Одновимірні та багатовимірні масиви.

1. Що таке масив? Визначення масиву

Масив – набір змінних однакового типу. Доступ до цих змінних здійснюється з допомогою одного й того ж імені. Це ім'я називається іменем масиву. Масиви використовуються для групування зв'язаних змінних між собою.

2. Визначення одновимірних та багатовимірних масивів.

Масиви можуть бути одновимірними та багатовимірними. В одновимірних масивах для доступу до елементу масиву використовується один індекс. В багатовимірних масивах для доступу до елементу масиву використовується декілька індексів.

3. Опис одновимірного масиву. Приклади опису одновимірних масивів

Загальна форма опису одновимірного масиву:

тип **ім'я_масиву[*розмір*];**
У вищенаведеному описі:

- *тип* – це тип елементів масиву. Він ще називається базовим типом. Базовий тип визначає кількість даних кожного елементу, що складає масив. Тип елементів масиву може бути як **базовим типом** так і складеним (структура). Детально про базові типи даних C++ описується **тут**.
- *розмір* – кількість елементів в масиві;
- *ім'я_масиву* – безпосередньо ім'я масиву, за яким здійснюється доступ до елементів масиву.

Після опису масиву, значення елементів масиву може бути нульовим або невизначеним.

Приклад 1. Опис масиву з 10 цілих чисел (тип int) з іменем ***A***.
int A[10];
У результаті, в пам'яті комп'ютера виділяється 10 комірок типу int. Якщо одна комірка займає 2 байти, то всього буде виділено 20 байт пам'яті. Номер першої комірки починається з нуля. Ці комірки об'єднані спільним іменем A.

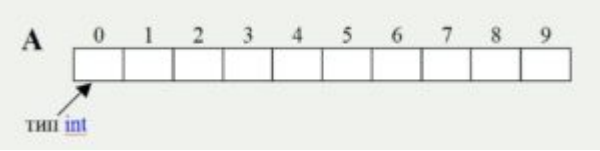

Рисунок 1. Масив з 10 цілих чисел

Приклад 2. Опис масиву з іменем M, що містить 20 елементів типу char.

```
char M[20];
```

4. Доступ до елементів одновимірного масиву. Приклади

Доступ до окремого елементу масиву здійснюється з допомогою індексу. Індекс визначає позицію елементу в масиві. Перший елемент масиву має нульовий індекс.

Щоб доступитись до окремого елементу масиву за його індексом, потрібно після імені масиву в квадратних дужках вказати номер цього елементу.

Приклад 1. Дано масив з іменем A з 10 цілих чисел. Записати число 5 в перший та останній елементи масиву.

```
// опис масиву A
int A[10];

A[0] = 5; // перший елемент масиву
A[9] = 5; // останній елемент масиву
```

На рисунку 2 видно результат роботи вищенаведеного програмного коду.

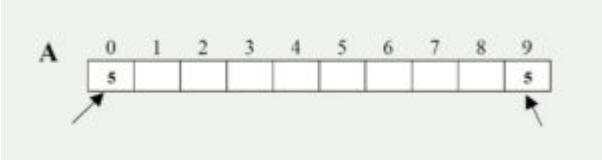

Рисунок 2. Результат роботи фрагменту коду

Приклад 2. Масив з 10 елементів типу char.

```
char M[10];

M[3] = 'a';
M[7] = '0';
M[8] = ';';
```

Рисунок 3. Масив з 10 елементів типу char

Приклад 3. Масив з 30 дійсних чисел.

```
double M[30]; // масив з 30 дійсних чисел

// занулення масиву M
for (int i=0; i<30; i++)
    M[i] = 0.0;
```

5. Як визначити розмір одновимірного масиву?

Розмір одновимірного масиву визначається за формулою:

розмір масиву = розмір типу в байтах × кількість елементів
Приклад.
Якщо в масиві 20 елементів типу double (8 байт), то розмір масиву буде розмір = 20 × 8 = 160 байт

6. Особливості обробки масивів компілятором в C++. Межі масиву

В C++ не ведеться строгого контролю за доступом до елементів за межами масиву. Якщо описати масив з 100 елементів, то можна зчитати або змінити 101-й, 102-й і наступні елементи. На цих позиціях можуть бути комірки пам'яті, які були виділені для інших змінних або навіть для вашої програми. Це може призвести до знищення програми при відсутності яких-небудь зауважень з боку компілятора C++.

