Массивы

Что такое массив в C++?

Массив — это структура данных, которая содержит множество значений, относящихся к одному и тому же типу.

- Каждому элементу массива отводится одна ячейка памяти.
- Элементы одного массива занимают последовательно расположенные ячейки памяти.
- Все элементы имеют одно имя - имя массива и отличаются индексами – порядковыми номерами в массиве.
- Количество элементов в массиве называется его размером.

Массивы расположены в непрерывном участке памяти, доступ к которым осуществляется по индексу (индексам).

Массивы в C++ – это производные типы данных.

Можно объявлять переменные типов-массивов.

Объявление переменной-массива

Примеры определений переменных-массивов:

```
// Объявление без инициализации
int a[5];
// Объявление с инициализацией.
// Число элементов можно не указывать компилятор посчитает сам
char hello[] = {'h', 'e', 'l', 'l', 'o'};
// Объявление с меньшим количеством
// значений в списке инициализации
int a[5] = {1, 2, 3};
// Оставшиеся два элемента проинициализируются нулями
```

Указание размера массива

Размер массива должен быть константным выражением, т.е. компилятор должен уметь его вычислить во время компиляции

```
// Целочисленное выражение
int arr[10 + 5*3];
// Операция sizeof() выполняется во время
    компиляции!
int arr[sizeof(int)*8 + 1];
// Выражение с константными переменными
const int Size = 10;
int arr[Size/2];
// Константы, определённые посредством #define
#define MAX_PATH 255
char filename[MAX_PATH + 1];
```

Хранение данных массиве

Объявление массива:

int mass[12] ;

Размер массива:

sizeof(mass) == sizeof(int) * 12;

Адресация в массиве:

Обращение к элементам массива

происходит по индексу с помощью квадратных скобок

```
int a[10];
// Для компиляции индекс должен быть
// целочисленным выражением
a[i+j/2] = 3; // i и j имеют целочисленный тип
a[0] = 90; // обращение к первому элементу
a[9] = -90; // обращение к последнему элементу
// Для правильной работы индекс должен быть
// от 0 до размера массива минус 1
a[55] = 30; // обращение к несуществующему
   элементу!
a[-5] = 30; // обращение к несуществующему
   элементу!
```

Обращение к не существующим элементам массива

```
short a[2];
```

Виды массивов

Статический

тип_элемента имя_массива[количество_элементов] = {необязательные исходные значения};

```
int MyNumbers [5] = {34, 56, -21, 5002, 365};
```

Динамический

тип_элемента указатель на имя_массива = оператор выделения памяти тип_элементов [переменная];

```
int *p_darr = new int[num];
```

Одномерный или Многомерный

Статический массив

Статическими массивами (static array) – называются массивы, размер которых фиксируется программистом во время компиляции.

Такой массив не может принять больше данных, чем определил программист. Он также не может задействовать меньше памяти, если используется только наполовину или вообще не используется.

Динамический массив

Динамическими массивами (dynamic array) – называются массивы, размер которых может меняться во время исполнения программы.

```
int num = 10;
int mass[num]; // не верно!
```

Псевдодинамический массив

```
const int MAX_SIZE=100;//именованная константа
int mass[MAX_SIZE];
```

```
int num = 10;
int *mass = new int[num]; // Верно! Выделяем память.
delete []mass; // Освобождаем память.
```

Одномерные и многомерные массивы

Одномерный массив размерности 10.

```
int MyArray1[10];
```

20 одномерных массивов размерности 10.

```
int MyArray2[20][10];
```

30 двумерных массивов размерности 20*10.

```
int MyArray3[30][20][10];
```

Многомерные массивы в C++ рассматриваются как массивы, элементами которых являются массивы.

Многомерный массив это массив, элементами которого служат массивы. Например, массив с описанием int a[4][5] – это массив из 4 указателей типа int*, которые содержат адреса одномерных массивов из 5 целых элементов.

