Программирование на языке C++
Лекция 7

Массивы. Многомерные массивы

Статические многомерные массивы

Объявление массива

```
тип_элемента имя_массива[размер1][размер2]...[размерN];

int n[10][2];  // Двумерный массив
int a[10][5][7], b[2][2][2][2];
int c[5, 3];   // ОШИБКА

const int m = 4, n = 3;
double table[m][n]; // Только если m и n константы


#define SIZE 10   // Везде в коде SIZE заменять на 10
double array[SIZE][SIZE];
```

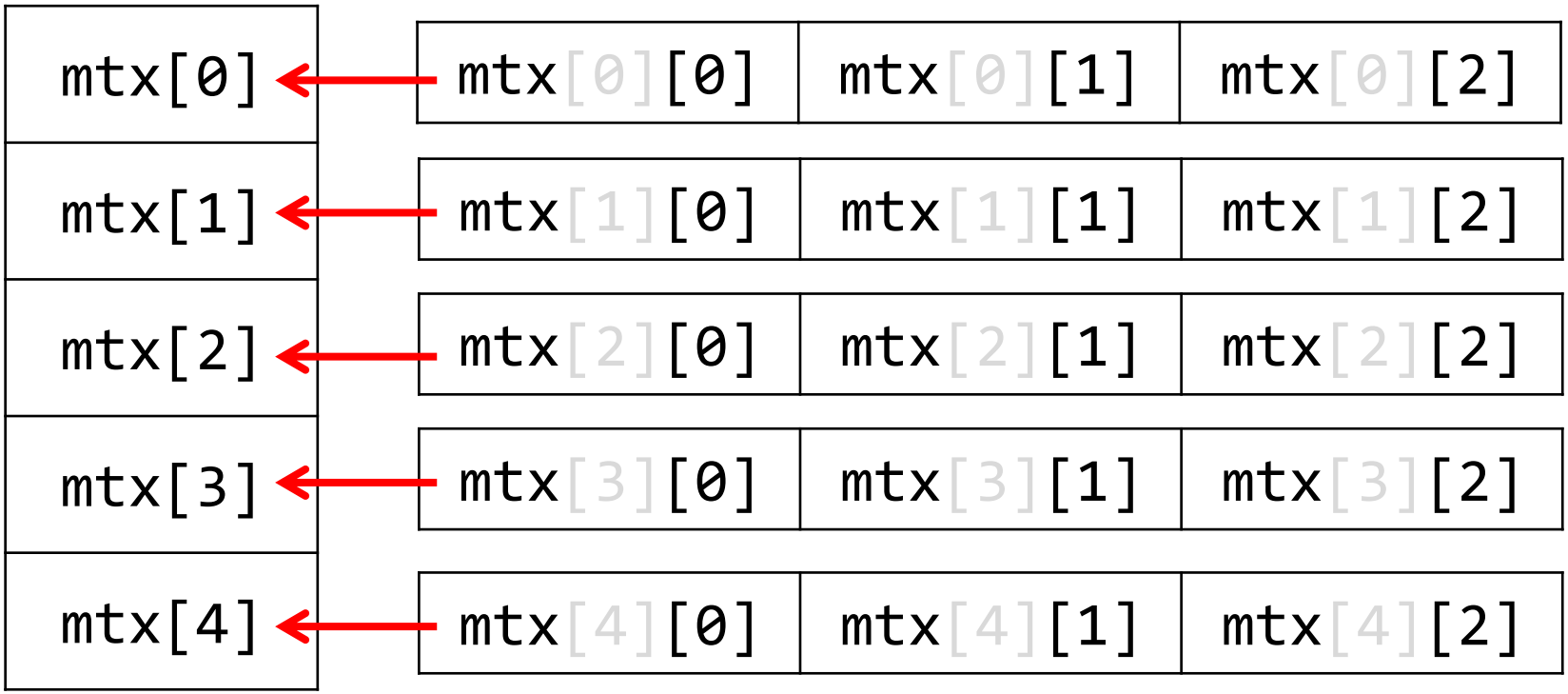
Двумерный массив как матрица

```
int mtx[5][3];
```

mtx[0][0]	mtx[0][1]	mtx[0][2]
mtx[1][0]	mtx[1][1]	mtx[1][2]
mtx[2][0]	mtx[2][1]	mtx[2][2]
mtx[3][0]	mtx[3][1]	mtx[3][2]
mtx[4][0]	mtx[4][1]	mtx[4][2]

