

Основы алгоритмизации и программирования

ФИСТ 1 курс

Власенко

Олег

Федосович

Лекция 11

Динамические структуры данных.

Списки.

Односвязный и двусвязный список.

Динамические структуры данных

«данные особой структуры, которые представляют собой отдельные элементы, связанные с помощью ссылок.

Каждый элемент (узел) состоит из двух областей памяти: поля данных и ссылок.

Ссылки – это адреса других узлов этого же типа, с которыми данный элемент логически связан.

В языке Си для организации ссылок используются переменные
- указатели.

При добавлении нового узла в такую структуру выделяется новый блок памяти и (с помощью ссылок) устанавливаются связи этого элемента с уже существующими.

Для обозначения конечного элемента в цепи используются нулевые ссылки (NULL).»

http://k504.khai.edu/attachments/article/762/devcpp_4.pdf

**Где и когда нужны динамические структуры
данных???**

Динамические структуры данных

Список односвязный

Список двусвязный

Циклический список

Дерево

Двоичное дерево

Двоичное дерево поиска

Графы

...

Еще раз о структурах

```
struct Line {  
    int x1, y1, x2, y2;  
};
```

```
struct Line newLine = {10, 10, 20, 10};  
struct Line * lines = NULL;
```

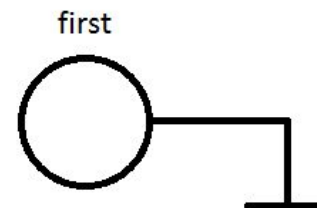
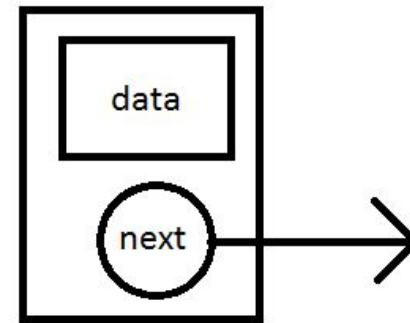
Еще раз о структурах (2)

```
struct Line {  
    int x1, y1, x2, y2;  
};
```

```
struct Line newLine = {10, 10, 20, 10};  
struct Line * lines = NULL;
```

```
struct Node {  
    int data;  
    struct Node * next;  
};
```

```
struct Node * first = NULL;
```

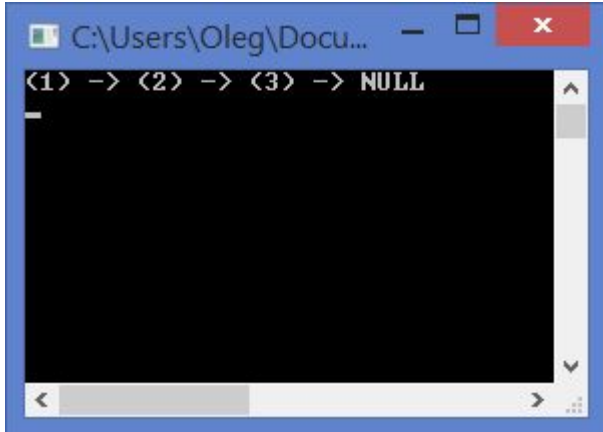


Отрабатываем навыки рисования

```
void main() {  
    struct Node node1 = {1, NULL};  
    struct Node node2 = { 2, NULL };  
    struct Node node3 = { 3, NULL };  
  
    first = &node1;  
    node1.next = &node2;  
    node2.next = &node3;  
  
    printList();  
}
```

Отрабатываем навыки рисования (2)

```
void printList() {  
  
    struct Node * ptr = first;  
    while (ptr != NULL) {  
        printf("(%d) -> ", ptr->data);  
        ptr = ptr->next;  
    }  
    printf("NULL\n");  
}
```

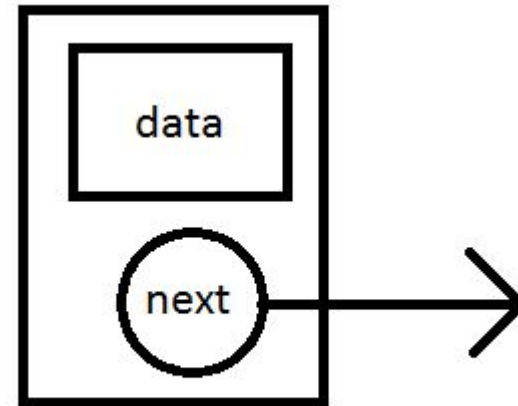


A screenshot of a Windows command prompt window. The title bar shows the path "C:\Users\Oleg\Docu...". The command prompt displays the output of the printList function: "<1> -> <2> -> <3> -> NULL". The window has a blue border and standard Windows window controls (minimize, maximize, close).

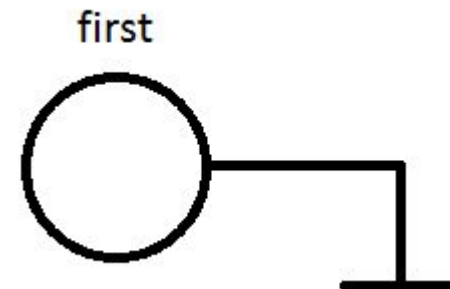
Связанный список в динамической памяти

```
#define _CRT_SECURE_NO_WARNINGS  
#include <stdio.h>
```

```
struct Node {  
    int data;  
    struct Node * next;  
};
```



```
struct Node * first = NULL;
```



СВЯЗАННЫЙ СПИСОК В ДИНАМИЧЕСКОЙ ПАМЯТИ

(2)

```
void printList() {  
    struct Node * ptr = first;  
    while (ptr != NULL) {  
        printf("(%d) -> ", ptr->data);  
        ptr = ptr->next;  
    }  
    printf("NULL\n");  
}
```

