

# Массивы

*Массив* – это упорядоченный набор однотипных элементов.

Массивы являются производными типами данных, создаваемыми из существующих типов данных.

**Объявление одномерного массива:**

`тип имя_массива[размерность];`

Количество элементов в массиве определяет размер массива и является **константным выражением**.

# Массивы

Не все компиляторы позволяют определять размер массива переменной величиной:

```
int n;  
cin >> n;  
float arr[n]; /* Неверно */
```

При создании **статического массива**, для указания его размера может использоваться только константа. Размер выделяемой памяти определяется на этапе компиляции и не может изменяться в процессе выполнения.

# Массивы

```
const int n=10;  
float arr[n]; /* Верно */
```

**Индекс массива** определяет используемый элемент массива и указывается в квадратных скобках после имени массива.

**Индекс массива** — это **целочисленное выражение**, значение которого может быть в диапазоне от 0 до значения, равного размерности, уменьшенной на 1.

arr[0]    arr[1]    ... arr[9]

# Массивы

Выход за границу массива – обращение к элементу массива, индексы которого выходят за указанные в объявлении массива пределы.

Для увеличения скорости работы программы не выполняется проверка, лежит ли индекс массива в указанных при объявлении пределах.

Такую проверку пришлось бы производить во время работы программы каждый раз при обращении к массиву с указанным индексом.

Если бы проверялась **допустимость** индекса массива, то программа работала бы медленнее.

Поэтому программист сам должен заботиться о том, чтобы индексы элемента массива лежали в пределах, указанных при его объявлении.

# Связь между указателями и массивами фиксированного размера

В Си понятие **массив** и **указатель** взаимосвязаны.

Имя массива, воспринимается как **адрес**, начиная с которого хранится массив. Этот адрес нельзя изменить, так как имя статического массива является *указателем-константой*.

Итак, *имя массива* – это адрес первого элемента массива (с индексом 0),

то есть для массива: `int a[10];`

*a* ~ **&a[0]**

Имя статического массива по определению имеет атрибут *const*, поэтому не может быть изменен, к нему не может быть применена операция инкремент: `a ++`

```
1 #include<iostream>
2 #include <stdlib.h>
3 using namespace std;
4 int main ()
5 { int A[] = {3, 5, 1, 6, 2, 4, 8, 3, 7, 2};
6   cout << "ukazatel na massiv =" << A;
7   cout<<endl;
8
9 }
```

F:\\_рърыртЕ\\_чрэ €ш ь\_яю\_т++\_2011\\_ЕшьхЕ\ яЁюуЁрьь ь ыхъЎш ь\22 |рёёшт\ ш

ukazatel na massiv =0x22fe78

-----  
Process exited with return value 0

Press any key to continue . . .

```

#include<iostream>
#include <stdlib.h>
using namespace std;
int main ()
{ const int n=10;
  int A[] = {3, 5, 1, 6, 2, 4, 7, 8, 9, 10};
  cout<<"elementi массива \n";
  for (int i=0; i<n; i++)
  {
    cout<<A[i]<<" ";
  }
  cout<<endl;
  system("pause");
}

```

имя массива указатель-константа, нельзя изменять

✖ cout<<A[i]<<" ";

Строка	Файл	Сообщение
1	D:\ковчег\Dev-Cpp\include\c++\3.4...	In file included from D:/ковчег/Dev-Cpp/include/c++/3.4.2/backward/iostream.h:31, fr...
1	D:\ковчег\Алутина\Программиров...	from D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\...
32:2	D:\ковчег\Dev-Cpp\include\c++\3.4...	#warning This file includes at least one deprecated or antiquated header. Please consider using o...
	D:\ковчег\Алутина\Программиров...	In function `int main()':
10	D:\ковчег\Алутина\Программиров...	ISO C++ forbids cast to non-reference type used as lvalue
10	D:\ковчег\Алутина\Программиров...	non-lvalue in assignment

# Связь между указателями и массивами фиксированного размера

В общем случае доступ к заданному элементу массива можно осуществлять двумя способами:

*имя\_массива*[номер элемента]

**a[3]** //привычный способ

ИЛИ

*\*(имя\_массива+номер элемента)*

**\*(a+3)** или **\*(3+a)** // через указатель

Например, обращение к элементу **a[i]** возможно как

**\*(a+i)** или **\*(i+a)**, а также **i[a]**

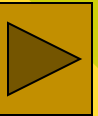


# Связь между указателями и массивами

Элемент массива  $a[i]$  есть элемент массива, на который указывает значение  $*(a+i)$ , где значение  $a$  является адресом элемента массива  $a[0]$ .

