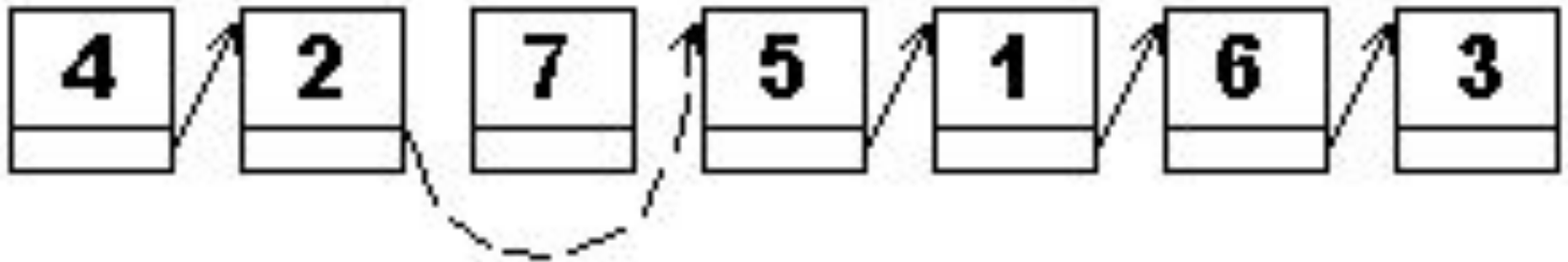


**Лекция 5.**  
**Линейный список**  
**в динамической памяти**

# Линейный список в виде одномерного массива



# Линейный список в динамической памяти



# Структура памяти языка Си

КОД ПРОГРАММЫ	Младшие адреса
РАЗДЕЛ ГЛОБАЛЬНЫХ ПЕРЕМЕННЫХ И КОНСТАНТ	
ДИНАМИЧЕСКАЯ ПАМЯТЬ	
СТЕК (локальные переменные)	Старшие адреса

# Статические и динамические переменные



# Функции для динамического распределения памяти

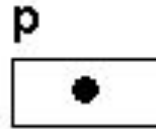
```
void *malloc(size_t size);
```

```
void *free(void *p);
```

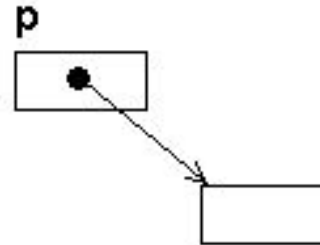
1. `char *c;`
2. `// Выделить память 10 байт`
3. `c=(char *)malloc(10);`
4. `// или`
5. `c=(char *)malloc(sizeof(char)*10);`

# Использование функций malloc() и free()

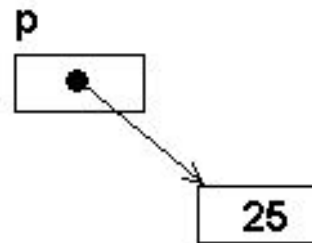
```
int *p;
```



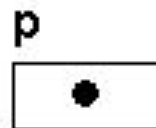
```
p=(int*)malloc(sizeof(int));
```



```
*p = 25;  
printf("%d", *p);
```



```
free(p);
```

