**1. Что такое массив? Определение массива**

Массив – набор переменных одинакового типа. Доступ к этим переменным осуществляется по одному имени. Это имя называется именем массива.

Массивы используются для группирования связанных переменных между собой.

**2. Определение одномерных и многомерных массивов.**

Массивы могут быть одномерными и многомерными. В одномерных массивах для доступа к элементу массива используется один индекс. В многомерных массивах для доступа к элементу массива используется несколько индексов.

**3. Описание одномерного массива. Примеры описания одномерных массивов**

Общая форма описания одномерного массива:

**тип имя\_массива[*размер*];**

В вышеприведенном описании:

* *тип* – это тип элементов массива. Он еще называется базовым типом. Базовый тип определяет количество данных каждого элемента, который составляет массив. Тип элементов массива может быть как [**базовым типом**](http://www.bestprog.net/ru/2016/08/16/03-%d0%b1%d0%b0%d0%b7%d0%be%d0%b2%d1%8b%d0%b5-%d1%82%d0%b8%d0%bf%d1%8b-%d0%b4%d0%b0%d0%bd%d0%bd%d1%8b%d1%85-visual-c/) так и составным (например, структура). Подробно о базовых типах данных C++ описывается [**здесь**](http://www.bestprog.net/ru/2016/08/16/03-%d0%b1%d0%b0%d0%b7%d0%be%d0%b2%d1%8b%d0%b5-%d1%82%d0%b8%d0%bf%d1%8b-%d0%b4%d0%b0%d0%bd%d0%bd%d1%8b%d1%85-visual-c/).
* *размер*– количество элементов в массиве;
* *имя\_массива* – непосредственно имя массива, по которому осуществляется доступ к элементам массива.

После описания массива, значение элементов может быть нулевым или неопределенным.

**Пример 1.** Описание массива из 10 целых чисел (тип int) с именем A.

int A[10];

В результате, в памяти компьютера выделяется 10 ячеек типа int. Если одна ячейка занимает 2 байта, то всего будет выделено 20 байт памяти. Номер первой ячейки начинается с нуля. Эти ячейки объединены общим именем A.

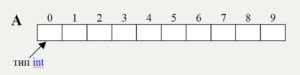
[](http://www.bestprog.net/wp-content/uploads/2017/02/05_02_02_04_02_01u.jpg)

Рисунок 1. Массив из 10 целых чисел

**Пример 2**. Описание массива с именем M, содержащим 20 элементов типа char.

char M[20];

**4. Доступ к элементам одномерного массива. Примеры**

Доступ к отдельному элементу массива осуществляется с помощью индекса. Индекс определяет позицию элемента в массиве. Первый элемент массива имеет нулевой индекс.

Чтобы получить доступ к отдельному элементу массива по его индексу, нужно после имени массива в квадратных скобках указать номер этого элемента.

**Пример 1.** Пусть дан массив с именем A, содержащий 10 целых чисел. Записать число 5 в первый и последний элементы массива.

// описание массива A

int A[10];

A[0] = 5; // первый элемент массива

A[9] = 5; // последний элемент массива

На рисунке 2 видно результат работы вышеприведенного программного кода.

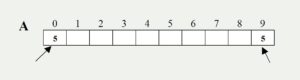
[](http://www.bestprog.net/wp-content/uploads/2017/02/05_02_02_04_02_02_.jpg)

Рисунок 2. Результат работы фрагмента кода

**Пример 2.** Массив из 10 элементов типа char.

char M[10];

M[3] = 'a';

M[7] = '0';

M[8] = ';';

[](http://www.bestprog.net/wp-content/uploads/2017/02/05_02_02_04_02_03_.jpg)  
Рисунок 3. Массив из 10 элементов типа char

**Пример 3.** Массив из 30 вещественных чисел.

double M[30]; // массив из 30 вещественных чисел

// обнуление массива M

for (int i=0; i<30; i++)

M[i] = 0.0;

**5. Как определить размер одномерного массива?**

Размер одномерного массива определяется по формуле:

**размер массива = размер типа в байтах × количество элементов**

**Пример.**

Если в массиве 20 элементов типа double (8 байт), то размер массива будет

**размер = 20 × 8 = 160 байт**

**6. Особенности обработки массивов компилятором в C++. Границы массива**

В C++ не ведется строгого контроля за доступом к элементам за пределами массива. Если описать массив из 100 элементов, то можно прочитать или изменить 101-й, 102-й и следующие элементы. На этих позициях могут быть ячейки памяти, которые были выделены для других переменных или даже для вашей программы. Это может привести к уничтожению программы при отсутствии каких-нибудь замечаний со стороны компилятора C++.

Значит, вся ответственность за соблюдение границ массивов лежит строго на программисте. Программист должен писать такой код, который гарантирует корректную работу с массивами. Это реализуется с помощью включения в программу специальных проверок.

