1 Массивы

1.1 Понятие массива.

Аппарат массивов имеется в любом алгоритмическом языке. Все языки реализуют одну и ту же концепцию. Каждый по-своему. Вот её положения.

1. Массив — это именованная пронумерованная совокупность однотипных элементов.

Важно: все элементы массива имеют ***одно и то же*** имя и принимают значения ***одного*** типа данных. К элементу можно обратиться, указав его имя и номер.

2. Массив должен быть объявлен в программе до первой ссылки на его элемент.
3. Объявление массива состоит из его имени, спецификации типа элемента и указания максимально допустимого числа элементов.
4. **Предполагается**, что массив занимает непрерывный блок памяти, достаточный для размещения всех элементов.
5. ***Программист гарантирует***, что программа в процессе обработки массива **не выйдет** за пределы этого блока.

Важно: Выход за объявленный в программе предел массива — это ошибка времени исполнения. **Ни одна система программирования не перехватывает эту ошибку.**

1.2 Объявление массива в С

```
<тип> <имя>[<число элементов>];
```

`тип` — любой базовый, производный или определённый пользователем.

`имя` — свободно выбираемый идентификатор.

`число элементов` — неотрицательное *константное выражение* типа **`int`**.

Элементы массива в Си **ВСЕГДА** индексируются **ЦЕЛЫМИ** числами от `0` до `<число элементов> - 1`.

Программист гарантирует (!!!), что программа в процессе исполнения не попытается обработать элемент с индексом, ***бóльшим*** или ***равным*** значению числа элементов.

Примеры

```
int arr[6];    /** создать массив с именем arr,
                   состоящий из шести элементов типа int.*/
float f_arr[5];
double d_arr[3];
```

1.3 Обработка объявления массива компилятором

1. Выделяется блок памяти, необходимый для размещения массива.

Например, для `arr` будет выделено `6*sizeof(int)` байт,

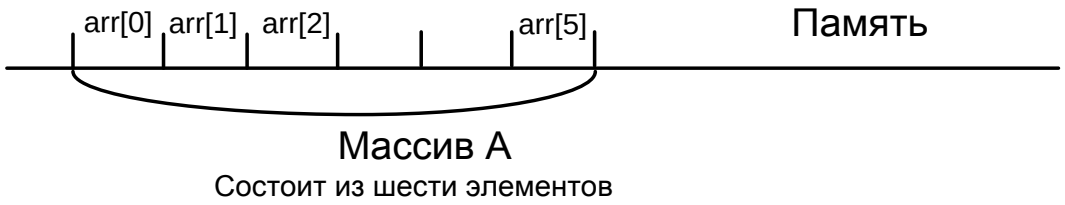

Рисунок 1- Размещение массива в памяти

для `f_arr – 5*sizeof(float)` байт,
для `d_arr – 3*sizeof(double)` байт.

2. Адрес младшего байта этого блока в дальнейшем считается именованной *адресной константой (константным указателем*[1]*).* Её *имя, совпадает с* объявленным *именем массива*. Значение этой именованной константы-указателя *не может быть изменено* в ходе исполнения программы.
3. При построении объектного кода любая ссылка на имя массива заменяется соответствующим адресом.

Объявленный так массив имеет **фиксированную** локализацию и **фиксированный** размер. Ни то, ни другое не могут изменяться в ходе исполнения программы.

1.4 Инициализация массива

Инициализация массива в Си **НЕ ВЫПОЛНЯЕТСЯ** автоматически. Его можно инициализировать только явно. Допускается несколько вариантов инициализации. В таблице 1 приведены примеры допустимых и некоторых недопустимых вариантов инициализации.

[1] *Указатель* – это имя, принимающее значения адресов оперативной памяти.

Таблица 1 – Варианты инициализации массивов в Си-программах

Объявление	**Созданный массив**
`int arr[5] = {1, 1, 1, 1, 1};`	1 1 1 1 1
`int arr[5] = {1, 2, 3};`	1 2 3 NULL NULL
`int arr[5] = {0};`	0 0 0 0 0
`int arr[] = {1, 3, 5, 7, 9};`	1 3 5 7 9 Размер массива определяется размером инициализатора.
`int arr[5] = {1, 1, 1, 1, 1, 1};`	Некоторые компиляторы считают синтаксической ошибкой
`int arr[];`	Синтаксическая ошибка. Сообщение компилятора: …**`.array size missing in`** …

1.5 Обращение к элементу массива

Обратиться к элементу массива можно двумя способами:

по *индексу* или по *адресу*

Индекс элемента должен быть целым неотрицательным числом.

Первый элемент массива **ВСЕГДА** имеет индекс **0**. Индекс пятого элемента равен **4**. Индекс `n`-того — `n-1`.

