

# **Основы программирования - Java**

**ФИСТ 1 курс**

**Власенко Олег Федосович**

**Лекция 6  
Списки**

# Списки

- Вспоминаем Си
- Односвязный список
- Двусвязный список
- Java – специфика реализации списков в Java

# 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.

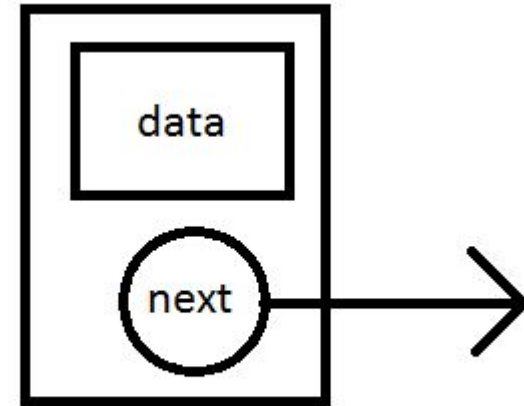
Для обозначения конечного элемента в цепи используются нулевые ссылки (NULL).»

[http://k504.khai.edu/attachments/article/762/devcpp\\_4.pdf](http://k504.khai.edu/attachments/article/762/devcpp_4.pdf)

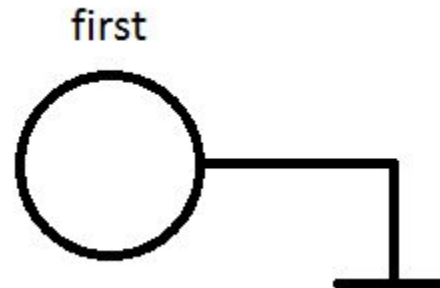
**Где и когда нужны динамические  
структуры данных???**

# Односвязный список

```
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();  
}
```

# 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)

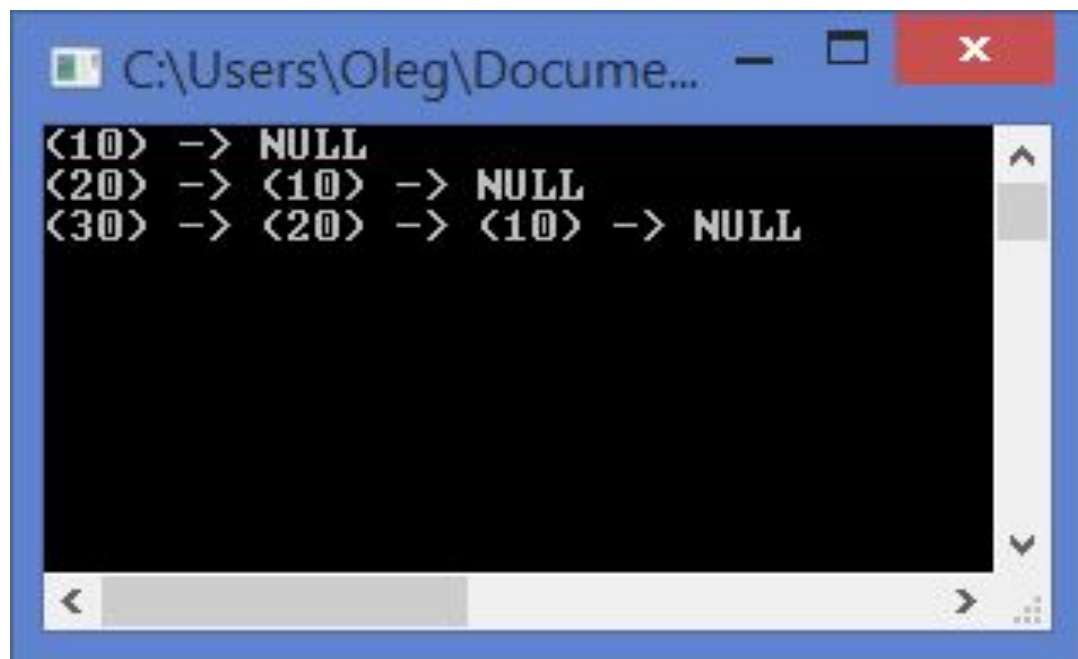
```
int deleteFromHead()
{
    int value = first->data;
    struct Node * delNode = first;

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

    return value;
}
```

