

Лабораторна робота №8

Тема: Масиви

Мета: Оволодіння практичними навичками роботи з масивами.

Теоретичні відомості

Поняття масиву. Опис та ініціалізація масиву

Масив - це сукупність елементів одного типу, звернення до яких здійснюється за допомогою імені масиву та індексу (тобто порядкового номера елемента).

Елементи масиву пронумеровані. Завдяки нумерації можна звернутися до будь-якого елемента масиву як до простого значення базового типу. У C++ нумерація елементів починається з нуля.

i	0	1	2	3	4
A[i]	-4	3	2	7	-39

У таблиці зображений масив цілих чисел, названий **A**. Цей масив містить 5 елементів. Таким чином, перший елемент масиву **A** указують як **A[0]**, другий елемент — як **A[1]**, п'ятий - як **A[4]** тощо. Тим самим - імена масивів повинні задовольняти тим самим вимогам, які ставляться до інших імен змінних. Отже, зображений масив міг бути заповнений у такий спосіб: $A[0] = -4; A[1] = 3; \dots A[4] = -39$

Номер позиції, зазначений всередині квадратних дужок, називається індексом. Індекс повинен бути цілим додатним числом або математичним виразом, результатом обчислення якого є ціле додатне число. Якщо в якості індексу записаний математичний вираз, то вираз обчислюється з метою визначення індексу.

Приклад 1. Якщо змінна **p** дорівнює 3, а змінна **t** дорівнює 1, то оператор $A[p + t] += 7$ додає 7 до елемента масиву **A[4]**.

Приклад 2. Надрукувати суму перших трьох елементів масиву **A** можна в такий спосіб:

```
cout << A[0] + A[1] + A[ 2];
```

Приклад 3. Щоб поділити значення останнього елемента масиву **A** на 3 і присвоїти результат змінній **y**, необхідно написати:

```
y = A[4] / 3;
```

Кожен масив займає певну область у пам'яті комп'ютера. Програміст повинен вказати тип елемента, кількість елементів, необхідних для кожного масиву, тоді компілятор зможе зарезервувати відповідний обсяг пам'яті. Наприклад, щоб компілятор зарезервував пам'ять для 5 елементів масиву цілих чисел **A**, можна використати таке оголошення:

```
int A [5];
```

Пам'ять для декількох масивів може бути зарезервована за допомогою одного оголошення. Наступне оголошення резервує пам'ять для 200 елементів масиву цілих чисел **b** й 15 елементів масиву цілих чисел **y**:

```
int b[200], y[15];
```

Масиви можуть складатися з елементів не лише типу **int**, але й інших типів даних. Але всі елементи одного масиву повинні бути одного типу! Наприклад, для зберігання рядка символів можна використати масив типу **char**:

```
char st [50] ;
```

Приклад 1. Використаємо оператори циклу *for* для присвоєння початкових нульових значень елементам масиву **B**, що містить 8 цілих чисел, і для його друкування:

Контроль за виходом за межі масиву C++ не здійснює, тому треба бути дуже уважним при використанні масивів!

```
#include<iostream.h> //Програма 5.1
#include<conio.h>
int main()
{
int B[8];
for (int i = 0; i < 8; i++) //Присвоєння
    B[i] = 0; // початкових значень
cout<<"Nomer"<<'\\t'<<"Znachennja"<<'\\n';
for(int i = 0; i < 8; i++) // Виведення
    cout<<i<<'\\t'<<B[i]<<'\\n'; // масиву
getch();
return 0;
}
```

Щоб переконатися в цьому, спробуйте при випробуванні програми 5.1 у другому циклі замість ... ; i < 8 ;... записати: ... ; i < 20;... Компілятор не виявить помилки й видасть на екран вісім нульових значень, а далі будуть виведені випадкові числа, які в даний час знаходяться у відповідних комірках пам'яті.

Перед використанням масиву його елементам необхідно присвоїти певні значення, в іншому випадку будемо мати масив з тими значеннями, які випадково опинилися у відповідних комірках пам'яті. Спробуйте, експериментуючи з програмою 5.1, вилучити рядки з присвоєнням початкових значень.

Зверніть увагу, що не можна безпосередньо вивести масив на екран. Якщо у цій програмі другий цикл замінити командою **cout<<B**, то замість елементів масиву на екрані побачимо щось на зразок 1245032. Це — адреса, за якою у пам'яті розміщений початок масиву. Саме тому для виведення всіх елементів масиву було використано цикл.

Оголошуючи масив, його елементам можна присвоїти початкові значення за допомогою вміщеного за оголошенням списку, взятого у фігурні дужки.

Приклад 2. У даній програмі елементам масиву цілих чисел присвоюються вісім значень, після чого масив друкується.

