

## Методические указания

### Тематическое занятие 16 **Массивы указателей.**

#### Содержание

<b>Создание и использование массива указателей.....</b>	<b>1</b>
<i>Массив индексов.....</i>	<i>1</i>
<i>Создание массива указателей .....</i>	<i>2</i>
<i>Доступ к элементам массива указателей через индексы.....</i>	<i>2</i>
<i>Доступ к значениям элементов исходных массивов .....</i>	<i>3</i>
<b>Адресная арифметика для массива указателей.....</b>	<b>4</b>
<i>Доступ к элементам массива указателей .....</i>	<i>4</i>
<i>Доступ к адресам элементов исходных массивов.....</i>	<i>4</i>
<i>Доступ к значениям элементов исходных массивов .....</i>	<i>4</i>
<i>Применения адресной арифметики для массива указателей.....</i>	<i>5</i>
<i>Общая схема доступа к элементам исходных массивов .....</i>	<i>6</i>
<b>Упражнения.....</b>	<b>6</b>
<i>Упражнение 16.1.....</i>	<i>6</i>
<i>Упражнение 16.2.....</i>	<i>6</i>

## Создание и использование массива указателей

### **Массив индексов**

Пусть имеется неупорядоченный массив целых чисел:

```
int a[5]={39, 76, 26, 14, 52};
```

a[0]	a[1]	a[2]	a[3]	a[4]
39	76	26	14	52

и поставлена задача отсортировать по возрастанию числа, находящиеся в массиве *a*, но при этом запрещено переставлять элементы массива *a* и копировать их.

Кажется, что самым простым решением является создание нового массива, в котором будут храниться индексы элементов исходного массива в порядке возрастания значений этих элементов:

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

Используя массив индексов `ind` можно вывести на экран исходный массив `a`, отсортированный по возрастанию:

```
for (i=0; i<5; ++i)
    printf("%d\n",a[ind[i]]);
```

ind[0]	ind[1]	ind[2]	ind[3]	ind[4]
3	2	0	4	1
a[3]	a[2]	a[0]	a[4]	a[1]
14	26	39	52	76

## Создание массива указателей

Рассмотрим пример, когда имеются несколько массивов с различным количеством элементов, все элементы этих массивов нужно расположить в общей последовательности, упорядоченной по возрастанию:

```
int a[5]={39, 76, 26, 14, 52};
int b[3]={67, 15, 48};
int c[4]={82, 23, 59, 61};
```

В этом случае при использовании массива индексов необходимо различать, какому из исходных массивов принадлежит данный индекс. Это усложняет задачу сортировки.

В подобных ситуациях часто используют иной подход – создают общий **массив указателей**, в котором хранятся адреса всех элементов исходных массивов.

В рассматриваемом примере определим количество элементов общего массива указателей:  $5+3+4=12$ . Объявим массив с названием `arp` (от «**a**rray of **p**ointers»), элементы которого являются указателями на переменные типа `int`:

```
int *arp[12];
```

Заполним этот массив адресами элементов массивов `a`, `b` и `c` в порядке возрастания их значений:

arp[0]	arp[1]	arp[2]	arp[3]	arp[4]	arp[5]	arp[6]	arp[7]	arp[8]	arp[9]	arp[10]	arp[11]
&a[3]	&b[1]	&c[1]	&a[2]	&a[0]	&b[2]	&a[4]	&c[2]	&c[3]	&b[0]	&a[1]	&c[0]
14	15	23	26	39	48	52	59	61	67	76	82

Теперь в массиве указателей адреса упорядочены по возрастанию значений тех элементов массивов, на которые они ссылаются. При этом не нарушен порядок исходных массивов, в них не произошло ни одной перестановки элементов.

## Доступ к элементам массива указателей через индексы

При заполнении массива указателей `arp` обращаться к его элементам можно несколькими способами. Самый простой из них – это доступ через индексы элементов массива `arp`:

```
arp[0] = &a[3];
arp[1] = &b[1];
arp[2] = &c[1];
arp[3] = &a[2];
...
```

Элементами массива `arp` являются указатели, которые принимают значения адресов ячеек элементов массивов `a`, `b` и `c`.