# Связанный список в динамической памяти (5)

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



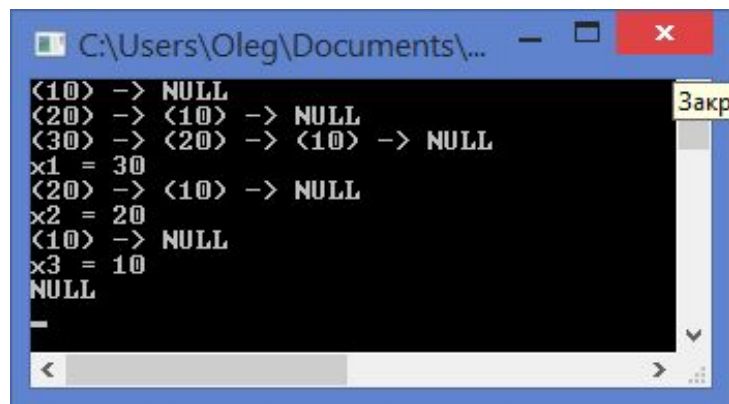
```
C:\Users\Oleg\Docume...  
<10> -> NULL  
<20> -> <10> -> NULL  
<30> -> <20> -> <10> -> NULL
```

# Связанный список в динамической памяти (6)

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

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

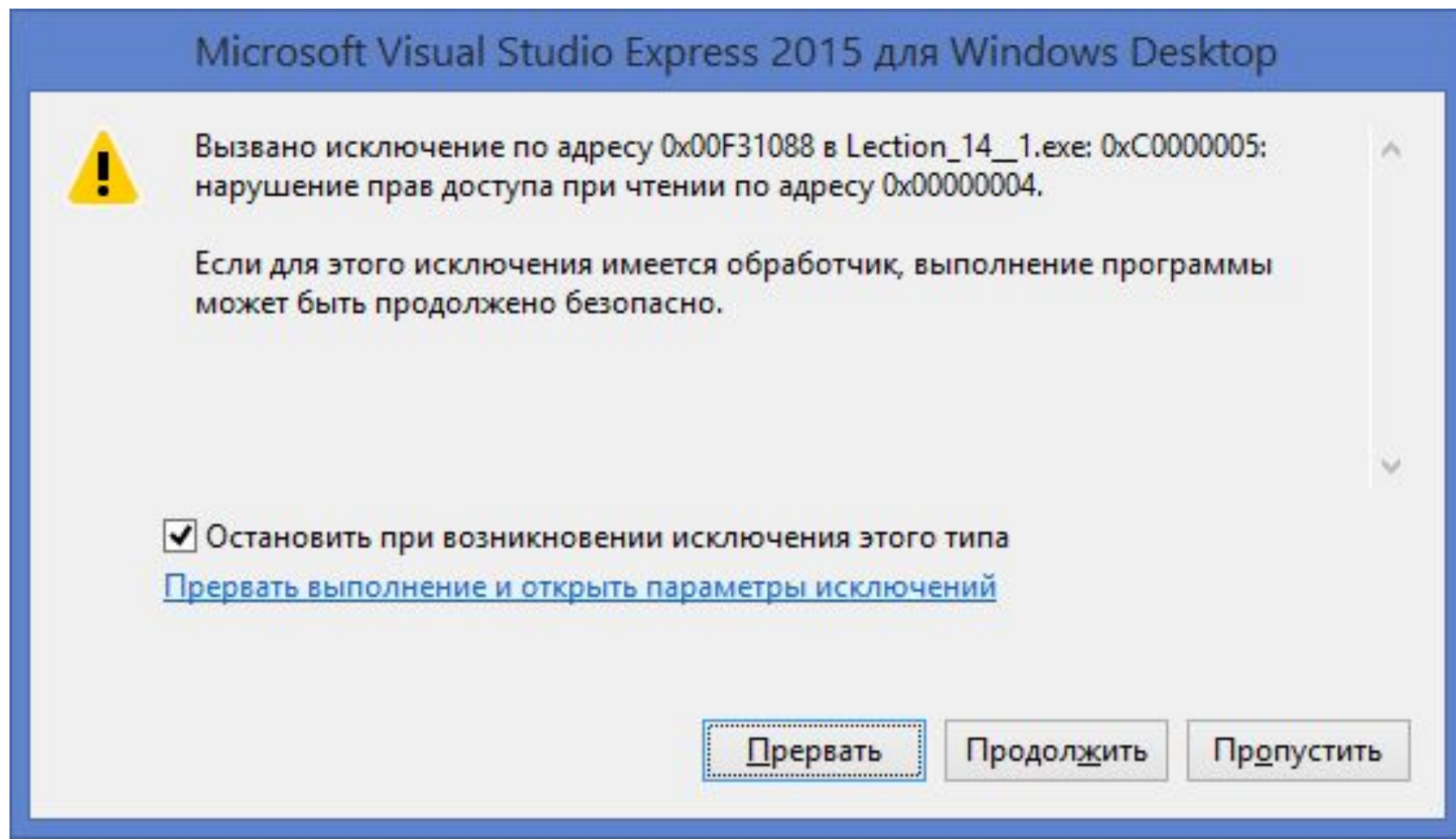
```
int x3 = deleteFromHead();  
printf("x3 = %d\n", x3);  
printList();
```