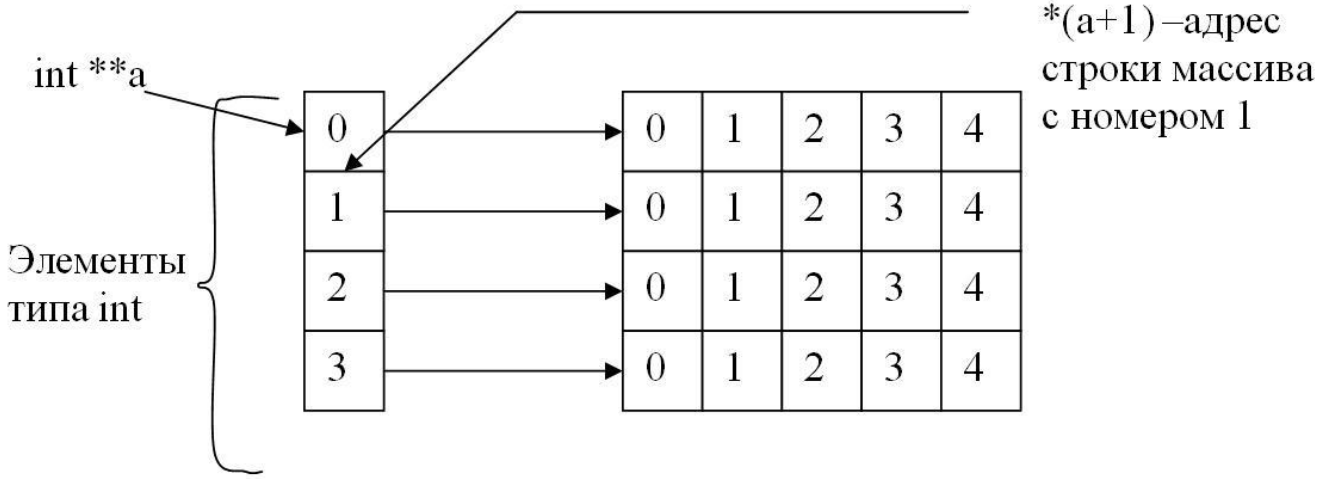
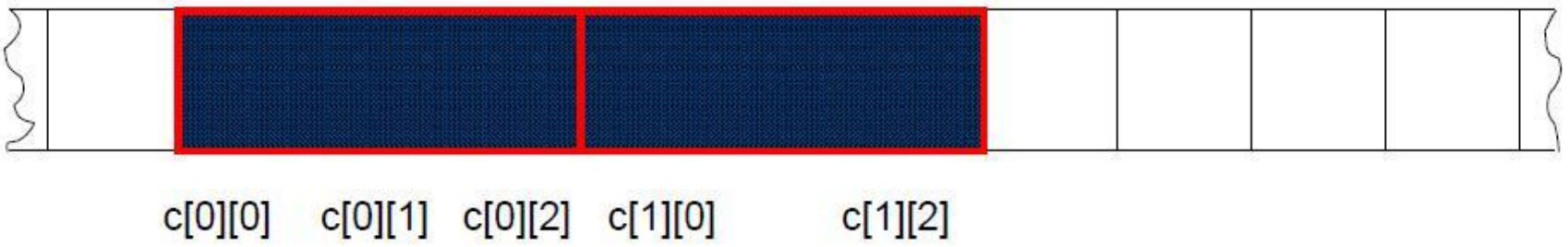


Рис . Организация многомерных массивов.

Многомерный массив также занимает непрерывный участок памяти

```
char c[2][3];
```

c[0][0]	c[0][1]	c[0][2]
c[1][0]	c[1][1]	c[1][2]



Адресация и инициализация многомерных массивов

int MyArray[3][3][**3**]={**0,1,2**,3,4,5,6,7,8,9,10,11};

Начальные значения получают следующие элементы трёхмерного массива:

MyArray[0][0][**0**] == **0**
MyArray[0][0][**1**] == **1**
MyArray[0][0][**2**] == **2**
MyArray[0][1][0] == 3
MyArray[0][1][1] == 4
MyArray[0][1][2] == 5
MyArray[0][2][0] == 6
MyArray[0][2][1] == 7
MyArray[0][2][2] == 8
MyArray[1][0][0] == 9
MyArray[1][0][1] == 10
MyArray[1][0][2] == 11

```
int MyArray[3][3][3] = {
                    {{0,1}},
                    {{100},{200,210},{300}},
                    {{1000},{2000,2100},{3000,3100,3200}}
                      };
MyArray[0][0][0] == 0
MyArray[0][0][1] == 1
MyArray[1][0][0] == 100
MyArray[1][1][0] == 200
MyArray[1][1][1] == 210
MyArray[1][2][0] == 300
MyArray[2][0][0] == 1000
MyArray[2][1][0] == 2000
MyArray[2][1][1] == 2100
MyArray[2][2][0] == 3000
MyArray[2][2][1] == 3100
MyArray[2][2][2] == 3200
```

Обработка одномерных массивов

Перебор элементов массива характеризуется:

- направлением перебора;
- количеством одновременно обрабатываемых элементов;
- характером изменения индексов.

По направлению перебора массивы обрабатывают:

- слева направо (от начала массива к его концу);
- справа налево (от конца массива к началу);
- от обоих концов к середине.

Индексы могут меняться:

- линейно (с постоянным шагом);
- нелинейно (с переменным шагом).

Присвоение 2011 четвертому элементу:

```
int MyNumbers[10];
MyNumbers [3] = 2011;
```

Перебор массива по одному элементу

Слева направо с шагом 1, используя цикл с параметром:

```
for(int I=0;I<n;I++){обработка a[I];}
```

Слева направо с шагом отличным от 1, используя цикл с параметром:

```
for (int I=0;I<n;I+=step){обработка a[I];}
```

Справа налево с шагом 1, используя цикл с параметром:

```
for(int I=n-1;I>=0;I--){обработка a[I];}
```

Справа налево с шагом отличным от 1, используя цикл с параметром:

```
for (int I=n-1;I>=0;I-=step){обработка a[I];}
```

Использование генератора случайных чисел для формирования массива.

```
int main(int argc, char* argv[])
{
int a[100];
int n;
cout<<"\nEnter the size of array:";
cin>>n;
for(int I=0;I<n;I++)
{a[I]=rand()%100-50; // случайные числа от -50 до 50
cout<<a[I]<<" "<<endl;
}
        system("pause");
        return 0;
}
```

