Одномерные массивы в С++

Одномерный массив — массив, с одним параметром, характеризующим количество элементов одномерного массива. Фактически одномерный массив — это массив, у которого может быть только одна строка, и n-е количество столбцов. Столбцы в одномерном массиве — это элементы массива. На рисунке 1 показана структура целочисленного одномерного массива **a**. Размер этого массива — 16 ячеек.

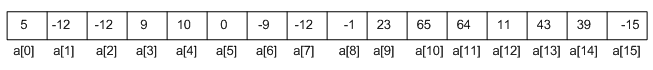


Рисунок 1 — Массивы в С++

Заметьте, что максимальный индекс одномерного массива **a** равен 15, но размер массива 16 ячеек, потому что нумерация ячеек массива всегда начинается с 0. Индекс ячейки – это целое неотрицательное число, по которому можно обращаться к каждой ячейке массива и выполнять какие-либо действия над ней (ячейкой).

|  |  |
| --- | --- |
| 1  2  3  4 | //синтаксис  объявления одномерного массива в С++:  /\*тип данных\*/  /\*имя одномерного массива\*/[/\*размерность одномерного массива\*/];  //пример объявления одномерного массива, изображенного на рисунке 1:  int a[16]; |

где, int—целочисленный [тип данных](http://cppstudio.com/obuchenie_cpp/tipy_dannyh);

а —  имя одномерного массива;  
16 — размер одномерного массива, 16 ячеек.

Всегда сразу после имени массива идут квадратные скобочки, в которых задаётся размер одномерного массива, этим массив и отличается от всех остальных переменных.

|  |  |
| --- | --- |
| 1  2 | //ещё один способ объявления одномерных массивов  int mas[10], a[16]; |

Объявлены два одномерных массива  masи а размерами 10 и 16 соответственно. Причём в таком способе объявления все массивы будут иметь одинаковый тип данных, в нашем случае — int.

|  |  |
| --- | --- |
| 1  2 | // массивы могут быть инициализированы при объявлении:  int a[16] = { 5, -12, -12, 9, 10, 0, -9, -12, -1, 23, 65, 64, 11, 43, 39, -15 }; // инициализация одномерного массива |

Инициализация одномерного массива выполняется в фигурных скобках после знака **равно**, каждый элемент массива отделяется от предыдущего запятой.

|  |  |
| --- | --- |
| 1 | int a[]={5,-12,-12,9,10,0,-9,-12,-1,23,65,64,11,43,39,-15}; // инициализации массива без определения его размера. |

В данном случае компилятор сам определит размер одномерного массива. Размер массива можно не указывать только при его инициализации, при обычном объявлении массива обязательно нужно указывать размер массива. Разработаем простую программу на обработку одномерного массива.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // array.cpp: определяет точку входа для консольного приложения.    #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 }; // объявление и инициализация одномерного массива      cout << "indeks" << "\t\t" << "element massiva" << endl; // печать заголовков      for (int counter = 0; counter < 16; counter++)  //начало цикла      {      //вывод на экран индекса ячейки массива, а затем содержимого этой ячейки, в нашем случае - это целое число       cout << "array1[" << counter << "]" << "\t\t" << array1[counter] << endl;      }      system("pause");      return 0;  } |

В **строках 10 — 11**объявлен и проинициализирован целочисленный одномерный массив с именемarray1, размер которого равен 16 ячейкам, то есть такой массив может хранить 16 чисел. Любая обработка массива осуществима только совместно с циклами. Какой цикл выбрать для обработки массива — это вам решать. Но лучше всего для этой задачи подходит [цикл for](http://cppstudio.com/obuchenie_cpp/tsikl-for). Переменную-счётчик counterбудем использовать для обращения к элементам одномерного массива array1. В условии продолжения цикла forстоит строгий знак неравенства, так как шестнадцатого индекса в одномерном массиве array1 нет. А так как нумерация ячеек начинается с нуля, то элементов в массиве 16. В теле цикла for оператор  coutпечатает элементы одномерного массива (см. Рисунок 2).

