# Тема. Програмування масивів.

## 1.     Поняття масиву.

До цих під для опрацювання даних використовувались скалярні типи. Однак при обробці великих наборів даних використання скалярних величин стає громіздким. Тому для вирішення таких завдань використовуються структуровані величини. Одним зі структурованих типів є регулярний тип даних, або масив.

*Масив -* це впорядкований набір однотипових елементів. Впорядкованість означає, що елементи масиву розташовані в певному порядку, тобто, проіндексовані. Одно типовість означає належність елементів масиву одному типу даних. Це дозволяє використовувати опрацювання масивів за допомогою операторів циклу. Розглянемо опрацювання масивів на прикладі одновимірного масиву.

Масив визначається наступним чином:

ім’я типу **= array**[тип індексу] **of** тип компонентів

Визначення масиву починається зарезервованим словом **array** (масив), за яким слідує тип індексу, укладений у квадратні дужки. Після типу індексу іде зарезервоване слово **of**, за яким указується тип компонентів масиву. В ролі типу індексу може виступати будь-який скалярний обмежений тип, окрім real та integer. Разом з тим типом індексу може виступати діапазон типу integer. В ролі типу компонентів може виступати довільний тип, крім файлового. Розглянемо спочатку одновимірні масиви, тобто масиви, в яких типом компонентів є скалярний тип даних.

ПРИКЛАД:

type

massiv = array[1..10] of integer;

period = array[14..20] of real;

letters = array[char] of boolean;

Змінні структурованого типу вводяться звичайним шляхом в розділі опису змінних:

var

a:massiv;

t:period;

k:letters;

При цьому існує також можливість використання структурованих змінних безіменних типів:

var

b : array[1..10] of integer;

f : array[14..20] of real;

s : array[char] of boolean;

Однак при цьому змінні **a** і **b** несумісні (тобто мають різний тип), хоча фактично являють собою однакові масиви з однаковим типом індексу та типом компонентів. Тому відповідний тип доцільно оголосити у розділі опису типів програми і не використовувати змінні безіменних типів. Тоді дані такого типу простіше опрацьовувати.

Для масивів, як єдиного цілого, існує лише одна операція – операція присвоювання. В усіх інших випадках масиви опрацьовуються поелементно. Як правило, для цього використовуються цикли. Це можливо завдяки впорядкованості та однотиповості його елементів. В більшості випадків це цикл з параметром.

Для опрацювання елементу масиву необхідно вказати ім’я масиву та індекс його елементу. Так, запис a[1] означає, що опрацьовується перший елемент масиву а; a[k] – опрацьовується  k-тий елемент масиву а; a[2\*k-1] – опрацьовується той елемент, індекс якого дорівнює виразу 2\*k-1.

ПРИКЛАД:

const n=10;

type

massiv = array[1..n] of integer;

var

a,b,massiv;

i:integer;

begin

…

 a:=b; {правильний оператор}

 read(a);  {неправильний оператор -

 недопустима операція }

for i:=1 to n do read(a[i]); {правильний оператор}

 write(b); { неправильний оператор -

 недопустима операція }

for i:=1 to n do write(a[i]); {правильний оператор}

…

end.

При обробці масивів його елементам потрібно спочатку надати значення, наприклад, ввести їх з клавіатури ( в противному випадку він буде заповнений нулями).

Як уже відмічалось, базовим типом масиву може бути будь-який тип даних, у тому числі і масив. Структура, елементами якої є масиви, називається багатовимірним масивом. Такі масиви об’являються аналогічно одновимірним. Розглянемо порядок визначення та опрацювання багатовимірних масивів на прикладі двовимірних числових масивів.

ПРИКЛАД:

type

massiv = array[1..n] of integer;

matr = array[1..m, 1..n] of integer;

tabl = array[1..m] of massiv;

Помітимо, що типи matr і tabl – це різні типи, хоча вони визначають практично однакові масиви. В більшості випадків використовується опис масиву аналогічний типу matr, хоча в деяких ситуаціях більш доцільним є опис типу tabl.

Як і для випадку одновимірних масивів, для багатовимірних масивів, як єдиного цілого, використовується лише операція присвоювання. Для решти операцій необхідно опрацьовувати масиви покомпонентно, причому тип компонентів може бути лише скалярним. Для опрацювання компонентів масиву необхідно вказати ім’я масиву та індекси цих компонентів.

