Тема III. Базовые средства языка программирования *C/C++*

(Продолжение)

Лекция 10

Лекция 10:

§7. Массивы

Определение

Массив —конечная именованная последовательность однотипных величин.

§7.1. Одномерные массивы

- Форматы объявления одномерного массива:

```
тип имя_массива[размер_массива] [= {инициализаторы}];
```

```
тип имя_массива[] [= {инициализаторы}];
```

Пример:

```
float a[10];  //объявление массива из 10-ти вещественных
  чисел
int i[256];   //объявление массива из 256-ти целых чисел

unsigned short int d[]            //объявление и инициализация
        = {15, 255, 120, 0, 1}; //массива из 5-ти целых
  чисел
```

Примечания:

A. Элементы массива нумеруются с нуля.

B. При объявлении массива используются те же модификаторы, что и при объявлении простых переменных.

C. Элементы массива не могут быть функциями или элементами типа **`void`**.

D. Инициализирующие значения массива записывают в фигурные скобки {}.

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

int c[4] = {[3] = 4};          // c[0] = 0;
                               // c[1] = 0;
                               // c[2] = 0;
                               // c[3] = 4;
```

Доступ к элементам массива:

для доступа к элементу массива указывается имя массива и затем номер элемента в квадратных скобках, который называют индексом, например, **`b[4]`** (**`b`** — имя, **`4`** — индекс).

```
int oxen[SIZE] = {5,3,2,8};    // все в порядке
int yaks[SIZE];
yaks = oxen;          // не разрешено
yaks[SIZE] = oxen[SIZE];       // выход за пределы
        // диапазона
yaks [SIZE] = {5,3,2,8};       // не работает
```

Пример:

```
...
# define SIZE 8          // число эл-тов массива

int main()
{
int i, sum;
    int a[SIZE] = {3,4,5,6,4,3,2,1};  //объявление и инициализация
                                       //массива
    for (i = 0, sum = 0; i < SIZE; i++)
        sum += a[i];
    /* запись sum += a[i] эквивалентна sum = sum + a[i] */
    cout << "Сумма элементов: " << sum;
    return 0;
}
```

Результат выполнения программы:

☐ *Замечания!*

A. Память под массив выделяется на этапе компиляции.
B. Размерность массива может быть задана только целой положительной константой или константным выражением.
C. Размерность массивов рекомендуется задавать с помощью именованных констант (см. пример выше).
D. При обращении к элементам массива автоматический контроль выхода индекса за границу массива не производится, что может привести к ошибкам.

§7.2. Многомерные массивы

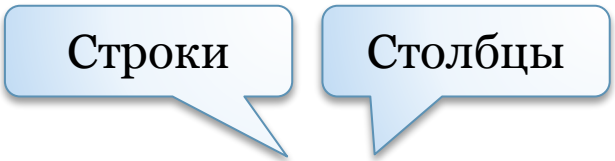
- Многомерные массивы объявляются указанием каждого измерения в отдельных квадратных скобках:

```
тип имя[размер_1][размер_2]...[размер_N] [={инициализаторы}];
```

- В качестве примера рассмотрим двумерный массив:

```
тип имя[число_строк][число_столбцов] [={инициализаторы}];
```

Пример:




```
int matr[5][4];    //объявление массива из 5-ти строк
                   //и 4-х столбцов
```

Примечания:

A. При инициализации многомерного массива он представляется либо как массив из массивов, либо задается общий список элементов в том порядке, в котором они располагаются в памяти:

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

или

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

B. В памяти многомерный массив располагается построчно (последовательно).

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

1_{00}	2_{01}
3_{10}	4_{11}
5_{20}	6_{21}

address	+ 1	+ 2	+ 3	+ 4	+ 5
1	**2**	**3**	**4**	**5**	**6**
00	01	10	11	20	21
Столбец 0	Столбец 1	Столбец 0	Столбец 1	Столбец 0	Столбец 1
Строка 0		Строка 1		Строка 2	

Примечания:

D. Для доступа к элементу многомерного массива следует указывать все его индексы:

```
matr[i][j];
*(matr[i] + j);
*(*(matr + i) + j);
```

§7.3. Динамические массивы
§7.3.1. Одномерные динамические массивы

- Создание динамического массива с помощью операции **new** (*C++*):

```
тип *имя = new тип [размер массива];
```

Пример:

```
int n = 100;                  //количество элементов массива
float *p = new float[n];      //объявление динамического массива
...                           //из n элементов с плавающей запятой
delete []p;                   //освобождение памяти
```

Доступ к элементам массива:

обращение к элементам динамического массива производится либо по индексу, либо через указатель:

```
float a, b;

a = p[5];       //обращение к 6-му элементу массива
                //с помощью индекса

b = *(p + 5); //обращение к 6-му элементу массива
                //через указатель
```

- Создание динамического массива с помощью функции **malloc** (*C*):

```
тип *имя =
     (приведение_типа*) malloc(размер_массива * размер_типа);
```

Пример:

```
int n = 100;                    //количество элементов массива
float *p = (float*) malloc(n * sizeof(float));
                                //объявление динамического массива
...                             //из n элементов с плавающей запятой
free (p);                       //освобождение памяти
```

Примечания:

A. Преобразование типа обязательно, поскольку функция **`malloc`** возвращает значение указателя типа **`void*`**.

B. Размер типа определять с помощью функции **`sizeof()`** необходимо, т.к. в некоторых случаях на некоторых ПК размер может отличаться от ожидаемого.

§7.3.2. Многомерные динамические массивы

- Для создания многомерного динамического массива необходимо

  указать в операции **new** все его размерности, например:

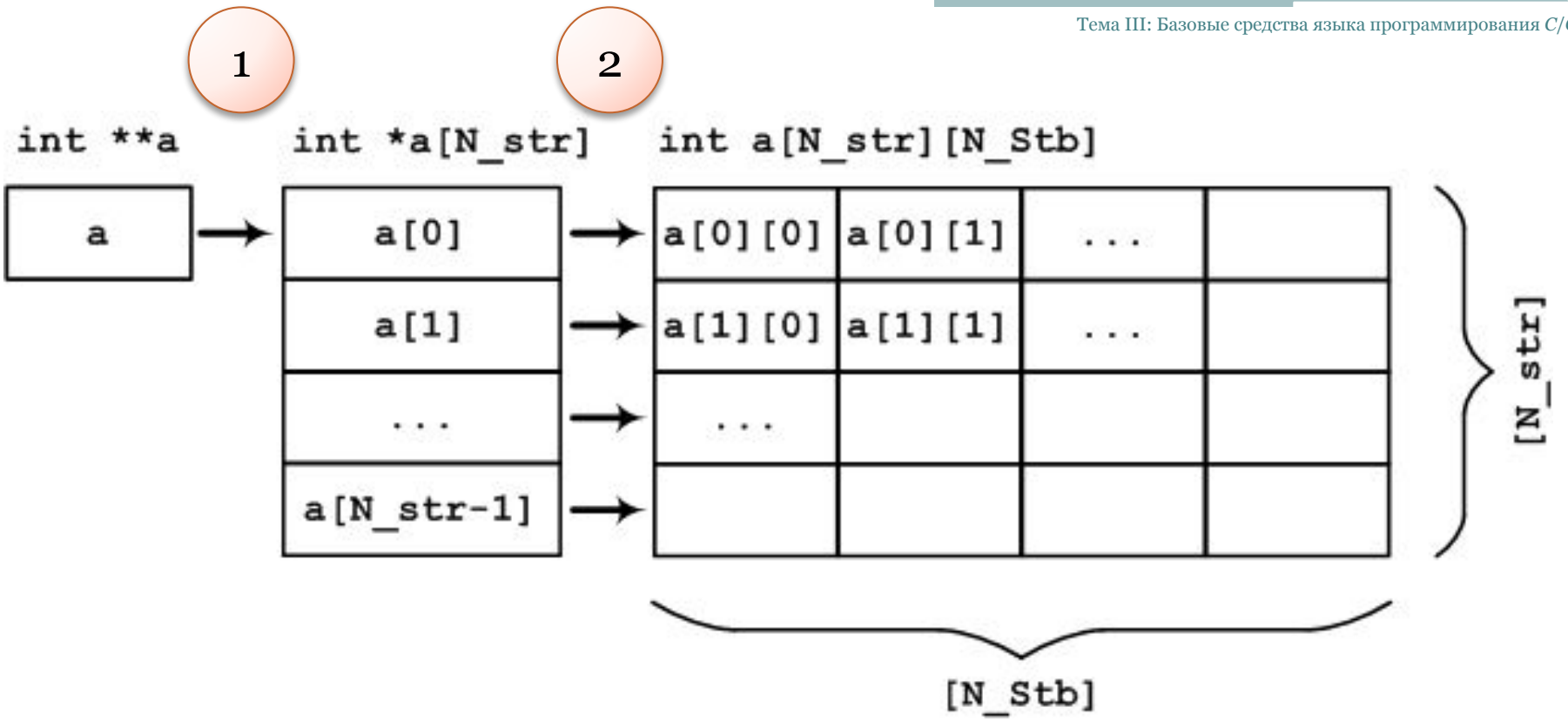
```
int N_str = 5;                                  //количество строк
int **m = (int**) new int[N_str][10]; //объявление двумерного
                                                //динамического массива
```

- Более универсальным (и безопасным) является способ создания массива, когда размерности задаются на этапе выполнения программы:

```
int N_str, N_stb;                                          // 1
cout << "Введите количество строк и столбцов: ";
cin  >> N_str >> N_stb;                                    // 2
int **a = new int *[N_str];                                // 3
for (int i = 0; i < N_str; i++)                            // 4
{
    a[i] = new int [N_stb];                                // 5
    /* Инициализация массива и обработка данных */
}
delete []a;                                                // 6
```

Здесь

6 — освобождение памяти.

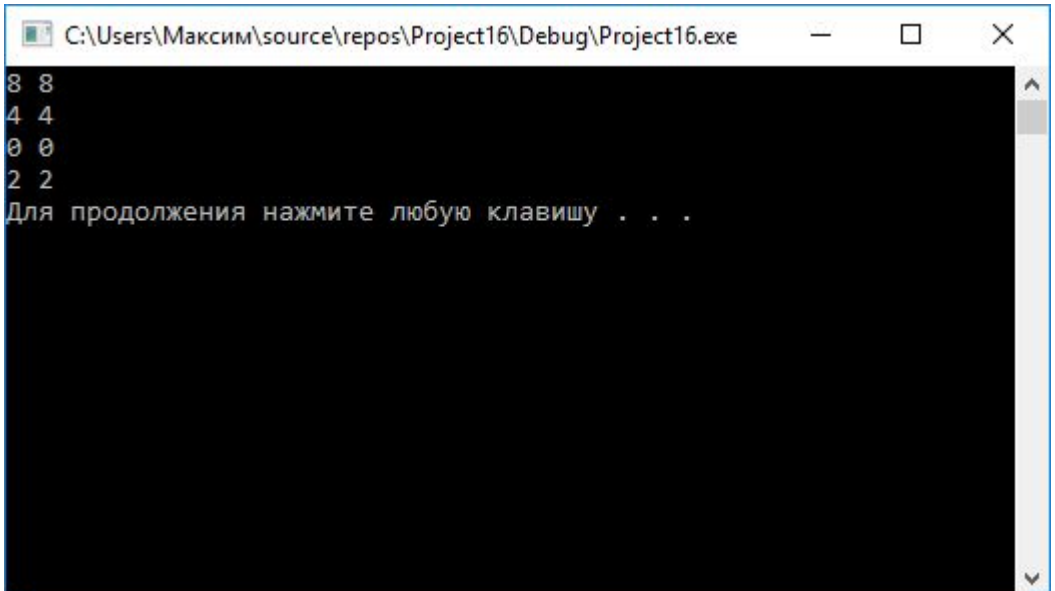


Шаг 1: объявляется указатель на массив указателей на **`int`** и выделяется память под массив указателей.

Шаг 2: в цикле выделяется память под каждую строку массива, каждому элементу массива указателей на строки присваивается адрес начала участка памяти, выделенного под строку.

```
#include <iostream>
using namespace std;

int main(void)
{
int ref[] = { 8, 4, 0, 2 };
int *ptr; int index;
for (index = 0, ptr = ref; index < 4; index++, ptr++)
printf("%d %d\n", ref[index], *ptr);
system("pause");
return 0;
}
```



? Вопросы:

Каковы значения *ptr и *(ptr + 2) в каждом из следующих случаев?

а)
```
int *ptr;
int torf [2] [2] = {12, 14, 16} ;
ptr = torf [0] ;
```

б)
```
int * ptr;
int fort[2] [2] = { {12}, {14, 16} } ;
ptr = fort[0];
```

? Вопросы:

Каковы значения **ptr и **(ptr + 1) в каждом из следующих случаев?

а)
```
int (*ptr)[2];
int torf [2] [2] = {12, 14, 16} ;
ptr = torf;
```

б)
```
int (*ptr)[2];
int fort[2] [2] = { {12}, {14, 16} } ;
ptr = fort;
```

? Вопросы:

Имеются следующие объявления:

```
float rootbeer[10], things[10][5], *pf, value = 2.2;
```

Укажите, какие из приведенных ниже операторов допустимы, а какие — нет:

```
а. rootbeer [2] = value;
б. scanf(“%f”, rootbeer);
в. rootbeer = value;
г. printf ("%f", rootbeer);
д. things [4][4] = rootbeer[3];
е. things[5] = rootbeer;
ж. pf = value;
з. pf = rootbeer;
```

§7.4. Символьные массивы (строки)

Определение

Строка представляет собой массив символов, заканчивающийся нуль-символом ‘\0’.

Примечание:

По положению нуль-символа определяется длина строки, т.е. длина символьного массива.

Форматы объявления символьного массива:

```
char имя_массива[размер_массива] [= "инициализатор"];
```

```
char имя_массива[] = "инициализатор";
```

Примеры:

1) Объявление символьного массива заданной длины и инициализация строковым литералом

```
char str[20] = "University"; //объявлен массив из 20-ти элементов
                             //с номерами от 0 до 19
```

Здесь под массив выделено 20 байт; символы массива записались в первые 11 байт, остальные ячейки памяти содержат нулевые значения:

2) Объявление символьного массива и инициализация строковым литералом без указания количества элементов массива

```
char str[] = "University"; //под массив выделено 11 байт
```

? Вопрос:

Что описывается следующими операторами:

```
char *str = "University";  ?
```

Примечания:

A. Если записать

```
char *str = "University";
```

то такая запись создаст указатель на строковую константу, но не символьный массив и не строковую переменную.

B. Указатель на константу удалить нельзя.