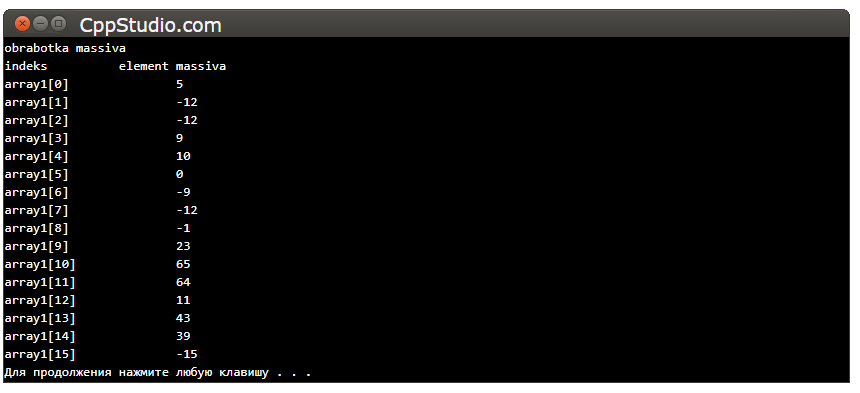


Рисунок 2 — Массивы в С++

Разработаем ещё одну программу на обработку одномерного массива в С++. Программа должна последовательно считывать десять введённых чисел с клавиатуры. Все введённые числа  просуммировать, результат вывести на экран.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | // array\_sum.cpp: определяет точку входа для консольного приложения.    #include "stdafx.h"  #include <iostream>  using namespace std;    int main(int argc, char\* argv[])  {      int array1[10]; // объявляем целочисленный массив      cout << "Enter elementi massiva: " << endl;      int sum = 0;      for ( int counter = 0; counter < 10; counter++ ) // цикл для считывания чисел          cin >> array1[counter];  // считываем вводимые с клавиатуры числа      cout << "array1 = {";      for ( int counter = 0; counter < 10; counter++ ) // цикл для вывода элементов массива          cout << array1[counter] << " ";  // выводим элементы массива на стандартное устройство вывода      for ( int counter = 0; counter < 10; counter++ ) // цикл для суммирования чисел массива          sum += array1[counter]; // суммируем элементы массива      cout << "}\nsum = " << sum << endl;      system("pause");      return 0;  } |

Перед тем как выполнять обработку массива его необходимо объявить, причём размер одномерного массива равен 10, так как это оговорено условием задачи. В переменной sumбудем накапливать сумму элементов одномерного массива. Первый цикл forзаполняет объявленный одномерный массив, введёнными с клавиатуры числами, **строки 12 — 13**. Переменная счётчик counterиспользуется для последовательного доступа к элементам одномерного массива array1, начиная с индекса 0 и до 9-го включительно. Второй цикл for выводит на экран элементы массива, **строки 15 — 16**. Третий циклforпоследовательно считывает элементы одномерного массива и суммирует их, сумма накапливается в переменнойsum,**строки 17 — 18**. Результат работы программы смотреть на рисунке 3.

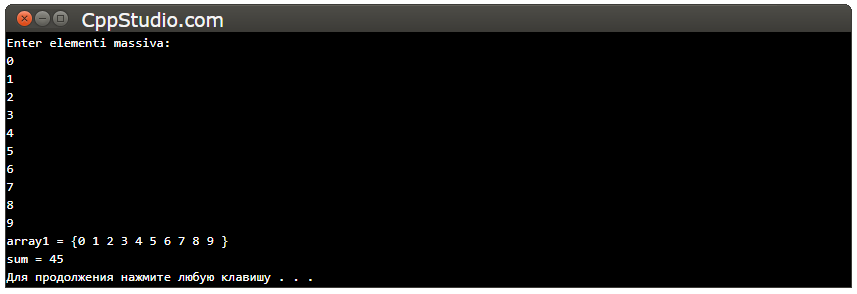
Для продолжения нажмите любую клавишу . . .

Рисунок 3 — Массивы в С++

Как найти время работы программы на С++*.*

У многих начинающих программистов рано или поздно возникает вопрос: «Как найти время работы программы?». В интернете много ответов на данный вопрос: написать свой мини-дебаггер, посчитать количество тактов и т. д. Самый простой вариант — это посчитать разницу между начальным временем и конечным. То есть, есть начальное значение времени, после которого объявлен фрагмент кода, время выполнения которого необходимо измерить. После фрагмента кода фиксируется ещё одно, конечное, значение времени. После чего, из конечного значения времени вычитаем начальное время и получим время, за которое выполнился измеряемый фрагмент кода или вся программа. Время работы программы  необходимо найти для того, чтобы проверить, насколько эффективнее стал работать отдельный алгоритм или программа в целом! Как это сделать,смотрите ниже.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | // Как найти время работы фрагмента кода?    // заголовочный файл с прототипом функции clock()  #include <ctime>  //         ...      unsigned int start\_time =  clock(); // начальное время      // здесь должен быть фрагмент кода, время выполнения которого нужно измерить      unsigned int end\_time = clock(); // конечное время      unsigned int search\_time = end\_time - start\_time; // искомое время |

