

МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Программирование

Отчет по лабораторной работе №4

по теме: «Линейные списки»

Выполнили студенты группы М3О-211Б-21

Мороз А.И.

Гордеева М.Д.

Проверила:

доцент каф. 304, к.т.н. Дмитриева Е.А.

Москва 2023 г.

Содержание

[**Задание** 3](#_Toc152467399)

[**Структурные схемы алгоритмов** 4](#_Toc152467400)

[Модуль main 4](#_Toc152467401)

[Функция PrintIndex 6](#_Toc152467402)

[Функция PrintList 7](#_Toc152467403)

[Функция Append 8](#_Toc152467404)

[Функция DeleteList 9](#_Toc152467405)

[Функция Search 10](#_Toc152467406)

[Функция Add 12](#_Toc152467407)

[Функция DeleteFirst 14](#_Toc152467408)

[**Код программы** 15](#_Toc152467409)

[**Результаты работы программы** 19](#_Toc152467410)

[Создание списка, вывод элементов списка с их индексами 19](#_Toc152467411)

[Удаление списка 20](#_Toc152467412)

[1 пункт 20](#_Toc152467413)

[5 пункт 21](#_Toc152467414)

[6 пункт 21](#_Toc152467415)

[**Вывод** 22](#_Toc152467416)

# **Задание**

Реализовать заданный согласно варианту задания линейный список, состоящий из 20 элементов заданного типа. Интерфейс должен включать **для всех вариантов** следующие операции**:**

- создание списка;

- вывод на экран и/или в файл значений элементов списка с их индексами (номерами);

- удаление списка,

а также **некоторые из дополнительных операций** (согласно варианту задания):

1. Поиск в списке наличия элемента с заданным значением с получением его номера в списке (повторное вхождение одного и того же значения может быть разрешено или запрещено – см. вариант).
2. Поиск в списке элемента с максимальным значением с получением его номера в списке (повторное вхождение одного и того же значения может быть разрешено или запрещено – см. вариант).
3. Включение нового элемента в начало списка.
4. Включение нового элемента в конец списка.
5. Включение нового элемента в позицию списка с заданным в программе номером.
6. Удаление элемента из начала списка.
7. Удаление элемента из конца списка.
8. Удаление элемента из позиции списка с заданным в программе номером.

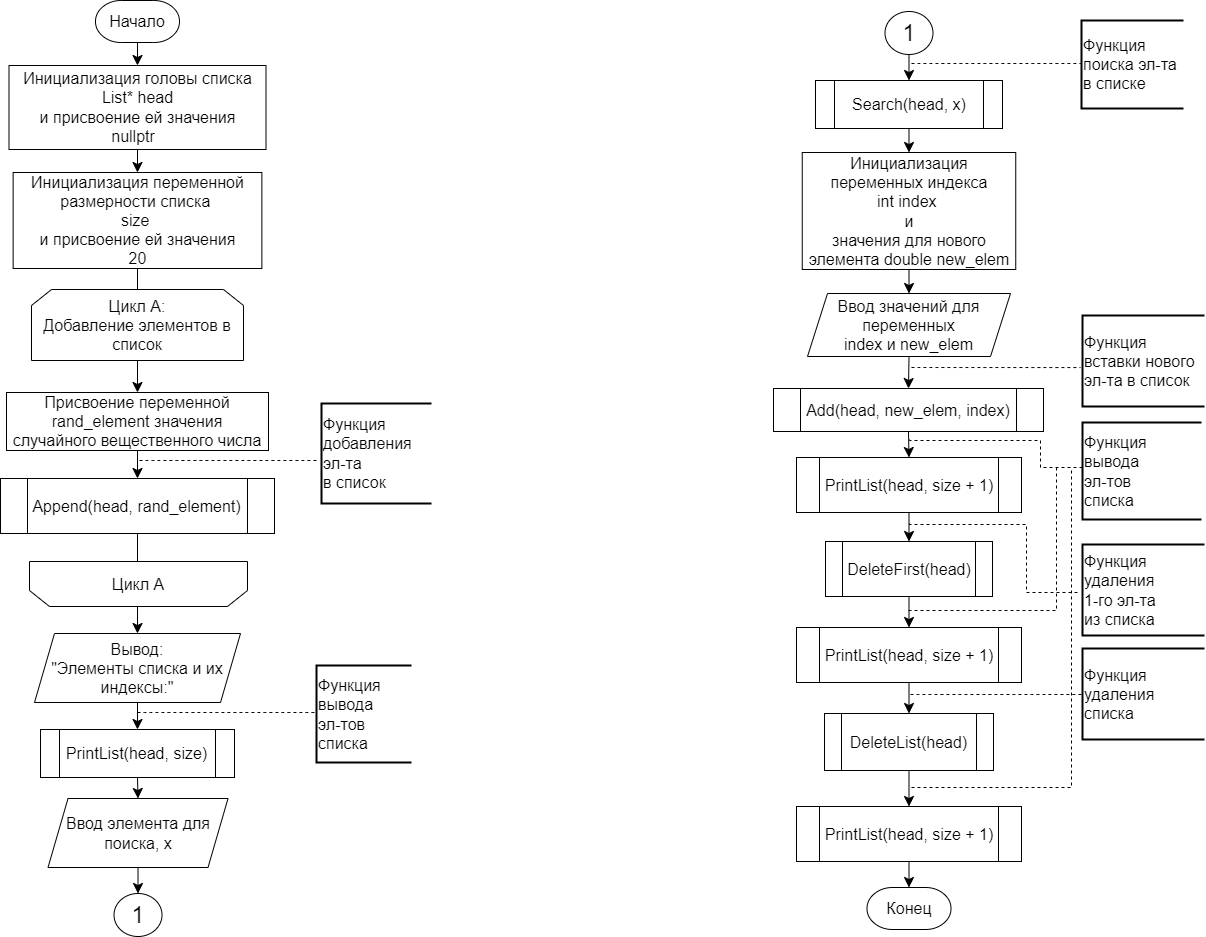
После выполнения операций включения или удаления вывести содержимое списка. Выполнение операций организовать с помощью меню.