СВЯЗАННЫЙ СПИСОК В ДИНАМИЧЕСКОЙ ПАМЯТИ

(3)

```
void addToHead(int value) {  
  
    struct Node * newNode;  
    newNode = (struct Node*)malloc(sizeof(  
        struct Node));  
  
    newNode->next = first;  
    newNode->data = value;  
  
    first = newNode;  
}
```

СВЯЗАННЫЙ СПИСОК В ДИНАМИЧЕСКОЙ ПАМЯТИ

(4)

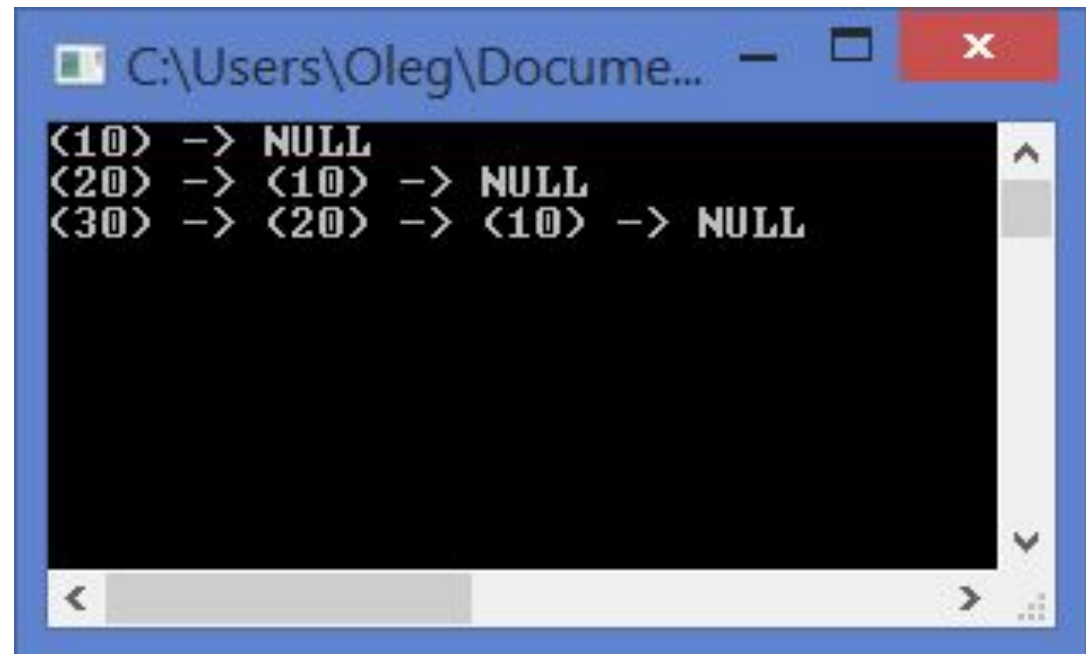
```
int deleteFromHead()
{
    int value = first->data;
    struct Node * delNode = first;

    first = first->next;
    free(delNode);

    return value;
}
```

СВЯЗАННЫЙ СПИСОК В ДИНАМИЧЕСКОЙ ПАМЯТИ (5)

```
void main() {  
  
    addToHead(10);  
    printList();  
  
    addToHead(20);  
    printList();  
  
    addToHead(30);  
    printList();  
}
```



```
C:\Users\Oleg\Docume...  
<10> -> NULL  
<20> -> <10> -> NULL  
<30> -> <20> -> <10> -> NULL
```

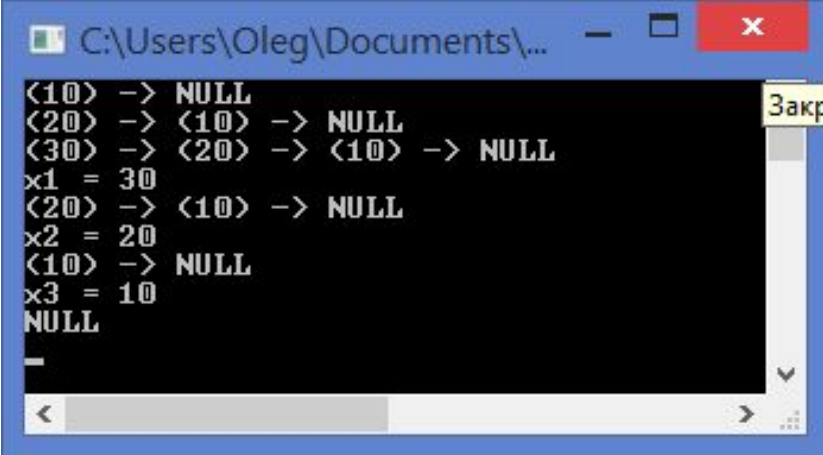
СВЯЗАННЫЙ СПИСОК В ДИНАМИЧЕСКОЙ ПАМЯТИ

(6)

```
int x1 = deleteFromHead();  
printf("x1 = %d\n", x1);  
printList();
```

```
int x2 = deleteFromHead();  
printf("x2 = %d\n", x2);  
printList();
```