Якщо початкових значень менше, ніж елементів у масиві, елементи, що

```
#include<iostream.h> //Програма 5.2
#include<conio.h>
int main()
{
int B[8] = {1, 3, 5, 7, 11, 13, 17, 19};
cout<<"Nomer"<<'\\t'<<"Znachennja"<<'\\n';
for (int i = 0; i < 8; i++)
    cout<<i<<'\\t'<<B[i]<<'\\n';
getch();return 0;
}
```

залишилися, автоматично одержують нульові початкові значення. Всім елементам масиву **n** можна надати нульові початкові значення за допомогою оголошення:

```
int n[10] = {0};
```

яке явно надає нульове початкове значення першому елементу й неявно - нульові початкові значення дев'яти елементам, що залишилися, тому що початкових значень тут менше, ніж елементів масиву.

Якщо розмір масиву не зазначений в оголошенні, то кількість елементів масиву буде дорівнювати кількості елементів у списку початкових значень. Наприклад, для створення масиву **B** з п'яти елементів запишемо:

```
int B[] = {1, 2, 3, 4, 5};
```

Приклад 3. У наступній програмі присвоюються початкові цілі значення 1, 2, 3, ... 10 елементам масиву **M** з десяти елементів і на екран виводяться елементи масиву з парними номерами:

Зверніть увагу, що у прикладі 5.3 для задання розміру масиву **M** в оголошенні **int M[n]** використовується іменована константа **n**.

```
#include<iostream.h> //Програма 5.3
#include<conio.h>
int main()
{
const int n = 10;
int M[n];
for (int k = 0; k < n; k++)
M[k] = k + 1;
cout<<"Element"<<"\t"<<"Znachennja"<<"\n";
for (int k = 0; k < n; k += 2)
cout<<k<<"\t"<<M[k]<<"\n";
getch();return 0;
}
```

Використання іменованих констант для задання розмірів масивів робить програму зручною для редагування.

У прикладі 5.3 перший цикл **for** заповнюватиме 200-елементний масив, якщо просто змінити значення **n** на початку програми з 10 на 200. Якби ми не використали іменовану константу **n**, потрібно було б змінити програму в трьох різних місцях, щоб застосувати її для обробки масиву з 200 елементів.

Передача масивів у функції

Програми можуть передавати масиви у функції так само, як і будь-які інші змінні. Передаючи масив у функцію, ви повинні вказати тип масиву. Оскільки в C++ не контролюється розмір масиву, то у функцію слід передати також параметр, що містить кількість елементів у масиві:

Такий прийом дозволяє однією функцією обробляти масиви різних розмірів.

```
void fun (int A[], int n);
```

У наступній програмі масиви різних розмірів передаються у функцію **show_array**, яка використовує цикл **for** для виведення значень масивів:

```
#include<iostream.h> //Програма 5.4
#include<conio.h>
void show_array(int A[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout << A[i] << ' ';
    cout << endl;
}
int main()
{
    int little[5] = {1,2,3,4,5};
    int big[3] = { 1000, 2000, 3000 };
    show_array(little, 5);
    show_array(big, 3);
    getch();return 0;
}
```

Як бачите, у функцію передається ім'я масиву, а також вказується параметр, що повідомляє функції кількість елементів у масиві:

```
show_array (little, 5);
```

У наступній програмі для уведення з клавіатури значень у масив створена функція **get_values**:

```
#include<iostream.h> //Програма 5.5
#include<conio.h>
void get_values (int A[], int n)
{
    int i;
    for (i = 0; i < n ; i++)
    {
        cout << "Vvedit' " << i << ": ";
        cin >> A[i];
    }
}

int main()
{
    int N[3];
    get_values(N, 3);
    cout << "Znachennya masivu:" << endl;
    for (int i = 0; i < 3; i++)
        cout << N[i] << endl;
    getch();return 0;
}
```

Складання програм із масивами

Сума елементів масиву

Задача 1. Знайти суму всіх елементів масиву цілих чисел A розмірністю n . Дані вводяться в програму користувачем із клавіатури.

```
#include<iostream.h> //Програма 5.4
#include<conio.h>
int main()
{
const int n = 10;
int A[n];
cout<<"Vvedi masiv iz "<<n<<" chisel:\n";
for (int j = 0; j < n; j++) //Уведення масиву
{
cout<<"A["<<j<<"]="<<cin >> A[j];
}
int S=0; //Знаходження суми й друк результатів
for (int j = 0; j < n; j++)
S += A[j];
cout<<"Suma="<<S<<'\n';
getch();return 0;
}
```

Перша частина програми служить для уведення елементів масиву в пам'ять комп'ютера. У C++ не можна вказувати масив безпосередньо у команді уведення даних (наприклад, `cin>>A;`), бо це призводить до помилки компіляції. Тому для заповнення масиву з клавіатури використано цикл **for**, в якому значення кожного з елементів масиву вводиться окремо.

Для знаходження суми елементів масиву виділимо змінну з іменем **S**. Спочатку запишемо в неї значення **0** (додавання ще не здійснювалося). Потім, впродовж циклу, в змінній **S** будемо накопичувати результат підсумовування елементів масиву. Для цього щоразу до значення **S** додаватимемо черговий елемент масиву **A[j]** і результат вміщуватимемо знову в змінну **S**:

S += A[j]

Нагадуємо, що запис **S += A [j]** рівносильний **S = S + A [j]**.

