NATIONAL RESEARCH
UNIVERSITY

АБСТРАКТНЫЕ ТИПЫ ДАННЫХ ПОСЛЕДОВАТЕЛЬНОСТИ, СПИСКИ

Максименкова Ольга Вениаминовна

Младший научный сотрудник МНУЛ ИССА

Старший преподаватель Департамента программной инженерии
Факультета компьютерных наук

Цели

Познакомиться

- С контейнерами и последовательностями, как с абстрактными типами данных;
- С динамическими структурами данных;
- Со способами реализации этих представлений на языке C++

Получить навыки

- Реализации некоторых АТД на языке C++

Соглашения о терминологии

- **Абстрактный Тип Данных** (*АТД*) [*abstract logic design*] – функциональное описание некоторого класса сущностей в терминах операций, которые могут выполняться над ними.
- **Интерфейс АТД** – формальное и однозначное описание синтаксиса и семантики операций, которые могут выполняться над экземплярами АТД.

ТАК ЖЕ, КАК ОПИСАНИЕ ЯЗЫКА ПРОГРАММИРОВАНИЯ НЕ ОПРЕДЕЛЯЕТ ОСОБЕННОСТИ ЕГО РЕАЛИЗАЦИИ, ТАК И ИНТЕРФЕЙС АТД НЕ ОПРЕДЕЛЯЕТ РЕАЛИЗАЦИЮ АТД.

Контейнер

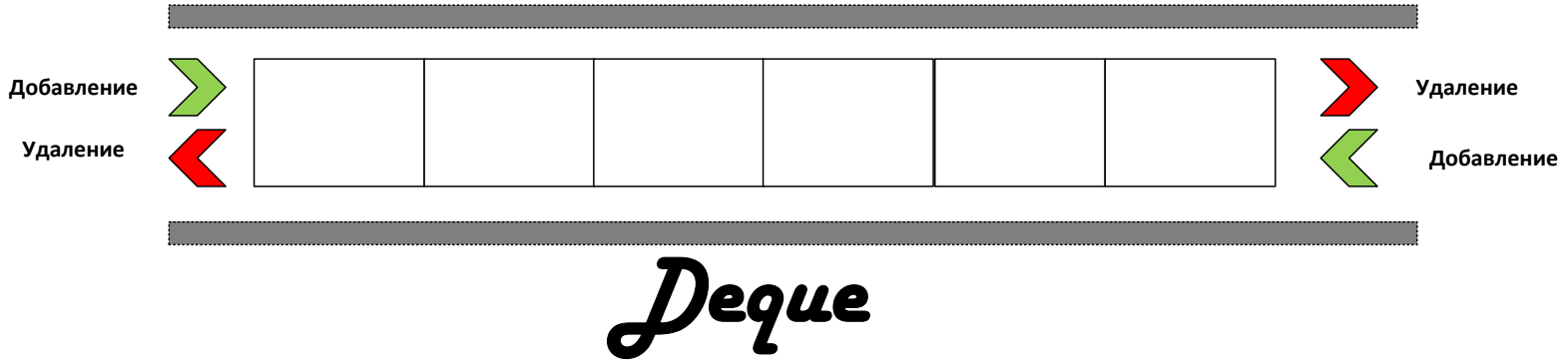
- **Контейнер** [*container*] – Абстрактный тип данных, представляющий собой структурированную коллекцию информационных элементов, доступ к которым определяется структурой контейнера.
- **Добавление** и **удаление** элементов контейнера назовём его **трансформацией**.
- **Доступ к элементу контейнера** – операция получения или изменения значения этого элемента.
- **Последовательность** [*sequence*] – контейнер, в котором элементы упорядочены по *индексам* (пронумерованы).