Найти максимальный элемент массива

```
#include <clx.h>
#pragma hdrstop
#include<iostream.h>
#include<stdlib.h>
#pragma argsused

int main(int argc, char* argv[])
{
int a[100];
int n;
cout<<"\nEnter the size of array:";cin>>n;
for(int I=0;I<n;I++){a[I]=rand()%100-50; cout<<a[I]<<" ";}
int max=a[0];
for(int I=1;I<n;I++){ if (a[I]>max)max=a[I];}
cout<<"\nMax="<<max << endl;
      system("pause");
      return 0;
}
```

Классы задач по обработке массивов

К задачам 1 класса относятся задачи, в которых выполняется однотипная обработка всех или указанных элементов массива.

К задачам 2 класса относятся задачи, в которых изменяется порядок следования элементов массива.

К задачам 3 класса относятся задачи, в которых выполняется обработка нескольких массивов или подмассивов одного массива. Массивы могут обрабатываться по одной схеме – синхронная обработка или по разным схемам – асинхронная обработка массивов.

К задачам 4 класса относятся задачи, в которых требуется отыскать первый элемент массива, совпадающий с заданным значением – поисковые задачи в массиве.

Задачи 1-ого класса

нахождение максимального элемента массива или среднего арифметического массива

```
#include<iostream.h>
#include<stdlib.h>
void main()
{
int a[100];
int n;
cout<<”\nEnter the size of array:”;cin>>n;
for(int I=0;I<n;I++){a[I]=rand()%100-50;cout<<a[I]<<” “;}
int Sum=0;
for(I=0;I<n;I++)
Sum+=a[I];
Cout<<"Среднее арифметическое= "  << Sum/n”;
}
```

Задачи 2-ого класса
обмен элементов внутри массива

Пример
Перевернуть массив.

```
for(int i=0,j=n-1;i<j;i++,j--)
{int r=a[i];
a[i]=a[j];
a[j]=r;}
```

Поменять местами пары элементов в массиве: 1и2, 3 и 4, 5 и

Задачи 3-ого класса
синхронная и асинхронная обработка массивов

Пример. Заданы два массива из n целых элементов. Получить массив c, где c[I]=a[I]+b[I].

```
For(int I=0;I<n;I++)c[I]=a[I]+b[I];
```

При асинхронной обработке массивов индекс каждого массива меняется по своей схеме.

В массиве целых чисел все отрицательные элементы перенести в начало массива.

```
int b[10];//вспомогательный массив
int i,j=0;
for(i=0;i<n;i++)
    if(a[i]<0){b[j]=a[i];j++;}//переписываем из a в b все отрицательные элементы
for(i=0;i<n;i++)
    if(a[i]>=0){b[j]=a[i];j++;}// переписываем из a в b все положительные элементы
for(i=0;i<n;i++) cout<<b[I]<<” “;
```

Удалить из массива все четные числа

```
int b[10];


    int i,j=0;
    for(i=0;i<n;i++){
        if(a[i]%2!=0){b[j]=a[i];j++;}
        for(i=0;i<j;i++){ cout<<b[i]<<" ";
            cout<<"\n"; };};
```

Задачи 4-ого класса
найти элемент по условию

Найти первое вхождение элемента К в массив целых чисел.

```
int k;
cout<<"\nK=?";cin>>k;
int ok=0;//признак найден элемент или нет
int i,nom;
for(i=0;i<n;i++)
    if(a[i]==k){ok=1;nom=i;break;}
if(ok==1)
        cout<<"\nnom="<<nom;
else
    cout<<"\n there is no such element!";
```

Сортировка массивов

Сортировка – это процесс перегруппировки заданного множества объектов в некотором установленном порядке.

Простые методы подразделяются на три основные категории:

- сортировка методом простого включения;
- сортировка методом простого выделения;
- сортировка методом простого обмена;

Сортировка методом простого включения (вставки).

```
int i,j,x;
    for(i=1;i<n;i++)
    {
    x=a[i];//запомнили элемент, который будем вставлять
    j=i-1;
    while(x<a[j] && j>=0)//поиск подходящего места
    {
        a[j+1]=a[j];//сдвиг вправо
        j--;
    }
    a[j+1]=x;//вставка элемента
    }
```

Сортировка методом простого выбора

```
int i,min,n_min,j;
    for(i=0;i<n-1;i++)
    {
        min=a[i];n_min=i;//поиск
минимального
        for(j=i+1;j<n;j++)
            if(a[j]<min){min=a[j];n_min=j;}
            a[n_min]=a[i];//обмен
            a[i]=min;
    }
```

Сортировка методом простого обмена

Сравниваются и меняются местами пары элементов, начиная с последнего. В результате самый маленький элемент массива оказывается самым левым элементом массива. Процесс повторяется с оставшимися элементами массива.

```
for(int i=1;i<n;i++)
for(int j=n-1;j>=i;j--)
if(a[j]<a[j-1])
{int r=a[j];a[j]=a[j-1];a[j-1]=r;}
}
```