



УКАЗАТЕЛИ

Лекция №7

ОБЩИЕ СВЕДЕНИЯ ОБ УКАЗАТЕЛЯХ

- указатель - переменная, которая содержит адрес другой переменной
- Указуемая переменная – переменная, адрес которой хранится в указателе



ИСХОДНЫЕ ПРЕДПОСЫЛКИ

- Физическая память компьютера – одномерный массив
- Необходимость гибкой работы с адресами оперативной памяти
- Си находится между Паскалем и Ассемблером
- Широкое применение Си для специализированных ЭВМ с ограниченными ресурсами
- Необходимость управления памятью – резервирование и освобождение в процессе работы программы



ФОРМАТ ОПИСАНИЯ УКАЗАТЕЛЯ

- ТИП *ИМЯ;
 - * в описании – признак указателя
 - тип относится к указуемой переменной

- Примеры
 - `int *p;`
 - `float *q;`
 - `char *s;`



СВОЙСТВА УКАЗАТЕЛЕЙ

- освобождают от необходимости помнить адреса ячеек памяти
- поддерживают операции адресной арифметики
 - перемещение от одной ячейки памяти к другой
 - сравнение указателей



ИНИЦИАЛИЗАЦИЯ УКАЗАТЕЛЕЙ

- `int A[5]={2,1,5,3,4};`
- `int *p=&A[0];//или int *p=A;`



ОБРАЩЕНИЕ К СОДЕРЖИМОМУ

- $a = *p$
- * в выражении означает доступ к указуемому содержимому



ЭТАПЫ РАБОТЫ С УКАЗАТЕЛЕМ

- Определение указателя
- Присваивание указателю адреса другой переменной
- Работа с переменной через указатель

- `int a=5, *p,b=0;`
- `p=&a;`//обязательно присвоить адрес!!!
- `b=*p;`

- !!! Использование указателя со случайным адресом ведет к непредсказуемым результатам

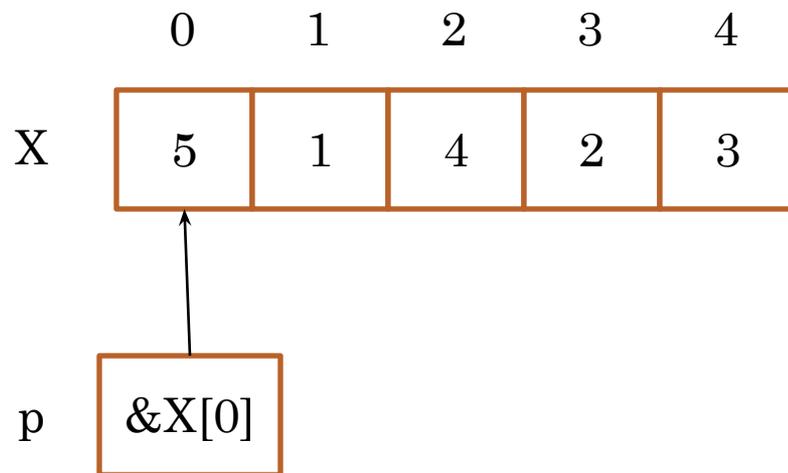


ПУСТОЙ УКАЗАТЕЛЬ

- `int *t=NULL;`
- Пустой указатель – нулевой адрес
- При обращении к нему формируется системное сообщение об ошибке



РАБОТА С МАССИВАМИ С ПОМОЩЬЮ УКАЗАТЕЛЕЙ



```
int X[]={5,1,4,2,3}, *p, i=2, k=0;
```

```
p=&X[0];
```

```
p++;
```

```
k=*(p+i);
```



ОПЕРАЦИИ АДРЕСНОЙ АРИФМЕТИКИ

- Обращение к содержимому со смещением
 - $*(p+i)$ – содержимое ячейки на i элементов вперед
 - $*(p-i)$ – содержимое ячейки на i элементов назад
 - смещение должно быть целым
 - размер содержимого учитывается автоматически
 - указатель не изменяется

