НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
Кафедра информатики и процессов управления (№17)

Дисциплина «Информатика» (углубленный уровень), 1-й курс, 2-й семестр.

Методические указания

Тематическое занятие 14

Массивы указателей.

Содержание

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 «**ar**ray 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.

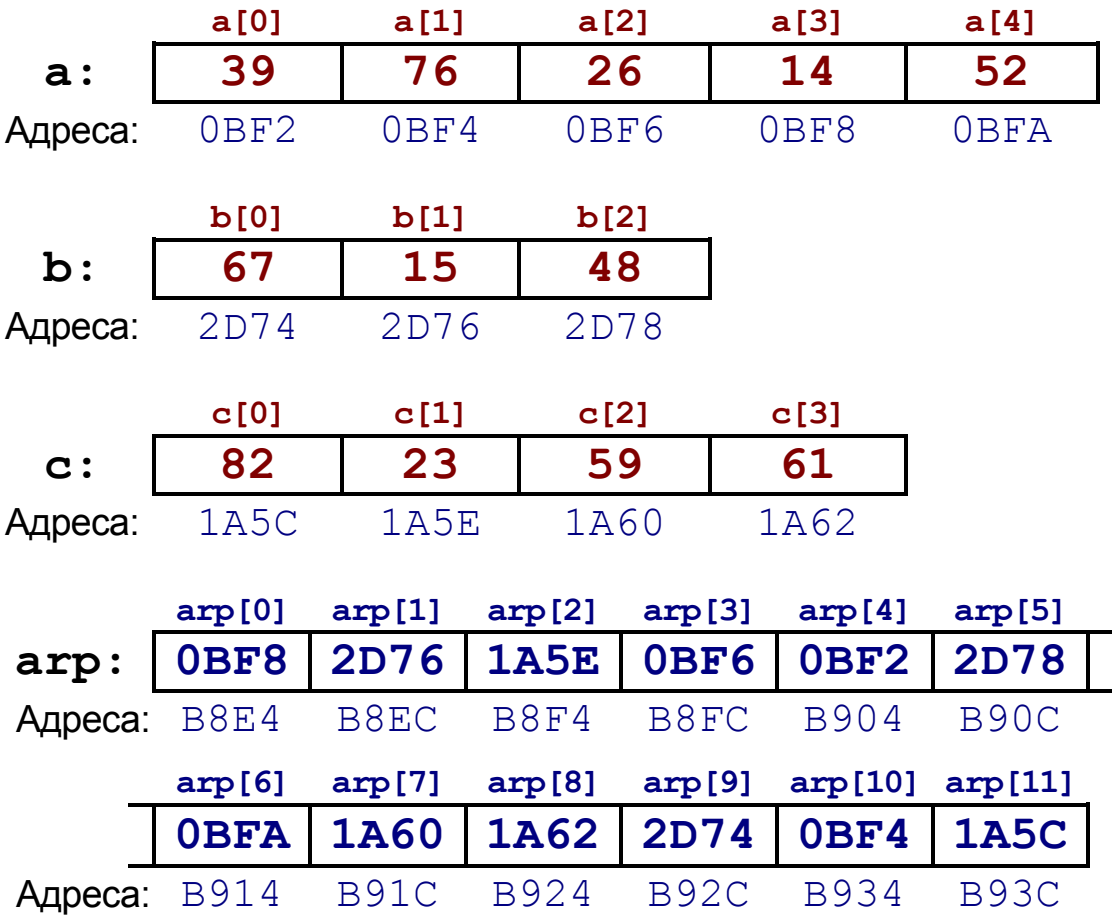
Доступ к элементам массива указателей через индексы

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:



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≤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.