Вариант задания

|  |  |  |  |
| --- | --- | --- | --- |
| № вар. | Вид списка | Тип элементов | Дополнительные операции |
| 6 | двунаправленный  кольцевой | вещественный | 1 (разрешено), 5, 6 |

# **Структурные схемы алгоритмов**

## Модуль main



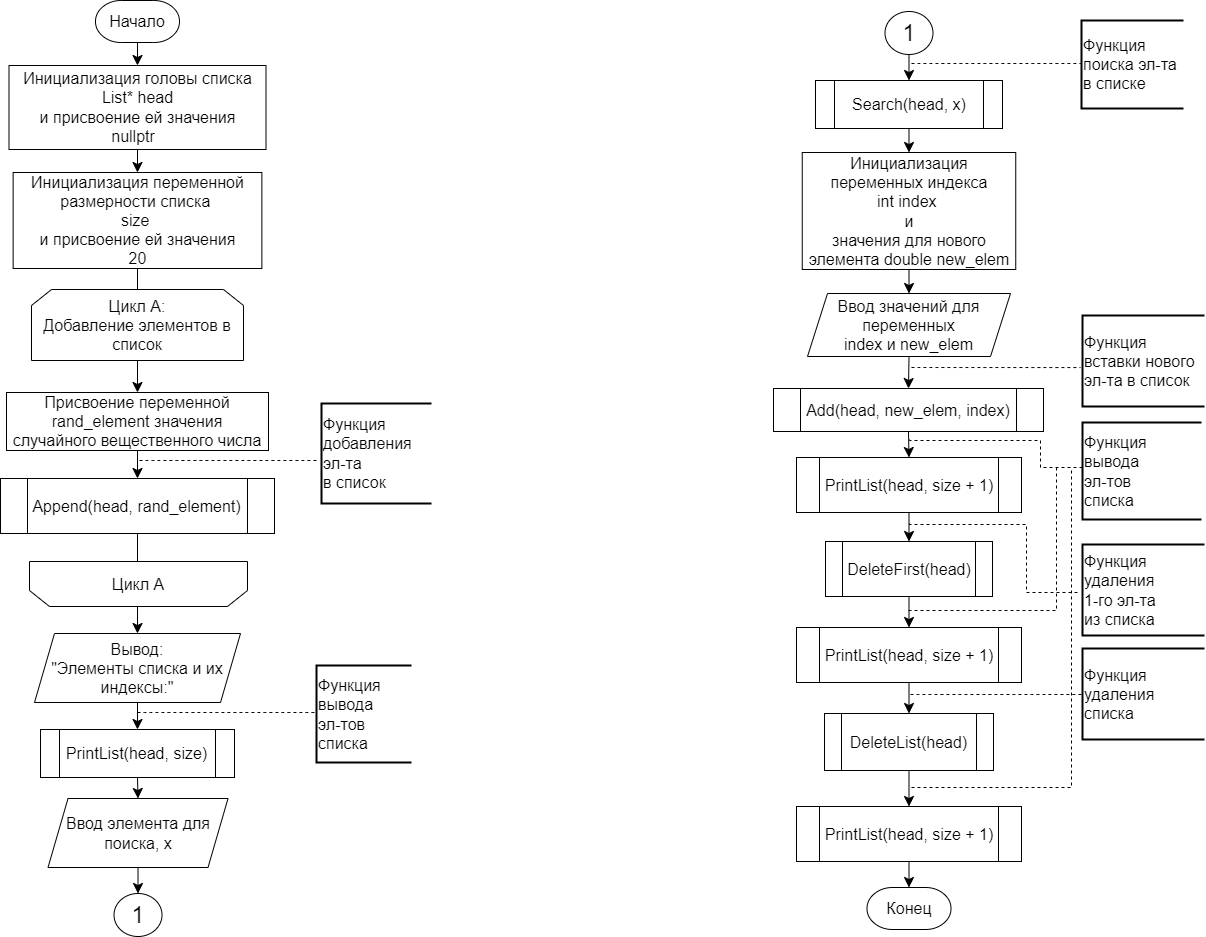


