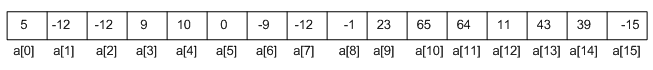
Одномірні масиви в С++

Одновимірна масив - масив, з одним параметром, що характеризує кількість елементів одновимірного масиву. Фактично одновимірний масив - це масив, у якого може бути тільки один рядок, і n-е кількість стовпців. Стовпці в одновимірному масиві - це елементи масиву. На малюнку 1 показана структура целочисленного одновимірного масиву a. Розмір цього масиву - 16 комірок.

Малюнок 1 - Масиви в С++

Зауважте, що максимальний індекс одновимірного масиву a дорівнює 15, але розмір масиву 16 комірок, тому що нумерація осередків масиву завжди починається з 0. Індекс осередки - це ціле невід'ємне число, за яким можна звертатися до кожної клітинки масиву і виконувати будь-які дії над ній (коміркою).

|  |  |
| --- | --- |
| 1  2  3  4 | // синтаксис оголошення одновимірного масиву в С++:  /\* тип даних\*/  /\* ім'я одновимірного масиву\*/[/\* розмірність одновимірного масиву \*/];  // приклад оголошення одновимірного масиву, зображеного на малюнку 1:  int a[16]; |

де, int—цілочисельний тип даних;

а - ім'я одновимірного масиву;

16 - розмір одновимірного масиву, 16 комірок.

Завжди відразу після імені масиву йдуть квадратні скобочки, в яких задається розмір одновимірного масиву, цим масив і відрізняється від всіх інших змінних.

|  |  |
| --- | --- |
| 1  2 | // ще один спосіб оголошення одновимірних масивів  int mas[10], a[16]; |

Оголошено два одновимірних масиву mas і а розмірами 10 і 16 відповідно. Причому в такому способі оголошення все масиви матимуть однаковий тип даних, в нашому випадку— int.

|  |  |
| --- | --- |
| 1  2 | // масиви можуть бути ініційовані при оголошенні:  int a[16] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 23, 65, 64, 11, 43, 39, -15 }; //  ініціалізація одновимірного масива |

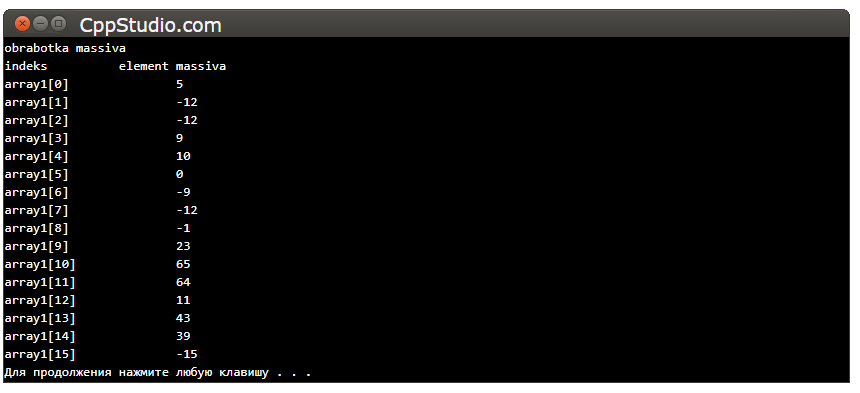
Ініціалізація одновимірного масиву виконується в фігурних дужках після знака **дорівнює**, кожен елемент масиву відокремлюється від попереднього коми.

|  |  |
| --- | --- |
| 1 | int a[]={5,-12,-12,9,10,0,-9,-12,-1,23,65,64,11,43,39,-15}; // ініціалізації масиву без визначення його  розміру. |

В даному випадку компілятор сам визначить розмір одновимірного масиву. Розмір масиву можна не вказувати тільки при його ініціалізації, при звичайному оголошенні масиву обов'язково потрібно вказувати розмір масиву. Розробимо просту програму на обробку одновимірного масиву.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // array.cpp: визначає точку входу для консольного застосування.    #include "stdafx.h"  #include <iostream>  using namespace std;    int main(int argc, char\* argv[])  {      cout << "obrabotka massiva" << endl;      int array1[16] = { 5, -12, -12, 9, 10, 0, -9,                      -12, -1, 23, 65, 64, 11, 43, 39, -15 }; // оголошення і ініціалізація одновимірногомасиву    cout << "indeks" << "\t\t" << "element massiva" << endl; // друк заголовків      for (int counter = 0; counter < 16; counter++)  //початок цикла      {      // висновок на екран індексу осередку масиву, а потім вмісту цього осередку, в нашому випадку - це ціле число     cout << "array1[" << counter << "]" << "\t\t" << array1[counter] << endl;      }      system("pause");      return 0;  } |