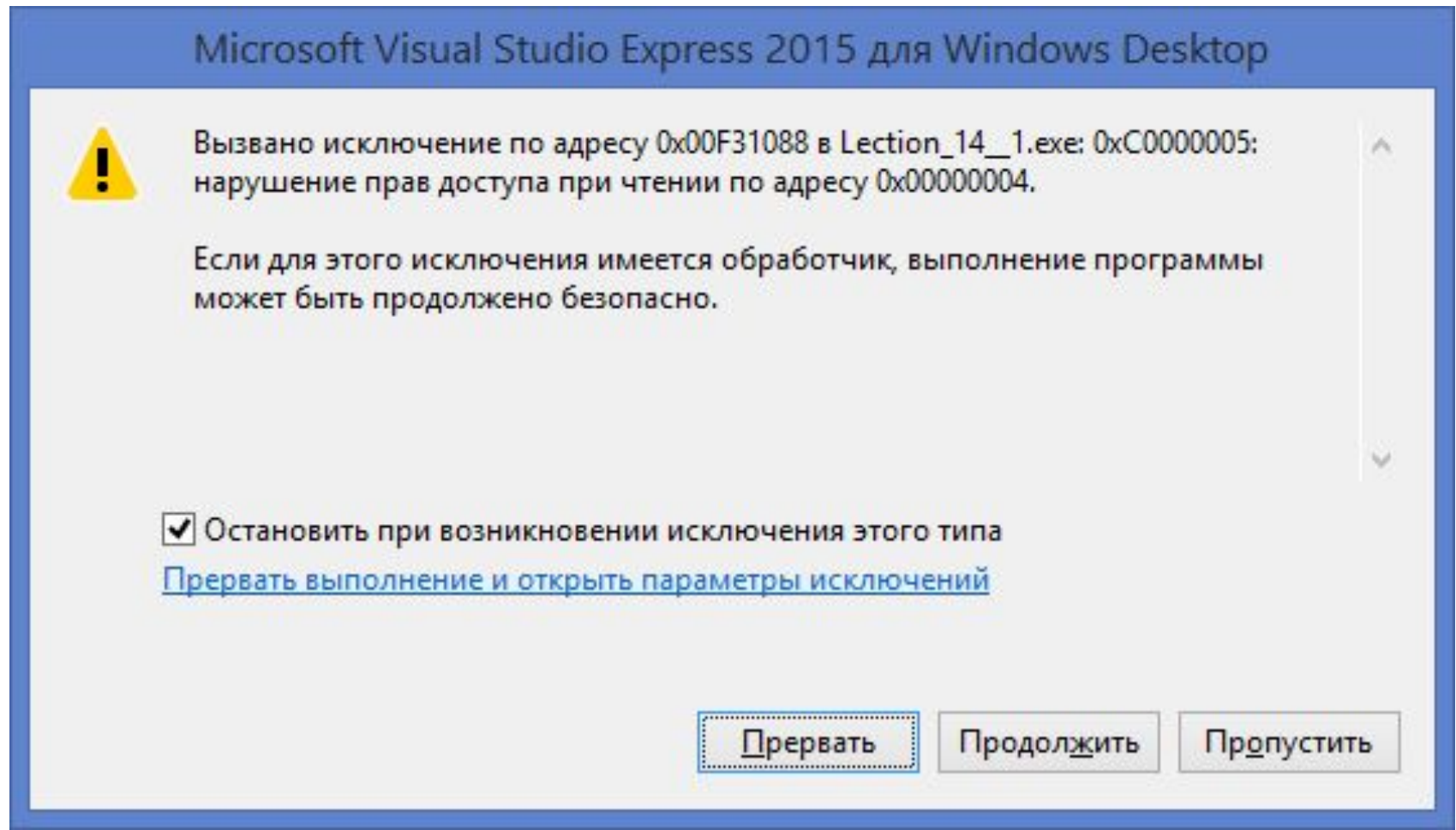
```
int x3 = deleteFromHead();  
printf("x3 = %d\n", x3);  
printList();
```



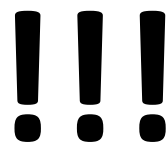
```
<10> -> NULL  
<20> -> <10> -> NULL  
<30> -> <20> -> <10> -> NULL  
x1 = 30  
<20> -> <10> -> NULL  
x2 = 20  
<10> -> NULL  
x3 = 10  
NULL
```

Связанный список в динамической памяти (7)

```
int x4 = deleteFromHead();  
printf("x4 = %d\n", x4);  
printList();  
}
```



И снова – урок рисования



Очистка всего списка

```
void clearList()
{
    while (first != NULL)
    {
        struct Node * delNode = first;
        first = first->next;
        free(delNode);
    }
}
```

Трассировка!!!

Защита от пустого списка

```
int deleteFromHead()
{
    if (first == NULL) {
        return -1;
    }

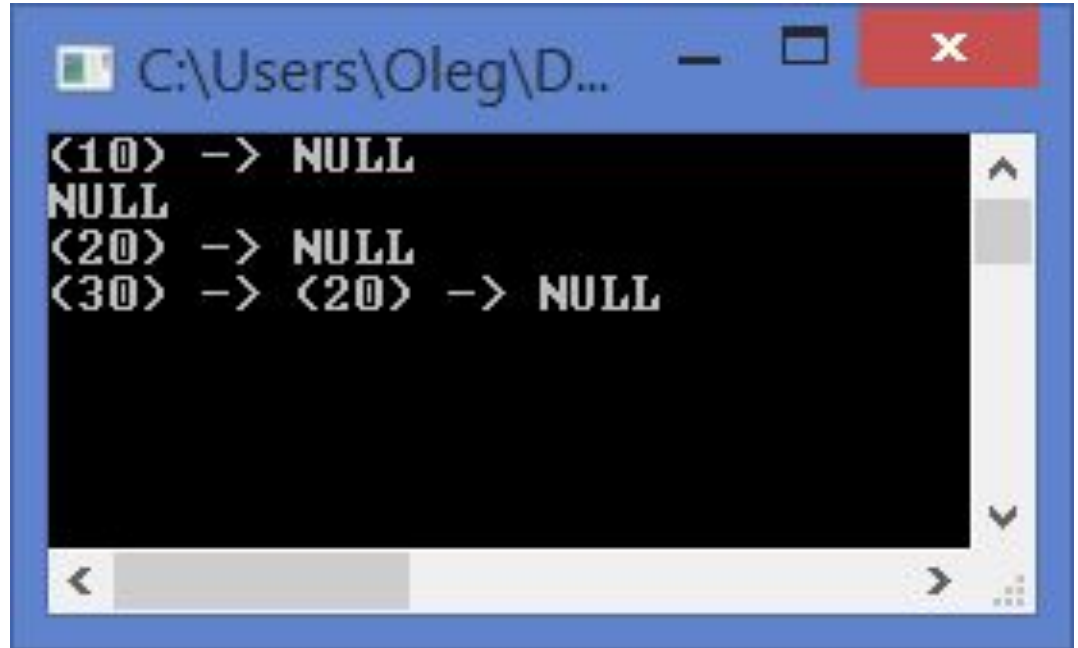
    int value = first->data;
    struct Node * delNode = first;

    first = first->next;
    free(delNode);

    return value;
}
```