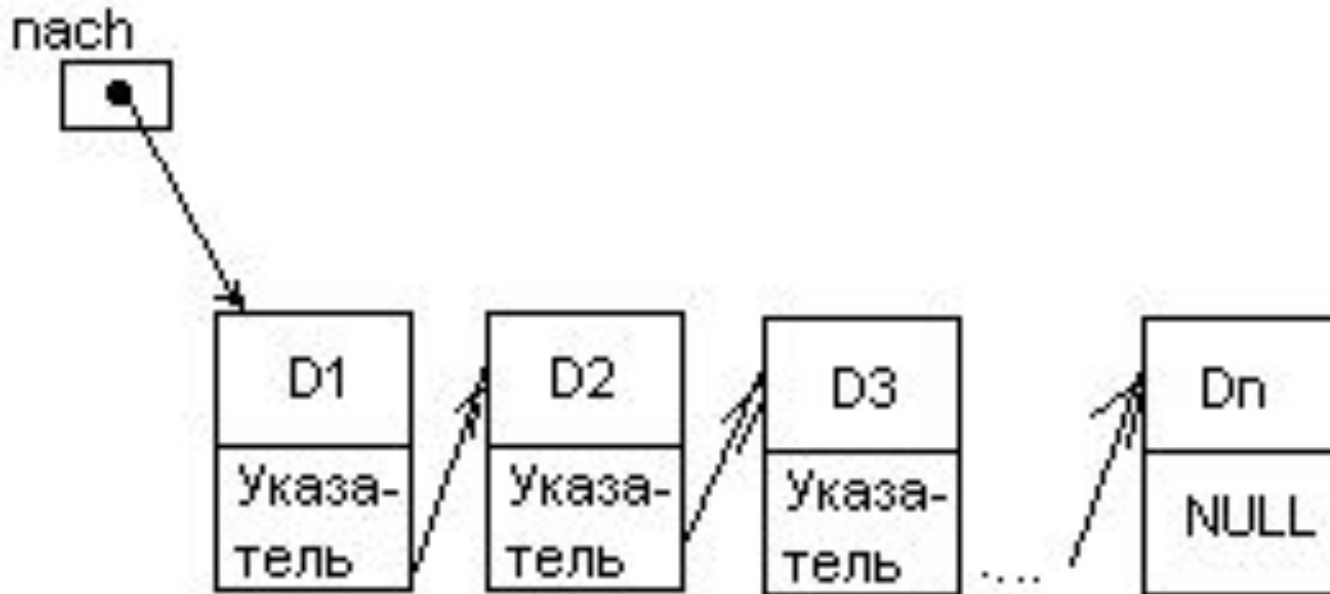


# Пример 1. Создание массива в динамической памяти

```
1. int *p, i;  
2. p = (int *) malloc(100 * sizeof(int));  
3. if (p==NULL)  
4.     {  
5.         printf("Недостаточно памяти\n");  
6.         return 0;  
7.     }  
8. for (i = 0; i < 100; i++)  
9.     *(p+i) = i;  
10. for (i = 0; i < 100; i++)  
11.     printf("%d ", p[i] );  
12. free(p);
```

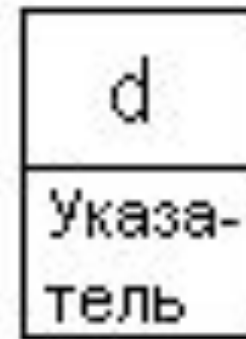


# Общее представление о списке в динамической памяти



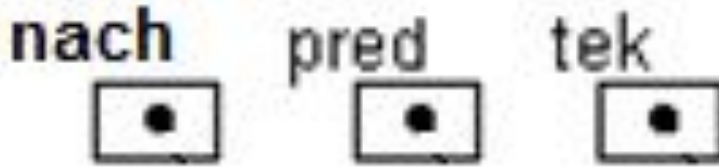
# Описание типа для построения линейного списка

```
struct element  
{  
    int d;  
    struct element *link;  
};
```