Проиллюстрируем рассматриваемый пример схемой ячеек памяти с их конкретными адресами:

	<b>a[0]</b>	<b>a[1]</b>	<b>a[2]</b>	<b>a[3]</b>	<b>a[4]</b>
<b>a:</b>	<b>39</b>	<b>76</b>	<b>26</b>	<b>14</b>	<b>52</b>
Адреса:	0BF2	0BF4	0BF6	0BF8	0BFA

	<b>b[0]</b>	<b>b[1]</b>	<b>b[2]</b>
<b>b:</b>	<b>67</b>	<b>15</b>	<b>48</b>
Адреса:	2D74	2D76	2D78

	<b>c[0]</b>	<b>c[1]</b>	<b>c[2]</b>	<b>c[3]</b>
<b>c:</b>	<b>82</b>	<b>23</b>	<b>59</b>	<b>61</b>
Адреса:	1A5C	1A5E	1A60	1A62

	<b>arp[0]</b>	<b>arp[1]</b>	<b>arp[2]</b>	<b>arp[3]</b>	<b>arp[4]</b>	<b>arp[5]</b>
<b>arp:</b>	<b>0BF8</b>	<b>2D76</b>	<b>1A5E</b>	<b>0BF6</b>	<b>0BF2</b>	<b>2D78</b>
Адреса:	B8E4	B8EC	B8F4	B8FC	B904	B90C
	<b>arp[6]</b>	<b>arp[7]</b>	<b>arp[8]</b>	<b>arp[9]</b>	<b>arp[10]</b>	<b>arp[11]</b>
	<b>0BFA</b>	<b>1A60</b>	<b>1A62</b>	<b>2D74</b>	<b>0BF4</b>	<b>1A5C</b>
Адреса:	B914	B91C	B924	B92C	B934	B93C

Следует заметить, что в данном упрощенном примере для хранения значения переменной типа `int` используется 2 байта, а для хранения указателя (адреса памяти) – 8 байт. Однако реальная программа должна корректно выполняться в разных вычислительных системах, где количество байт для хранения значений переменных и адресов может быть различным.

## Доступ к значениям элементов исходных массивов

Для получения доступа к значениям исходных массивов `a`, `b` и `c` (на которые ссылаются элементы массива указателей `arp`) достаточно применить операцию раскрытия ссылки (разыменования) к адресу, хранящемуся в ячейке массива указателей `arp`. При этом получаются следующие тождественные выражения, которые возвращают одинаковые значения:

```
*arp[0] == *(&a[3]) == *a[3] == a[3] == 14
*arp[1] == *(&b[1]) == *b[1] == b[1] == 15
*arp[2] == *(&c[1]) == *c[1] == c[1] == 23
*arp[3] == *(&a[2]) == *a[2] == a[2] == 26
...
```

Таким образом, через массив указателей `arp` можно изменять значения исходных массивов `a`, `b` и `c`, например:

```
*arp[5] = 44; /* значение b[2] изменится с 48 на 44 */
```

# Адресная арифметика для массива указателей

## Доступ к элементам массива указателей

Другим способом доступа к элементам массива указателей `arp` является использование операции разыменования и адресной арифметики для указателя `arp`:

```
*arp      = &a[3];
*(arp+1)  = &b[1];
*(arp+2)  = &c[1];
*(arp+3)  = &a[2];
...
```

Здесь имя массива `arp` является синонимом указателя на первый элемент массива, который сам является указателем на четвертый элемент массива `a`. Таким образом, `arp` – синоним указателя на указатель и содержит адрес `&arp[0]` 1-го из 12-и элементов массива. Его разыменование `*arp` даст значение этого адреса (которое само является адресом 4-го элемента массива `a`):

```
*arp == *(&arp[0]) == *&arp[0] == arp[0] == &a[3] == 0BF8
```

Применение адресной арифметики к идентификатору `arp` позволяет перемещаться по массиву указателей `arp`. Например, выражение `arp+1` ссылается на 2-й элемент массива указателей `arp`:

```
arp+1 == &arp[1]
```

Его разыменование даст значение 2-го из 12-и элементов массива `arp`, которое является адресом 2-го элемента массива `b`)

```
*(arp+1) == *(&arp[1]) == *&arp[1] == arp[1] == &b[1] == 2D76
```

## Доступ к адресам элементов исходных массивов

Поскольку элементами массива указателей `arp` являются адреса элементов исходных массивов (`a`, `b` или `c`), то к ним также можно применять адресную арифметику. При этом перемещение будет происходить по одному из исходных массивов `a`, `b` или `c`.

