

Числовые массивы в языке программирования C++

АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

Цели и задачи лекции

Цель лекции – изучить использование массивов в языке программирования C++

Теоретическая часть

В языке программирования C заложены средства для задания последовательностей упорядоченных данных. Такие последовательности называются массивами. В массивах должны быть упорядочены данные одного и того же типа. В данной лабораторной работе будут рассматриваться массивы с целыми и вещественными типами данных, т.е. типы **int**, **float** или **double**.

Теоретическая часть

Массивы данных могут быть одномерными (векторами размера $1 \times n$ или $n \times 1$), двумерными (матрицами размера $n \times m$) или многомерными (размера $n \times m \times p \dots$). В частности, для векторов и матриц в приведенной записи первый индекс означает количество строк, а второй (число или буква) – это количество столбцов.

Теоретическая часть

Для названия массива может быть использована переменная, состоящая из букв (буквы), букв с цифрами, букв с цифрами и знаком подчеркивания и т.д. в соответствии с правилами объявления переменных, принятых в языке C++. Если размерность массива меньше, чем требуется, то компилятор не выдаст сообщения об ошибке. Выход за границы массивов должен следить только сам программист.

Одномерные массивы

Одномерный массив – это список связанных однотипных переменных.

Общая форма записи одномерного массива:

тип имя_массива[размер];

В приведенной записи элемент тип объявляет базовый тип массива. Количество элементов, которые будут храниться в массиве с именем имя_массива, определяется элементом размер.

Одномерные массивы

В языке C++ индексация массива начинается с нуля. Например, если размер массива определен величиной 9, то в массиве можно хранить 10 элементов с индексацией 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Доступ к отдельному элементу массива осуществляется с помощью индекса. Индекс описывает позицию элемента внутри массива.

Одномерные массивы

Все массивы занимают смежные ячейки памяти, т.е. элементы массива в памяти расположены последовательно друг за другом. Ячейка памяти с наименьшим адресом относится к первому элементу массива, а с наибольшим – к последнему.

Одномерные массивы

Для одномерных массивов общий размер массива в байтах вычисляется по формуле:

всего байт = размер типа в байтах * количество элементов

В языке C++ нельзя присвоить один массив другому. Для передачи элементов одного массива другому необходимо выполнить присвоение поэлементно.

Двухмерные массивы, матрицы

Двухмерный массив представляет собой список одномерных массивов.

Общая форма записи двухмерного массива:

тип имя_массива[размер1] [размер2];

В приведенной записи размер1 означает количество строк двухмерного массива, а размер2 – количество столбцов.

Двухмерные массивы, матрицы

В двухмерном массиве позиция любого элемента определяется двумя индексами. Индексы каждого из размеров массива начинаются с 0 (с нуля).

Место хранения для всех элементов массива определяется во время компиляции. Память, выделенная для хранения массива, используется в течение всего времени существования массива.

Двухмерные массивы, матрицы

Для двухмерных массивов общий размер массива в байтах вычисляется по формуле:

всего байт = число строк * число столбцов *
размер типа в байтах

Многомерные массивы

Общая форма записи многомерного массива:

тип имя_массива[размер1] [размер2]...
[размерN];

Индексация каждого размера начинается с нуля. Элементы многомерного массива располагаются в памяти в порядке возрастания самого правого индекса. Поэтому правый индекс будет изменяться быстрее, чем левый (левые).

Многомерные массивы

При обращении к многомерным массивам компьютер много времени затрачивает на вычисление адреса, так как при этом приходится учитывать значение каждого индекса. Следовательно, доступ к элементам многомерного массива происходит значительно медленнее, чем к элементам одномерного. В этой связи использование многомерных массивов встречается значительно реже, чем одномерных или двухмерных массивов.

Многомерные массивы

Для многомерных массивов общий размер многомерного массива в байтах вычисляется по формуле:

**всего байт = размер1* размер2*...* размерN *размер
типа в байтах**

Очевидно, многомерные массивы способны занять большой объем памяти, а программа, которая их использует, может очень быстро столкнуться с проблемой нехватки памяти.

Многомерные массивы

Для определения размера типа в байтах применяется функция `sizeof()`, которая возвращает целое число. Например, `sizeof(float)`.

Инициализация массивов

Инициализация массивов

В языке C массивы при объявлении можно инициализировать.

Общая форма инициализации массива:

```
тип    имя_массива[размер1] * [размерN] =  
{список_значений};
```

Инициализация массивов

В список_значений входят константы, разделенных запятыми. Типы констант должны быть совместимыми с типом массива.

Пример инициализации одномерного массива:

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

При этом $A[0] = 1$, $A[1] = 2$ и т.д.

Инициализация массивов

При инициализации многомерного массива для улучшения наглядности элементы инициализации каждого измерения можно заключать в фигурные скобки.

Инициализация массивов

Пример инициализации двумерного массива:

```
int MN[3][4] = {  
    {1, 2, 3, 4},  
    {5, 6, 7, 8},  
    {9, 10, 11, 12}  
};
```

Инициализация массивов

Массив `MN[3][4]` – это матрица, у которой 3 строки и 4 столбца.

Для многомерных массивов инициализацию можно также проводить с указанием номера инициализируемого элемента.

Инициализация массивов

Пример инициализации трехмерного массива:

```
int XYZ[2][3][4] = {  
  { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} },  
  { {13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} }  
};
```

Инициализация массивов

Как видно, массив XYZ содержит два блока, каждый из которых есть матрица размера 3 x 4, т.е. 3 строки и 4 столбца.

В языке C возможна инициализация безразмерных массивов. Например, для одномерного массива:

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

Инициализация массивов

В многомерном массиве размер самого левого измерения также можно не указывать. В частности, для инициализации массива MN[3][4] допустима следующая запись:

```
int MN[][4] = {  
    {1, 2, 3, 4},  
    {5, 6, 7, 8},  
    {9, 10, 11, 12}  
};
```


Инициализация массивов

При инициализации многомерных массивов необходимо указать все данные (размерности) за исключением крайней слева размерности. Это нужно для того, чтобы компилятор смог определить длину подмассивов, составляющих массив, и смог выделить необходимую память. Рассмотрим пример безразмерной инициализации для трехмерного массива целых чисел:

Инициализация массивов

```
int XYZ[][3][4] = {  
{  
{1, 2, 3, 4},  
{5, 6, 7, 8},  
{9, 10, 11, 12}  
},  
{  
{13, 14, 15, 16},
```

Инициализация массивов

```
{17, 18, 19, 20},  
{21, 22, 23, 24}  
}  
};
```

Инициализация массивов

Вывод трехмерного массива на консоль (дисплей) можно выполнить по следующей программе:

```
#include <stdio.h>
#include <conio.h>
int main (void) {
    int i, j, k;
    int XYZ[][3][4] = {
{ {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} }, // 1-й
```

Инициализация массивов

```
{ {13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24} } }; //
```

2-й

```
for (i = 0; i < 2; ++i) { printf("\n");  
    for (j = 0; j < 3; ++j) { printf("\n");  
        for (k = 0; k < 4; ++k)  
            printf(" %3d", XYZ[i][j][k]);  
        }  
    }
```

Инициализация массивов

```
printf("\n\n Press any key: ");  
_getch();  
return 0;  
}
```

Спасибо за внимание