Собираем все вместе

```
void main() {  
  
    addToHead(10);  
    printList();  
  
    clearList();  
    printList();  
  
    addToHead(20);  
    printList();  
  
    addToHead(30);  
    printList();  
}
```



```
<10> -> NULL  
NULL  
<20> -> NULL  
<30> -> <20> -> NULL
```

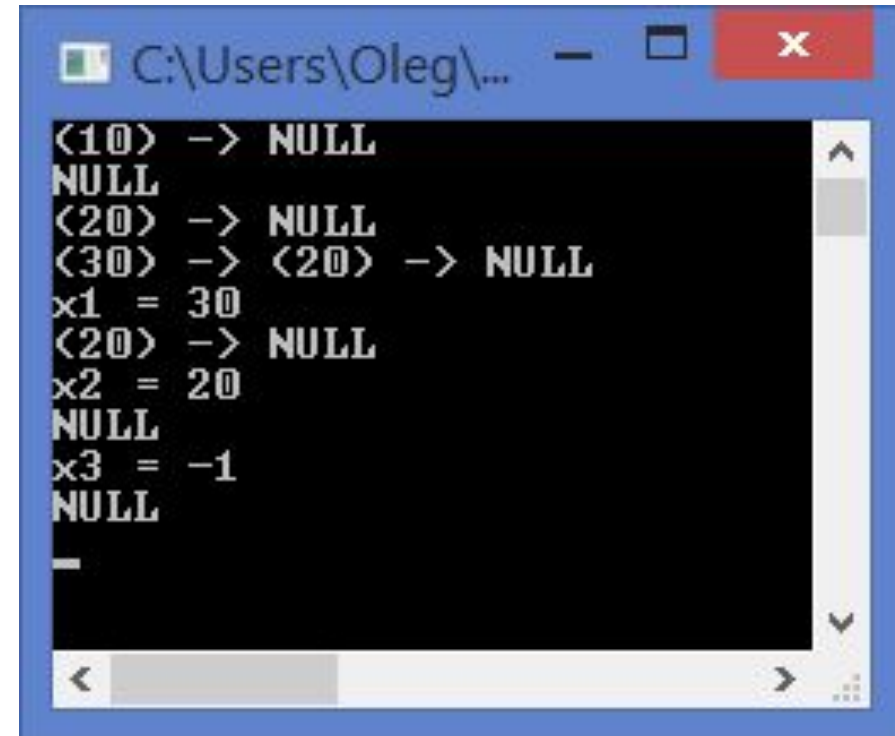
Собираем все вместе (2)

```
int x1 = deleteFromHead();  
printf("x1 = %d\n", x1);  
printList();
```

```
int x2 = deleteFromHead();  
printf("x2 = %d\n", x2);  
printList();
```

```
int x3 = deleteFromHead();  
printf("x3 = %d\n", x3);  
printList();
```

```
}
```



```
C:\Users\Oleg\...  
<10> -> NULL  
NULL  
<20> -> NULL  
<30> -> <20> -> NULL  
x1 = 30  
<20> -> NULL  
x2 = 20  
NULL  
x3 = -1  
NULL  
-
```

Вставляем элемент в конец списка

```
void addToTail(int value) {  
  
    struct Node * newNode;  
    newNode = (struct Node*)malloc(  
                sizeof(struct Node));  
    newNode->data = value;  
    newNode->next = NULL;  
}
```

Вставляем элемент в конец списка (2)