Рис. 1 - Структурная схема модуля main

## Функция PrintIndex

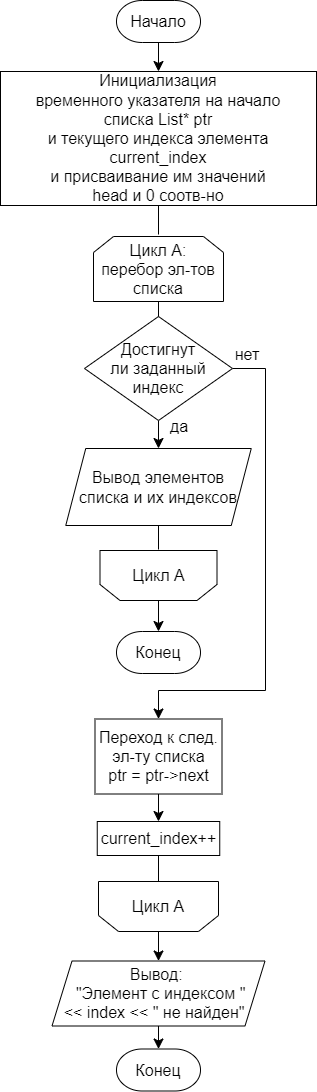


Рис. 2 - Структурная схема функции PrintIndex

## Функция PrintList

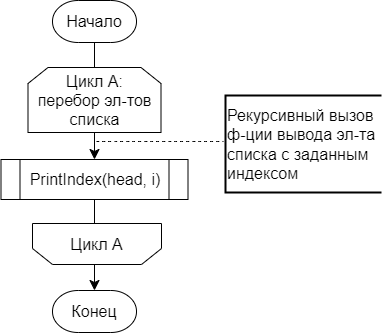