- «Перемещение» вдоль памяти
 - $p++$ - на 1 элемент вперед
 - $p--$ - на 1 элемент назад
 - $p+i$ – на i элементов вперед
 - $p-i$ – на i элементов назад
 - указатель изменяется



ОПЕРАЦИИ АДРЕСНОЙ АРИФМЕТИКИ

□ Сравнение указателей

- $r < q$; адрес в указателе r ближе к началу, чем адрес в указателе q ?
- $r == q$; указатели указывают на одну и ту же ячейку?

□ Вычитание указателей

- $k = r - q$;
 - результат – целое число, показывающее, на сколько элементов адрес в указателе r дальше от начала памяти относительно адреса в указателе q



ПРИМЕРЫ ОПЕРАЦИЙ НАД УКАЗАТЕЛЯМИ

```
int X[]={5,1,4,2,3}, *p,*q, i=0, k=0,j=0;
```

```
p=&X[2];
```

```
q=&X[4];
```

```
p--;
```

```
k=*(p-1);//k=?
```

```
if(p<q)
```

```
    j=q-p;//j=?
```

```
if(*p>*q)
```

```
    i=(*p)-(*q);//i=?
```



СРАВНЕНИЕ МАССИВОВ И УКАЗАТЕЛЕЙ

□ Сходства

- Обеспечивают доступ к элементам по номеру
- Тип данных учитывает размер элементов

□ Различия

- Адрес массива изменить нельзя- массив привязан к конкретной области памяти
- Адрес, записанный в указателе, можно изменить с помощью операций адресной арифметики



ЭКВИВАЛЕНТНОСТЬ ОПЕРАЦИЙ НАД УКАЗАТЕЛЯМИ И МАССИВАМИ

- `int X[5], *p, k=2;`
- `p=X;`

- $X \sim \&X[0]$
- $X+k \sim \&X[0]$
- $p+k \sim \&p[k]$
- $*(p+k) \sim p[k]$

- Работу с указателями можно сделать почти неотличимой от работы с массивами



ОБРАБОТКА МАССИВА С ПОМОЩЬЮ УКАЗАТЕЛЯ

```
int X[]={5,1,4,2,3}, *p,*q, i=0, n=5,s=0;
```

```
//Способ 1
```

```
p=X;
```

```
for(i=0;i<n,i++)
```

```
    s=s+p[i];// указатель на месте, индекс двигается
```

```
//Способ 2
```

```
for(p=X;p<X+n,p++)
```

```
    s=s+(*p);// указатель двигается
```



ПОИСК ЭЛЕМЕНТА С ПОМОЩЬЮ УКАЗАТЕЛЕЙ

```
int X[]={5,1,4,2,3}, *p,*q, i=0, n=5,s=0;  
for(p=X;p<X+n;p++)//что делает фрагмент?  
    if(*p==4)  
        q=p;  
s=*q;
```

```
q=X;  
for(p=X+1;p<X+n;p++)//что делает фрагмент?  
    if(*p<*q)  
        q=p;  
s=*q;
```



ПЕРЕСТАНОВКИ ЭЛЕМЕНТОВ С ПОМОЩЬЮ УКАЗАТЕЛЕЙ

```
int X[]={5,1,4,2,3}, *p,*q, i=0, n=5,s=0;
```

```
p=X;
```

```
q=p+4;
```

```
s=*p;
```

```
*p=*q;
```

```
*q=s;
```

```
for(q=p;q<p+n;q++)
```

```
    printf("%d",*q);
```



Выводы

- Указатели позволяют гибко работать с оперативной памятью
- Указатель позволяет работать с памятью как с массивом с помощью адресной арифметики
- Указатель позволяет «перемещаться» вдоль памяти, массив жестко привязан к участку памяти
- Тип данных при работе с указателями учитывается автоматически
- Указатели позволяют динамически работать с памятью во время работы программы