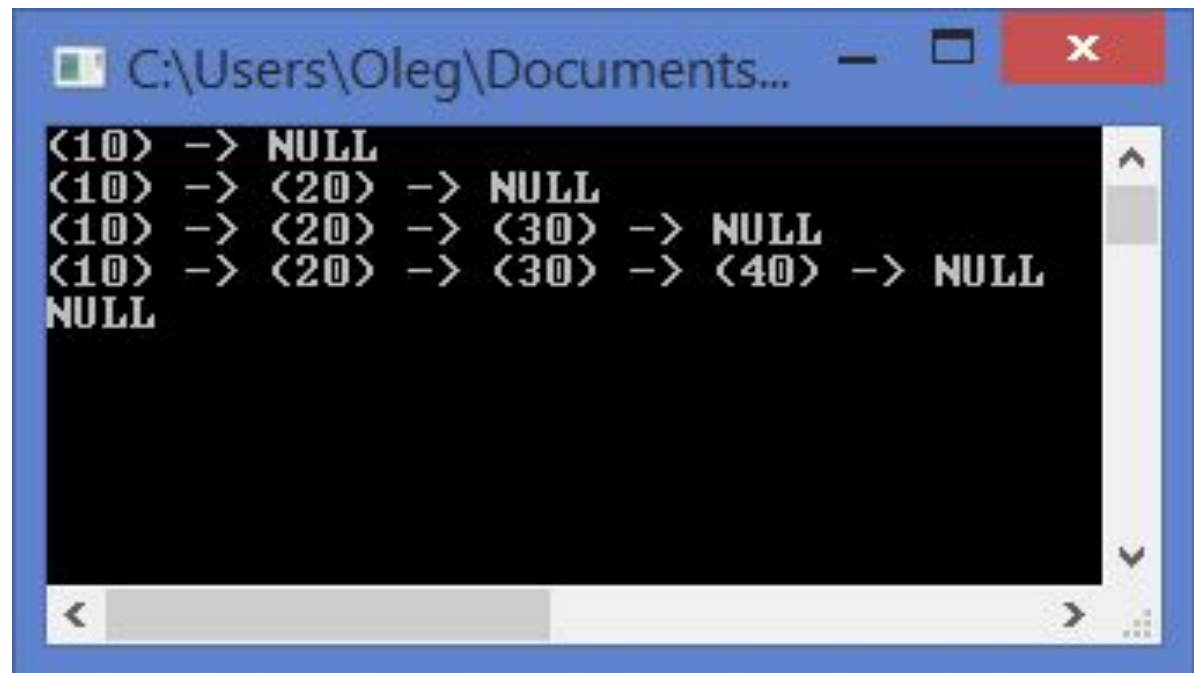
```
if (first == NULL) {  
    first = newNode;  
    return;  
}
```

```
if (first->next == NULL) {  
    first->next = newNode;  
    return;  
}
```

```
struct Node * last = first;  
while (last->next != NULL) {  
    last = last->next;  
}  
last->next = newNode;  
}
```

Вставляем элемент в конец списка (3)

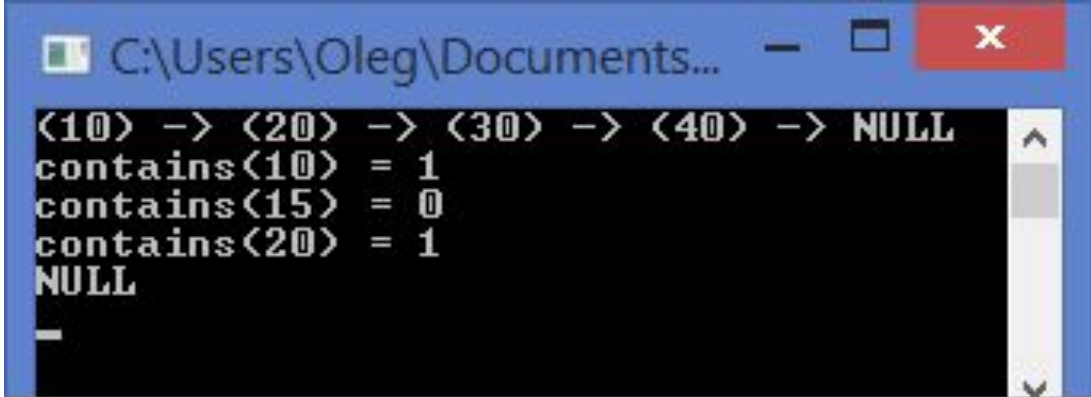
```
void main() {  
    addToTail(10);  
    printList();  
  
    addToTail(20);  
    printList();  
  
    addToTail(30);  
    printList();  
  
    addToTail(40);  
    printList();  
  
    clearList();  
    printList();  
}
```



```
C:\Users\Oleg\Documents...  
<10> -> NULL  
<10> -> <20> -> NULL  
<10> -> <20> -> <30> -> NULL  
<10> -> <20> -> <30> -> <40> -> NULL  
NULL
```

Проверка, есть ли элемент в списке

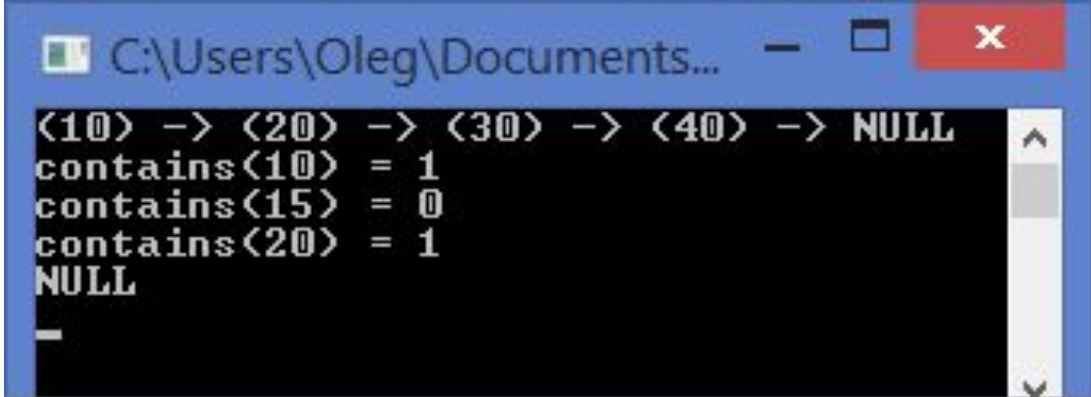
```
void main() {  
    addToTail(10);  
    addToTail(20);  
    addToTail(30);  
    addToTail(40);  
    printList();  
  
    printf("contains(10) = %d\n", contains(10));  
    printf("contains(15) = %d\n", contains(15));  
    printf("contains(20) = %d\n", contains(20));  
  
    clearList();  
    printList();  
}
```



```
<10> -> <20> -> <30> -> <40> -> NULL  
contains<10> = 1  
contains<15> = 0  
contains<20> = 1  
NULL  
-  
Users\Oleg\Documents\Visual Studio 2015\Projects\Lecti
```