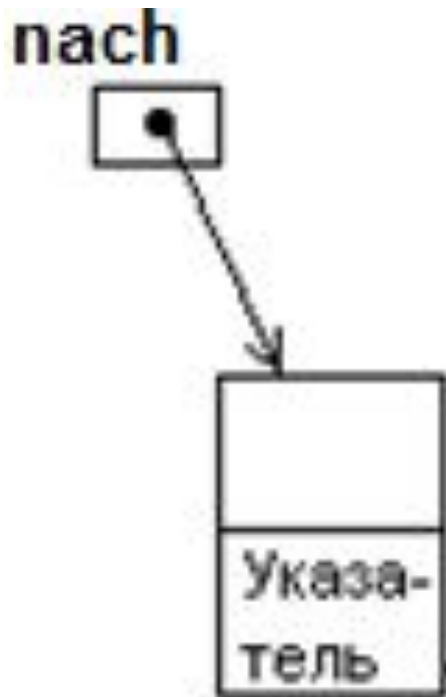


# Создание элементов в динамической памяти

```
struct element *nach, *pred, *tek;
```

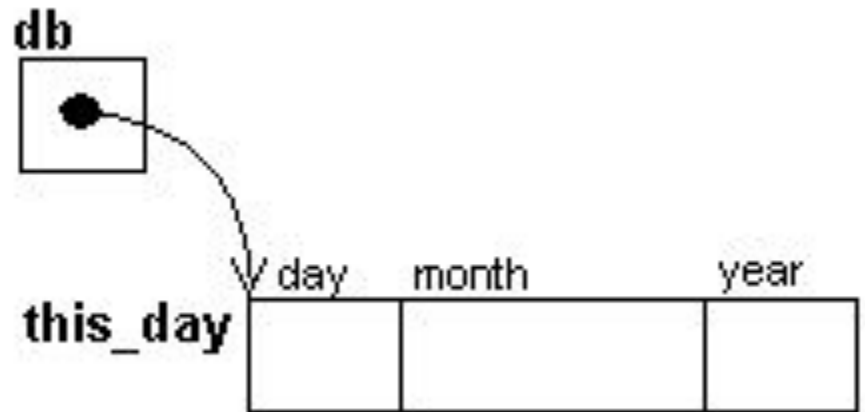


```
nach=(struct element *)malloc(sizeof(struct element));
```



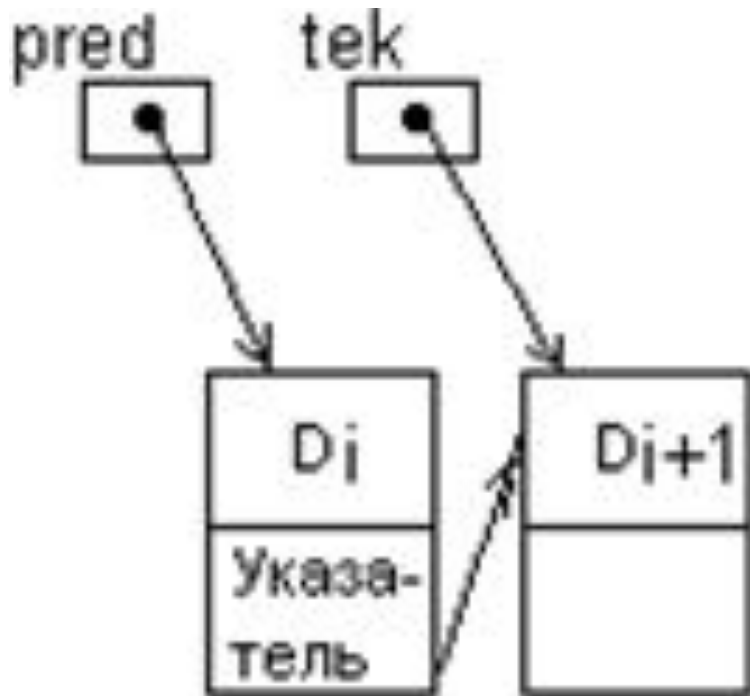
# Обращение к полям структуры через указатель (в статической памяти)

```
1. struct date {  
2.     int day;  
3.     char month[10];  
4.     int year; };  
5. struct date this_day, *db;  
6. db = &this_day;  
   ...  
7. (*db).day = 25;  
8. db -> day = 25;
```