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

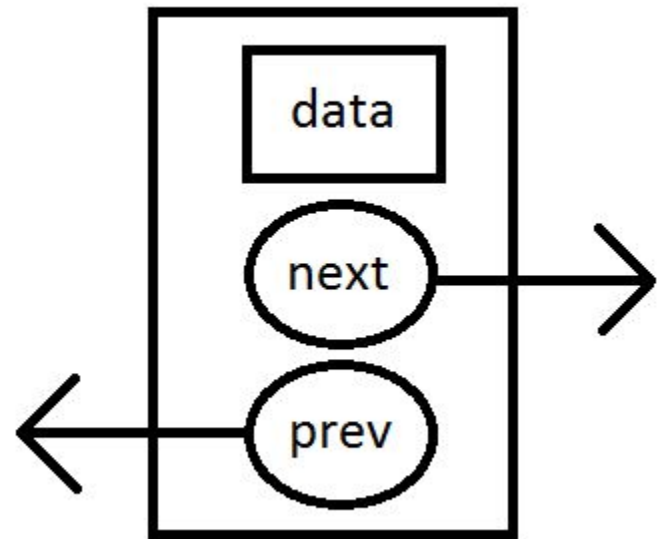
# Связанный список в динамической памяти (7)

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

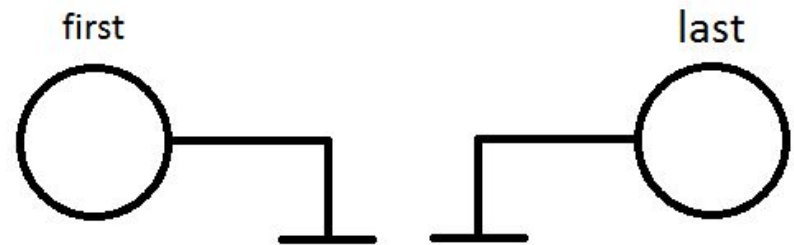


# Двусвязный список

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



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



# Список на Java

# Интерфейс списка

```
public interface IList {  
  
    void insertToHead(int key);  
    void deleteFromHead();  
    int getHeadElement();  
    boolean contains(int key);  
    String toString();  
  
}
```



# Класс узла

```
class Node {  
    int key;  
  
    Node next;  
    Node prev; // previous  
  
    public Node(int key, Node next, Node prev) {  
        this.key = key;  
        this.next = next;  
        this.prev = prev;  
    }  
}
```

# Класс списка (1)

```
public class List implements IList {  
    Node head; // first  
    Node tail; // last  
  
    public List() {  
        head = new Node(0, null, null);  
        tail = new Node(0, head, head);  
        head.next = tail;  
        head.prev = tail;  
    }  
}
```

# Класс списка (2)

@Override

```
public String toString() {  
    String str = "<<";  
  
    Node p = head.next;  
    while (p != tail) {  
        str = str + p.key + " ";  
        p = p.next;  
    }  
  
    str = str + ">>";  
    return str;  
}
```

# Класс списка (3)

@Override

```
public void insertToHead(int key) {  
    Node p = new Node(key, head.next, head);  
    head.next.prev = p;  
    head.next = p;  
  
}
```