Рис. 3 - Структурная схема функции PrintList

## Функция Append

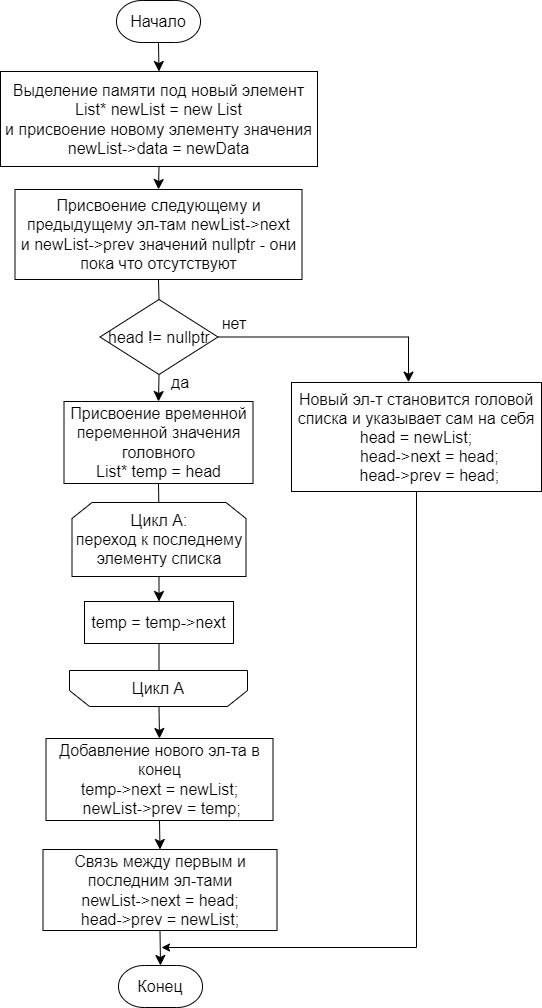


Рис. 4 - Структурная схема функции Append

## Функция DeleteList

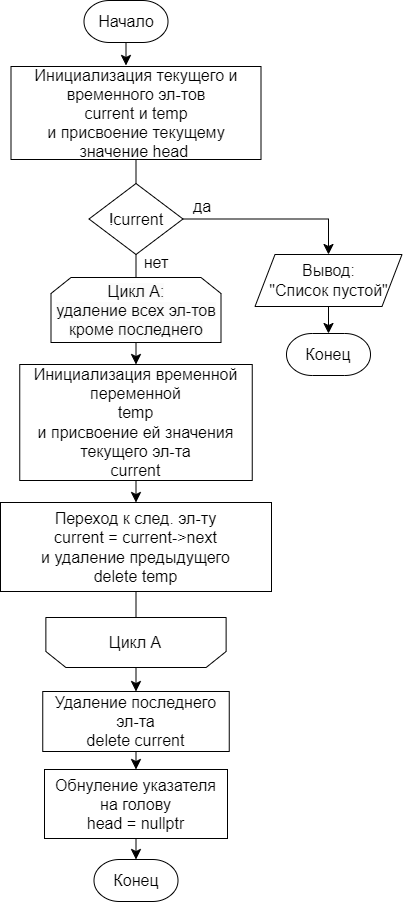
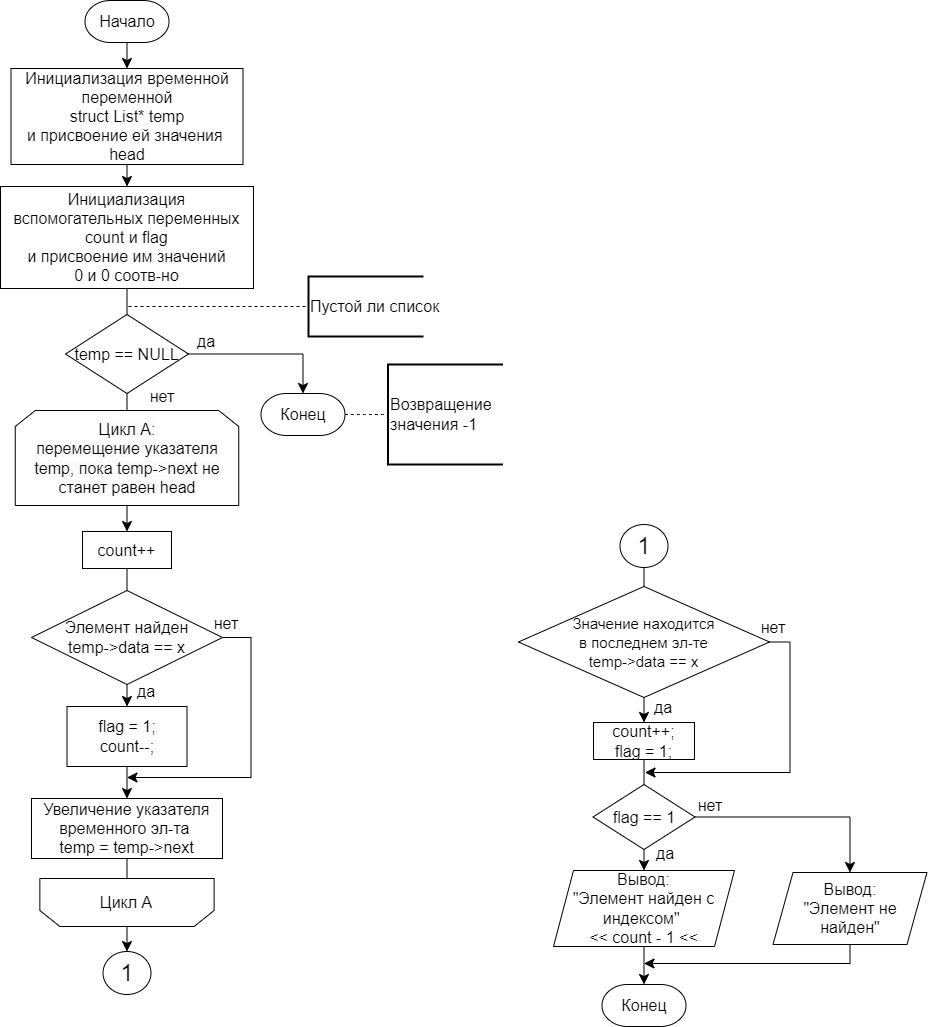


