Односвязные списки - продолжение.

Вставка элемента списка

Предполагается, что список (линейный односвязный) существует: есть элемент, есть «голова», ссылка с «головы» установлена на элемент

Два случая вставки элемента списка — после текущего и перед текущим.

Схему действий при вставке после текущего уже рассматривали — функция `insert_after()`.

Нужно обработать ситуацию, когда текущий элемент — последний (его `*next = NULL`).

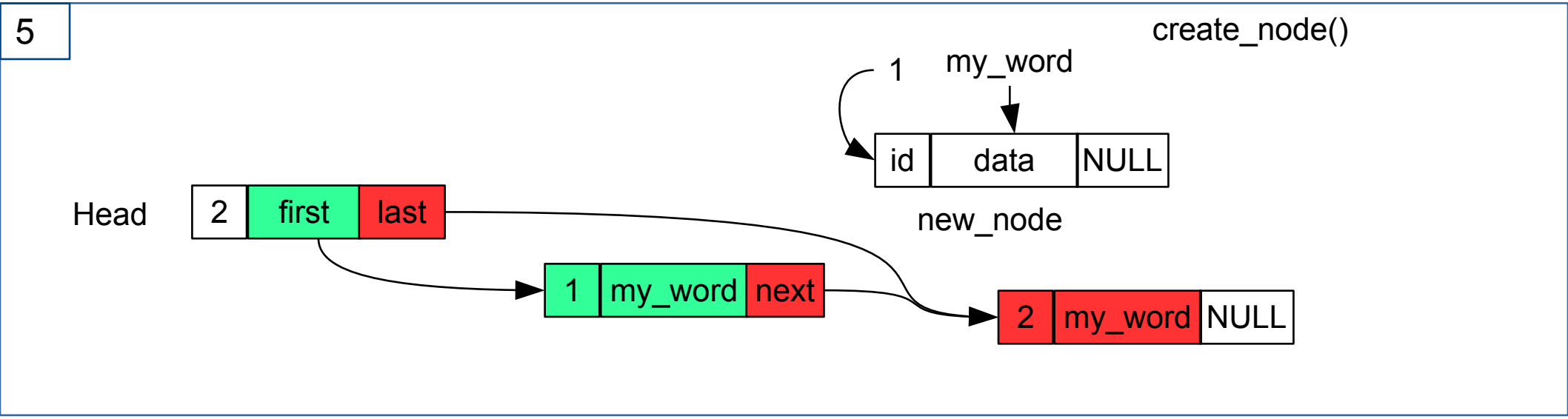
Вставка элемента списка

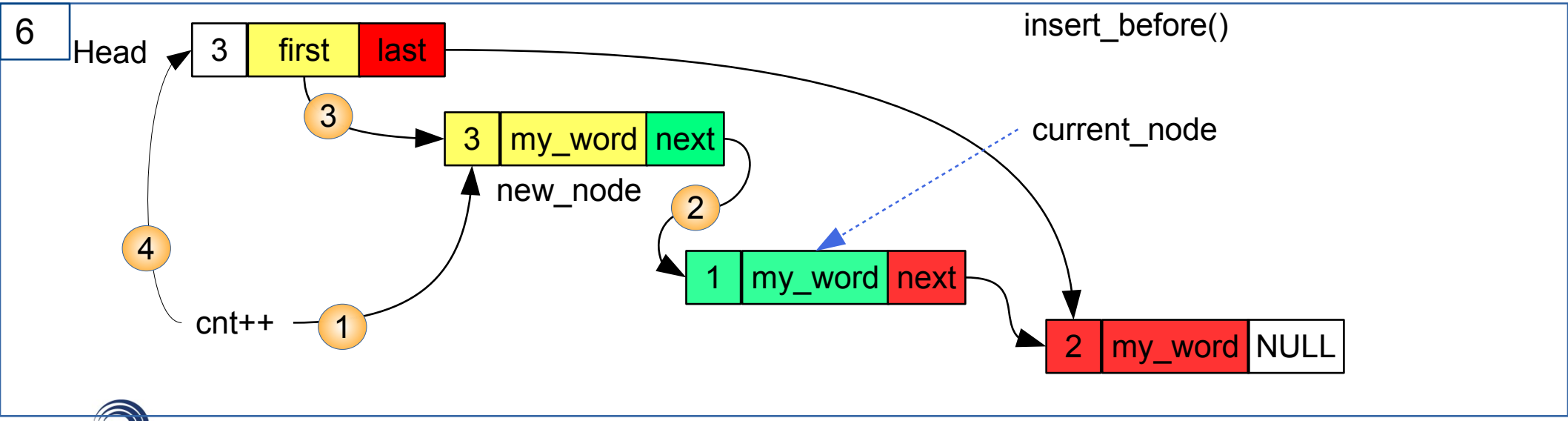
При вставке элемента перед текущим есть две ситуации