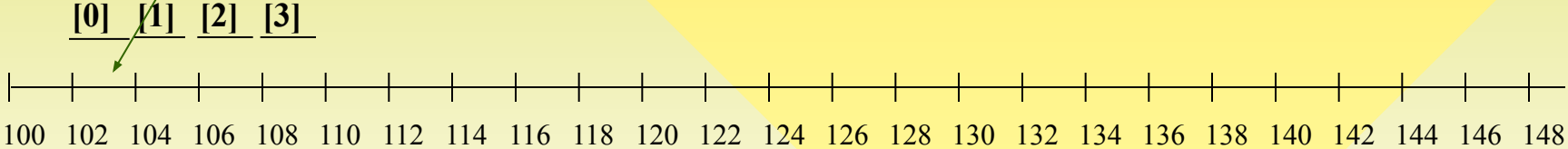
Выражение  $a+i$  является примером арифметических действий с указателями — целое значение  $i$  складывается со значением указателя, адресом первого элемента массива.

Значение этого выражения есть  $a$  плюс объем памяти, занимаемый  $i$  элементами массива.



```
int ma[4], i=0;  
ma[i] = i;  
*(ma + i) = i;  
i[ma] = i;
```

Имя	Адрес
ma	&ma[0]=102
ma+1	&ma[1]=104
ma+2	&ma[2]=106
...	...



```
#include<iostream>
#include <stdlib.h>
using namespace std;
int main ()
{
    int a[] = {3, 5, 1, 6, 2, 4, 8, 3, 7, 2};
    cout<<"elementi massiva \n ";
    for (int i=0; i<sizeof(a)/sizeof(int); i++)
        {
            cout<<i[a]<<" ";
        }
    cout<<endl;
    system("pause");
}
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям