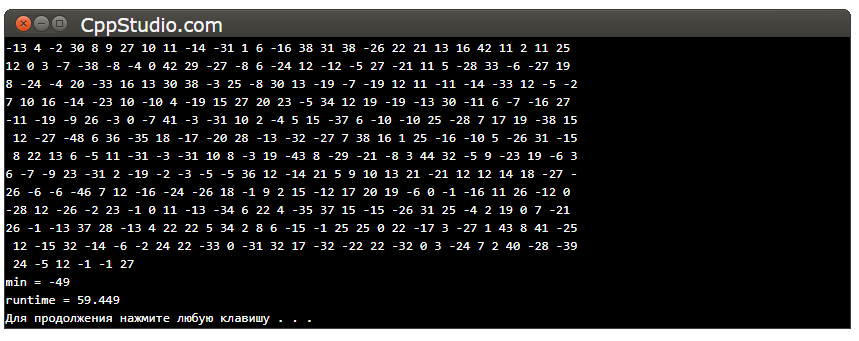
Для того, чтобы найти время работы программы, нужно воспользоваться функцией clock(). Прототип функции clock()находится в заголовочном файле <ctime>, который нужно подключить, **строка 4**. Функция clock() возвращает значение времени в миллисекундах (1с = 1000млс). Причём  отсчёт времени начинается с момента запуска программы. Если надо измерить работу всей программы, то в конце программы, перед оператором return 0;  нужно запустить функцию clock(), которая покажет рабочее время. Для поиска времени работы фрагмента кода нужно найти разницу между конечным и начальным временем, как показано выше.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | // Как найти время работы программы?        // заголовочный файл с прототипом функции clock()      #include <ctime>      //         ...      // здесь должен быть код программы, время выполнения которой нужно измерить      unsigned int end\_time = clock(); // время работы программы |

Разработаем программу, в которой с помощью функции clock() вычислим время работы программы. Программа ищет минимальное значение в массиве размером в 200000 элементов. Размер массива специально выбран большим, для того, чтобы было заметно, как работает программа. Так как числа генерируются случайно, то при каждом запуске получается новый случай, и время может не совпадать. К тому же, время выполнения программы зависит от того, насколько загружен компьютер и от того, какая у компьютера вычислительная мощность. На разных машинах по-разному  будет затрачиваться время на выполнение программы, на более мощных компьютерах затрачиваемое время будет меньше и наоборот.

|  |  |
| --- | --- |
| 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: определяет точку входа для консольного приложения.  // Как найти время работы программы?    #include "stdafx.h"  #include <iostream>  #include <ctime>  using namespace std;    int main(int argc, char\* argv[])  {      srand(time(0));      const int array\_size = 200000; // размер одномерного массива      int array1[array\_size]; // объявление одномерного массива      for (int counter = 0; counter < array\_size; counter++)      {       array1[counter] = rand() % 50 - rand() % 50; // заполняем массив случайными значениями в диапазоне от -49 до 49 включительно       cout << array1[counter] << " "; // печать элементов одномерного массива array1      }      int min = array1[0]; // переменная для хранения минимального значения      for (int counter = 1; counter < array\_size; counter++)      {       if ( min > array1[counter] ) // поиск минимального значения в одномерном массиве           min = array1[counter];      }      cout << "\nmin = " << min << endl;      cout << "runtime = " << clock()/1000.0 << endl; // время работы программы      system("pause");      return 0;  } |

В **строке 26** запускается функция clock(), которая скажет сколько потребовалось время программе. Разбирать [алгоритм поиска](http://cppstudio.com/obuchenie_cpp/poisk-v-massivah) не нужно, так как это совсем другая тема. Главное, нужно понять, как использовать функцию clock(), для поиска времени работы программы или отдельного фрагмента кода. А именно, в **строке 26**, после основного кода программы, но до оператора return 0; объявлена функция clock(), которая вернёт значение времени. Результат работы программы (см. Рисунок 1).



На **рисунке 1** видно, что время выполнения программы приблизительно равно 59 секунд. Даже если элементы массива неизменны, время всё равно будет немного отличаться при повторном запуске программы, так как некоторые ресурсы компьютера постоянно будут заняты, а значит, не доступны программе. Если Вы запустите эту программу у себя на компьютере, то время выполнения программы может быть совсем другим, так как наши машины не идентичны.