

Лекция 4



А.Ф. ЗУБАИРОВ

Структуры в C++



- Структура – производный тип данных; создается из элементов других типов.

```
struct имя_структуры {  
    тип имя_элемента;  
    тип имя_элемента;  
    ...  
    тип имя_элемента;  
};
```

Объявление:

```
struct имя_структуры перемен1, перемен2, ...;
```

Структуры в C++



Создание структуры:

```
struct person {  
    unsigned short age;  
    char name[255];  
    char sex;  
};
```

Объявление переменных:

```
struct person student, people[52], *man;
```

Структуры C++



- Создание типа:

```
typedef struct имя_структуры {  
    тип имя_элемента;  
    тип имя_элемента;  
    ...  
    тип имя_элемента;  
} имя_типа;
```

Структуры в C++



Создание типа:

```
typedef struct person {  
    unsigned short age;  
    char name[255];  
    char sex;  
} person;
```

Объявление переменных:

```
person student, people[52], *man;
```

Операции со структурами



1. Присваивание переменных структур переменным того же типа:
`student = people[4];`
2. Взятие адреса структуры: `&student`
3. Применение операции `sizeof` для определения размера структуры:
`sizeof(student)`
4. Обращение к элементам структуры.

Обращение к элементам структуры



- **Инициализация:**

```
person student = {4, "Петя", 'м'};  
person *woman;
```

- **Обращение к элементам:**

- **Через имя переменной – операция-точка (.):**

```
student.age = 5; d = student.age;
```

- **Через указатель – операция-стрелка (->):**

```
woman->age = 7; d = woman->age;
```

```
(*woman).age = 7; d = (*woman).age;
```

```
(&student)->age = 100;
```

Связанные структуры данных

Последовательное распределение

Массив `queue` из n элементов

Адрес	Содержимое
<code>queue + 1</code>	элемент1
<code>queue + 2</code>	элемент2
<code>queue + 3</code>	элемент3
...	...
<code>queue + n</code>	элементn

Связанное распределение

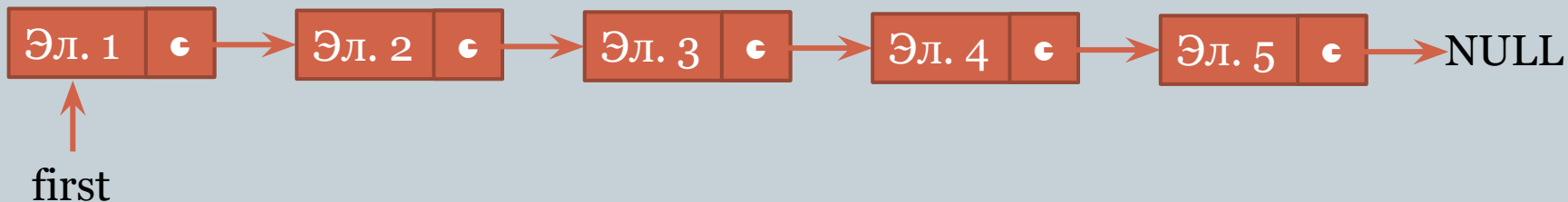
Список

Адрес	Содержимое	
A	элемент1	B
B	элемент2	C
C	элемент3	D
	...	
N	элементn	NULL

Связанные структуры данных



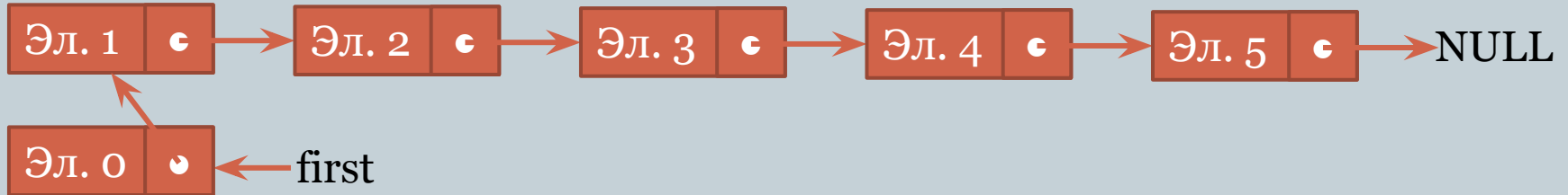
- В программе для реализации связанной структуры данных кроме значения элемента необходимо хранить адрес следующего элемента.



- `first` – переменная связи, указывающая на первый узел списка.