- Текущий элемент — первый (на него указывает **`Head→first`**)
- Текущий элемент — произвольный.

Схемы действий рассмотрены далее — функция **`insert_before()`**.

Вставка элемента списка – 3

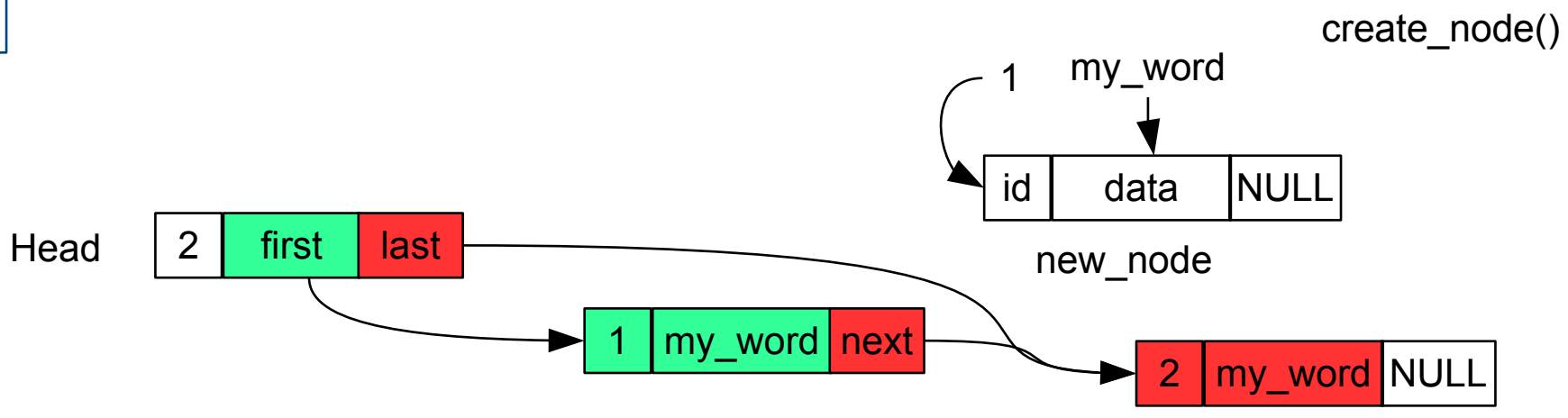




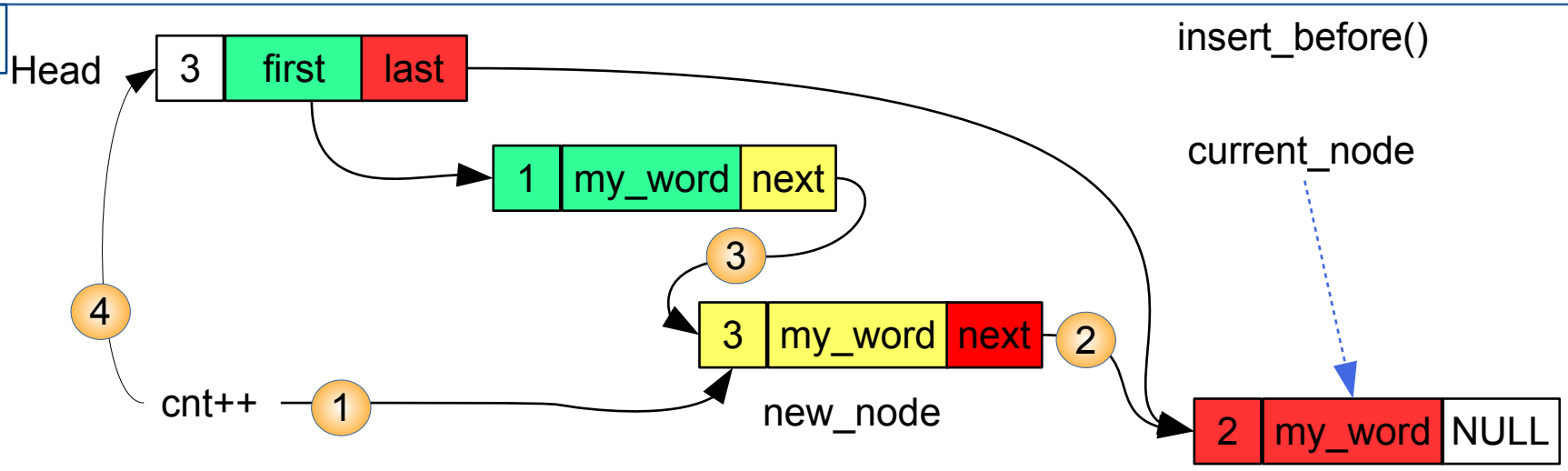
см. пример lect-14-01.c

Вставка элемента списка – 4

12



13



см. пример lect-14-01.c

Вставка элемента списка

Если текущий элемент — произвольный, то можно пройти по списку, начиная с «головы», найти элемент, который указывает на текущий, и для найденного элемента выполнить `insert_after()` (функция уже есть).

Вставка элемента списка перед текущим

```
void insert_before(Head *my_head, Node *new_node, Node
*current_node)
{
    Node *q=NULL,*q1=NULL;
    int n;
    if(my_head && new_node && current_node)
    {
        if(myHead->first==current_node)
        {
            n=my_head->cnt+1;
            new_node->id=n;
            new_node->next=current_node;
            my_head->first=new_node;
            my_head->cnt=n;
        }
         /* else - на следующей странице */
```

Вставка элемента списка перед текущим

```
/* функция insert_before() - продолжение */
        else
        {   /* проход по списку */
            q=my_head->first;
            while(q!=NULL)
            {
                if(q->next==current_node)
                {
                    q1=q;
                    insert_after(my_head,new_node,q1);
                    q=NULL;
                }
                else q=q->next;
            }
        }
    }
}
```

Поиск элемента по номеру (позиции)

В цикле проходим по элементам, начиная с первого (**`Head→first`**).

Цикл for с количеством повторений, соответствующим искомой позиции.

Для поиска элементов они должны быть в списке (список д.б. сформирован).

См. пример lect-14-02.c – функция **`select_by_order()`**.

Поиск элемента по номеру (позиции)

```
Node *select_by_order(Head *my_head, int n)
{
    Node *q;
    int i,k;

    k=my_head->cnt; /* last node id */
    q=my_head->first;
    if((n>0)&&(n<(k+1)))
    {
        for(i=1;i<n;i++) q=q->next;

    }
    else q=NULL;
    return q;
}
```

Поиск элемента по значению поля

Точно так же делается проход по списку от первого элемента (`Head→first`), сравниваются значения заданного поля.

