я логічного файлу, X - або змінна, або масив, або рядок, або множина , або запис із таким же описом, який має компонента файлу.

Виконання процедури Read(f,X) полягає в читанні із зовнішнього пристрою одного компонента файлу й запис її в X. Повторне застосування процедури Read(f,X) забезпечить читання наступного компонента файлу й запис її в X.

Виконання процедури Write(f,X) полягає в записі X на зовнішній пристрій як однієї компоненти. Повторне застосування цієї процедури забезпечить запис X як наступного компонента файлу.

Для роботи з компонентними файлами уведена розширена форма операторів уведення й висновку:

Read(f,X1,X2,...XK)

Write(f,X1,X2,...XK)

Тут f - компонентний файл, а змінні X1, X2,...XK повинні мати той-же тип, що й оголошений тип компонентів файлу f.

20. Б Е З Т И П О В І Ф А Й Л И

Безтипові файли дозволяють записувати на диск довільні ділянки памяті ЕОМ і зчитувати їх з диска. Операції обміну з безтиповими файлами здійснюються за допомогою процедур **BlokRead** і **BlockWrite**. Крім того, уводиться розширена форма процедур Reset і Rewrite. В іншому принципи роботи залишаються такими ж, як і з компонентними файлами.

Перед використанням логічний файл

var f: File;

повинен бути пов'язаний з фізичним за допомогою процедури Assign. Далі файл повинен бути відкритий для читання або для запису процедурою Reset або Rewrite, а після закінчення роботи закритий процедурою Close.

При відкритті файлу довжина буфера встановлюється за замовчуванням в 128 байт. TURBO PASCAL дозволяє змінити розмір буфера уведення - виводу, для чого варто відкривати файл розширеним записом процедур

Reset(var f: File; BufSize: Word)

або

Rewrite(var f: File; BufSize: Word)

Параметр BufSize задає число байтів, зчитувальних з файлу або записуваних у нього за один обіг. Мінімальне значення BufSize – 1 байт, максимальне - 64 До байт.

Читання даних з безтипового файлу здійснюється процедурою

BlockRead(var f: File; var X; Count: Word; var QuantBlock: Word);

Ця процедура здійснює за один обіг читання в змінну X кількості блоків, задана параметром Count, при цьому довжина блоку дорівнює довжині буфера. Значення Count не може бути менше 1. За одне звертання не можна прочитати більше, ніж 64 До байтів.

Запис даних у безтиповий файл виконується процедурою

BlockWrite(var f: File; var X; Count: Word; var QuantBlock: Word);

яка здійснює за один обіг запис зі змінної X кількості блоків, задана параметром Count, при цьому довжина блоку дорівнює довжині буфера.

Необов'язковий параметр QuantBlock повертає число блоків (буферів), записаних успішно поточною операцією BlockWrite.

21. ПОСЛІДОВНИЙ І ПРЯМИЙ ДОСТУП

Зміст послідовного доступу полягає в тім, що в кожний момент часу доступний лише один компонент із всієї послідовності. Для того, щоб звернутися (одержати доступ) до компонента з номером К, необхідно переглянути від початку файлу К-1 попередню компоненту. Після звертання до компонента з номером К можна звертатися до компонента з номером К+1. Звідси витікає , що процеси формування (запису) компонентів файлу й перегляду (читання) не можуть довільно чергуватися. Таким чином, файл спочатку будується за допомогою послідовного додавання компонент у кінець, а потім може послідовно проглядатися від початку до кінця.

Розглянуті раніше засобу роботи з файлами забезпечують послідовний доступ.

TURBO PASCAL дозволяє застосовувати до компонентного й безтипового файлам, записаним на диск, спосіб прямого доступу. Прямий доступ означає можливість заздалегідь визначити у файлі блок, до якого буде застосована операція уведення - виводу. У випадку безтипових файлів блок дорівнює розміру буфера, для компонентних файлів блок - це один компонент файлу.

Прямий доступ припускає, що файл являє собою лінійну послідовність блоків. Якщо файл містить n блоків, то вони нумеруються від 1 через 1 до n. Крім того, вводиться поняття умовної границі між блоками, при цьому умовна границя з номером 0 розташована перед блоком з номером 1, границя з

номером 1 розташована перед блоком з номером 2 і, нарешті, умовна границя з номером n перебуває після блоку з номером n.

Реалізація прямого доступу здійснюється за допомогою функцій і процедур *FileSize, FilePos, Seek і Truncate*.

Функція **FileSize**(var f): Longint повертає кількість блоків у відкритому файлі f.