Некоторые виды последовательностей

- **Вектор** [*vector*]
  - последовательность, в которой возможен доступ к любому элементу по *индексу* элемента
- **Дек**
- **Стек**
- **Очередь**

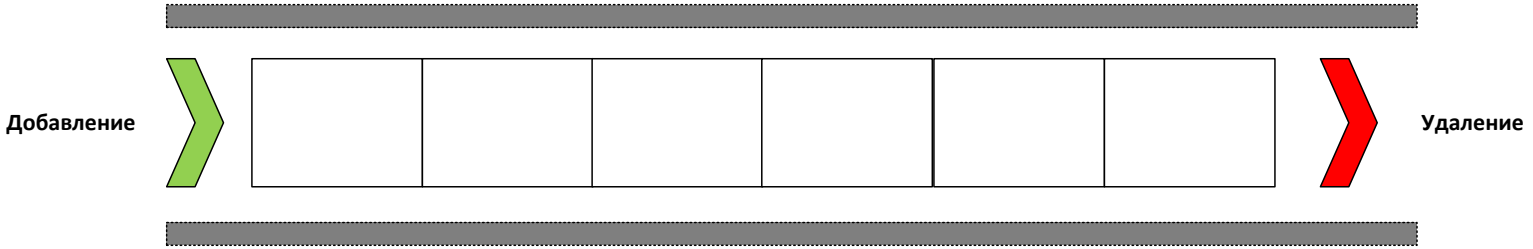
Дек

- **Дек** [*deque*, *double ended queue*] – последовательность, в которой возможны только:
  1. **доступ**: к концевым элементам;
  2. **добавление**: до начального и после конечного элемента;
  3. **удаление**: концевых элементов.



Очередь

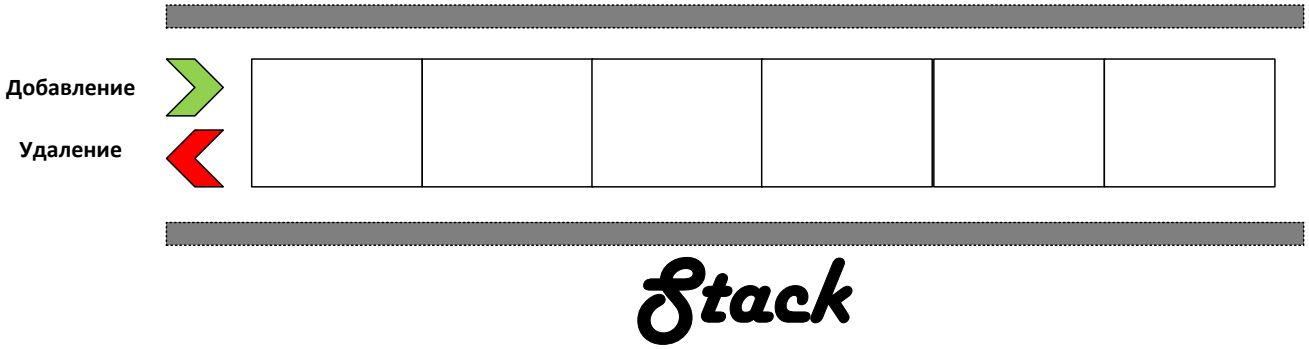
- **Очередь** [queue] – дек, в котором возможны только:
  1. **доступ**: к начальному элементу;
  2. **добавление**: после конечного элемента;
  3. **удаление**: начального элемента.
- Конечный элемент очереди часто называют **хвостом очереди**, а начальный – **головой очереди**.



Queue

Стек

- **Стек** [stack] – дек, в котором возможны только:
  1. **доступ**: к конечному элементу;
  2. **добавление**: после конечного элемента;
  3. **удаление**: конечного элемента.
- Конечный элемент стека называют **вершиной стека**.



NATIONAL RESEARCH
UNIVERSITY

Списки

Односвязный

Двусвязный

Список

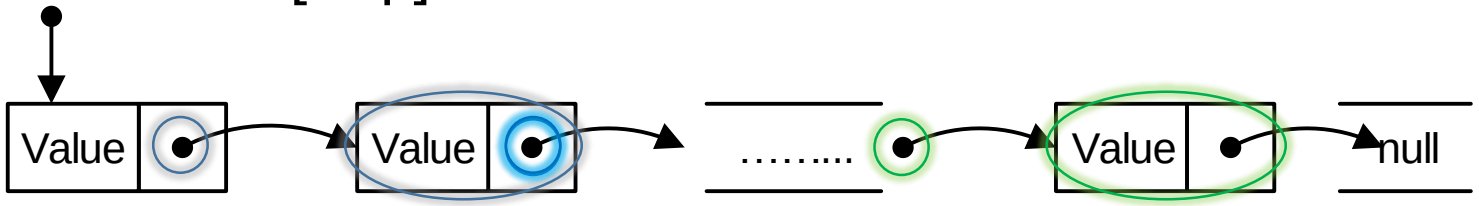
- **Список** представляет собой такую реализацию последовательности однотипных элементов, когда элементы связаны друг с другом посредством ссылок.
- Каждый элемент списка содержит не только информационное поле, но одно или более полей со ссылками на другие элементы.