```
elementi massiva
3 5 1 6 2 4 8 3 7 2
Для продолжения нажмите любую клавишу . . .
```

# Указатели на многомерные массивы

Многомерные массивы в языке Си – это массивы, элементами которых являются массивы. При объявлении таких массивов в памяти компьютера создается несколько различных объектов.

Пусть  $x$  – имя двумерного массива.

$$x \sim \&x[0][0]$$

Массивы хранятся записанными по строкам, элементы каждой строки занимают непрерывную область памяти.

Таким образом  $x[i]$  является указателем на строку массива  $x$  (подмассив).

$x[i]$  – адрес первого элемента  $i$ -ой строки,

т.е.  $(x+i)$

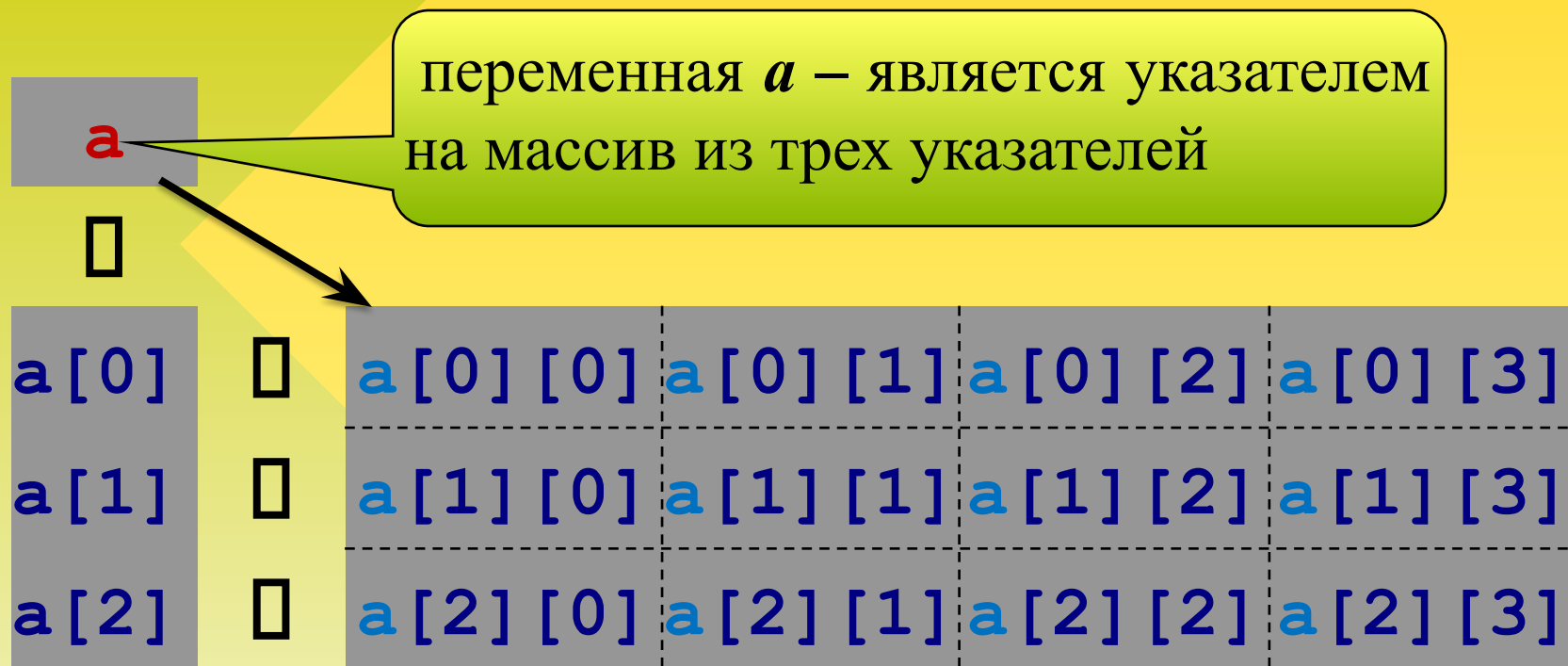
# Указатели на многомерные массивы

Например, при объявлении двумерного массива

```
int a[3][4] ;
```

в памяти выделяется участок для хранения значения переменной *a*, которая является указателем на массив из трех указателей *a[0]*, *a[1]*, *a[2]* на три строки.

# Указатели на многомерные массивы



каждый из трех указателей содержит адрес массива из четырех элементов типа *int*

# Связь между указателями и массивами

Для двумерного массива

*int a [3][4];*

число элементов в строке

обращение к *a [2][3]* можно заменить на *\*(a+2\*4+3)*

число столбцов

для *a [i][j]* заменить на *\*(a+i\*4+j)* или *\*(\*(a+i)+j)*

По индукции для ссылки на элемент трехмерного массива *x[i][j][k]* справедливо выражение: *\*(\*(\*(x+i)+j)+k)*

```

#include<iostream>
#include <stdlib.h>
using namespace std;
int main ()
{
int a[3][4];
for ( int i = 0; i < 3; i++ )
    for ( int j = 0; j < 4; j++ )
        *(a[i] + j) = 10*i+j;           // a[i][j] = 10*i+j;
for ( int i = 0; i < 3; i++ )
    for ( int j = 0; j < 4; j++ )
        cout << (*(a + i) + j) <<" ";
    cout << endl;
for ( int i = 0; i < 3; i++ )
    cout << (a + i) <<" ";
system("pause");
}

```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занят...

```

0      1      2      3      10      11      12      13      20      21      22      23
0x22ff10  0x22ff20  0x22ff30  Для продолжения нажмите л

```