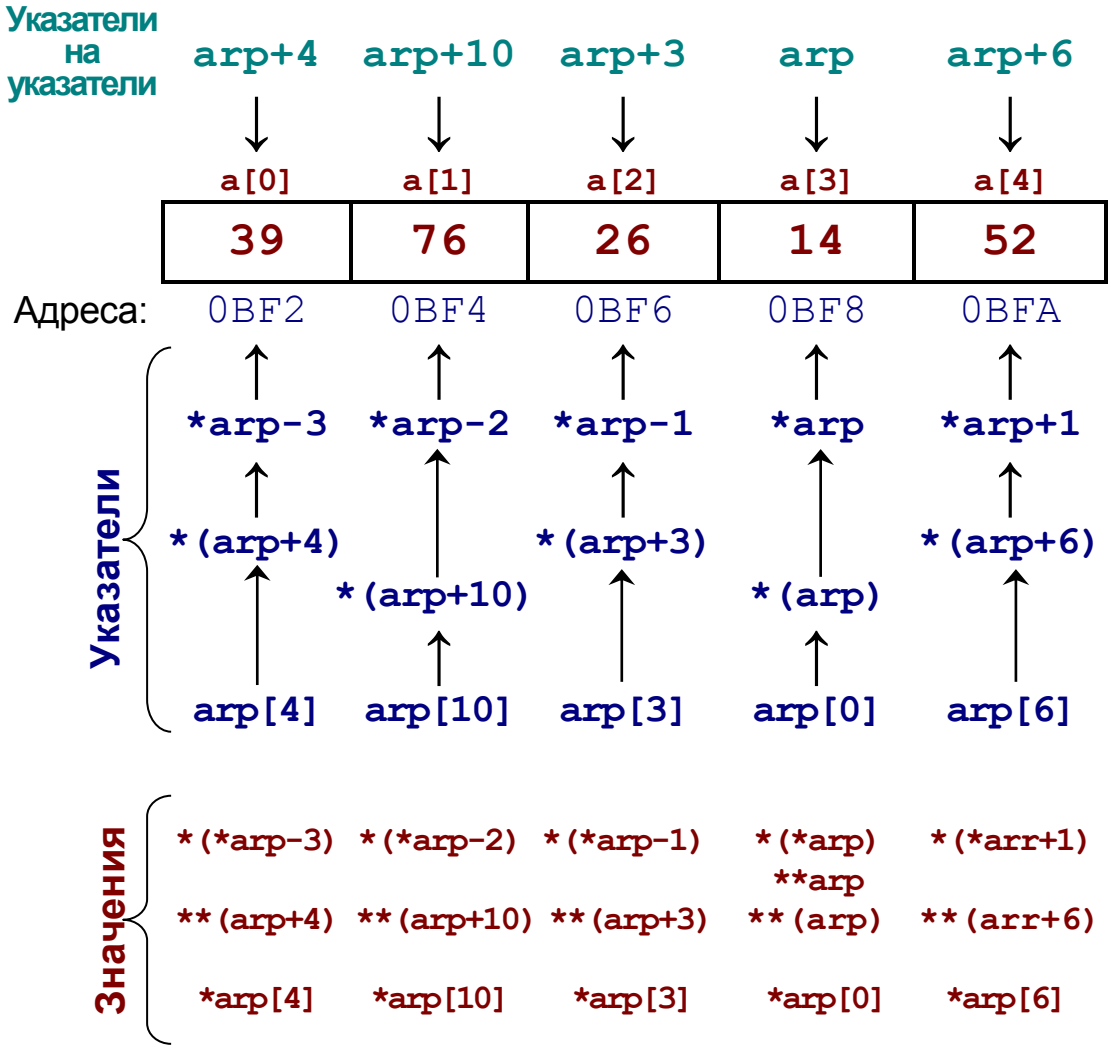
Применения адресной арифметики для массива указателей

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 `a`, `b` и `c`	число-вое
arp	адрес 1-го элемента `&arp[0]`		`B8E4`
arp+2	адрес 3-го элемента `&arp[2]`		`B8F4`
***arp**	значение 1-го элемента `arp[0]`	адрес 4-го элемента `&a[3]` массива `a`	`0BF8`
***(arp+2)**	значение 3-го элемента `arp[2]`	адрес 2-го элемента `&c[1]` массива `c`	`1A5E`
***arp+1**	значение 7-го элемента `arp[6]`	адрес 5-го элемента `&a[4]` массива `a`	`0BFA`
***(arp+2)+1**	значение 8-го элемента `arp[7]`	адрес 3-го элемента `&c[2]` массива `c`	`1A60`
****arp**		значение 4-го элемента `a[3]` массива `a`	**14**
***(*(arp+2))**		значение 2-го элемента `c[1]` массива `c`	**23**
***(*arp+1)**		значение 5-го элемента `a[4]` массива `a`	**52**
***(*(arp+2)+1)**		значение 3-го элемента `c[2]` массива `c`	**59**

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 14.1

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

Упражнение 14.2

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