Наиболее распространённые списки

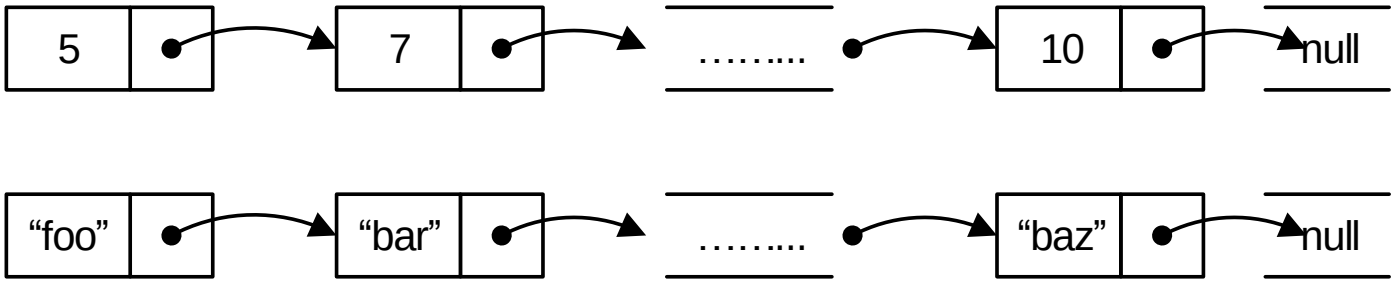
- **односвязные списки**
  - каждый элемент, кроме последнего, содержит ссылку на следующий, последний элемент ссылается на **null**
- **двухсвязные списки**
  - каждый элемент, кроме первого и последнего, содержит ссылки на предыдущий и следующий

Односвязный список

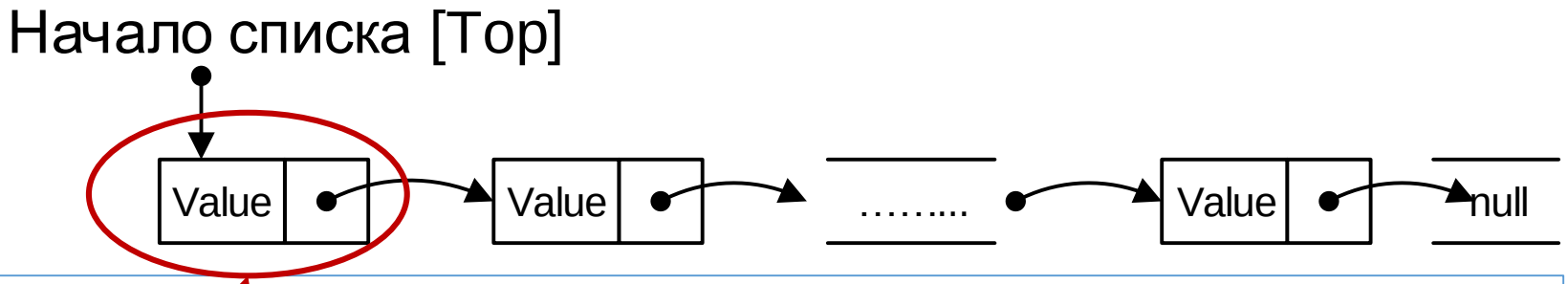
Начало списка [Top]



Примеры



Модель элемента односвязного списка




```
#include "iostream"
using namespace std;
struct ListItem {
  int n;
  ListItem* next;
};

void main() {
  ListItem head; // создаём структуру
  head.n = 1; // значение поля структуры
  head.next = NULL; // значение поля типа указатель
}
```

Как добавить элемент в односвязный список?