- Для числовых полей — равенство или попадание в интервал
- Для текстовых полей — совпадение строки или подстроки.

Все алгоритмы проверки уже известны.

См. пример lect-14-03.c — поиск по значению поля `id`, функция `select_by_id()`.

Поиск элемента по значению поля

```
Node *select_by_id(Head *my_head, int n)
{
    Node *q;
    int k;

    q=my_head->first;
    k=my_head->cnt; /* last node id */
    if((n>0)&&(n<=k))
    {
        while((q->id)!=n) q=q->next;
    }
    else q=NULL;
    return q;
}
```

Подсчет количества элементов списка

- Устанавливаем указатель на первый элемент (**`Head→first`**) и счетчик в 1.
- Проходим по списку пока **`next`** не **`NULL`**, прибавляем на каждом шаге 1 к счетчику.

Результат — количество элементов в списке, т. е. длина списка.

Удаление элемента списка

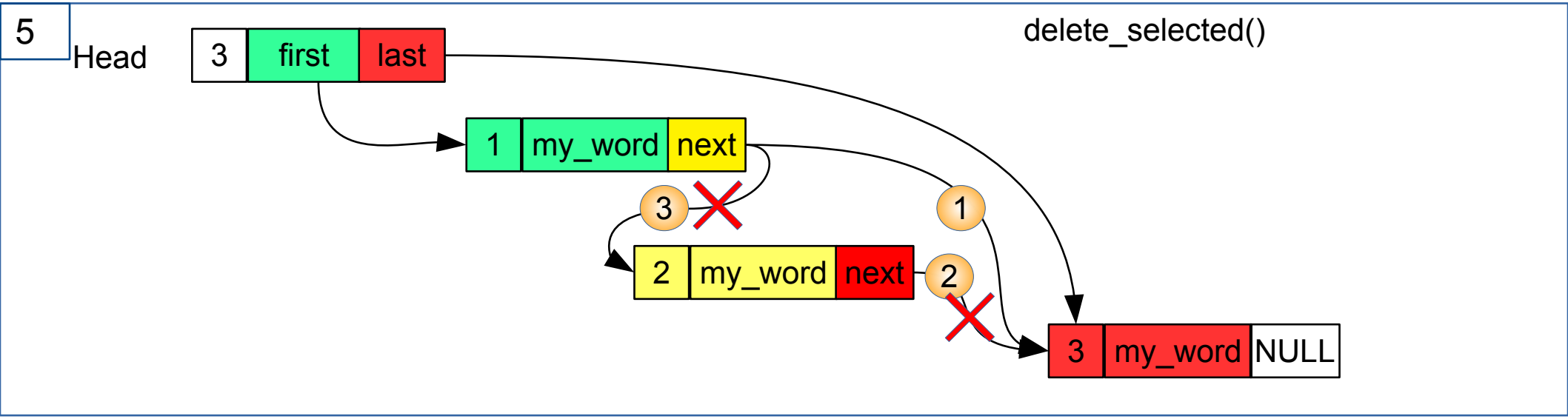
Чтобы элемент удалить, его надо найти (по порядку или по значению поля).