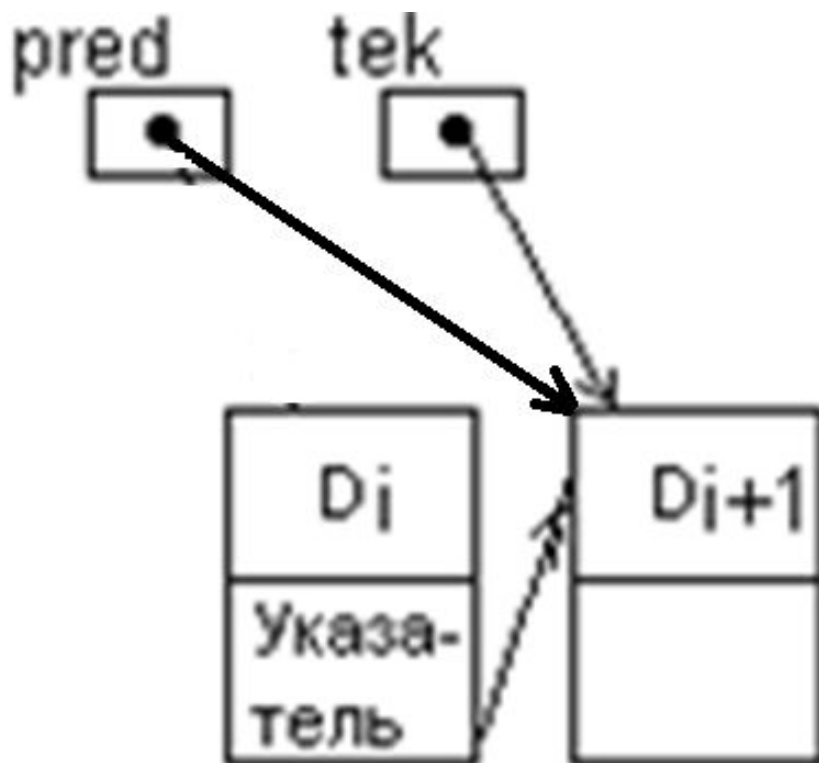
# Связывание элементов

`pred -> link=tek; // обращение к полю link`  
`// структуры через указатель`



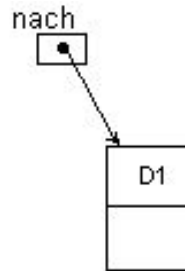
# Переприсваивание указателей

`pred=tek; // pred и tek указывают на  
// один и тот же элемент`

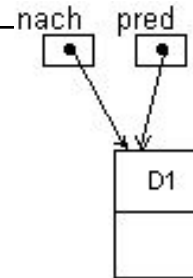


# Алгоритм построения списка в прямом порядке

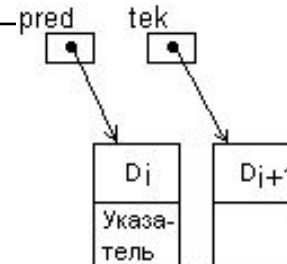
1. Создать в динамической памяти первый элемент nach;  
Ввести nach->d;



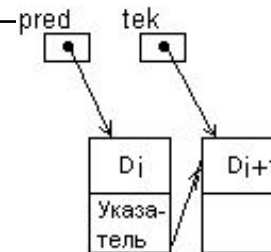
2. pred=nach;



3. Создать текущий элемент tek:  
Ввести tek->d;

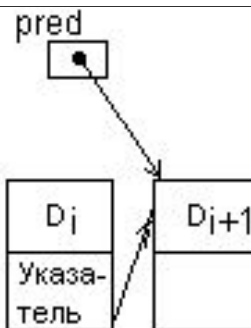


4. Установить связь предыдущего с текущим:  
pred->link=tek;



# Продолжение. Алгоритм построения списка в прямом порядке

5. Текущий элемент становится предыдущим: `pred=tek;`



6. Если не конец ввода то перейти на 3.