Например:

```
*arp+1 == (*arp)+1 == arp[0]+1 == &a[3]+1 == &a[4] == 0BFA
```

```
*(arp+1)+1 == (*(arp+1))+1 == arp[1]+1 == &b[1]+1 == &b[2] == 2D78
```

## Доступ к значениям элементов исходных массивов

Разыменование этих указателей позволяет получить доступ к элементам исходных массивов `a`, `b` или `c`:

```
**arp == *(*arp) == *arp[0] == *&a[3] == a[3] == 14
```

```
*(*arp+1) == *(arp[1]) == *arp[1] == *&b[1] == b[1] == 15
```

```
*(*arp+1) == *(arp[0]+1) == *(&a[3]+1) == *&a[4] == a[4] == 52
```

```
*(*arp+1)+1 == *(arp[1]+1) == *(&b[1]+1) == *&b[2] == b[2] == 48
```

Общая формула такого двойного разыменования для идентификатора `arp` массива указателей из `N` элементов:

**`*(*(arp+n)+i) == *(arp[n]+i),`**

где  $0 \leq n < N$ ,

**`i`** – смещение индекса в одном из исходных массивов.

Следует обратить внимание, что в рассматриваемом примере изменение переменной `n` на единицу приводит к смещению в памяти на 8 байт, поскольку происходит переход к соседней ячейке с адресом. А изменение переменной `i` на единицу приводит к смещению на 2 байта, поскольку переход происходит на соседнюю ячейку, хранящую значение переменной типа `int`.

Конечно, величина смещения в байтах может быть различной в зависимости от конкретной вычислительной системы, на которой выполняется рассматриваемый программный код.

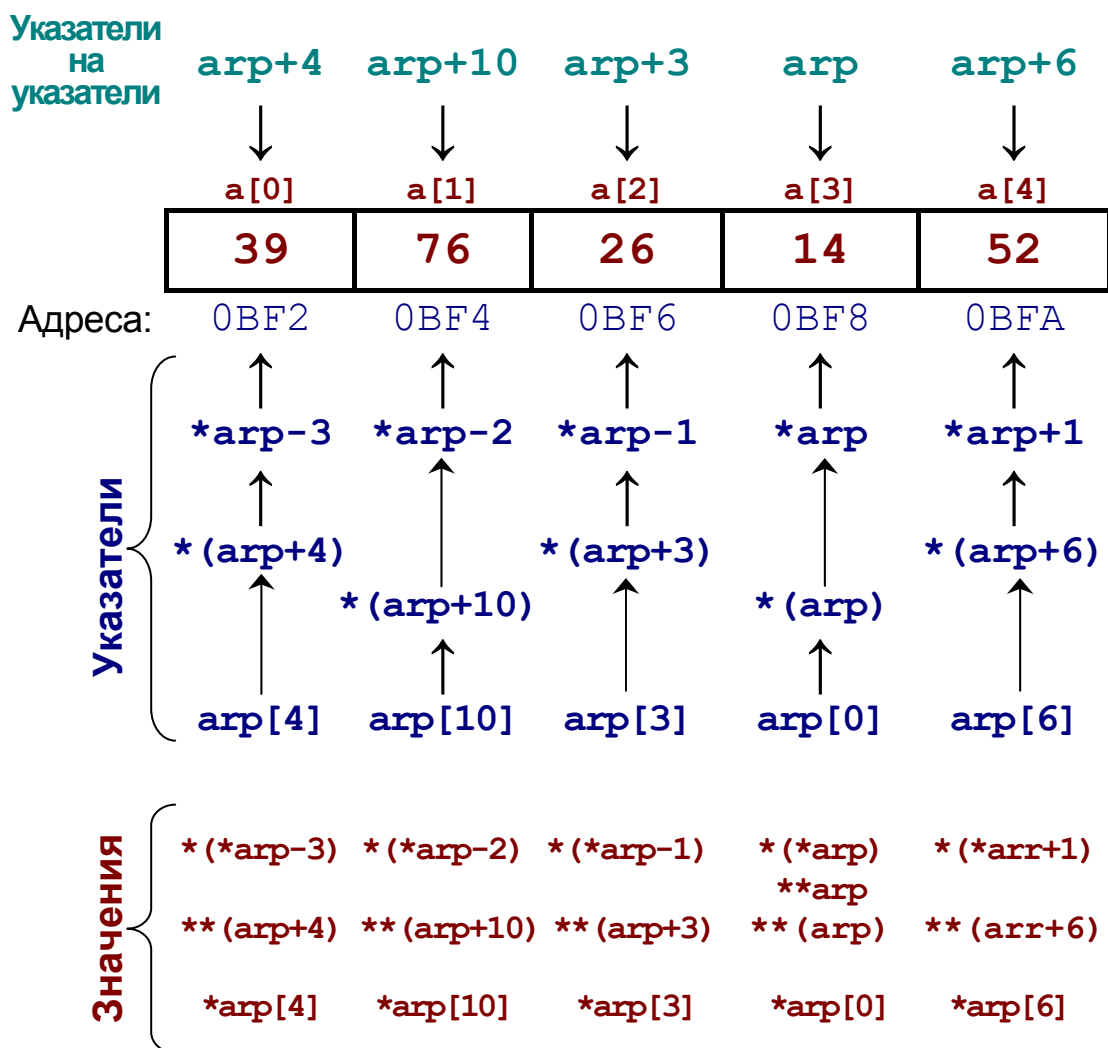
### **Применения адресной арифметики для массива указателей**