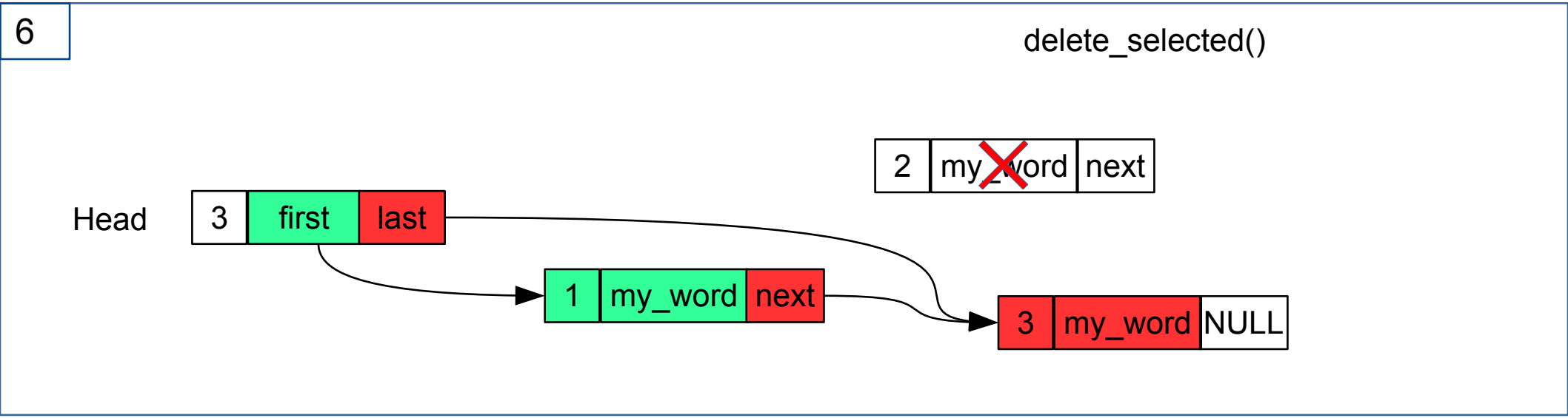
Три ситуации:

- Удаляемый элемент – в середине списка
- Удаляемый элемент — первый (**`Head→first`**)
- Удаляемый элемент — последний (**`Head→last`**)

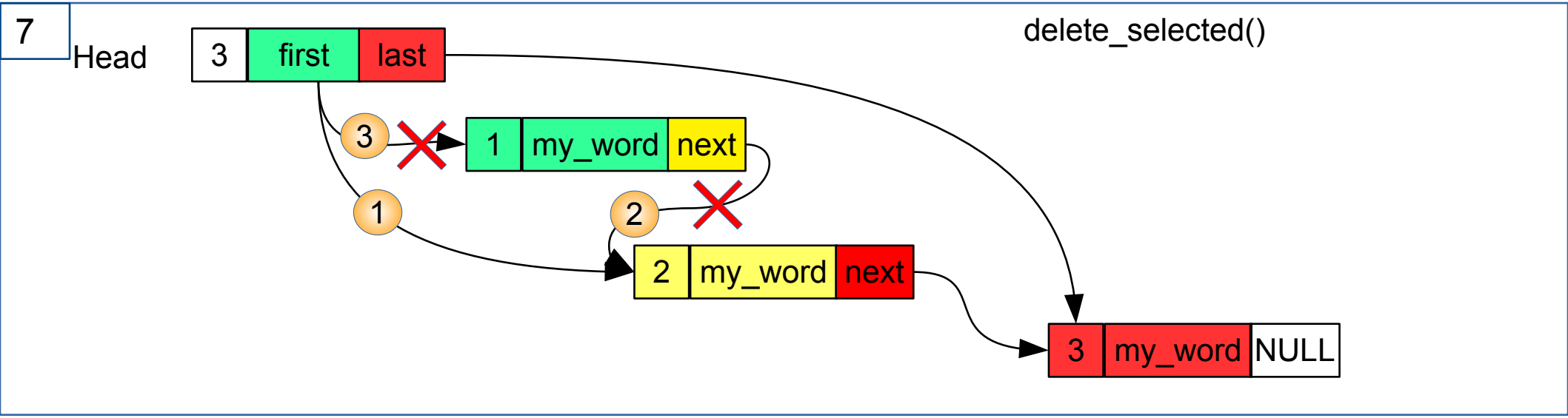
См. пример lect-14-04.c — функция **`delete_selected()`**.