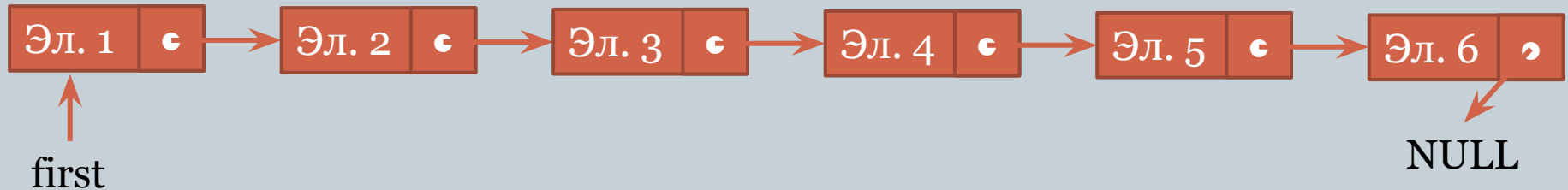
Вставка нового узла в список

Вставка в начало:



- 1) Создать новый элемент 0
- 2) Сделать ссылку у элемента 0 на 1
- 3) Сменить указатель first на элемент 0

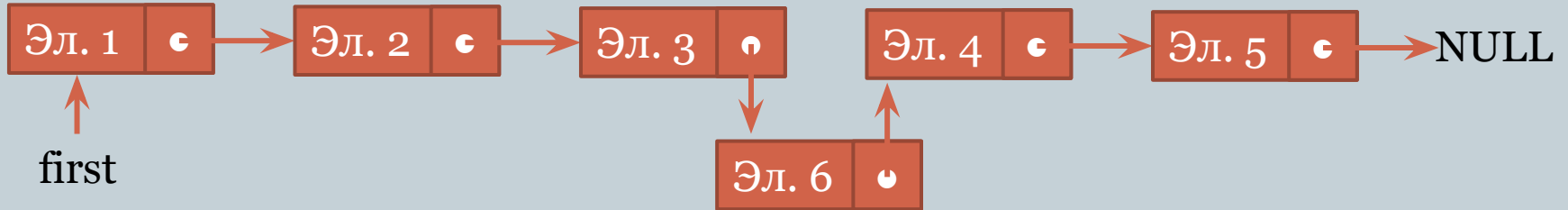
Вставка в конец после k элемента:



- 1) Создать новый элемент k+1
- 2) Сделать ссылку у k на k+1
- 3) Сделать ссылку у k+1 на NULL

Вставка нового узла в список

Вставка в середину элемента k:



- 1) Создать новый элемент k
- 2) Сделать ссылку у k на k+1
- 3) Сделать ссылку у k-1 на k

Удаление узла из списка



Удаление первого элемента:

- 1) Сменить указатель `first` на элемент 1
- 2) Уничтожить элемент 1

Удаление последнего k элемента:

- 1) Сменить указатель элемента $k-1$ на `NULL`
- 2) Уничтожить последний k элемент

Удаление k элемента из середины:

- 1) Сменить указатель элемента $k-1$ на элемент $k+1$
- 2) Уничтожить k элемент

Реализация связанного списка



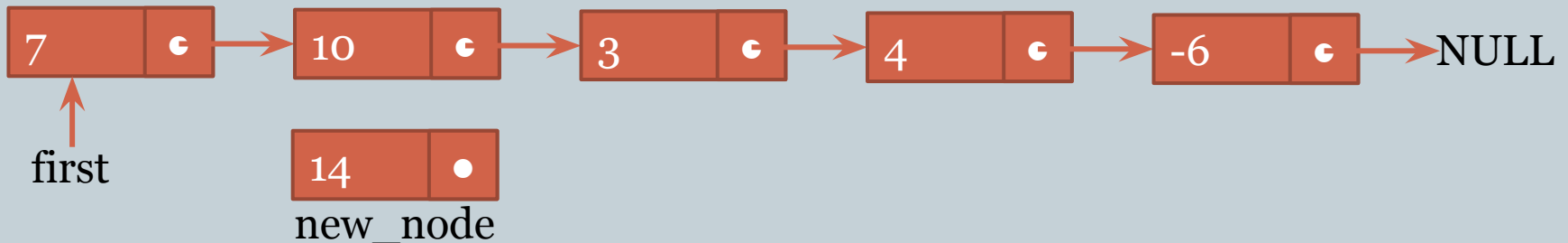
- Элемент списка реализуется в виде структуры:

```
struct имя_структуры {  
    элемент;  
    указатель_на_структуру_текущего_типа;  
};
```

```
struct list {  
    int info;  
    struct list *link;  
};
```

Реализация связанного списка

- `first` типа `list` – указатель на первый элемент списка;
- `new_node` типа `list` – указатель на новый элемент списка;
- Задача: вставить `new_node` типа `list` после `first`.



`first->info` возвращает информационную часть первого элемента списка (7)

`first->link` возвращает указатель на второй элемент списка

`first->link->info` возвращает информационную часть второго элемента (10)

`first->link->link` возвращает указатель на третий элемент списка

и т.д.

```
new_node->link = first->link;
```

```
first->link = new_node;
```