Применение операций адресной арифметики к идентификатору массива указателей `arp` позволяет перемещаться как по самому массиву указателей, так и по исходным массивам `a`, `b` и `c`:

Выражение	Значение выражения		
	в массиве указателей	в исходных массивах <code>a</code> , <code>b</code> и <code>c</code>	числовое
<code>arp</code>	адрес 1-го элемента <code>&amp;arp[0]</code>		<code>B8E4</code>
<code>arp+2</code>	адрес 3-го элемента <code>&amp;arp[2]</code>		<code>B8F4</code>
<code>*arp</code>	значение 1-го элемента <code>arp[0]</code>	адрес 4-го элемента <code>&amp;a[3]</code> массива <code>a</code>	<code>0BF8</code>
<code>*(arp+2)</code>	значение 3-го элемента <code>arp[2]</code>	адрес 2-го элемента <code>&amp;c[1]</code> массива <code>c</code>	<code>1A5E</code>
<code>*arp+1</code>	значение 7-го элемента <code>arp[6]</code>	адрес 5-го элемента <code>&amp;a[4]</code> массива <code>a</code>	<code>0BFA</code>
<code>*(arp+2)+1</code>	значение 8-го элемента <code>arp[7]</code>	адрес 3-го элемента <code>&amp;c[2]</code> массива <code>c</code>	<code>1A60</code>
<code>**arp</code>		значение 4-го элемента <code>a[3]</code> массива <code>a</code>	<b><code>14</code></b>
<code>*(*(arp+2))</code>		значение 2-го элемента <code>c[1]</code> массива <code>c</code>	<b><code>23</code></b>
<code>*(*arp+1)</code>		значение 5-го элемента <code>a[4]</code> массива <code>a</code>	<b><code>52</code></b>
<code>*(*(arp+2)+1)</code>		значение 3-го элемента <code>c[2]</code> массива <code>c</code>	<b><code>59</code></b>

## Общая схема доступа к элементам исходных массивов

Для наглядности изобразим общую схему различных способов доступа к элементам исходного массива `a`:



На данной схеме в столбцах (для каждой ячейки памяти) одним цветом обозначены тождественные выражения (которые возвращают одинаковые значения). Например, для 1-го элемента массива `a`:

- тождественные выражения для указателей:

`arp[4] == *(arp+4) == *arp-3 == &a[0] == 0BF2`

- тождественные выражения для значений (типа `int`):

`*arp[4] == ** (arp+4) == *(arp-3) == a[0] == 39`

Подобные схемы можно изобразить для массивов `b` и `c`.

## Упражнения

### Упражнение 16.1

По аналогии с общей схемой различных способов доступа к элементам массива `a` составить описание различных способов доступа к элементам массива `b`.

### Упражнение 16.2

По аналогии с общей схемой различных способов доступа к элементам массива `a` составить описание различных способов доступа к элементам массива `c`.