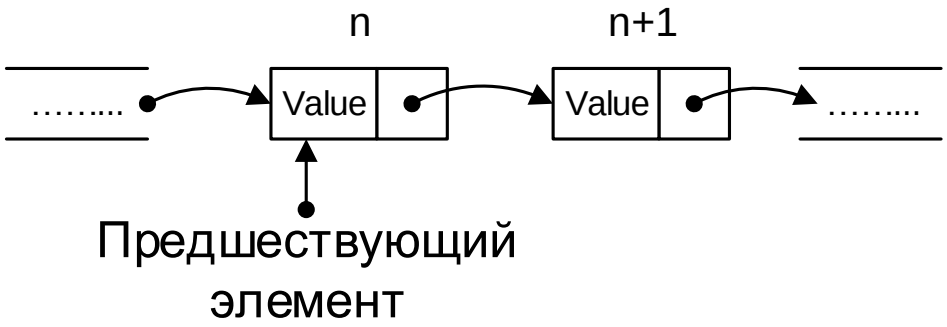
```
// добавим элемент
ListItem newItem;
newItem.n = 2;
newItem.next = NULL;
// head должен "указывать" на новый элемент
head.next = &newItem;

// пройдём по списку, нужен "указатель"
ListItem* curr = &head;
while (curr != NULL) {
  cout << curr->n << endl;
  curr = curr->next;
}
system("pause");
```

Самостоятельно напишем функцию, добавления элемента в конец односвязного списка → следующий слайд

Вставка элемента в односвязный список

Пусть имеется некоторый односвязный список и ссылка **p** на элемент, после которого мы хотим вставить новый элемент.

Состояние списка перед добавление элемента

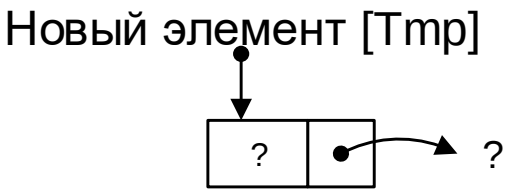


Алгоритм добавления элемента

1. создание нового элемента
2. присвоение значений его полям
3. добавление связи с элементом, следующим за **P**
4. добавление связи с элементом **P**

Создание и инициализация нового элемента

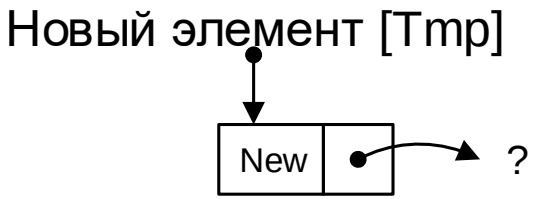
1. Создание нового элемента




```
ListItem Tmp;
```

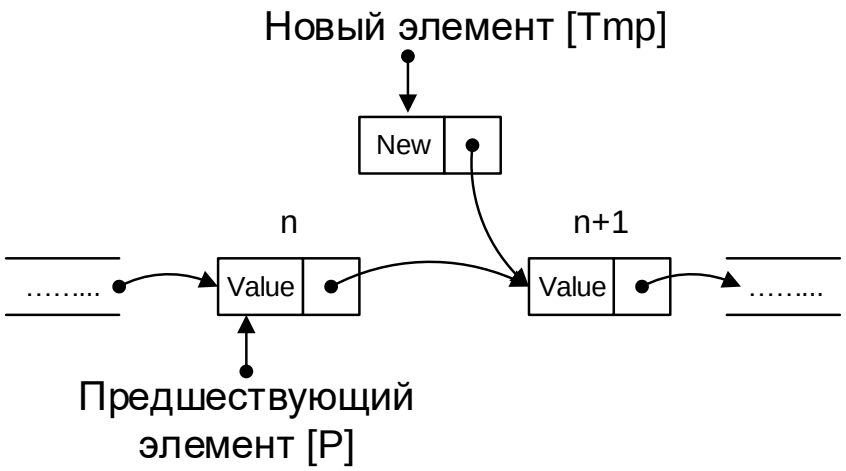
2. Присвоение значений его полям

```
Tmp.n = 2;
Tmp.next = NULL;
```



