Практическая работа № 1

ЛИНЕЙНЫЕ СВЯЗНЫЕ СПИСКИ

Постановка задачи

Составить программу создания линейного односвязного списка (ЛОС) и реализовать основные алгоритмы работы с ЛОС, обеспечивающие следующие действия:

1 - Добавить элемент

1.3.2 – Перед заданным элементом

2 - Удалить элемент

2.3.2 – Все вхождения

6 - Вычислить длину ЛОС

7 - Вывести (распечатать) ЛОС на экран

Перечисленные действия оформить в виде самостоятельных режимов работы созданного ЛОС . Выбор режимов производить с помощью пользовательского меню.

Провести тестирование программы. Тест-примеры определить самостоятельно. Результаты тестирования в виде скриншотов экранов включить в отчет по выполненной работе.

Оформить отчет с подробным описанием созданного ЛОС, принципов программной реализации алгоритмов работы с ЛОС, описанием текста исходного кода и проведенного тестирования программы.

Сделать выводы о проделанной работе, основанные на полученных результатах.

1. Описание алгоритма

Алгоритм программы состоит из функции main и вызываемых в ней вспомогательных функций:

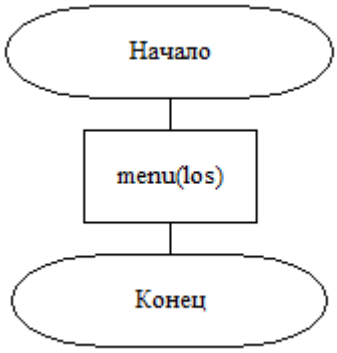

Рис.1 Схема алгоритма функции main

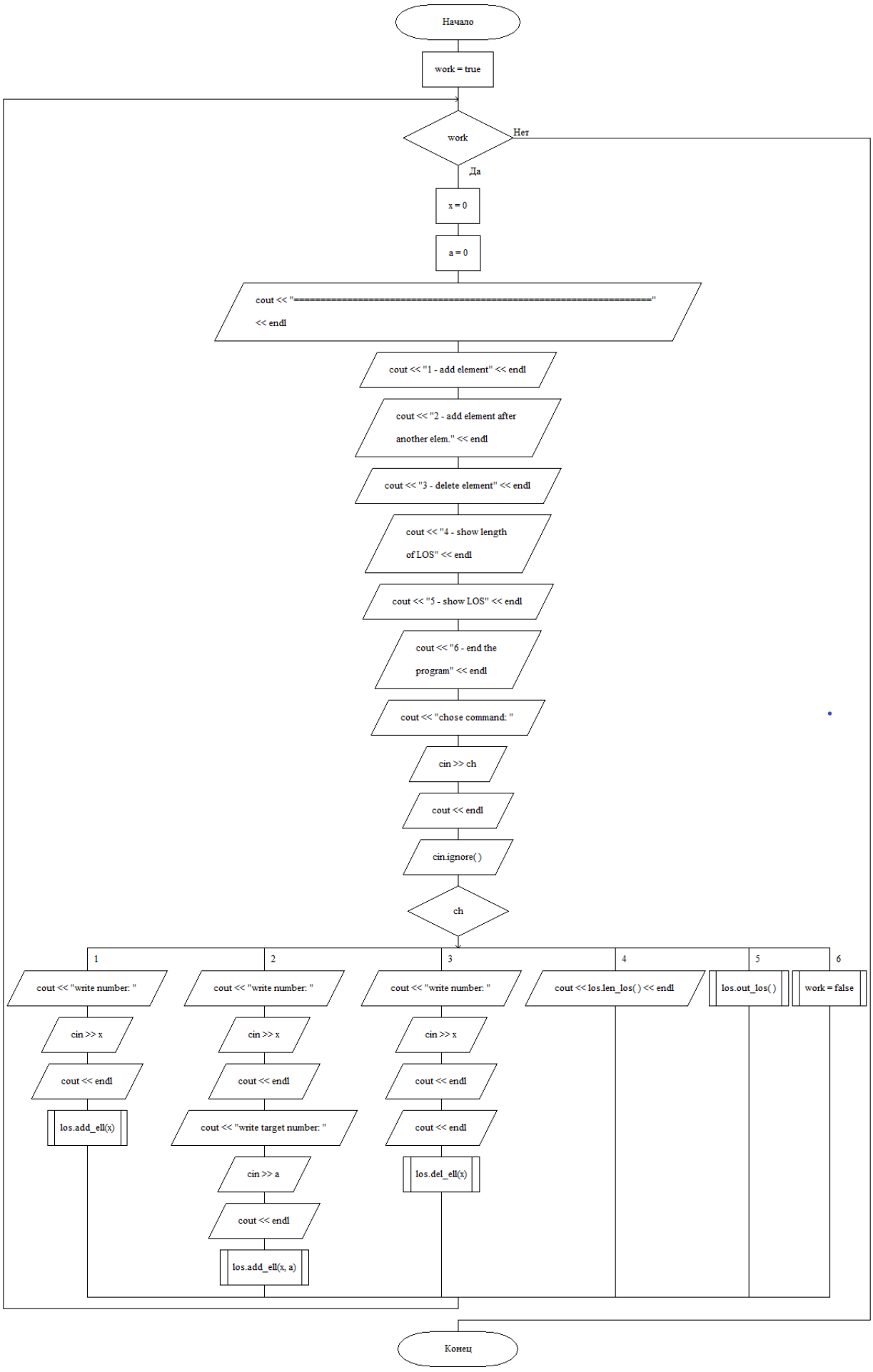
Начало
work = true
work
Нет
Да
x = 0
a = 0
cout << "=="
<< endl
cout << "1 - add element" << endl
cout << "2 - add element after
another elem." << endl
cout << "3 - delete element" << endl
cout << "4 - show length
of LOS" << endl
cout << "5 - show LOS" << endl
cout << "6 - end the
program" << endl
cout << "chose command: "
cin >> ch
cout << endl
cin.ignore()
ch
1
2
3
4
5
6
cout << "write number: "
cin >> x
cout << endl
los.add_ell(x)
cout << "write number: "
cin >> x
cout << endl
cout << "write target number: "
cin >> a
cout << endl
los.add_ell(x, a)
cout << "write number: "
cin >> x
cout << endl
cout << endl
los.del_ell(x)
cout << los.len_los() << endl
los.out_los()
work = false
Конец