Проверка, есть ли элемент в списке - код

```
int contains(int value)
{
    struct Node * ptr = first;
    while (ptr != NULL) {
        ...
    }
    return 0; // если так и не нашли элемента ==value
}
```



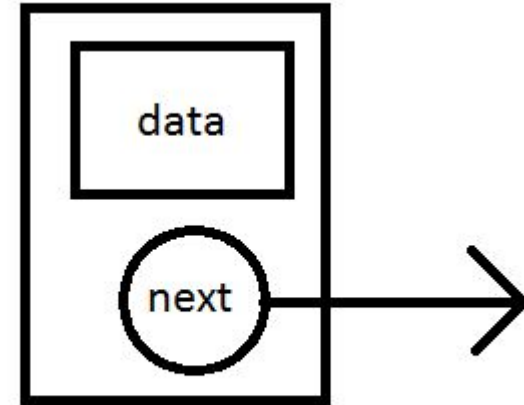
The screenshot shows a Windows command prompt window with a blue title bar. The title bar text is "C:\Users\Oleg\Documents...". The window content displays the following text:

```
<10> -> <20> -> <30> -> <40> -> NULL
contains<10> = 1
contains<15> = 0
contains<20> = 1
NULL
-
```

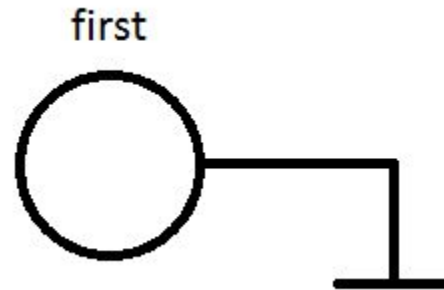
At the bottom of the image, there is a partial path: "Users\Oleg\Documents\Visual Studio 2015\Projects\Lecti".

Односвязный список

```
struct Node {  
    int data;  
    struct Node * next;  
};
```

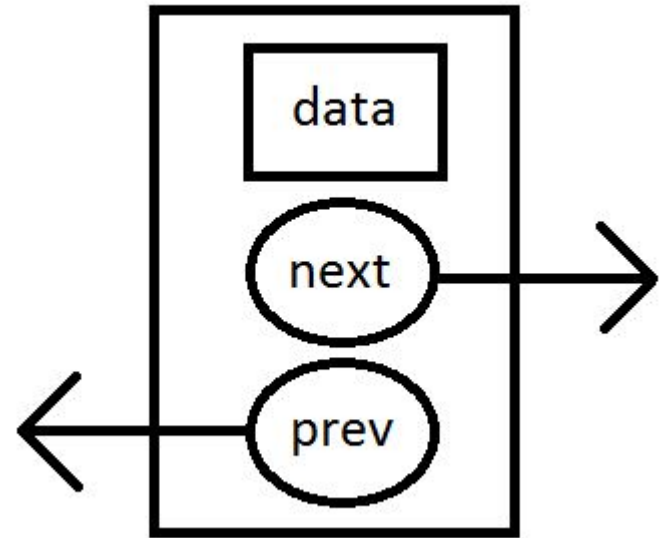


```
struct Node * first = NULL;
```

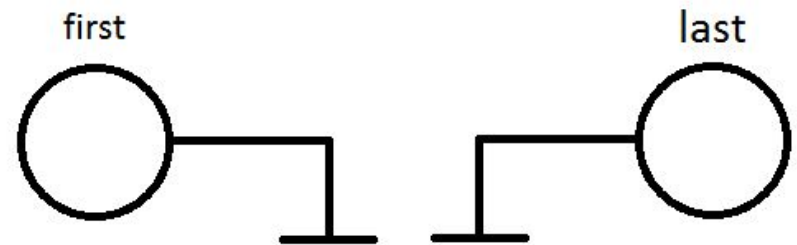


Двусвязный список

```
struct Node2 {  
    int data;  
    struct Node2 * next;  
    struct Node2 * prev;  
};
```



```
struct Node2 * first = NULL;  
struct Node2 * last = NULL;
```

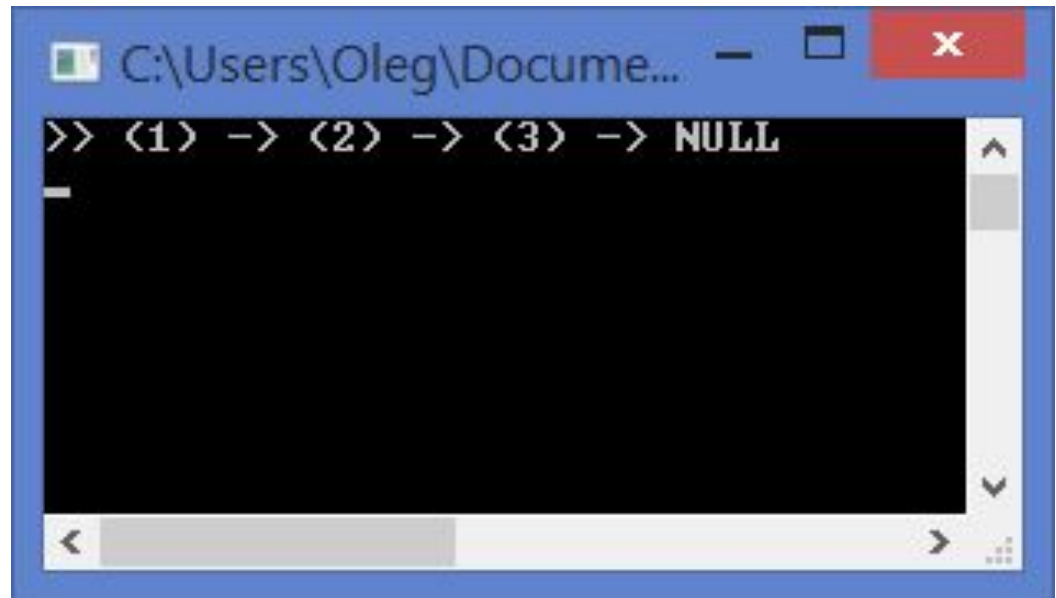


Отрабатываем навыки рисования

```
void main() {  
    struct Node2 node1 = { 1, NULL, NULL };  
    struct Node2 node2 = { 2, NULL, NULL };  
    struct Node2 node3 = { 3, NULL, NULL };  
    first = &node1;  
    node1.next = &node2;  
    node2.next = &node3;  
    last = &node3;  
    node3.prev = &node2;  
    node2.prev = &node1;  
  
    printList();  
    printListRev();  
}
```