```
#include<iostream h>
#include <stdlib.h>
using namespace std;
int main ()
{ const int n=10;
  int s=0, A[] = {3, 5, 1, 6, 2, 4, 8, 3, 7, 2};
  cout<<"elementi massiva \n ";
  int *p, *q;
  p=A; q=A+10;
  for (; p<q; p++)
    {
      cout<<*p<<" ";
    }
  cout<<endl;
  system("pause");
}
```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\mas

elementi massiva

3 5 1 6 2 4 8 3 7 2

Для продолжения нажмите любую клавишу . . .

# Динамические массивы

Часто возникают ситуации, когда заранее не известно, сколько объектов – чисел, строк текста и прочих данных будет хранить программа.

В этом случае используется динамическое выделение памяти, когда память занимает и освобождается в процессе исполнения программы.

Динамическое выделение памяти необходимо для **эффективного использования памяти** компьютера.

# Функции распределения памяти

Функция управления памятью	Действие	Заголовочный файл в VC++ 3.1	Заголовочный файл в DevC++
malloc()	Распределение	stdlib.h alloc.h	stdlib.h malloc.h
calloc()	Распределение		
realloc()	Перераспределение		
free()	Освобождение		

# Динамические массивы

Каждая из функций `malloc()` и `calloc()` резервирует непрерывный блок ячеек для хранения указанного объекта и возвращает бестиповый указатель на первую ячейку этого блока.

Функция *`free(указатель)`*; освобождает ранее резервированный блок и возвращает эти ячейки в динамическую область для последующего использования.

# Динамические массивы

При динамическом распределении памяти для массивов необходимо *сначала описать указатель на массив.*

I:

1. Описать указатель:

*тип \*имя\_указателя; //указатель на одномерный массив*

2. Затем присвоить указателю значение одной из функций:

*имя\_указателя = (тип \*) функция;*

II. Другой вариант описание с инициализацией:

*тип \*имя\_указателя = (тип \*) функция;*

# Динамические массивы

Прототипы функций:

*void \*calloc*( кол-во\_элементов, размер\_элементов) ;

*void \*malloc*( суммарный\_размер\_элементов) ;

# Динамическое размещение массивов с помощью функции: *calloc()*

Выделить память под одномерный массив *a[10]* из элементов типа *int* можно следующим образом:

```
#include <stdlib.h> //подключить заголовочный файл библиотеки
```

объявление указателя  
на одномерный массив

размерность  
массива

размерность  
элементов массива

```
{ int *a;  
  a=(int *) calloc(10, sizeof(int));
```

```
...
```

```
free(a);
```

```
}
```

ненужную для дальнейшей работы  
программы память необходимо освободить

Для определения необходимого объема памяти используется функция *sizeof*:

*sizeof* (выражение);      *sizeof* (тип);

# Массивы

Выделение памяти под одномерный динамический массив:

```
{ float *pf;  
  int n=30;  
  pf=(float *) malloc(n* sizeof(float));  
  ...  
  free (pf); // освобождение памяти:  
}
```



```

int main ()
{
    float **a;
    int n, m, i, j;
    scanf("%d %d", &n, &m);
    a=(float **)calloc(n, sizeof(float *));
    for (i=0; i<n; i++)
        a[i]=(float *)calloc(m, sizeof(float));
    for (i=0; i<n; i++)
    {
        for (j=0; j<m; j++)
            {a[i][j]=i+j;
            printf("%8.1f", a[i][j]);
            }
        printf("\n");
    }
    for (i=0; i<n; i++)
        free( a[i]);
    free(a);
}

```

необходимо  
одномерные  
массивы.

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям

4 5

0.0	1.0	2.0	3.0	4.0
1.0	2.0	3.0	4.0	5.0
2.0	3.0	4.0	5.0	6.0
3.0	4.0	5.0	6.0	7.0

Для продолжения нажмите любую клавишу . . .

# Массивы

Выделение памяти под двумерный динамический массив:

```
{ float **b;  
  int n=5, m=8;  
  b=(float **) malloc(n*sizeof(float*));  
  for (i=0; i<n; i++)  
    *(b+i) = (float *) malloc(m*sizeof(float));  
  ...  
  for (i=0; i<n; i++)  
    free(*(b+i));  
  free (b); // освобождение памяти:  
}
```

```

#include <stdio.h>
#include <stdlib.h>
int main ()
{ float **a;
  int n, m, i, j;
  scanf("%d %d", &n, &m);
  a=(float **)malloc(n*sizeof(float *));
  for (i=0; i<n; i++)
    a[i]=(float *)malloc(m *sizeof(float));
  for (i=0; i<n; i++)
  { for (j=0; j<m; j++)
    {a[i][j]=i+j;
     printf("%8.1f", a[i][j]);
    }
    printf("\n");
  }
  for (i=0; i<n; i++)
    free( a[i]);
  free(a);
system("pause");

```

Кол-во эл-тов умножить на размер эл-тов

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\динамическая Памят

6 8

0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0
1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0
2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0
3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0
5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0

Для продолжения нажмите любую клавишу . . .

# Массивы

Функции *new()* и *delete* встроенные функции C++, поэтому подключение дополнительной библиотеки не требуется.