Рис. 5 - Структурная схема функции DeleteList

## Функция Search



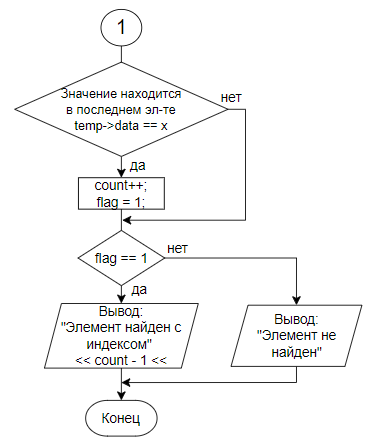
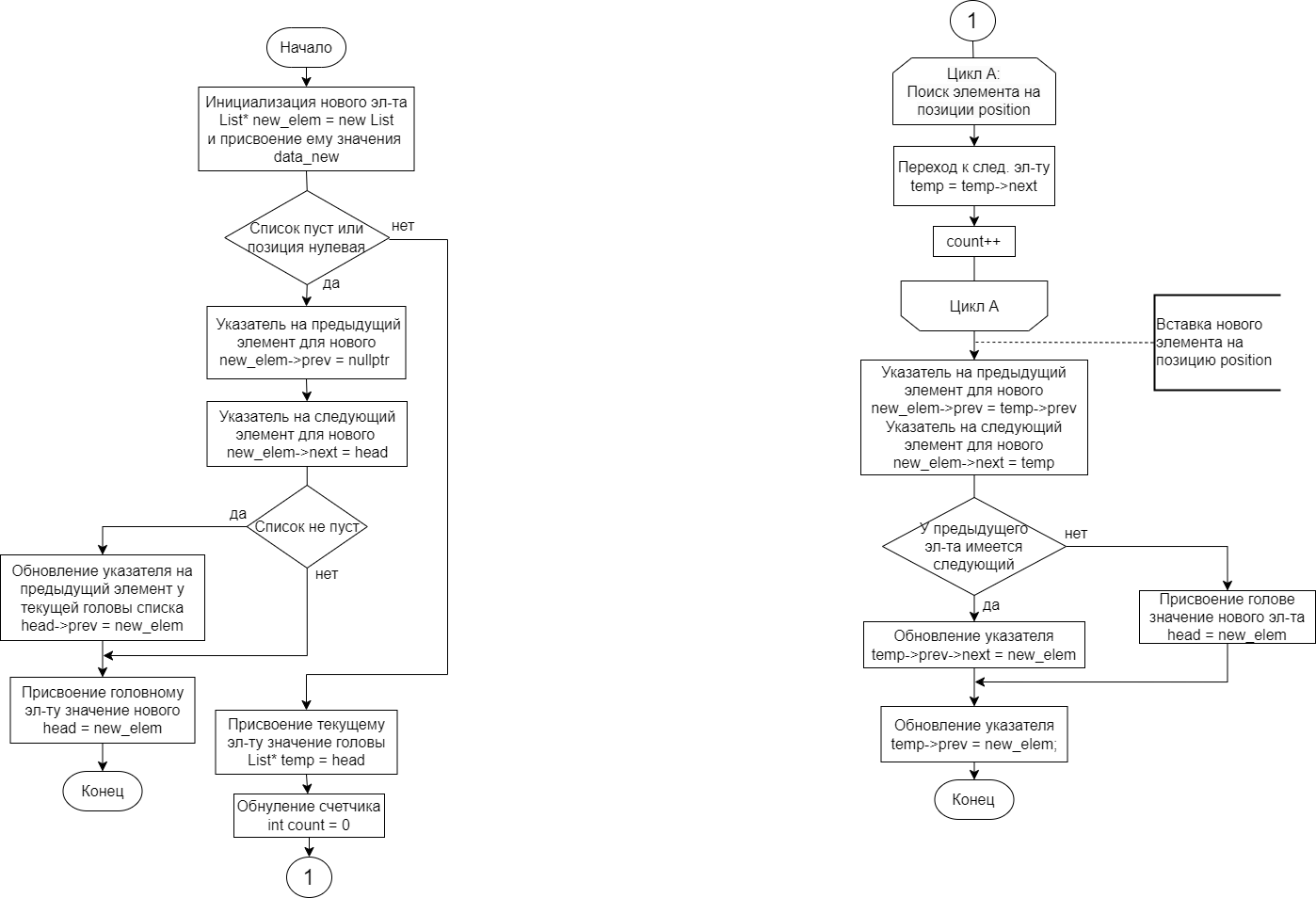