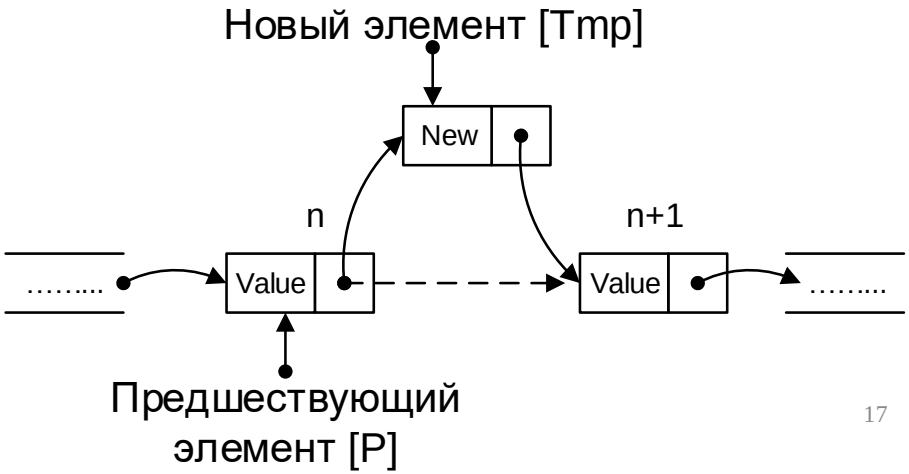
Добавление связей

3. добавление связи с элементом, следующим за P




```
Tmp.next = p.next;
```

4. добавление связи с элементом P

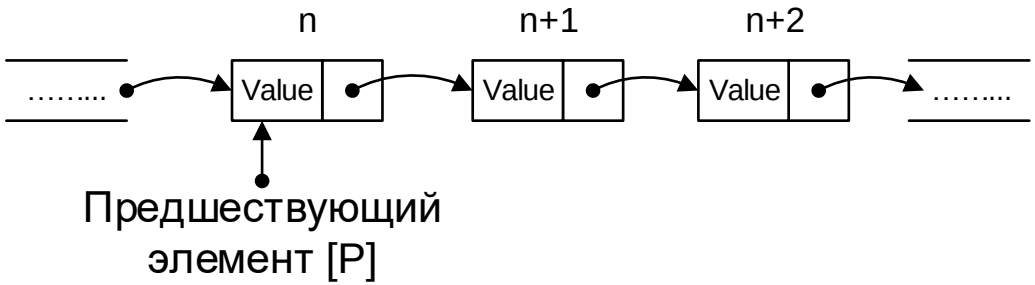



```
p.next = &Tmp;
```

Удаление элемента из односвязного списка

Пусть имеется некоторый односвязный список и ссылка **p** на элемент, предшествующий тому, который мы хотим удалить.

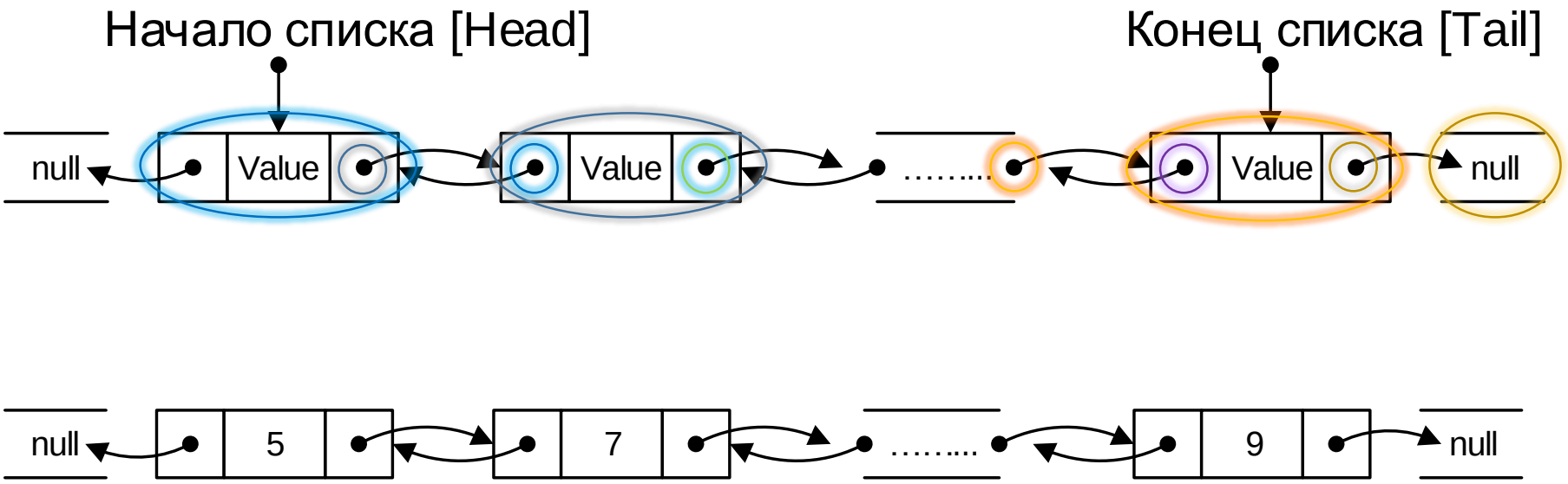
Состояние списка перед удалением элемента