# Класс списка (4)

@Override

```
public void deleteFromHead() {
```

```
    if (head.next == tail) {
```

```
        return;
```

```
    }
```

```
    Node delNext = head.next.next;
```

```
    delNext.prev = head;
```

```
    head.next = delNext;
```

```
}
```

# Класс списка (5)

```
@Override
```

```
public int getHeadElement() {
```

```
    return head.next.key;
```

```
}
```

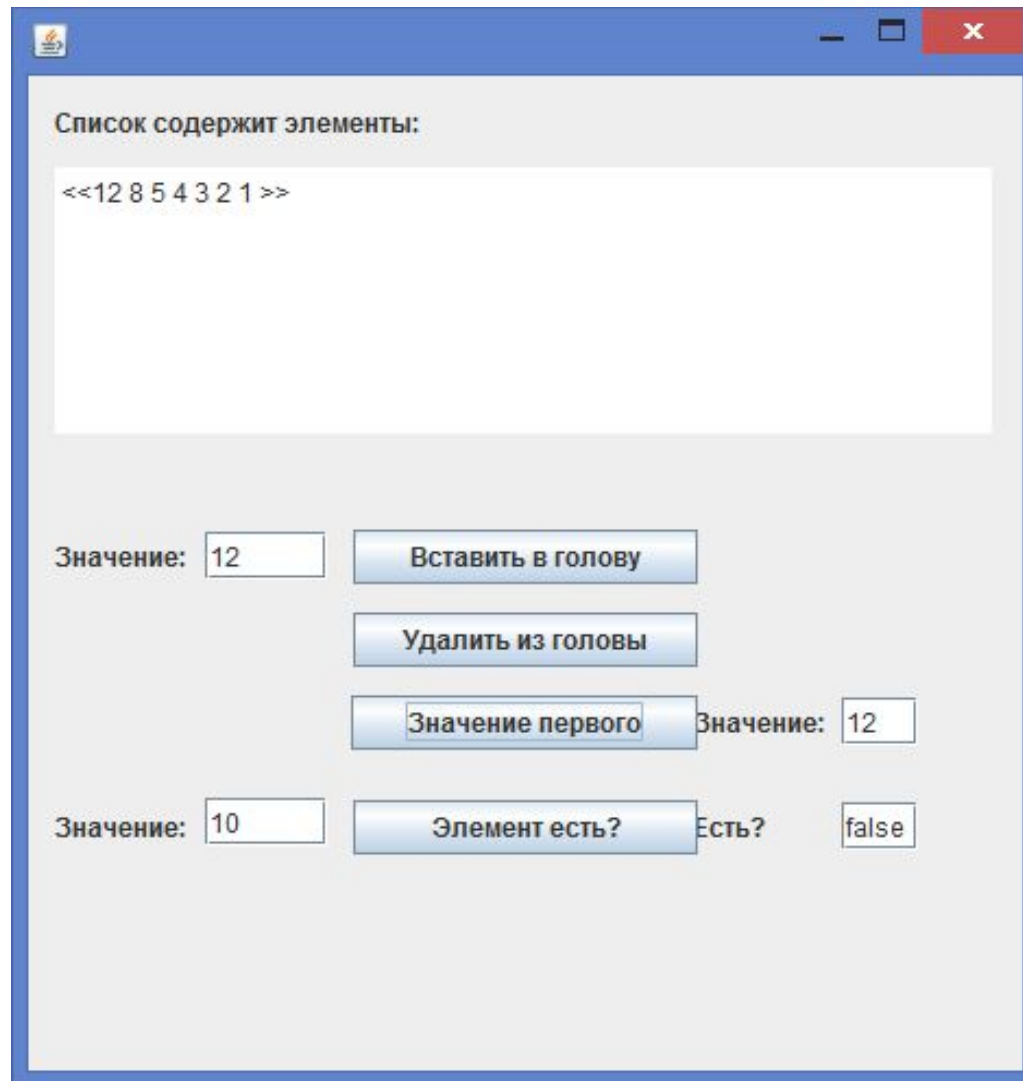
# Класс списка (6)

@Override

```
public boolean contains(int key) {  
    Node p = head.next;  
    while (p != tail) {  
        if (p.key == key) {  
            return true;  
        }  
        p = p.next;  
    }  
    return false;  
}
```