Це продовжуватимемо доти, поки змінна циклу **j** не стане дорівнювати **n**. Як тільки це станеться — цикл завершиться і у змінній **S** залишиться результат підсумовування.

Пошук елементів із заданою властивістю

Задача 2. Підрахувати і надрукувати кількість від'ємних елементів у масиві з 10 цілих чисел x .

```

#include<iostream.h>                //Програма 5.5
#include<conio.h>
int main()
{
const int n = 10;                    // Розмірність масиву
int x[n];                            // Оголошення масиву
for(int i = 0; i < n; i++)
{
cout<<i<<" element:\t";
cin>>x[i];                          //Уведення елементів масиву
}
// Підрахунок від'ємних
int col = 0;
for(int i = 0; i < n; i++)
    if(x[i] < 0)                      // Перевірка
        col++;                       // Лічильник
cout<<endl;
for(int i = 0; i < n; i++)
    cout<<x[i]<<"\t";                //Друк елементів масиву
cout<<"\nvidjemnih elementiv:\t"<<col<<endl;
getch();return 0;
}

```

У цій програмі, на відміну від попередньої, до змінної з ім'ям **col** буде додаватися одиниця, кожного разу, коли буде зустрічатися від'ємний елемент масиву. Відбір відбувається за допомогою умовного оператора **if (x [i] < 0)**.

Знаходження мінімального й максимального елементів

Змінним **Min** та **Max** на початку програми присвоїмо значення першого елементу масиву. Потім у циклі будемо перебирати по черзі всі наступні елементи масиву і виконувати перевірку:

- якщо поточний елемент масиву більший, ніж **Max**, то змінній **Max** присвоюється значення поточного елемента;
- якщо поточний елемент масиву менший, ніж **Min**, то змінній **Min** присвоюється значення поточного елемента.

```

#include<iostream.h> //Програма 5.6
#include<conio.h>
int main()
{
const int n = 10; //Розмірність масиву
float a[n]; //Оголошення масиву
for(int i = 0; i < n; i++)
{
cout<<"Vvedit' "<<i<<" element:\t";
cin>>a[i]; //Уведення елементів масиву
}
float Min, Max;
Min=Max=a[0]; //Ініціалізація 1-м ел-том масиву
// Порівняння з поточним елементом
for(int i = 1; i < n; i++)
if(a[i] > Max)
Max = a[i]; //Поточний Max елемент
else
if(a[i] < Min)
Min = a[i]; // Поточний Min елемент
cout<<"\nMin=\t"<<Min<<"\nMax=\t"<<Max<<endl;
getch();
return 0;
}

```

Хід роботи:

Завдання 1.

1. Випробуйте програму 5.1. Зробіть кожен елемент масиву рівним 5.
2. Випробуйте програму 5.2. Як змінити програму, щоб останні 4 елементи дорівнювали нулю? Спробуйте не вказувати розмір масиву в оголошенні.
3. Випробуйте програму 5.3. Задайте розмір масиву рівним 15. Виведіть на екран елементи масиву з непарними номерами.
4. Випробуйте програми 5.4.- 5.5.
5. Збережіть програми у власній папці.

Завдання 2.

1. Випробуйте програму 5.4 для $n=10$ та $n=5$. Знайдіть добуток всіх елементів таблиці А, яка містить тільки додатні цілі числа.
2. Випробуйте програму 5.5. Підрахуйте і надрукуйте кількість додатних елементів масиву цілих чисел х.
3. Випробуйте програму 5.6. Уважно дослідіть роботу програми. Зменшіть значення кожного елемента масиву на величину мінімального елемента.
4. Збережіть програми у власній папці.

Контрольні запитання:

1. Що таке масив?
2. Для чого елементи масиву нумерують?
3. Як звернутися до елемента масиву?
4. Як нумеруються елементи масиву?
5. Яким повинен бути індекс масиву?
6. Чи можливо у якості індексу використовувати математичні вирази?
7. Які імена можуть мати масиви?
8. Для чого описують тип масиву?
9. Чи можуть у одному масиві використовуватися дані різних типів?
10. Що буде, якщо не присвоїти початкові значення всім елементам масиву?
11. Чи здійснюється у C++ контроль за виходом за межі масиву?
12. Чи можна таким чином вивести масив B на екран: `cout<<B`? Що при цьому ми побачимо на екрані?
13. Які способи присвоєння значень елементам масиву ви знаєте?
14. Що таке іменована константа? Для чого їх використовують при роботі з масивами?
15. Як передати масив у функцію?
16. Чи можна увести елементи масиву таким чином: `cin>>A`?
17. Який оператор використовують для послідовної обробки елементів масиву?
18. Опишіть алгоритм знаходження суми елементів масиву.
19. Опишіть алгоритм підрахунку кількості від'ємних елементів у масиві.
20. Опишіть алгоритм знаходження найбільшого елемента в масиві.