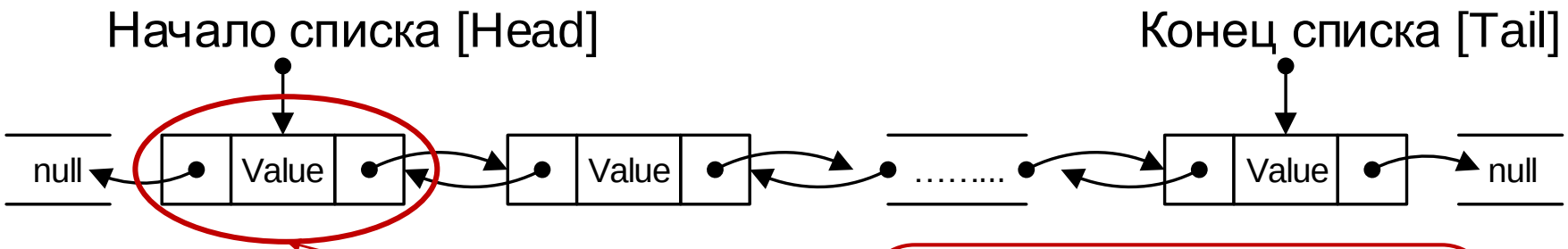
Алгоритм удаления элемента

1. добавление связи между **P** и следующим за удаляемым элементом
2. Освобождение памяти из под удалённого элемента!!!

Двусвязный список



Модель элемента двусвязного списка




```
struct DblListItem {
  int n;
  ListItem* next;
  ListItem* prev;
};

void main() {
  DblListItem head;
  head.n = 1;
  head.next = NULL;
  head.prev = NULL;
}
```

Задание: многомерные массивы

1. Опишите функцию **`structure_create()`**, которая формирует структуру данных **X**, основанную на массиве целочисленных массивов, по следующему правилу: на вход функции подаётся количество элементов 0<**N**<1000, хранящихся в **X**. Каждый массив из **X**, начиная со второго содержит на один элемент больше предыдущего, последний массив может содержать произвольное количество элементов.
2. Опишите функцию **`structure_fill()`** заполнения значениями уже созданной структуры вида **X**.
3. Опишите функцию **`structure_print()`** для отформатированного вывода на экран структуры вида **X**.

Пример заполнения X числами от 1 *до* 12

```
1
2 3
4 5 6
7 8 9 10
11 12
```

Задание: односвязный список

1. Описать функцию добавления нового элемента в конец односвязного списка (как узнать, в какой список добавить элемент?).
2. Описать функцию вывода на экран всех элементов списка, начиная с головы.
3. Описать функцию добавления элемента в произвольное место в односвязном списке (как задавать это место?)
4. Описать функцию удаления элемента из списка.
5. Описать функцию обращения/инверсии списка. Функция заменяет порядок следования элементов на обратный.

Задание: двусвязные и закольцованные списки

1. Опишите набор функций для управления двусвязным списком: добавление элемента списка, удаление элемента списка, добавление элемента в конец списка, добавление элемента в начало списка; вывод списка в прямом и обратном порядке.
2. *Каким образом реализовать закольцованный список?*
3. * Опишите функции для управления односвязным и двусвязным закольцованными списками.
4. ** Реализуйте алгоритм сортировки слиянием для односвязного списка.

NATIONAL RESEARCH UNIVERSITY

Спасибо за внимание!

Максименкова Ольга Вениаминовна

Старший преподаватель Департамента программной инженерии, ФКН

E-mail: omaksimenkova@hse.ru

Blog: Stop To Scale (http://stoptoscale.blogspot.ru)