Вывод списка

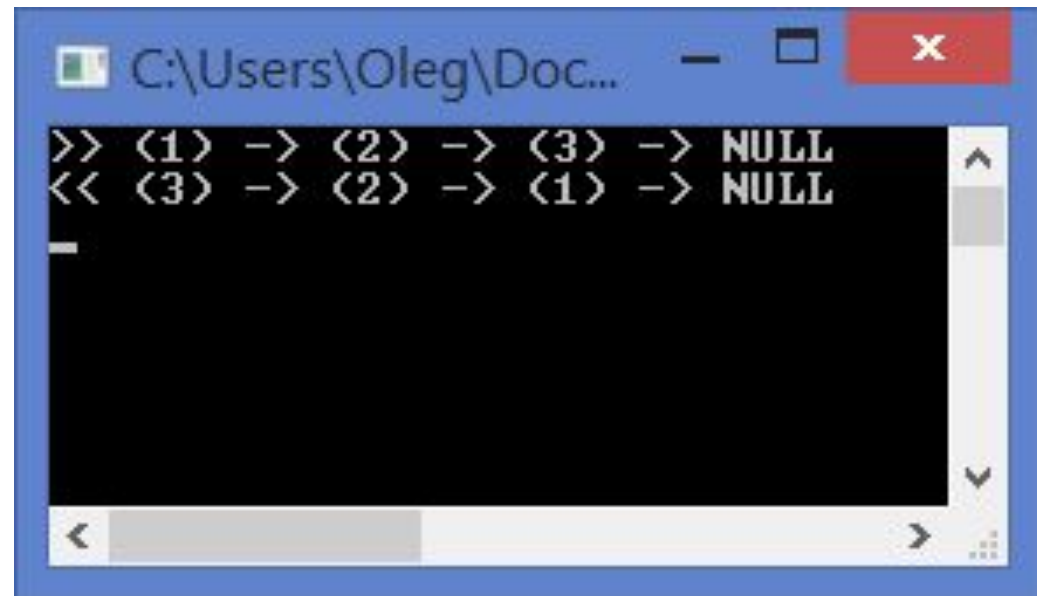
```
void printList() {  
    printf(">> ");  
  
    struct Node2 * ptr = first;  
    while (ptr != NULL) {  
        printf("(%d) -> ", ptr->data);  
        ptr = ptr->next;  
    }  
    printf("NULL\n");  
}
```



The screenshot shows a Windows command prompt window with a blue title bar. The title bar text is partially visible as "C:\Users\Oleg\Docume...". The command prompt displays the output of the printList function: ">> <1> -> <2> -> <3> -> NULL". The output is displayed in white text on a black background. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Вывод списка от конца в начало

```
void printListRev() {  
    printf("<< ");  
  
    struct Node2 * ptr = last;  
    while (ptr != NULL) {  
        printf("(%d) -> ", ptr->data);  
        ptr = ptr->prev;  
    }  
    printf("NULL\n");  
}
```



The screenshot shows a Windows command prompt window with the following output:

```
>> <1> -> <2> -> <3> -> NULL  
<< <3> -> <2> -> <1> -> NULL
```

The window title bar indicates the path C:\Users\Oleg\Doc... and includes standard Windows window controls (minimize, maximize, close).

Добавление элемента в голову списка

```
void addToHead(int value) {
    struct Node2 * newNode = (struct Node2*)
        malloc(sizeof(struct Node2));
    newNode->next = first;
    newNode->data = value;
    newNode->prev = NULL;

    if (first == NULL) {
        last = newNode;
        first = newNode;
    } else {
        first->prev = newNode;
        first = newNode;
    }
}
```

Добавление элемента в хвост списка

```
void addToTail(int value) {  
    struct Node2 * newNode = (struct Node2*)  
        malloc(sizeof(struct Node2));  
    newNode->next = NULL;  
    newNode->data = value;  
    newNode->prev = last;  
  
    if (last == NULL) {  
        first = newNode;  
        last = newNode;  
    } else {  
        last->next = newNode;  
        last = newNode;  
    }  
}
```


Пример добавления элементов в список

```
void main() {  
    addToHead(10);  
    addToHead(20);  
    addToHead(30);  
    addToHead(40);  
  
    addToTail(110);  
    addToTail(120);  
    addToTail(130);  
    addToTail(140);  
  
    printList();  
    printListRev();  
}
```

Что будет выведено???