Рис. 6 - Структурная схема функции Search

## Функция Add



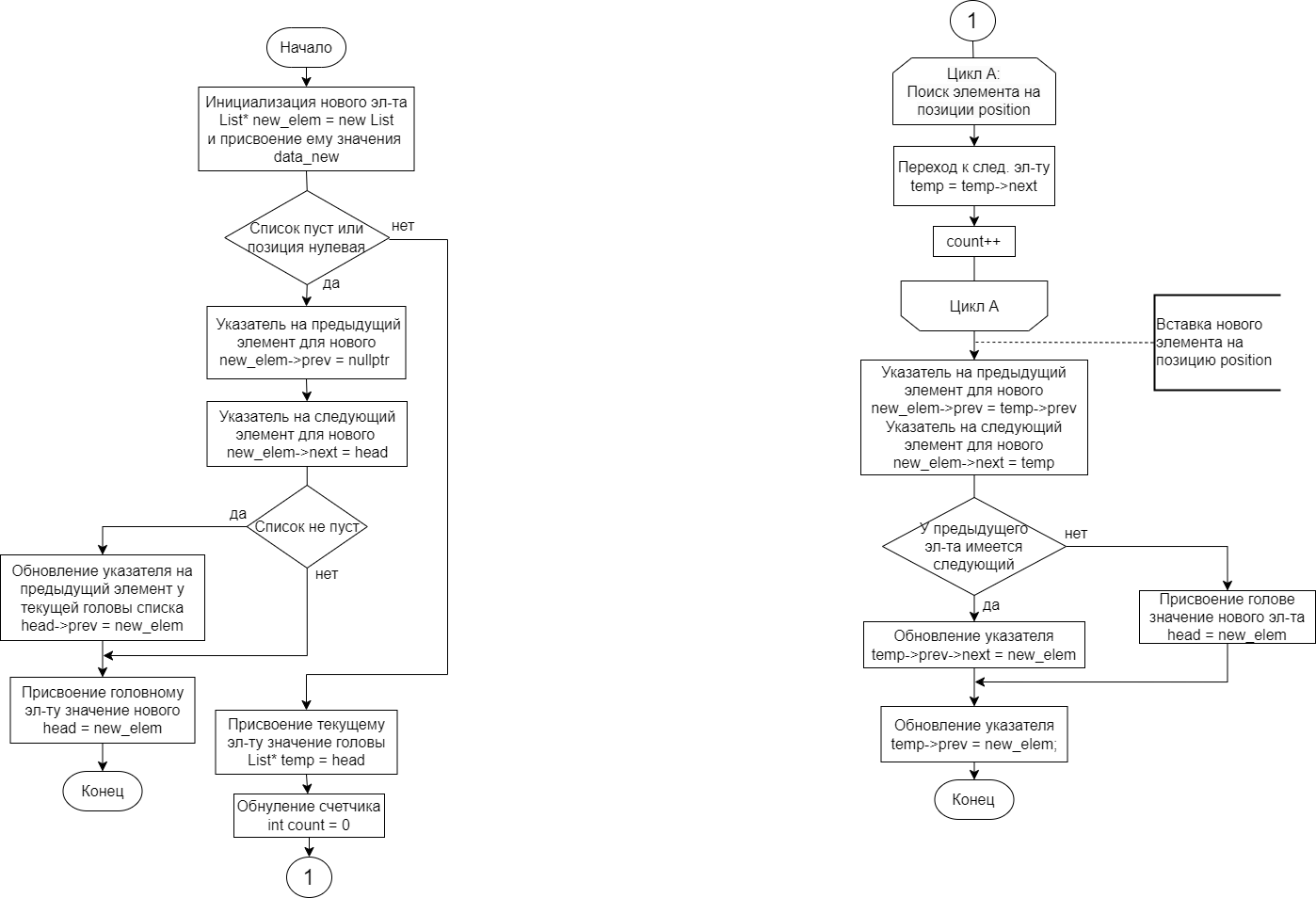


Рис. 7 - Структурная схема функции Add

## Функция DeleteFirst

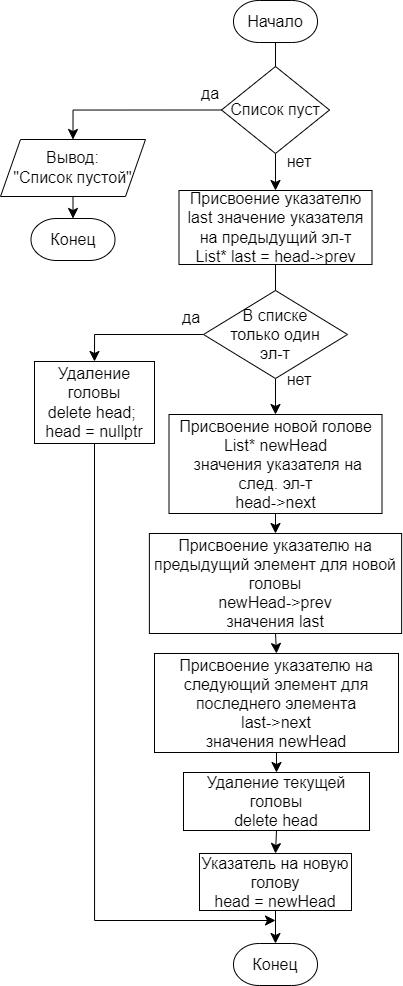


Рис. 8 - Структурная схема функции DeleteFirst

# **Код программы**

#include <iostream>

#include <iomanip>

using namespace std;