Рис.2 Схема алгоритма функции menu

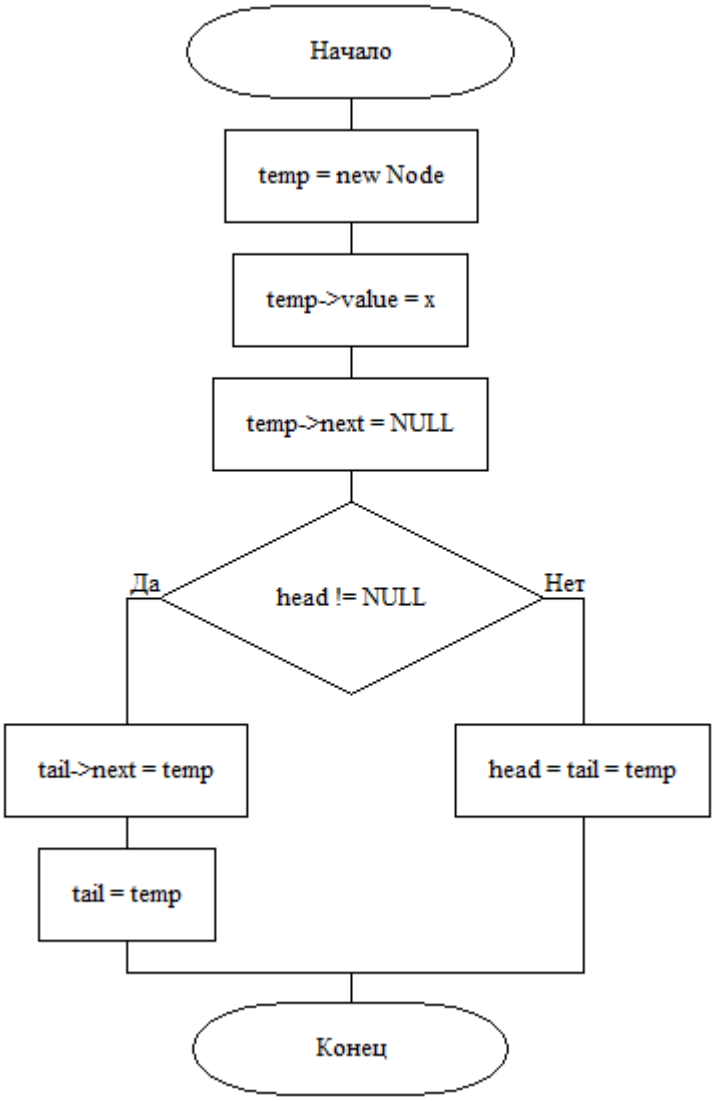

Рис.3 Схема алгоритма функции add_ell

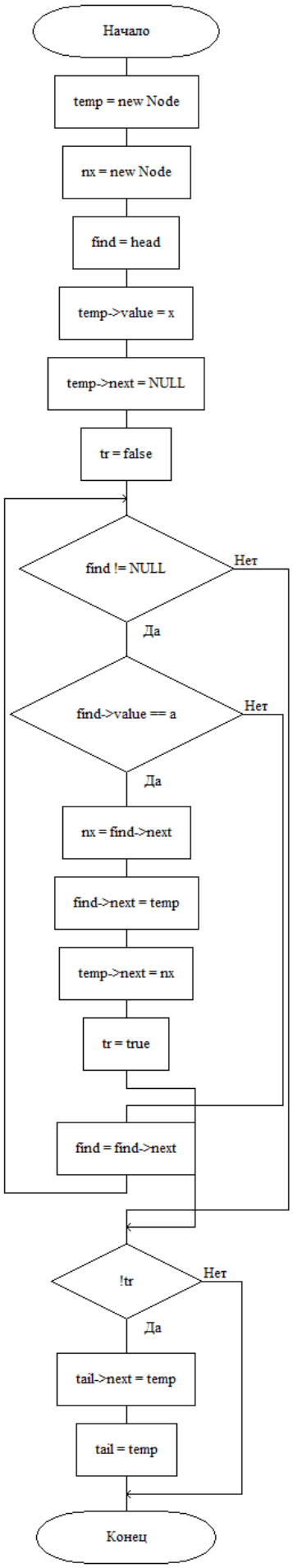
Начало
temp = new Node
nx = new Node
find = head
temp->value = x
temp->next = NULL
tr = false
find != NULL
Нет
Да
find->value == a
Нет
Да
nx = find->next
find->next = temp
temp->next = nx
tr = true
find = find->next
!tr
Нет
Да
tail->next = temp
tail = temp
Конец