7. Обнулить адресную часть последнего элемента:  
`tek->link=NULL;`





# Глобальные описания

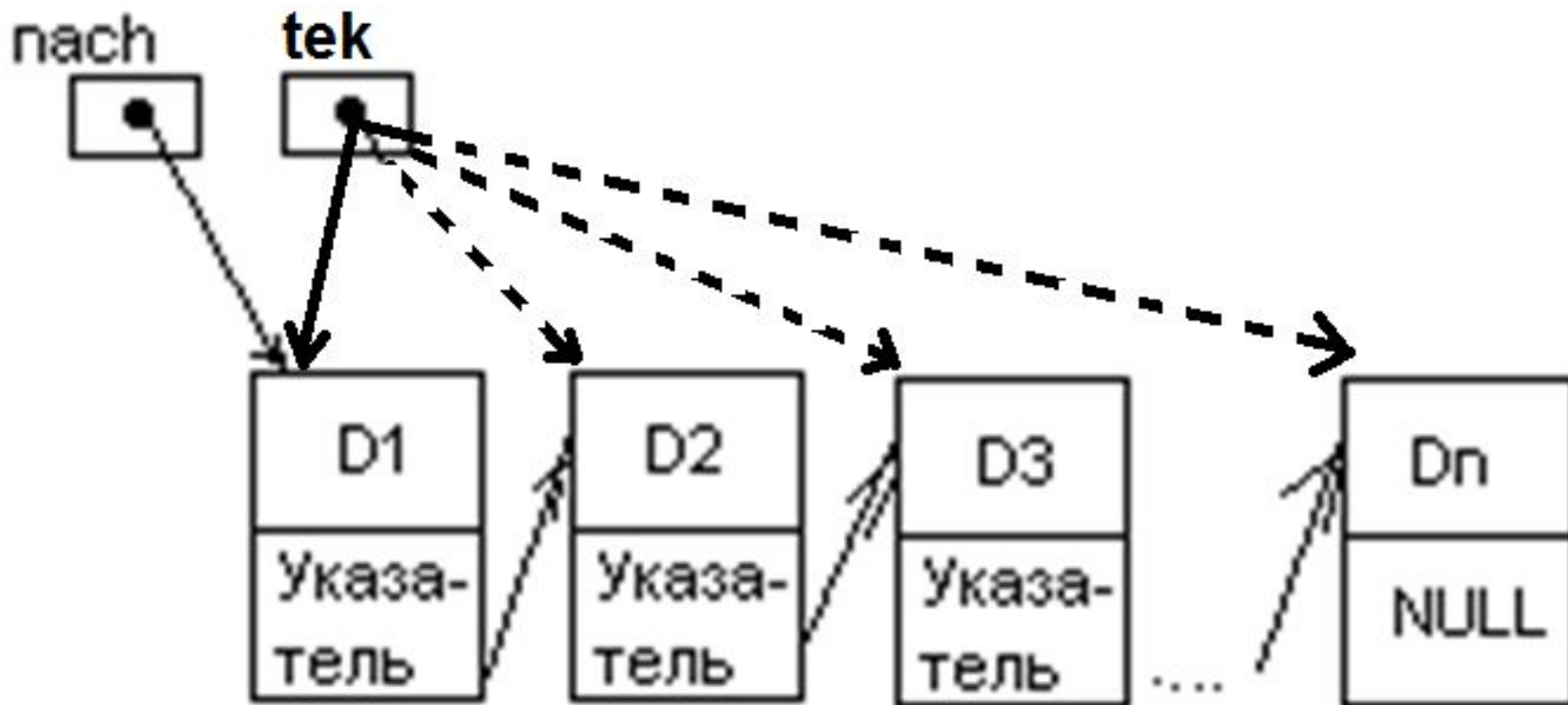
```
struct element
{
    int d;
    struct element *link;
};
struct element *nach; // Указатель на
// начало списка
```