Двумерный массив как массив массивов

```
int mtx[5][3];
```



Двумерный массив в оперативной памяти

```
int mtx[5][3];
```

[0][0]	[0][1]	[0][2]
[1][0]	[1][1]	[1][2]
[2][0]	[2][1]	[2][2]
[3][0]	[3][1]	[3][2]
[4][0]	[4][1]	[4][2]

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Инициализация массива

```
int trash[10][10]; // Значения массива не известны

// Каждая строка в отдельных скобочках
int dig[5][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9},
                 {10, 11, 12}, {13, 14, 15}};
int dig2[3][3][3] = {{{1, 2, 3}, {1, 2, 3}, {1, 2, 3}},
                     {{4, 5, 6}, {4, 5, 6}, {4, 5, 6}},
                     {{7, 8, 9}, {7, 8, 9}, {7, 8, 9}}};
int dig3[5][3] = {{1}, {2}, {3}, {}};

// Или можно скобочки не указывать
int dig4[5][3] = {1, 2, 3, 4, 5, 6, 7, 8, 9,
                  10, 11, 12, 13, 14, 15};
```

Инициализация массива

```
// Можно не указывать размер в первых скобочках
int dig[][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9},
                {10, 11, 12}, {13, 14, 15}};


// int dig[4][3] = {{1, 2, 3}, {4, 0, 0}, {0, 0, 0},
//                  {5, 0, 0}};
int dig[][3] = {{1, 2, 3}, {4}, {}, 5};


// В C++11 знак '=' можно не писать
double nums[2][2] {{1.5, 3.1}, {1, -1.5}};
```

Работа со статическим многомерным массивом

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

// Массив всегда начинается с нулевого индекса
int num = 1 + dig[4][0]; // num == 14
cout << dig[1][0];      // 4

dig[1][1] = 0; // {{…}, {4, 0, 6}, {…}, {…}, {…}}
dig[0][0] *= -1; // {{-1, 2, 3}, {…}, {…}, {…}, {…}}

// Выход за границы массива не проверяется
array[-1][0] = 3;
array[0][3] = 3;
```

Работа со статическим многомерным массивом

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

int dest[2][2];
dest = {{1, 2}, {4, 5}}; // Ошибка

dest = src; // Ошибка
dest[0] = src[0]; // Ошибка

// Копируем значения из src в dest
for (int i = 0; i < 2; i++)
    for (int j = 0; j < 2; j++)
        dest[i][j] = src[i][j];
```

Вывод массива на экран

```
#include <iostream>

int main(){
    int array[2][3] = {{1, 2, 3}, {4, 5, 6}};
    for (int i = 0; i < 2; i++){
        for (int j = 0; j < 2; j++)
            std::cout << array[i][j] << '\t';
        std::cout << '\n';
    }
    system("pause");
    return 0;
}
```

Вывод массива на экран

```
int array[2][3] = {{1, 2, 3}, {4, 5, 6}};
```

```
1    2    3
4    5    6
Для продолжения нажмите любую клавишу . . .
```

Поведение массива в коде

Имя массива в коде эквивалентно константному указателю на тип элемента массива.

Значение этого указателя зафиксировано на элементе массива с индексом нуль.

```
// Двумерный массив 4x3 элемента типа double
double array[4][3];
// Константный указатель на
// массив из 3х элементов типа double
double (*const p)[3] = array;
```

Поведение массива в коде

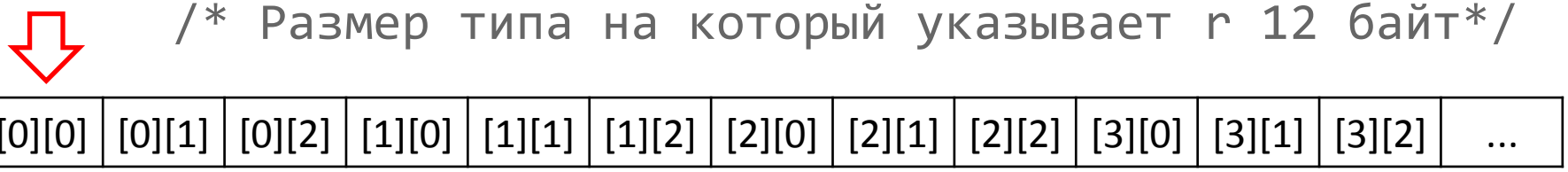
```
double array[4][3][10];

double (*const p)[3][10] = array;
```