Функція **FilePos**(var f): Longint повертає поточну позицію у файлі f. Позиція у файлі - це номер умовної границі. Для тільки що відкритого файлу поточною позицією буде границя з номером 0. Це значить, що можна записати або прочитати блок з номером 1. Після читання або запису першого блоку поточна позиція переміститься на границю з номером 1, і можна буде звертатися до блоку з номером 2. Після прочитання останнього запису значення FilePos дорівнює значенню FileSize.

Процедура **Seek**(var f; N: Longint) забезпечує призначення поточної позиції у файлі (присохлий). У параметрі N повинен бути заданий номер умовної границі, що передує блоку, до якого буде вироблятися наступний обіг. Наприклад, щоб працювати із блоком 4, необхідно задати значення N, рівне 3. Процедура Seek працює з відкритими файлами.

Процедура **Truncate**(var f) установлює в поточній позиції ознаку кінця файлу й видаляє (стирає) всі наступні блоки.

Приклад.

Нехай на диску є текстовий файл ID.DAT, що містить числові значення дійсного типу по двох числа в кожної рядку - значення аргументу й функції відповідно. Кількість пар чисел не більше 200. Скласти програму, що читає файл, значення аргументу й функції записує в одномірні масиви, підраховує їхня кількість, виводить на екран дисплея й записує у файл компонентного типу RD.DAT.

```
Program F;
 var
  rArg, rF: Array[1..200] of Real;
    inf: Text;
    outf: File of Real;
    n, l: Integer;
```

```
begin
  Assign(inf,'ID.DAT');
  Assign(outf,'RD.DAT');
  Reset(inf);
  Rewrite(outf);
  n:=0;
  while not EOF(inf) do
    begin
      n:=n+1;
      ReadLn(inf,rArg[n],rF[n])
    end;
  for l:=1 to n do
   begin
    WriteLn(l:2,rArg[l]:8:2,rF[l]:8:2);
    Write(outf,rArg[l], rF[l]);
   end;
  close(outf)
end.
```

22. В К А З І В Н И К И

Операційна система MS - DOS весь адресованний простір ділить на сегменти. **Сегмент** - це ділянка пам'яті розміром 64 До байт. Для задання адреси необхідно визначити адресу початку сегмента й зсув відносно початку сегмента.

В TURBO PASCAL визначений адресний тип Pointer - покажчик. Змінні типу Pointer

var p: Pointer;

містять адресу якого - небудь елемента програми й займають 4 байти, при цьому адреса зберігається як два слова, одне з них визначає сегмент, друге - зсув.

Змінну типу покажчик можна описати іншим способом.

type NameType= ^T;

var p: NameType;

Тут p - змінна типу покажчик, пов'язана з типом T с допомогою імені типу NameType. Описати змінну типу покажчик можна безпосередньо в розділі опису змінних:

var p: ^T;

Необхідно розрізняти змінну типу покажчик і змінну, на яку цей покажчик посилається. Наприклад якщо p - посилання на змінну типу T, то p^ - позначення цієї самої змінної.

Для змінних типу покажчик уведений стандартне значення NIL, що означає, що покажчик не посилається ні до якого об'єкта. Константа NIL використовується для будь-яких покажчиків.

Над покажчиками не визначено ніяких операцій, крім перевірки на рівність і нерівність.

Змінні типу покажчик можуть бути записані в лівій частині оператора присвоювання, при цьому в правій частині може перебувати або функція визначення адреси Addr(X), або вираження @ X, де @ - унарная операція узяття адреси, X - ім'я змінної будь-якого типу, у тому числі процедурного.

Змінні типу покажчик не можуть бути елементами списку уведення - виводу.

23. Д И Н А М І Ч Н І З М І Н Н І

Статичною змінною (статично розміщеною) називається описана явно в програмі змінна, звертання до неї здійснюється по імені. Місце в пам'яті для розміщення статичних змінних визначається при компіляції програми.

На відміну від таких статичних змінних у програмах, написаних мовою ПАСКАЛЬ, можуть бути створені динамічні змінні. Основна властивість динамічних змінних полягає в тім, що вони створюються й пам'ять для них виділяється під час виконання програми.

Розміщаються динамічні змінні в динамічній області пам'яті (heap - області).

Динамічна змінна не вказується явно в описах змінних і до неї не можна звернутися по імені. Доступ до таких змінних здійснюється за допомогою покажчиків і посилань.

Робота з динамічною областю пам'яті в TURBO PASCAL реалізується за допомогою процедур і функцій *New, Dispose, GetMem, FreeMem, Mark, Release, MaxAvail, MemAvail, SizeOf.*

Процедура **New**(var p: Pointer) виділяє місце в динамічній області пам'яті для розміщення динамічної змінної p^ і її адреса привласнюється покажчику p.