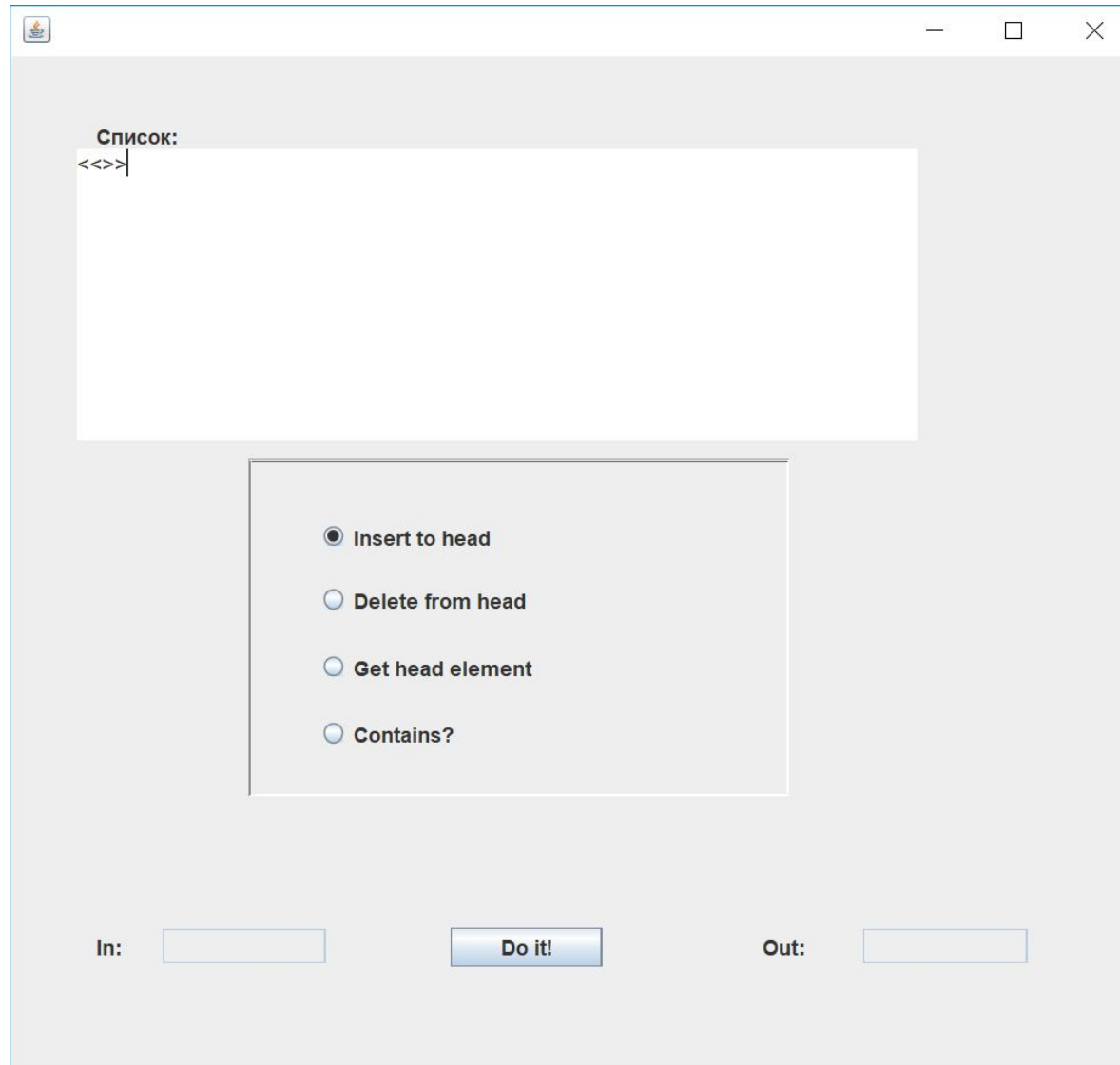
```
} // public class List implements IList {
```

# GUI для проб со списком





# GUI для проб со списком



# Спасибо за внимание!

Власенко Олег Федосович

E-mail: [vlasenko.oleg@gmail.com](mailto:vlasenko.oleg@gmail.com)

Vk: [vk.com/oleg.f.vlasenko](https://vk.com/oleg.f.vlasenko)

Телефон: +7 902 246 05 47