Такое поведение компилятора C++ можно объяснить по следующим причинам. Язык C++ предназначен для профессионалов. Задача языка – дать возможность профессиональным программистам создавать максимально эффективный код. Если в компилятор C++ включить код проверки, то это существенно замедлит выполнение программы и код не будет максимально эффективным.

В любом случае, у программиста есть выбор — осуществлять проверку на корректность номера индекса в диапазоне индексов массива или нет. В этом и есть вся прелесть C++.

**7. Как осуществляется инициализация массива в C++. Примеры**

В C++ поддерживается два вида инициализации массивов:

* инициализация с заданием размера массива;
* «безразмерная» инициализация.

Общий вид инициализации с заданием размера массива:

**тип имя\_массива[*размер*] = { *список\_значений* };**

где

* тип – тип элементов массива;
* размер – количество элементов массива указанного типа;
* список\_значений – список значений инициализации элементов массива. Элементы массива разделяются символом ‘,’ (запятая).

Общий вид «безразмерной» инициализации:

**тип имя\_массива[] = { *список\_значений* };**

В этом случае размер массива определяется количеством элементов, которые описаны в *список\_значений*.

**Пример 1.** Массив B инициализирован с заданием размера.

// инициализация массива B

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

**Пример 2.** Массив C инициализирован на основе списка значений («безразмерная» инициализация).

// инициализация массива C без задания размера

float C[] = { -3.9, 2.8, -1.6, 2.2 };

**8. Инициализация символьных массивов. Пример**

Для символьных массивов можно использовать сокращенный вариант инициализации:

**char имя\_массива[*размер*] = "строка";**

В этом случае каждому элементу массива присваивается один символ строки.

**Пример.** Инициализация символьного массива с именем str.

// символьный массив - сокращенный вариант инициализации

char str[] = "Hello!";

// другой вариант инициализации символьного массива

char str2[] = { 'H', 'e', 'l', 'l', 'o', '!' };

В вышеприведенном примере массивы str и str2 содержат одинаковые значения.

**9. Присвоение одного массива другому. Пример**

В языке программирования C++ (в отличие от других языков) нельзя присваивать непосредственно один массив другому. Присваивать можно только поэлементно с использованием оператора цикла. При этом оба массива должны иметь одинаковый тип элементов.

Пусть заданы два массива целых чисел. Фрагмент кода, который присваивает один массив другому:

// описание массивов A и B

int A[10], B[10];

int i;

//A = B; // ошибка!

for (i=0; i<10; i++)

B[i] = 0;

for (i=0; i<10; i++)

A[i] = B[i];

**1. Нахождение сумм и произведений элементов массива. Примеры**

**Пример 1.** Задан массив A, содержащий 100 целых чисел. Найти сумму элементов этого массива. Фрагмент кода, решающего эту задачу

// сумма элементов массива A из 100 целых чисел

int A[100];

int suma; // переменная, содержащая сумму

int i; // дополнительная переменная

// ввод массива A

// ...

// Вычисление суммы

suma = 0; // обнулить сумму

for (i=0; i<100; i++)

    suma += A[i];

Перебор всех элементов массива выполняется в цикле for.

Переменная sum сохраняет результирующее значение суммы элементов массива. Переменная i есть счетчиком, определяющим индекс элемента массива A[i].

**Пример 2.** Задан массив B, содержащий 20 вещественных чисел. Найти сумму элементов массива, которые лежат на парных позициях. Считать, что позиции 0, 2, 4 и т.д. есть парными.

// сумма элементов массива B

// лежащих на парных позициях

float B[20];

float sum; // переменная, содержащая сумму

int i; // дополнительная переменная

// ввод массива

// ...

// Вычисление суммы

sum = 0; // обнулить сумму

for (i=0; i<20; i++)

    if ((i%2)==0)

        sum += B[i];

В этом примере выражение

(i%2)==0

определяет парную позицию (парный индекс) массива B. Если нужно взять нечетные позиции, то нужно написать

(i%2)==1

**Пример 3**. Задан массив, который содержит 50 целых чисел. Найти сумму положительных элементов массива.

// сумма положительных элементов массива

int A[50];

int sum; // переменная, содержащая сумму

int i; // дополнительная переменная

// ввод массива

// ...

// Вычисление суммы

sum = 0; // обнулить сумму

for (i=0; i<50; i++)

    if (A[i]>0)

        sum = sum + A[i];

Пример 4. Задан массив из 50 целых чисел. Найти произведение элементов массива, которые есть нечетными числами.

// произведение нечетных элементов массива

int A[50];

int d; // переменная, содержащая произведение

int i; // вспомогательная переменная

// ввод массива

// ...

// Вычисление произведения

d = 1; // начальная установка переменной d

for (i=0; i<50; i++)

    if ((A[i]%2)==1)

        d = d \* A[i];

Чтобы определить, есть ли элемент массива A[i] нечетным, нужно проверить условие

(A[i]%2)==1

Если условие выполняется, то элемент массива есть нечетное число.