# Шаг по связи

`tek = tek->link;`



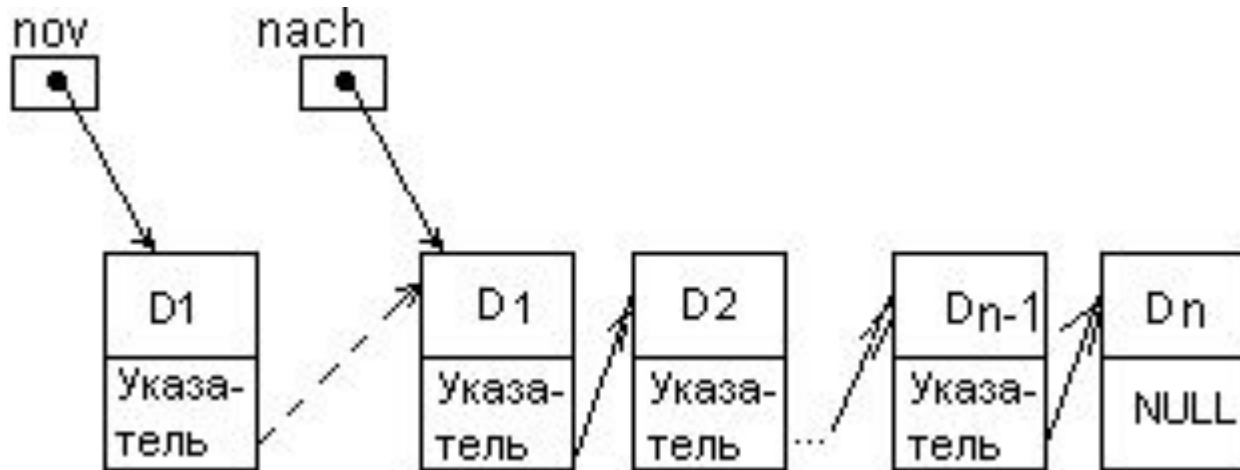
# Просмотр списка



## Пример 2. Функция просмотра списка:

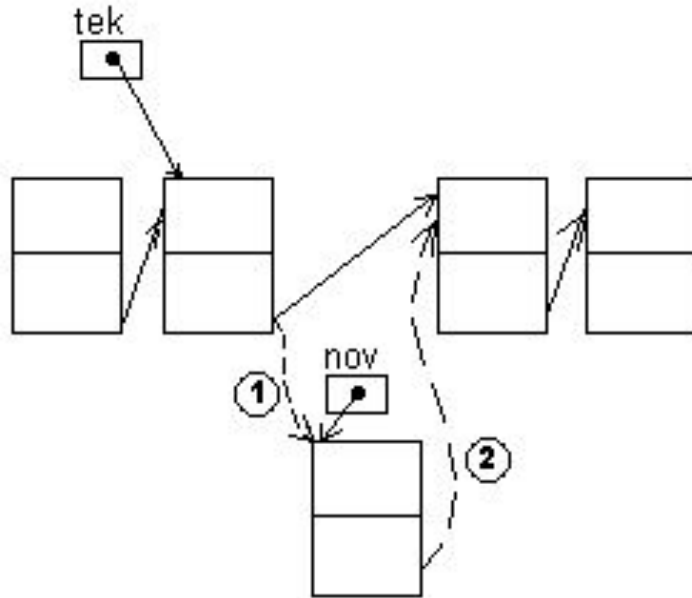
```
1. void prosmotr()
2. {
3.     struct element *tek;
4.     tek = nach;      // Встали на начало списка
5.     while(tek != NULL) // Пока не конец списка
6.     {
7.         printf("%d", tek->d);
8.         tek = tek->link; // Шаг по связи
9.     }
10. }
```

# Добавление элемента в начало списка



1. Создать новый элемент `nov`.  
Ввести информационную часть `nov->d`.
2. Построить связь:  
`nov->link=nach;`
3. Новый элемент сделать начальным:  
`nach=nov;`