Вся відповідальніть за дотримання меж масивів лежить строго на програмісті. Програміст повинен писати такий код, який гарантує коректну роботу з масивами. Це реалізується з допомогою включення у програму спеціальних перевірок.

7. Як здійснюється ініціалізація масиву в C++. Приклади

У C++ підтримується два види ініціалізації масивів:

- ініціалізація з розміром;
- "безрозмірна" ініціалізація.

Загальний вигляд ініціалізації з заданням розміру масиву:

***тип* ім'я_масиву[*розмір*] = { список_значень };**
де

-
- *тип* – тип елементів масиву;
- *розмір* – кількість елементів масиву вказаного типу;
- *список_значень* – список значень ініціалізації елементів масиву. Елементи масиву розділяються символом ',' (кома).

Загальний вигляд "безрозмірної" ініціалізації:

тип ім'я_масиву[] = { список_значень };

У цьому випадку розмір масиву визначається кількістю елементів, що описані в список_значень.

Приклад 1. Масив B ініціалізований з задаванням розміру.

```
// ініціалізація масиву B
int B[10] = { 5, 6, 9, -8, 3, 2, 4, -90, -103, 0 };
```

Приклад 2. Масив C ініціалізований на основі списку значень (“безрозмірна” ініціалізація).

```
// ініцілізація масиву C без задавання розміру
float C[] = { -3.9, 2.8, -1.6, 2.2 };
```

8. Ініціалізація символьних масивів. Приклад

Для символьних масивів можна використовувати скорочений варіант ініціалізації:

char ім’я_масиву[*розмір*] = "рядок";

У цьому випадку кожному елементу масиву присвоюється один символ рядка. Значення *розмір* задає розмір пам’яті, що виділяється для масиву. Значення розмір повинно бути не менше довжини рядка, в іншому випадку компілятор видасть повідомлення про помилку.

Приклад. Ініціалізація символьного масиву з іменем str.

```
// символьний масив - скорочений варіант ініціалізації
char str[] = "Hello!";

// інший варіант ініціалізації символьного масиву
char str2[] = { 'H', 'e', 'l', 'l', 'o', '!' };
```

У вищенаведеному прикладі масиви str та str2 містять однакові значення

9. Присвоєння одного масиву іншому. Приклад

У мові програмування C++ (на відміну від інших мов) не можна присвоювати безпосередньо один масив іншому. Присвоювати можна тільки поелементно з використанням оператору циклу. При цьому обидва масиви повинні мати однаковий тип елементів.

Нехай задано два масиви цілих чисел. Фрагмент коду, що присвоює один масив іншому:

```
// опис масивів A та B
int A[10], B[10];
int i;

//A = B; // помилка!

for (i=0; i<10; i++)
   B[i] = 0;
```

```
for (i=0; i<10; i++)
   A[i] = B[i];
```

10. Опис масиву структур. Приклад

Нехай дано опис структури, що містить інформацію про книгу:

```
struct BOOK
{
   char title[50];
   char author[50];
   int year;
   float price;
};
```

Цей опис визначає новий тип struct BOOK. При такому описі пам'ять не виділяється. Це тільки інформація про новий тип даних. Структура повинна бути описана за межами визначення будь-якого класу на початку визначення простору імен.
В програмі масив з 5 книг (структур) можна описати так:

```
struct BOOK Books[5];
```

Доступ до елементів структури в програмі:

```
strcpy(Books[0].title, "Title-1");
strcpy(Books[0].author, "Author-1");
Books[0].year = 1970;
Books[0].price = 28.85;
```

У вищенаведеному коді використано функцію роботи з рядками

```
strcpy(str1, str2);
```

Ця функція копіює рядок str2 в рядок str1. Щоб використовувати цю та інші функції роботи з рядками на початку програми (перед описом простору імен) потрібно вказати рядок

```
#include <cstring>
```

або

```
#include <string.h>
```