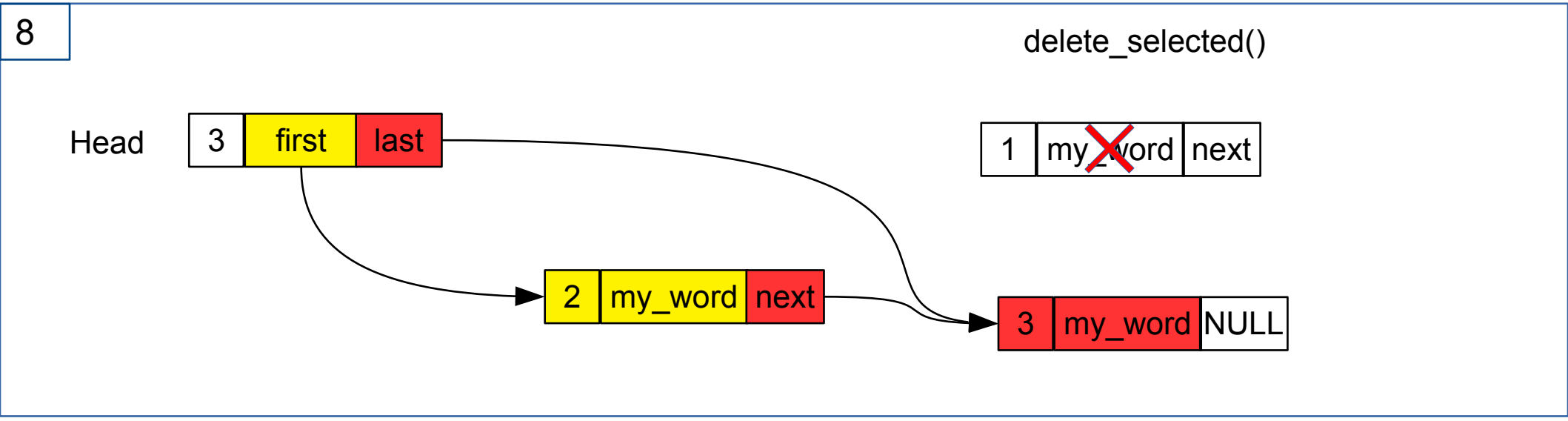
Удаление элемента списка



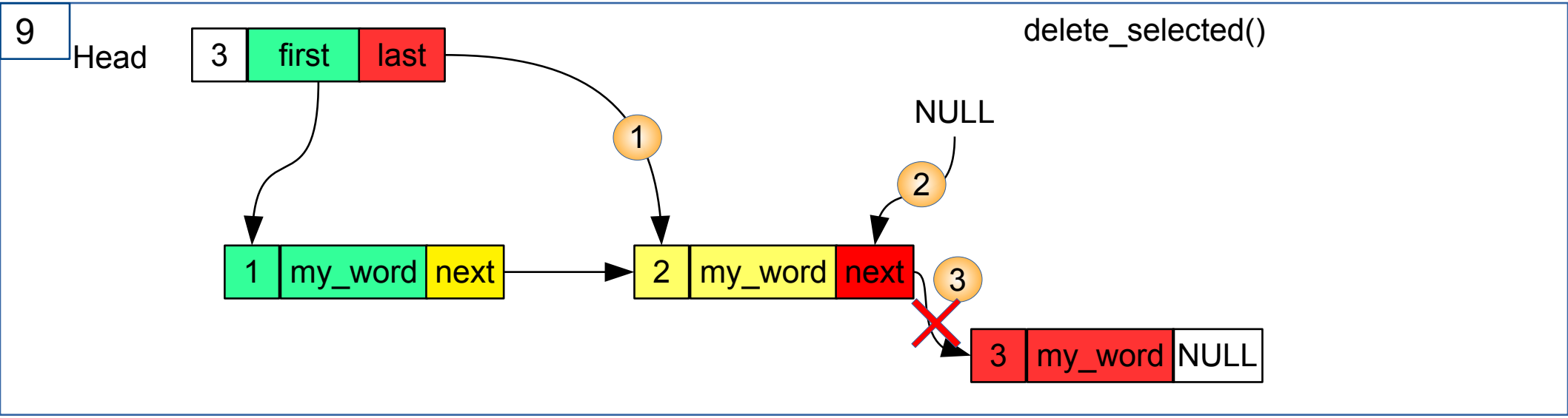


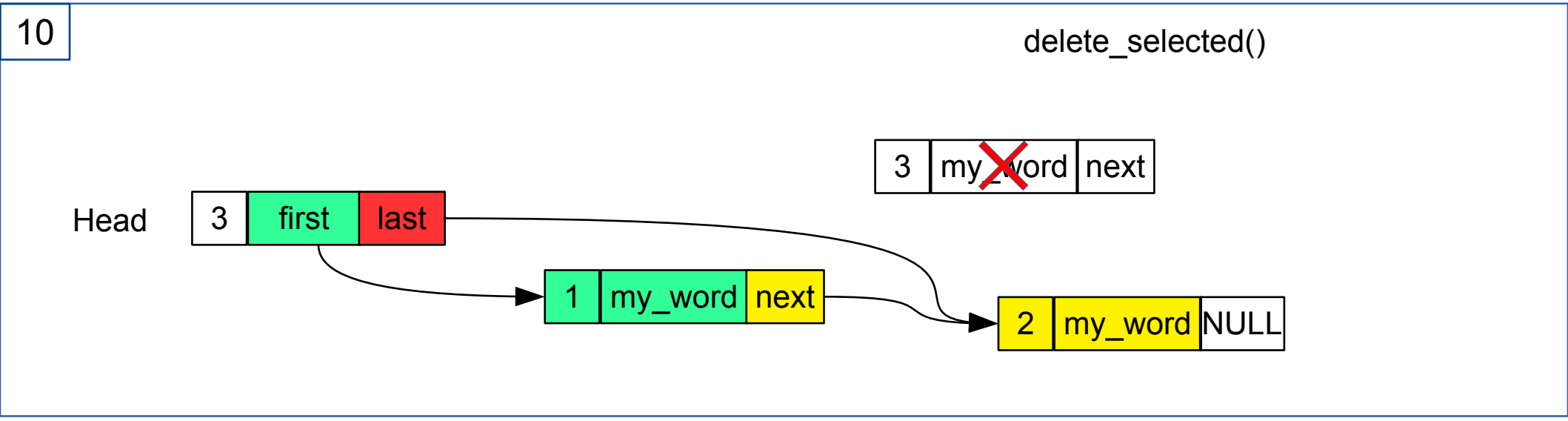
Удаление элемента списка





Удаление элемента списка





Удаление элемента списка

```
void delete_selected(Head *my_head, Node *current_node)
{
    Node *q, *q1;

    q=my_head->first;
    q1=my_head->last;
    if(current_node==q)
    {
        my_head->first=current_node->next;
        current_node->next=NULL;
        free(current_node);
    }
    else /* на следующем слайде */
```