// cтруктура для представления элемента списка

struct List {

double data; // элемент

List\* next; // указатель на следующий элемент

List\* prev; // указатель на предыдущий элемент

};

// рекурсивная ф-я вывода элемента списка с заданным индексом

void PrintIndex(List\* head, int index)

{

List\* ptr = head; // временный указатель на начало списка

int current\_index = 0; // текущий индекс элемента

// перебор элементов списка

while (ptr != nullptr)

{

if (current\_index == index)

{

// при достижении заданного индекса - вывод значения и завершение цикла

cout << fixed << setprecision(1) << "[" << index << "] -- " << ptr->data << endl; // с одной цифрой после запятой

return;

}

ptr = ptr->next; // переход к следующему элементу списка

current\_index++;

}//while

cout << "Элемент с индексом " << index << " не найден" << endl;

}//PrintIndex

// вывод списка

void PrintList(List\* head, int size)

{

for (int i = 0; i < size; i++)

{

PrintIndex(head, i); // рекурсивный вызов функции

}

}

// добавление нового элемента в конец кольцевого двусвязного списка

void Append(List\*& head, double newData)

{

// создание нового элемента

List\* newList = new List; // выделение памяти под новый элемент

newList->data = newData; // присвоение новому элементу значения

// следующий и предыдущий элементы пока что отсутствуют

newList->next = nullptr;

newList->prev = nullptr;

// проверка, пустой ли список

if (head != nullptr)

{

List\* temp = head;

// переход к последнему элементу списка

while (temp->next != head)

{

temp = temp->next;

}

// добавление нового элемента в конец

temp->next = newList;

newList->prev = temp;

// связь между первыми и последним элементами (т.к. кольцевой список)

newList->next = head;

head->prev = newList;

}

else

{

// если список пуст, новый элемент становится головой списка

// и будет указывать сам на себя (т.к. кольцевой список)

head = newList;

head->next = head;

head->prev = head;

}

}//Append

// удаление списка

void DeleteList(List\*& head)

{

List\* current = head; // текущий элемент

List\* temp; // временный указатель для удаляемого элемента

// проверка, пустой ли список

if (!current)

{

cout << "Список пустой \n";

return;

}

// удаление всех элементов кроме последнего

while (current->next != head)

{

temp = current; // временно сохраняем текущий элемент

current = current->next; // переход к следующему

delete temp; // удаление предыдущего элемента

}

delete current; // удаление последнего элемента

head = nullptr; // обнуление указателя на голову

}//DeleteList

// поиск элемента в списке

int Search(struct List\* head, double x)

{

struct List\* temp = head; // временная переменная

int count = 0;

int flag = 0;

if (temp == NULL)

return -1;

else

{

// перемещение указателя temp будет происходить

// до тех пор, пока temp->next не станет равен head

while (temp->next != head)

{

count++;

if (temp->data == x) // если элемент найден

{

flag = 1; // элемент найден

count--;

break;

}

temp = temp->next; // сдвиг на следующий элемент

}//while

// проверка, содержится ли значение в последнем элементе списка

if (temp->data == x)

{

count++;

flag = 1; // элемент найден

}

if (flag == 1)

{

cout << "Элемент " << x << " найден с индексом [" << count - 1 << "]" << endl;

}

else

{

cout << "Элемент " << x << " не найден" << endl;

}

}

}//Search

// вставка нового элемента в список

void Add(List\*& head, double data\_new, int position)

{

List\* new\_elem = new List; // создание нового элемента

new\_elem->data = data\_new; // присвоение значения

// если список пуст или позиция - 0,

// вставка происходит в начало списка

if (head == nullptr || position == 0)

{

new\_elem->prev = nullptr; // указатель на предыдущий элемент для нового

new\_elem->next = head; // указатель на следующий элемент для нового

// если список не пуст

if (head != nullptr)

{

// обновление указателя на предыдущий элемент у текущей головы списка

head->prev = new\_elem;

}

head = new\_elem; // новый элемент - голова списка

return;

}

List\* temp = head; // текущая голова списка

int count = 0;

// поиск элемента на позиции position

while (temp != nullptr && count < position)

{

temp = temp->next; // переход к следующему элементу

count++;

}

// вставка нового элемента на позицию position

new\_elem->prev = temp->prev; // указатель на предыдущий элемент для нового

new\_elem->next = temp; // указатель на следующий элемент для нового

// если у предыдущего элемента есть следующий,

// обновление его указателя на новый элемент

if (temp->prev != nullptr)

{

temp->prev->next = new\_elem;

}

else

{

head = new\_elem; // голова - новый элемент

}

temp->prev = new\_elem; // указатель на предыдущий элемент для текущего

}//Add

// удаление элемента из начала списка

void DeleteFirst(List\*& head)