У **рядках 10 - 11** оголошений і ініціалізованим першим цілочисельний одномірний масив з ім'ям array1, розмір якого дорівнює 16 комірок, тобто такий масив може зберігати 16 чисел. Будь-яка обробка масиву здійсненна тільки спільно з циклами. Який цикл вибрати для обробки масиву - це вам вирішувати. Але найкраще для цього завдання підходить [цикл for](http://cppstudio.com/obuchenie_cpp/tsikl-for). Змінну-лічильник counterбудемо використовувати для звернення до елементів одновимірного масиву array1. В умови продовження циклу for варто строгий знак нерівності, так як шістнадцятого індексу в одновимірному масиві array1 немає. А так як нумерація комірок починається з нуля, то елементів в масиві 16. У тілі циклу for оператор countдрукує елементи одновимірного масиву (див. Малюнок 2).

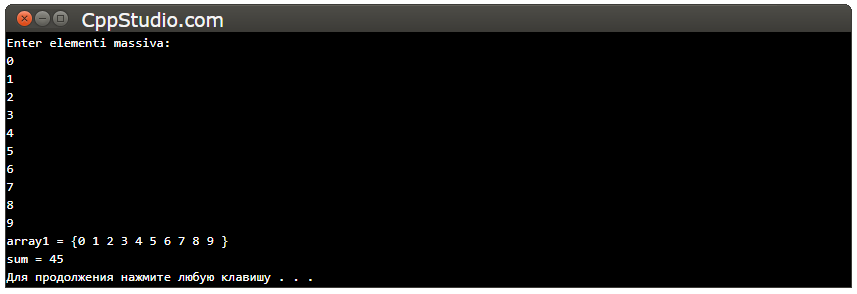


Малюнок 2 - Масиви в С++

Розробимо ще одну програму на обробку одновимірного масиву в С ++. Програма повинна послідовно зчитувати десять введених чисел з клавіатури. Всі введені числа підсумувати, результат вивести на екран.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // array\_sum.cpp: визначає точку входу для консольного застосування.    #include "stdafx.h"  #include <iostream>  using namespace std;    int main(int argc, char\* argv[])  {      int array1[10]; // оголошуємо цілочисельний масив      cout << "Enter elementi massiva: " << endl;      int sum = 0;      for ( int counter = 0; counter < 10; counter++ ) // цикл для зчитування чисел          cin >> array1[counter];  // зчитуємо вводяться з клавіатури числа      cout << "array1 = {";      for ( int counter = 0; counter < 10; counter++ ) // цикл для виведення елементів масиву          cout << array1[counter] << " ";  // виводимо елементи масиву на стандартний пристрій виводу      for ( int counter = 0; counter < 10; counter++ ) // цикл для підсумовування чисел масиву          sum += array1[counter]; // підсумовуємо елементи масиву      cout << "}\nsum = " << sum << endl;      system("pause");      return 0;  } |

Перед тим як виконувати обробку масиву його необхідно оголосити, причому розмір одновимірного масиву дорівнює 10, так як це обумовлено умовою завдання. В змінної sumбудемо накопичувати суму елементів одновимірного масиву. перший цикл forзаповнює оголошений одновимірний масив, введеними з клавіатури числами, **рядки 12 — 13**. Мінлива лічильник counterвикористовується для послідовного доступу до елементів одновимірного масиву array1, починаючи з індексу 0 і до 9-го включно.Другий цикл for виводить на екран елементи масиву, **рядки 15 — 16**.Третій цикл forпослідовно зчитує елементи одновимірного масиву і підсумовує їх, сума накопичується в змінної sum,**рядки 17 — 18**. Результат роботи програми дивитися на малюнку 3.

Для продолжения нажмите любую клавишу . . .

Малюнок 3 - Масиви в С++

Як знайти час роботи програми на С++*.*

У багатьох початківців програмістів рано чи пізно виникає питання: «Як знайти час роботи програми?». В інтернеті багато відповідей на це питання: написати свій міні-дебагер, порахувати кількість тактів і т. Д. Найпростіший варіант - це порахувати різницю між початковим часом і кінцевим. Тобто, є початкове значення часу, після якого оголошено фрагмент коду, час виконання якого необхідно виміряти. Після фрагмента коду фіксується ще одне, кінцеве, значення часу. Після чого, з кінцевого значення часу віднімаємо потрібний проміжок часу і отримаємо час, за яке виконався вимірюваний фрагмент коду або вся програма. Час роботи програми необхідно знайти для того, щоб перевірити, наскільки ефективніше почав працювати окремий алгоритм або програма в цілому! Як це зробити, дивіться нижче.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Як знайти час роботи фрагмента коду?    // заголовки з прототипом функції clock()  #include <ctime>  //         ...      unsigned int start\_time =  clock(); // початок часу      // тут повинен бути фрагмент коду, час виконання якого потрібно виміряти      unsigned int end\_time = clock(); // кінцевий час      unsigned int search\_time = end\_time - start\_time; // шуканий час |