# Добавление элемента в середину списка



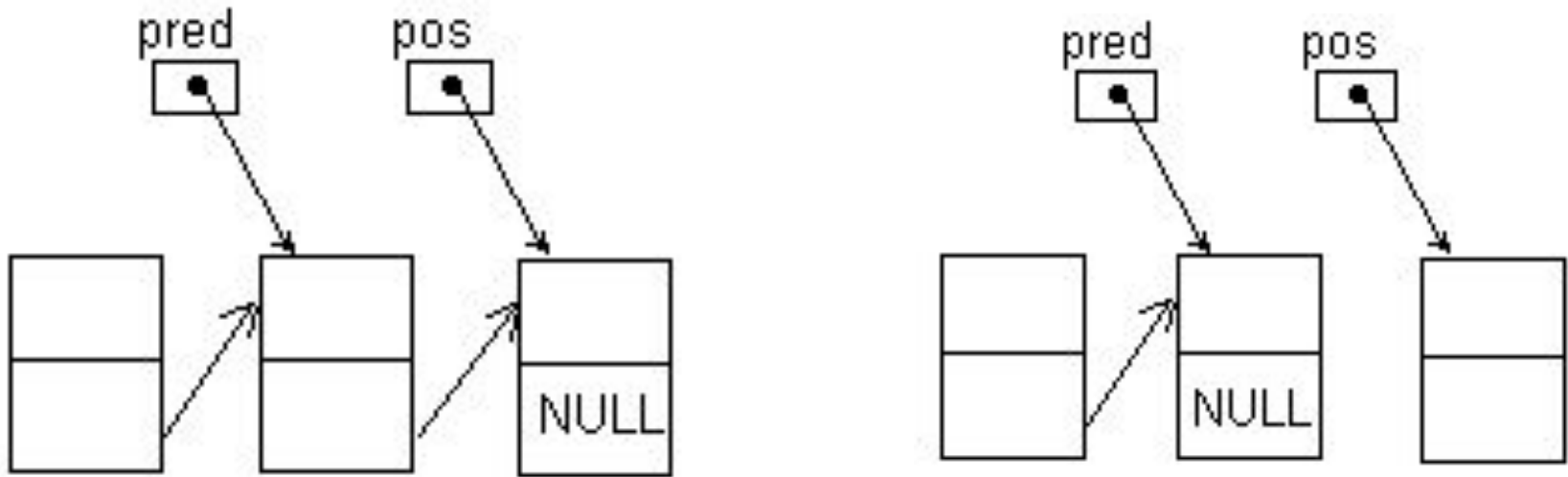
1. Заполнить связь 2 у элемента **nov**:

`nov->link=tek->link;`

2. Соединить **tek** и **nov** (связь 1):