{

// проверка, пустой ли список

if (head == nullptr)

{

cout << "Список пустой \n";

return;

}

List\* last = head->prev; // указатель last получаеь значение указателя на предыдущий элемент

// если в списке только один элемент

if (head == last)

{

delete head; // удаление головы (единственного элемента)

head = nullptr;

}

else

{

List\* newHead = head->next; // новая голова списка - следующий элемент для текущего

newHead->prev = last; // указатель на предыдущий элемент для новой головы

last->next = newHead; // указатель на следующий элемент для последнего элемента

delete head; // удаление текущей головы

head = newHead; // указатель на новую голову

}

}//DeleteFirst

int main()

{

List\* head = nullptr; // голова списка

setlocale(LC\_ALL, "Russian");

system("color F0");

int size = 20; // размерность списка

// добавление элементов в список

for (int i = 0; i < size; i++)

{

// случайное вещественное число

double rand\_element = round((2.0 + ((double)rand() / RAND\_MAX) \* 48.0) \* 10.0) / 10.0;

Append(head, rand\_element);

}

cout << "Элементы списка и их индексы" << endl;

PrintList(head, size);

// поиск

cout << "-------------Поиск элемента-------------\n";

double x;

cout << "Введите элемент для поиска: ";

cin >> x;

Search(head, x);

cout << "\n";

// вставка

cout << "------------Вставка элемента------------\n";

int index;

double new\_elem;

cout << "Введите индекс для вставки нового элемента: ";

cin >> index;

cout << "Введите значение элемента: ";

cin >> new\_elem;

Add(head, new\_elem, index);

PrintList(head, size + 1);

cout << "\n";

// удаление из начала

cout << "------Удаление элемента из начала------\n";

DeleteFirst(head);

PrintList(head, size);

cout << "\n";

// удаление списка

cout << "------------Удаление списка------------\n";

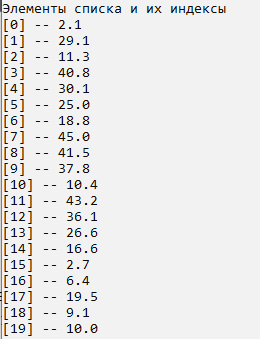
DeleteList(head);

PrintList(head, size);

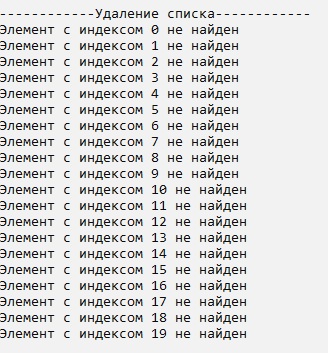
}//main

# **Результаты работы программы**

## Создание списка, вывод элементов списка с их индексами

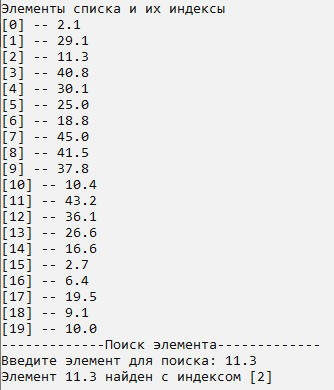


## Удаление списка



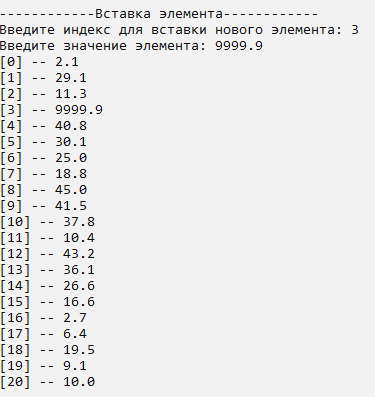
## 1 пункт

Поиск в списке наличия элемента с заданным значением с получением его номера в списке



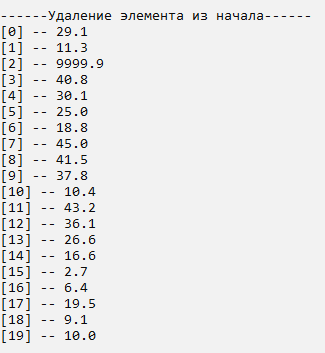
## 5 пункт

Включение нового элемента в позицию списка с заданным в программе номером



## 6 пункт

Удаление элемента из начала списка



## **Вывод**

Программа работает корректно на основе проделанных тестов. Были изучены принципы программной реализации динамической структуры данных «двунаправленный кольцевой список». Для вывода значения элемента списка с заданными индексом в формате с фиксированной точностью (одна цифра после запятой) были использованы функции библиотеки <iomanip>.