**2. Нахождение максимального (минимального) элемента массива. Примеры**

**Пример 1.** Задан массив из 30 вещественных чисел. Найти элемент (индекс), имеющий максимальное значение в массиве.

// поиск позиции (индекса), содержащего максимальное значение

float B[30];

float max; // переменная, содержащая максимум

int index; // позиция элемента, содержащего максимальное значение

int i; // дополнительная переменная

// ввод массива

// ...

// поиск максимума

// установить максимум как 1-й элемент массива

index = 0;

max = B[0];

for (i=1; i<30; i++)

    if (max<B[i])

    {

        max = B[i]; // запомнить максимум

        index = i;  // запомнить позицию максимального элемента

    }

В вышеприведенном примере переменная max содержит максимальное значение. Переменная indexсодержит позицию элемента, который имеет максимальное значение. В начале переменной maxприсваивается значение первого элемента массива. Затем, начиная со второго элемента, происходит прохождение всего массива в цикле for. Одновременно проверяется условие

if (max<B[i])

Если условие выполняется (найден другой максимум), тогда новое значение максимума фиксируется в переменных max и index.

Вышеприведенный пример находит только один максимум. Однако, в массивах может быть несколько максимальных значений. В этом случае для сохранения позиций (индексов) максимальных значений нужно использовать дополнительный массив как показано в следующем примере.

**Пример 2.** Задан массив содержащий 50 целых чисел. Найти позицию (позиции) элемента, который имеет минимальное значение. Если таких элементов несколько, сформировать дополнительный массив индексов.

// поиск позиций (индексов), содержащих минимальное значение

int A[50];

int min; // переменная, содержащая минимальное значение

int INDEXES[50]; // позиции элементов, содержащих минимальное значение

int n; // число одинаковых минимальных значений

int i; // дополнительная переменная

// ввод массива

// ...

// 1. Поиск минимального значения

// установить минимальное значение в первом элементе массива

min = A[0];

for (i=1; i<50; i++)

    if (min>A[i])

        min = A[i]; // запомнить минимальное значение

// 2. Формирование массива

n = 0; // обнулить счетчик в массиве INDEXES

for (i=0; i<50; i++)

    if (min == A[i])

        {

            n++; // увеличить число элементов в INDEXES

            INDEXES[n-1] = i; // запомнить позицию

}

listBox1->Items->Clear();

// 3. Вывод массива INDEXES в listBox1

for (i=0; i<n; i++)

listBox1->Items->Add(INDEXES[i].ToString());

В вышеприведенном листинге сначала ищется минимальное значение min.

На втором шаге формируется массив INDEXES, в котором число элементов записывается в переменную n. Происходит поиск минимального значения в массиве A с одновременным формированием массива INDEXES.

На третьем шаге приведен пример, как вывести массив INDEXES в элементе управления listBox1(ListBox).

**3. Сортировка массива методом «пузырька»**

**Пример.** Пусть дан массив A, содержащий 10 целых чисел. Отсортировать элементы массива в нисходящем порядке с помощью метода «пузырька».

// сортировка массива методом "пузырька"

int A[10];

int i, j; // дополнительные переменные - счетчики

int t; // дополнительная переменная

// ввод массива A

// ...

// сортировка

for (i=0; i<9; i++)

    for (j=i; j>=0; j--)

        if (A[j]<A[j+1])

        {

            // поменять местами A[j] и A[j+1]

            t = A[j];

            A[j] = A[j+1];

            A[j+1] = t;

        }

**4.  Поиск элемента в массиве. Примеры**

**Пример 1.** Определить, находится ли число k в массиве M состоящем из 50 целых чисел.

// определение наличия заданного числа в массиве чисел

int M[50];

int i;

int k; // искомое значение

bool f\_is; // результат поиска, true - число k есть в массиве, иначе false

// ввод массива M

// ...

// ввод числа k

// ...

// поиск числа в массиве

f\_is = false;

for (i=0; i<50; i++)

    if (k==M[i])

    {

        f\_is = true; // число найдено

        break; // выход из цикла, дальнейший поиск не имеет смысла

    }

// вывод результата

if (f\_is)

    label1->Text = "Число " + k.ToString() + " есть в массиве M.";

else

    label1->Text = "Числа " + k.ToString() + " нет в массиве M.";

**Пример 2.** Найти все позиции вхождения числа k в массиве M состоящим из 50 целых чисел.

// определение всех позиций заданного числа в массиве чисел

int M[50]; // массив чисел

int i; // вспомогательная переменная

int k; // искомое значение

int INDEXES[50]; // искомый массив позиций вхождения числа k

int n; // количество найденных позиций или количество элементов в массиве INDEXES

// ввод массива M

// ...

// ввод числа k

// ...

// поиск числа k в массиве M и одновременное формирование массива INDEXES

n = 0;

for (i=0; i<50; i++)

    if (k==M[i])

    {

        // число найдено

        n++;

        INDEXES[n-1] = i;

    }

// вывод результата в listBox1

listBox1->Items->Clear();

for (i=0; i<n; i++)

    listBox1->Items->Add(INDEXES[i].ToString());