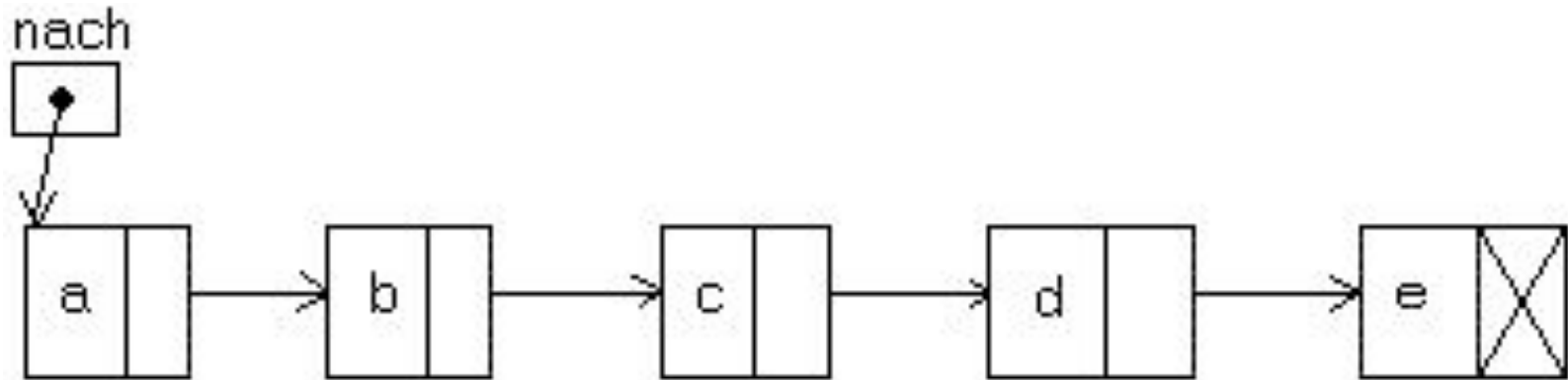
`tek->link=nov;`

# Добавление элемента в конец списка



1. `pred->link=NULL;`
2. `free(pos);`

# Пример 3. Перемещение по списку

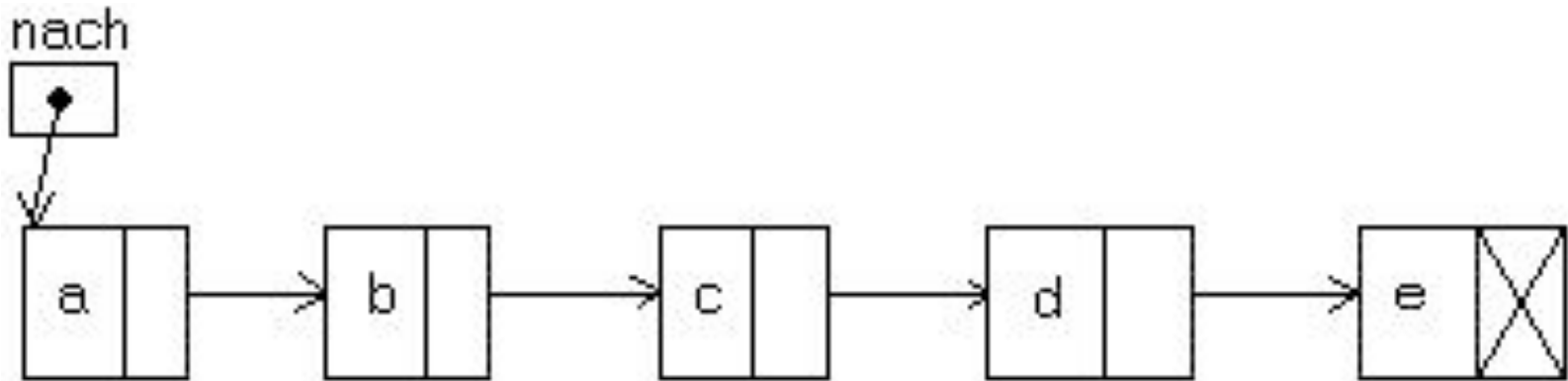


Фрагмент программы:

```
tek=nach;  
while(tek->d != 'c')  
    tek=tek->link;  
printf("%s", tek->d);
```



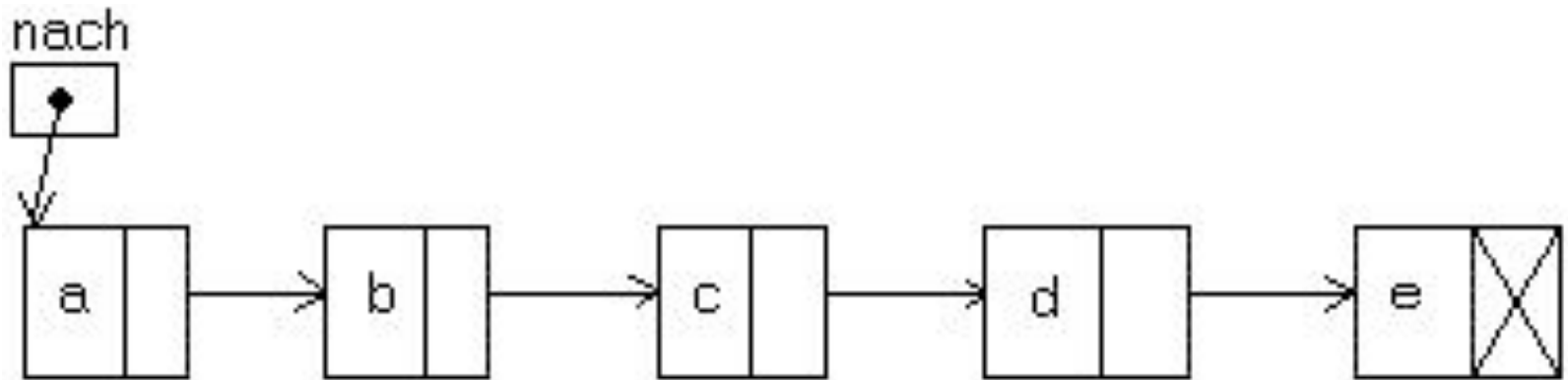
# Пример 4. Перемещение по списку



Что будет выведено на экран в результате выполнения фрагмента программы:

```
tek=nach;  
for(i=1; i<=4; i++)  
    tek=tek->link;  
printf("%s", tek->d);
```

# Пример 5. Перемещение по списку



Что будет выведено на экран в результате выполнения фрагмента программы:

```
tek=nach;  
while(tek->link != NULL)  
    tek=tek->link;  
printf("%s", tek->d);
```