Оператор []

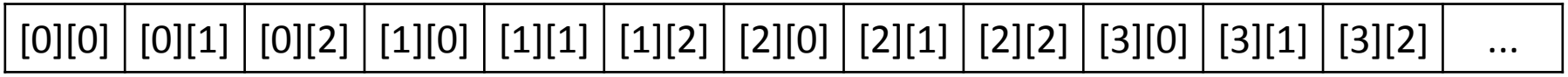
```
p[n] === *(p + n);

double arr[4][3];
int (*p)[3] = arr;
/* Размер типа на который указывает p 3 int */

cout << (p[1][1] == arr[1][1]);     // true
```

Оператор []

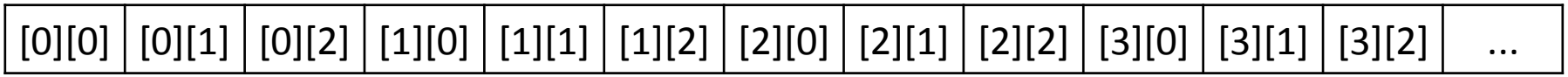
```
p[n] === *(p + n);
int arr[4][3];
int (*r)[3] = arr;
/* Размер типа на который указывает r 12 байт*/
```




```
int *c = r[1];
```




```
c[1] == r[1][1];
```



Массив как аргумент функции

```
#include <iostream>
int sum(int array[4][2], int size); // Прототип
int sum(int array[][2], int size);  // Прототип
int sum(int (*array)[2], int size); // Прототип

int main(){
    int arr[4][2] = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
    std::cout << sum(arr, 4);
}

int sum(int array[4][2], int size){
    int res = 0;
    size = size*2;
    while(--size >= 0) res += array[size/2][size%2];
    return res;
}
```

Массив как аргумент функции

```
#include <iostream>
int sum(int array[2], int size); // Прототип

int main(){
    int arr[4][2] = {{1, 2}, {3, 4}, {5, 6}, {7, 8}};
    std::cout << sum(arr[0], 4);
}

int sum(int array[2], int size){
    int res = 0;
    while(--size >= 0) res += array[size];
    return res;
}
```

Массив как аргумент функции

- В функцию массив передаётся по имени;
- Значения массива не копируются, копируется указатель;
- Размер первого измерения при передаче в функцию НЕ контролируется;
- Возврат массива из функции происходит тоже через указатель

Динамические многомерные массивы

Объявление I

```
// int array[10][5];
int **array = new int*[10];
for(int i=0; i<10; i++)
    array[i] = new int[5];

// Освобождаем память
for(int i=0; i<10; i++)
    delete[] array[i];
delete[] array;
```

Объявление II

```
// int array[10][5];
int **array = new int*[10];
array[0] = new int[10*5];
for(int i=1; i<10; i++)
    array[i] = array[i-1] + 5;

// Освобождаем память
delete[] array[0];
delete[] array;
```

Объявление III

```
// int array[10][5];
int (*array)[5] = new int[10][5];

// Освобождаем память
delete[] array;
```

Объявление III

```
// int array[10][5];
int (*array)[5] = new int[10][5];

// Освобождаем память
delete[] array;
```

Инициализация

```
// int array[2][5];
// Начиная с C++11 можно так:
int (*array)[5] = new int[2][5] {
                      {1, 2, 3, 4, 5}, {1}};

// Все скобочки обязательно должны быть
// тут ошибка
int (*array)[5] = new int[2][5] {
                      1, 2, 3, 4, 5, {1}};
```

Использование

Использование динамически объявленных массивов полностью идентично использованию статических массивов

После использования память необходимо освободить вручную с помощью оператора `delete[]`