Процедура **Dispose**(var p: Pointer) звільняє ділянку пам'яті, виділену для розміщення динамічною змінною процедурою New, і значення покажчика p стає невизначеним.

Проуедура **GetMem**(var p: Pointer; size: Word) виділяє ділянку пам'яті в heap - області, привласнює адресу початку покажчику p, розмір ділянки в байтах задається параметром size.

Процедура **FreeMem**(var p: Pointer; size: Word) звільняє ділянку пам'яті, адреса початку якої визначений покажчиком p, а розмір - параметром size. Значення покажчика p стає невизначеним.

Процедура **Mark**(var p: Pointer) записує в покажчик p адреса початку ділянки вільної динамічної пам'яті на момент її виклику.

Процедура **Release**(var p: Pointer) звільняє ділянку динамічної пам'яті, починаючи з адреси, записаної в покажчик p процедурою Mark, тобто, очищає ту динамічну пам'ять, що була зайнята після виклику процедури Mark.

Функція **MaxAvail**: Longint повертає довжину в байтах самої довгої вільної ділянки динамічної пам'яті.

Функція **MemAvail**: Longint повний обсяг вільної динамічної пам'яті в байтах.

Допоміжна функція **SizeOf**(X): Word повертає обсяг у байтах, займаний X, причому X може бути або ім'ям змінної будь-якого типу, або ім'ям типу.

24. ДИНАМІЧНІ СТРУКТУРИ ДАНИХ. СТЕКИ

Структуровані типи даних, такі, як масиви, множини, записи, являють собою статичні структури, тому що їхні розміри незмінні протягом усього часу виконання програми.

Часто потрібно, щоб структури даних міняли свої розміри в ході рішення завдання. Такі структури даних називаються динамічними, до них відносять стеки, черги, списки, дерева й інші. Опис динамічних структур за допомогою масивів, записів і файлів приводить до неощадливого використання пам'яті ЕОМ і збільшує час рішення завдань.

Кожний компонент будь-якої динамічної структури являє собою запис, що містить принаймні два поля: одне поле типу покажчик, а друге - для розміщення даних. У загальному випадку запис може містити не один, а кілька вказівників і кілька полів даних.

Поле даних може бути змінною, масивом, множиною або записом.

Стеком називається динамічна структура даних, додавання компонента в яку й виключення компонента з якої здійснюється з одного кінця, називаного вершиною стека. Стек працює за принципом

LIFO (Last-In, First-Out) -

поступивший останнім, обслуговується першим.

Звичайно над стеками виконується три операції:

- початкове формування стека (запис першого компонента);
- додавання компонента в стек;

- вибірка компонента (видалення).

Для формування стека й роботи з ним необхідно мати дві змінні типу покажчик, перша з яких визначає вершину стека, а друга - допоміжна. Нехай опис цих змінних має вигляд:

var pTop, pAux: Pointer;

де pTop - покажчик вершини стека;

pAux - допоміжний покажчик.

Приклад. Скласти програму, що формує стек, додає в нього довільну кількість компонентів, а потім читає всі компоненти і виводить їх на екран дисплея. Як дані взяти рядок символів. Уведення даних - із клавіатури дисплея, ознака кінця уведення – рядок символів END.

```
Program STACK;
 uses Crt;
 type
  Alfa= String[10];
  PComp= ^Comp;
  Comp= Record
         sD: Alfa;
         pNext: PComp
        end;
 var
  pTop: PComp;
  sC: Alfa;
 Procedure CreateStack(var pTop: PComp; var sC: Alfa);
  begin
   New(pTop);
   pTop^.pNext:=NIL;
   pTop^.sD:=sC
  end;
 Procedure AddComp(var pTop: PComp; var sC: Alfa);
  var pAux: PComp;
  begin
   NEW(pAux);
   pAux^.pNext:=pTop;
   pTop:=pAux;
   pTop^.sD:=sC
  end;
```

```
Procedure DelComp(var pTop: PComp; var sC:ALFA);
 begin
  sC:=pTop^.sD;
  pTop:=pTop^.pNext
 end;
begin
 Clrscr;
 writeln('  ВВЕДИ РЯДОК ');
 readln(sC);
 CreateStack(pTop,sC);
 repeat
  writeln('  ВВЕДИ РЯДОК ');
  readln(sC);
  AddComp(pTop,sC)
 until sC='END';
 writeln('****** ВИВІД РЕЗУЛЬТАТІВ ******');
 repeat
  DelComp(pTop,sC);
  writeln(sC);
 until pTop = NIL
end.
```