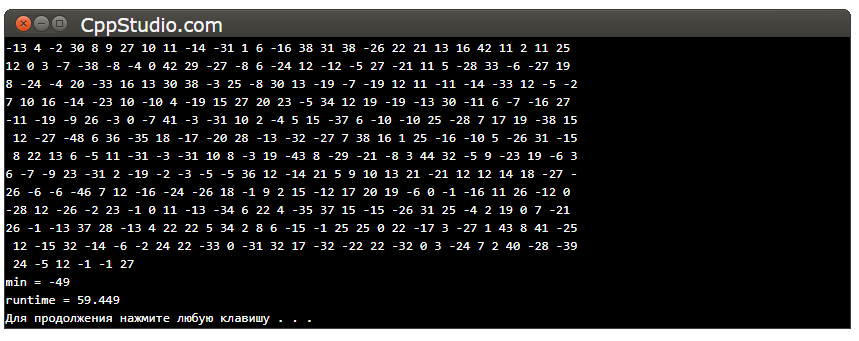
Для того, щоб знайти час роботи програми, потрібно скористатися функцією clock(). Прототип функції clock() знаходиться в заголовки <ctime>, який потрібно підключити, **рядок 4**. Функція clock() повертає значення часу в мілісекундах (1с = 1000млс). Причому відлік часу починається з моменту запуску програми. Якщо треба виміряти роботу всієї програми, то в кінці програми, перед оператором return 0; потрібно запустити функцію clock(), яка покаже робочий час. Для пошуку часу роботи фрагмента коду потрібно знайти різницю між кінцевим і початковим часом, як показано вище.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Як знайти час роботи програми?        // заголовки з прототипом функції clock()      #include <ctime>      //         ...      // тут повинен бути код програми, час виконання якої потрібно виміряти      unsigned int end\_time = clock(); // час роботи програми |

Розробимо програму, в якій за допомогою функції clock() обчислимо час роботи програми. Програма шукає мінімальне значення в масиві розміром в 200000 елементів. Розмір масиву спеціально обраний великим, для того, щоб було помітно, як працює програма. Так як числа генеруються випадково, то при кожному запуску виходить новий випадок, і час може не збігатися. До того ж, час виконання програми залежить від того, наскільки завантажений комп'ютер і від того, яка у комп'ютера обчислювальна потужність. На різних машинах по-різному буде витрачатися час на виконання програми, на більш потужних комп'ютерах витрачається час буде менше і навпаки.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | // runtime.cpp: визначає точку входу для консольного застосування.  // Як знайти час роботи програми?    #include "stdafx.h"  #include <iostream>  #include <ctime>  using namespace std;    int main(int argc, char\* argv[])  {      srand(time(0));      const int array\_size = 200000; // розмір одновимірного масиву      int array1[array\_size]; // оголошення одновимірного масиву      for (int counter = 0; counter < array\_size; counter++)      {       array1[counter] = rand() % 50 - rand() % 50; // заповнюємо масив випадковими значеннями в діапазоні від -49 до 49 включно       cout << array1[counter] << " "; // друк елементів одновимірного масиву array1    }      int min = array1[0]; // переменная для хранения минимального значения      for (int counter = 1; counter < array\_size; counter++)      {       if ( min > array1[counter] ) // пошук мінімального значення в одновимірному масиві           min = array1[counter];      }      cout << "\nmin = " << min << endl;      cout << "runtime = " << clock()/1000.0 << endl; // час роботи програми      system("pause");      return 0;  } |

У рядку **26** запускається функція clock(),яка скаже скільки було потрібно час програмою. розбирати алгоритм пошуку не потрібно, так як це зовсім інша тема. Головне, потрібно зрозуміти, як використовувати функцію clock(),для пошуку часу роботи програми або окремого фрагмента коду. А саме, в **рядку 26**, після основного коду програми, але до оператора return 0; оголошена функція clock(),яка поверне значення часу. Результат роботи програми (див. Малюнок 1).



На **малюнку 1** видно, що час виконання програми приблизно дорівнює 59 секунд. Навіть якщо елементи масиву незмінні, час все одно буде трохи відрізнятися при повторному запуску програми, так як деякі ресурси комп'ютера постійно будуть зайняті, а значить, не доступні програмі. Якщо Ви запустіть цю програму у себе на комп'ютері, то час виконання програми може бути зовсім іншим, так як наші машини не ідентичні.