```
{int n=20;
```

```
int *parray=new int [n]; //совместили объявление указателя и  
выделение памяти под массив
```

```
...
```

```
delete [] parray; }
```

# Массивы

*new <тип элементов массива>[число\_элементов]*

Операция *new* возвратит указатель, значением которого служит адрес первого элемента массива. При выделении динамической памяти для массива его размеры должны быть полностью определены:

*float (\*fp) [4]; // объявили указатель на массив*

~~*fp=new float [3][4]; //выделили память для двумерного массива*~~

~~Объявлен указатель на двумерный массив. В определении указателя круглые скобки обязательны. Указатель *fp* является средством доступа к участку динамической памяти с размерами  $3*4*sizeof(float)$  байтов.~~

~~Массив не имеет имя, указатель *fp* позволяет перемещаться по элементам массива. Для освобождения памяти:~~

~~*delete [ ] fp;*~~

~~освободит память, выделенную для двумерного массива, если *fp* адресует на его начало.~~

```

int main ()
{ int *ip, x=0, N, M, L;
  cout << "Kol-vo strok massiva: "; cin >> N;
  cout << "Kol-vo stolbhcov massiva: "; cin >> M;
  cout << "Kol-vo znacheniy v stolbhce: "; cin >> L;
  ip= new int [N*M*L]; //Резервирование места в куче для одномерного массива
  for (int i=0;i<N;i++)
    for (int j=0;j<M;j++)
      for (int k=0;k<L;k++)
        *(ip+i*(M*L)+j*L+k)=++x; //Заполнение массива значениями
  cout << "Poluchenniy massiv:" << endl;
  for (int i=0; i<N; i++)
  { cout << i << " stroka: " << endl;
    for (int j=0; j<M; j++)
    {cout << '\t' << j << " stolbezh: " << endl;
      cout << "\t\t znacheniya: " ;
      for (int k=0; k<L; k++)
        cout << *(ip+i*(M*L)+j*L+k) << " ";
      cout << endl;    }
    }
  delete [] ip;
}

```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к занятиям\massiv3.exe

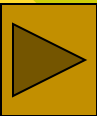
```

Kol-vo strok massiva: 2
Kol-vo stolbhcov massiva: 3
Kol-vo znacheniy v stolbhce: 5
Poluchenniy massiv:
0 stroka:
  0 stolbezh:
    znacheniya: 1 2 3 4 5
  1 stolbezh:
    znacheniya: 6 7 8 9 10
  2 stolbezh:
    znacheniya: 11 12 13 14 15
1 stroka:
  0 stolbezh:
    znacheniya: 16 17 18 19 20
  1 stolbezh:
    znacheniya: 21 22 23 24 25
  2 stolbezh:
    znacheniya: 26 27 28 29 30

```

Для продолжения нажмите любую клавишу . . .





```

#include<iostream>
#include <stdlib.h>
using namespace std;
int main ()
{
    int **a, i, j;
    a = new int*[3];
    for (i=0;i<3;i++ )
        a[i]=new int[4];

    for (i=0; i<3; i++)
    { for (j=0; j<4; j++)
      { *(*(a+i)+j) = 10*i+j;
        cout<< *(*(a+i)+j)<<"  ";
      }
      cout<< endl;
    }

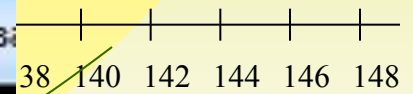
    delete (a);
    system("pause");
}

```

р двумерного  
атель на массив  
элементы

мя	Адрес
	&100

[3] [0] [1] [2] [3]



```

D:\ковчег\Алутина\Программирование\К занятиям\Задания к з
0 1 2 3
10 11 12 13
20 21 22 23
Для продолжения нажмите любую клавишу . . .

```