ПРИКЛАД:

var

  c, d :matr;

  i,j;integer;

begin

  …

  c:=d; { правильний оператор }

  c[1]:=d[2]; { правильний оператор }

  d[1,4]:=10; { правильний оператор }

  read(c); { неправильний оператор - недопустима операція }

  read(d[1]); {неправильний оператор - недопустима операція }

  for i:=1 to n do

  for j:=1 to n do read(d[i,j]); { правильний оператор }

  …

end.

Аналогічно задаються та опрацьовуються масиви інших розмірностей та з іншим типом індексів або компонентів.

## 2.     Основні алгоритми опрацювання масивів.

При опрацюванні масивів важливими є операції вводу значень масиву та виводу їх на екран. . Розглянемо виконання цих операцій. При цьому вважатимемо, що діє такий опис:

const n=...;

type massiv=array[1..n] of integer;

var

  a:massiv;

  i:integer;

2.1.  Введення елементів масиву з клавіатури на екрані.

for i:=1 to n do

begin

  write(‘Введіть,i,’-й елемент масиву:’);

  read(a[i]);

end;

2.2.  Заповнення елементів масиву випадковими цілими числами. Числа беруться з діапазону [0, 99].

randomize;

for i:=1 to n do a[i]:=random(100);

2.3. Форматний друк елементів масиву .

for i:=1 to n do write(a[i]:4);

writeln

Далі розглянемо типові алгоритми опрацювання масивів.

2.4. Пошук найменшого елемента масиву

Program pr1;

Uses crt;

const n=20;

type massiv=array[1..n] of integer;

var

  a:massiv;

  i:integer;

  min:integer;

begin

  clrscr;

  randomize;

  for i:=1 to n do a[i]:=random(100);

  writeln(‘Вихідні дані’);

  for i:=1 to n do write(a[i]:4);

  writeln;

  min:=a[1];

  for i:=2 to n do if a[i]<min then min:=a[i];

  writeln(‘Результат’);

  writeln(‘Найменший елемент min=’, min:4);

end.

2.5. Обчислення суми елементів масиву

Program pr2;

Uses сrt;

const n=20;

type massiv=array[1..n] of integer;

var

  a:massiv;

  i:integer;

  s:integer;

begin

  clrscr;

  randomize;

  for i:=1 to n do a[i]:=random(100);

  writeln(‘Вихідні дані’);

  for i:=1 to n do write(a[i]:4);

  writeln;

  s:=0;

  for i:=1 to n do s:=s+a[i];

  writeln(‘Результат’);

  writeln(‘Сума s=’, s:4);

end.

2.6.  Обчислення добутку елементів масиву з парними номерами.

Program pr3;

Uses crt;

const n=20;

type massiv=array[1..n] of integer;

var

  a:massiv;

  i:integer;

  p:longint;

begin

  clrscr;

  randomize;

  for i:=1 to n do a[i]:=random(10)+1;

  writeln(‘Вихідні дані’);

  for i:=1 to n do write(a[i]:4);

  writeln;

  p:=1;

  i:=2;

  repeat

  p:=p\*a[i];

  i:=i+2;

  until i>n

  writeln(‘Результат’);

  writeln(‘Добуток елементів з парними номерами p=’, p:4)

end.

# Узагальнення по темі.

*Масив -* це впорядкований набір однотипових елементів. Впорядкованість означає, що елементи масиву розташовані в певному порядку, тобто, проіндексовані. Одно типовість означає належність елементів масиву одному типу даних. Це дозволяє використовувати опрацювання масивів за допомогою операторів циклу.

Для визначення масиву потрібно задати тип індексу та тип елементів. . В ролі типу індексу може виступати будь-який скалярний обмежений тип, окрім real та integer. Разом з тим типом індексу може виступати діапазон типу integer. В ролі типу компонентів може виступати довільний тип, крім файлового.

Для масивів, як єдиного цілого, існує лише одна операція – операція присвоювання. В усіх інших випадках масиви опрацьовуються поелементно. В більшості випадків опрацювання здійснюється за допомогою оператору циклу з параметром.

Як уже відмічалось, базовим типом масиву може бути будь-який тип даних, у тому числі і масив. Структура, елементами якої є масиви, називається багатовимірним масивом. Такі масиви об’являються аналогічно одновимірним. Розглянемо порядок визначення та опрацювання багатовимірних масивів на прикладі двовимірних числових масивів.