Пример добавления элементов в список

```
void main() {  
    addToHead(10);  
    addToHead(20);  
    addToHead(30);  
    addToHead(40);  
  
    addToTail(110);  
    addToTail(120);  
    addToTail(130);  
    addToTail(140);  
  
    printList();  
    printListRev();  
}
```

```
C:\Users\Oleg\Documents\Visual Studio 2015\Projects\Lectio...  
>> <40> -> <30> -> <20> -> <10> -> <110> -> <120> -> <130> -> <140> -> NULL  
<< <140> -> <130> -> <120> -> <110> -> <10> -> <20> -> <30> -> <40> -> NULL  
-  
<
```

Очистка списка

```
void clearList()
{
    while (first != NULL) {
        struct Node2 * delNode = first;
        first = first->next;
        free(delNode);
    }

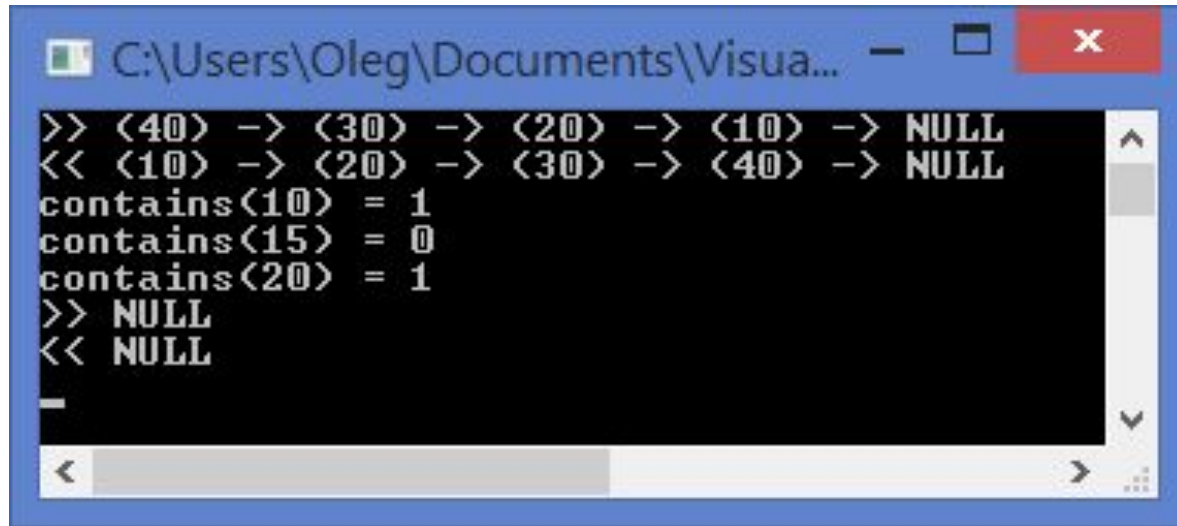
    last = NULL;
}
```

Список содержит элемент?

```
int contains(int value)
{
    struct Node2 * ptr = first;
    while (ptr != NULL) {
        if (ptr->data == value) {
            return 1;
        }
        ptr = ptr->next;
    }
    return 0;
}
```

Вызов clearList() и contains()

```
void main() {  
    addToHead(10);  
    addToHead(20);  
    addToHead(30);  
    addToHead(40);  
    printList();  
    printListRev();
```



```
>> <40> -> <30> -> <20> -> <10> -> NULL  
<< <10> -> <20> -> <30> -> <40> -> NULL  
contains<10> = 1  
contains<15> = 0  
contains<20> = 1  
>> NULL  
<< NULL
```

```
printf("contains(10) = %d\n", contains(10));  
printf("contains(15) = %d\n", contains(15));  
printf("contains(20) = %d\n", contains(20));
```

```
clearList();  
printList();  
printListRev();
```

```
}
```

Удаление элемента из головы

```
int deleteFromHead()
```

```
{
```

```
    if (first == NULL) {
```

```
        return -1;
```

```
    }
```

```
int value = first->data;
```

```
struct Node2 * delNode = first;
```

Удаление элемента из головы (2)

```
if (first == last) {  
    first = NULL;  
    last = NULL;  
    free(delNode);  
}  
else {  
    first = first->next;  
    first->prev = NULL;  
    free(delNode);  
}  
  
return value;  
}
```

Удаление элемента из хвоста (1)

```
int deleteFromTail()
{
    if (last == NULL) {
        return -1;
    }

    int value = last->data;
    struct Node2 * delNode = last;
```


Удаление элемента из хвоста (2)

```
if (first == last) {  
    first = NULL;  
    last = NULL;  
    free(delNode);  
}  
else {  
    last = last->prev;  
    last->next = NULL;  
    free(delNode);  
}  
  
return value;  
}
```

Тестирование удаления

```
void main() {  
    addToHead(10);  
    addToHead(20);  
    addToHead(30);  
    addToHead(40);  
    printList();  
    printListRev();  
  
    int x1 = deleteFromHead();  
    printf("deleteFromHead(): x1 = %d\n", x1);  
    printList();  
    printListRev();
```

Тестирование удаления (2)

```
int x2 = deleteFromTail();  
printf("deleteFromTail(): x2 = %d\n", x2);  
printList();  
printListRev();
```

```
int x3 = deleteFromHead();  
printf("deleteFromHead(): x3 = %d\n", x3);  
printList();  
printListRev();
```

```
}
```

```
C:\Users\Oleg\Documents\Vi...  
>> <40> -> <30> -> <20> -> <10> -> NULL  
<< <10> -> <20> -> <30> -> <40> -> NULL  
deleteFromHead(): x1 = 40  
>> <30> -> <20> -> <10> -> NULL  
<< <10> -> <20> -> <30> -> NULL  
deleteFromTail(): x2 = 10  
>> <30> -> <20> -> NULL  
<< <20> -> <30> -> NULL  
deleteFromHead(): x3 = 30  
>> <20> -> NULL  
<< <20> -> NULL
```

Домашнее задание

1. Делайте текущие лабораторные работы и сдавайте домашние работы!
2. Читайте про списки – см следующий слайд!

Источники информации

- <http://www.intuit.ru/studies/courses/648/504/lecture/11456>
- Динамические структуры данных:
однонаправленные и двунаправленные списки
- http://k504.khai.edu/attachments/article/762/devcpp_4.pdf -
Динамические структуры данных