1.5.1 Обращение к элементу по индексу

```
<имя массива>[<индекс элемента>]
```

Например:

```
int x, arr[5]={1, 2, 3, 4, 5};
x = arr[3];   // x примет значение 4
arr[0] = x;   // первый элемент массива arr примет значение 4
```

НЕОБХОДИМО ПОМНИТЬ

Индекс первого элемента **любого** массива **ВСЕГДА 0** (нуль).

Индекс последнего — **на единицу меньше числа элементов.**

Ни одна исполняющая система **не перехватывает** выход

за пределы памяти, выделенной под массив.

Следить за этим должен программист.

1.5.2 Обращение к элементу по адресу

Обращение к элементу массива по индексу – это *выражение*. Квадратные скобки в нём – *символ операции.* Эта операция формирует *адрес* указанного индексом элемента массива. Если обращение к элементу находится **в левой части оператора присваивания**, то по этому адресу *размещается* значение выражения, сформированное в правой части. Во всех других случаях *извлекается* значение, размещённое по этому адресу.

Например, выражение

```
x = arr[3]
```

эквивалентно выражению

```
x = *(arr + 3).
```

В этом выражении символ `'*'` обозначает операцию *разыменования*. Она применима только к *указателям*. Возвращает значение, **размещённое по адресу**, являющемуся значением её операнда. В нашем примере операнд имеет значение *адреса* четвёртого элемента массива. Разыменование возвратит *значение* этого элемента.

Реально и в первом, и во втором случаях выполняются следующие действия:

– вычисляется значение адреса первого байта четвёртого элемента массива `arr` вот так: `arr + 3*sizeof(int);`

– это значение интерпретируется как адрес значения типа `int;`

– извлекается значение, размещённое по этому адресу (операция *);

– это значение присваивается переменной `x.`

Аналогично обрабатывается и выражение `arr[3] = x.`

Таким образом, выражение `*(arr+i)` *эквивалентно* выражению `arr[i]` и может использоваться в программе для доступа к элементам массива *по адресу*.

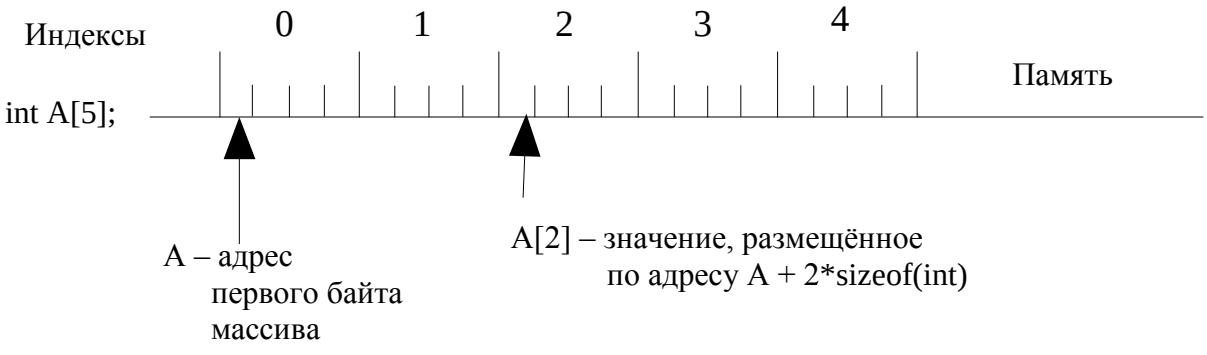

Рисунок 2 - Индексы и адреса элементов массива

1.6 Модель многомерного массива

Синтаксис языка Си определяет только одномерные массивы. Но, согласно определению, элементами массива могут быть массивы (однотипные!). Тогда можно, скажем, двумерный набор чисел (матрицу M×N) можно представить как одномерный массив из M *строк* – одномерных массивов, каждый из которых состоит из N чисел. Аналогично трёхмерный набор чисел M×N×P можно представить одномерным массивом P элементов – двумерных массивов и т.д.

Объявление двумерного массива:

```
<тип> <имя>[<число строк>][<число столбцов>];
```

Примеры:

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

В процессе обработки этого объявления (с инициализацией) компилятор зарезервирует блок памяти размером `15*sizeof(int)` байтов. Адрес первого байта этого блока будет считаться в дальнейшем значением именованной адресной константы `mtr`. Он же является адресом первой строки матрицы, т.е., первого блока байтов размером `5*sizeof(int)` байтов. То есть, выражения `mtr` и `mtr[0]` возвращают одно и то же значение. Выражение `mtr[1]` возвратит адрес первого байта второй строки. То есть, квадратные скобки здесь обозначают операцию сдвига в адресном пространстве на длину строки нашей матрицы: `mtr[i] эквивалентно *(mtr + i)`.

Выражение `mtr[0][0]` возвратит `1` – значение первого элемента первой строки, а выражение `mtr[0][1]` – значение второго элемента первой строки.

В общем случае

Выражение `mtr[i][j]` возвращает значение `j`-того элемента `i`-той строки и эквивалентно выражению `*(*(mtr + i) + j)`.