Удаление элемента списка

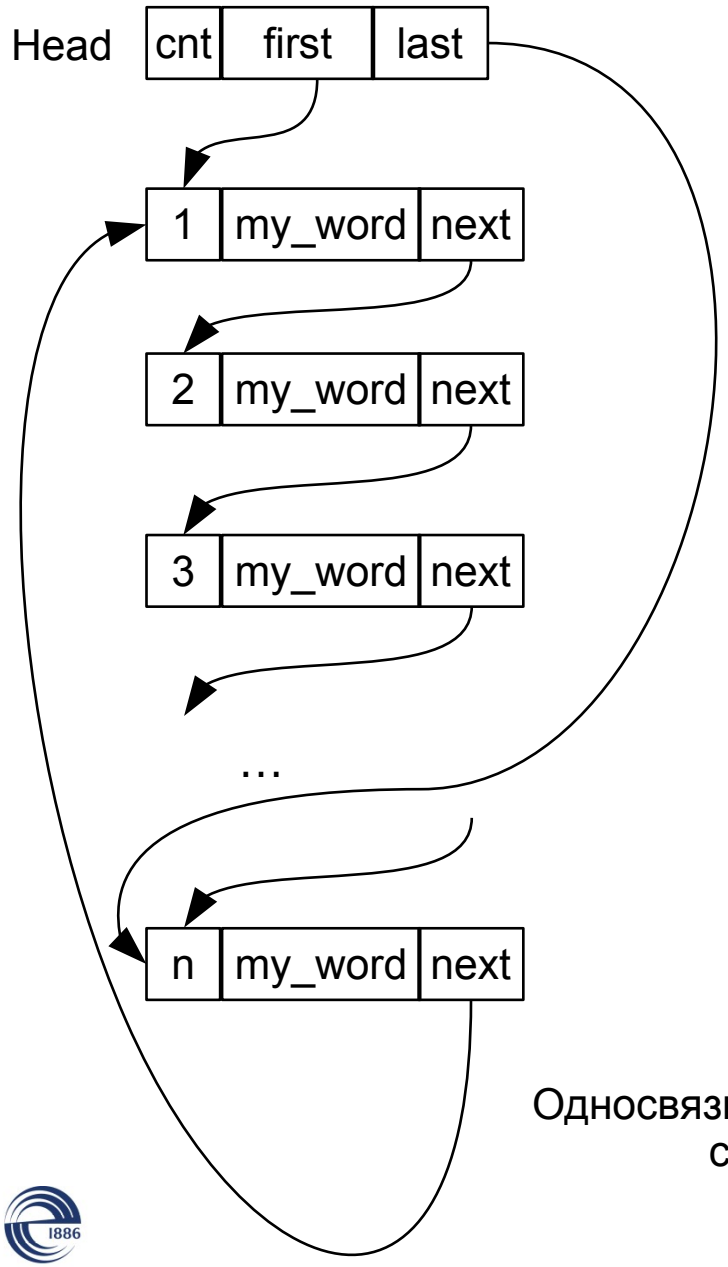
```
/* delete_selected() cont. */


    else
    {
        while(q!=NULL)
        {
            if(q->next==current_node)
            {
                if(current_node==q1) my_head->last=q;
                q->next=current_node->next;
                current_node->next=NULL;
                free(current_node);
            }
            else q=q->next;
        }
    }
}
```

Сортировка списка

Сортировка делается по значению какого-либо поля аналогично сортировке любого массива.

- Либо путем перестановки элементов (любой алгоритм, переназначение указателей) — все операции уже разобраны.
- Либо созданием из списка массива указателей на структуры, сортировка массива и обратное преобразование — тоже все понятно.

Кольцевой односвязный список



Односвязный кольцевой список

В односвязном кольцевом списке у «последнего» элемента **`*next`** является указателем на первый элемент

```
q=head→last;
q→next=head→first;
```

В кольцевом списке можно переставить **`head→first`** и **`head→last`** на любые два соседних элемента («перемещение головы»).

Кольцевой односвязный список

Особенности

- Проход по списку `while(q!=NULL)` ничего не дает. Нужно либо `while(q!=Head→first)`, либо использовать значение `cnt` из «головы» и цикл `for()`.
- При добавлении элемента перед первым в пустой список поле `next` нового элемента должно содержать его собственный адрес (если список не пустой — все как в линейном списке).
- При добавлении элемента после последнего в пустой список поле `next` нового элемента должно содержать его собственный адрес, а если если список не пустой – адрес первого элемента (`Head→first`) .
- При удалении элемента, если этот элемент является первым, необходимо изменить адрес, хранящийся в поле `next` последнего элемента кольцевого писка, на адрес элемента, следующего за удаляемым.