Рис.4 Схема алгоритма функции add_ell

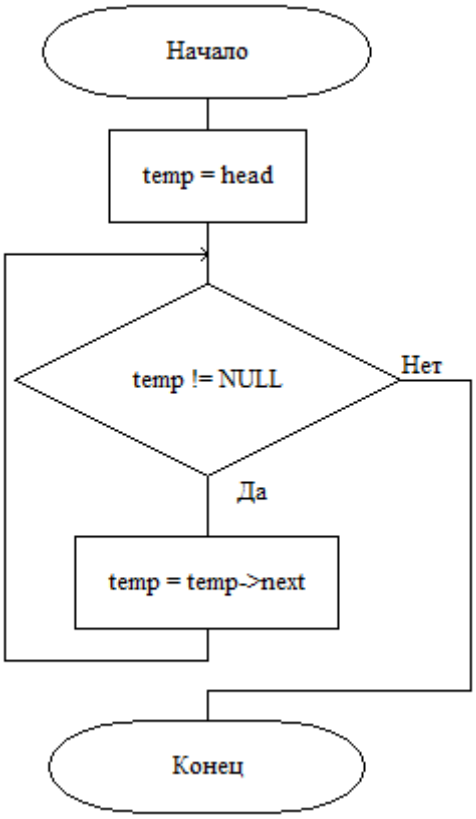

Рис.5 Схема алгоритма функции out_los

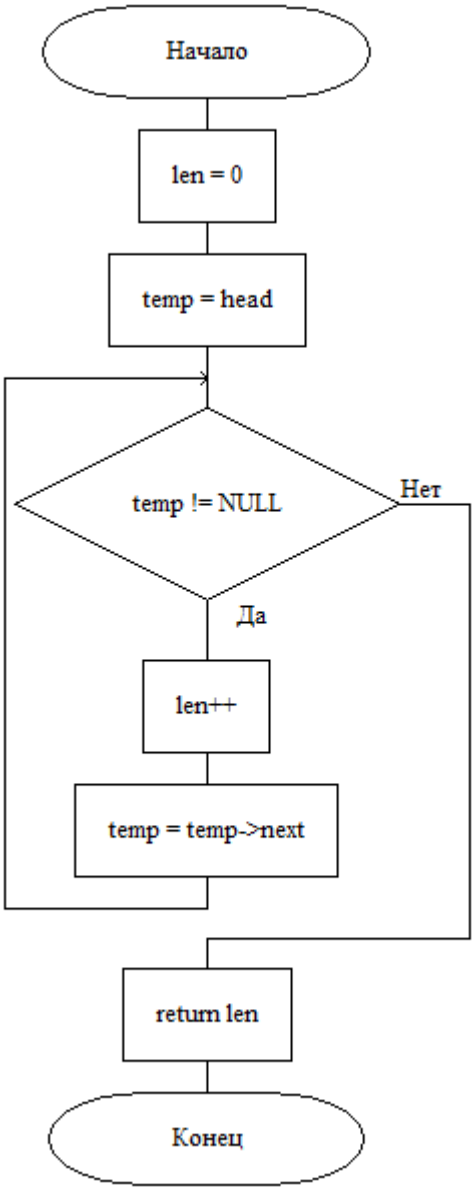

Рис.6 Схема алгоритма функции len_los

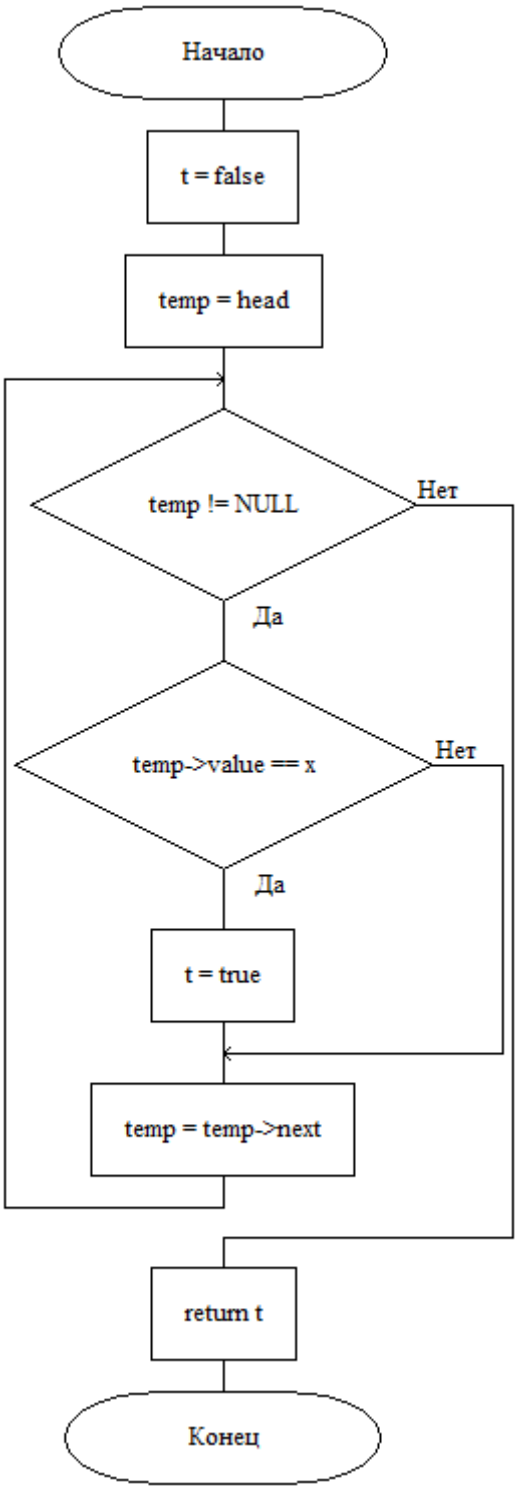

Рис.7 Схема алгоритма функции in_los

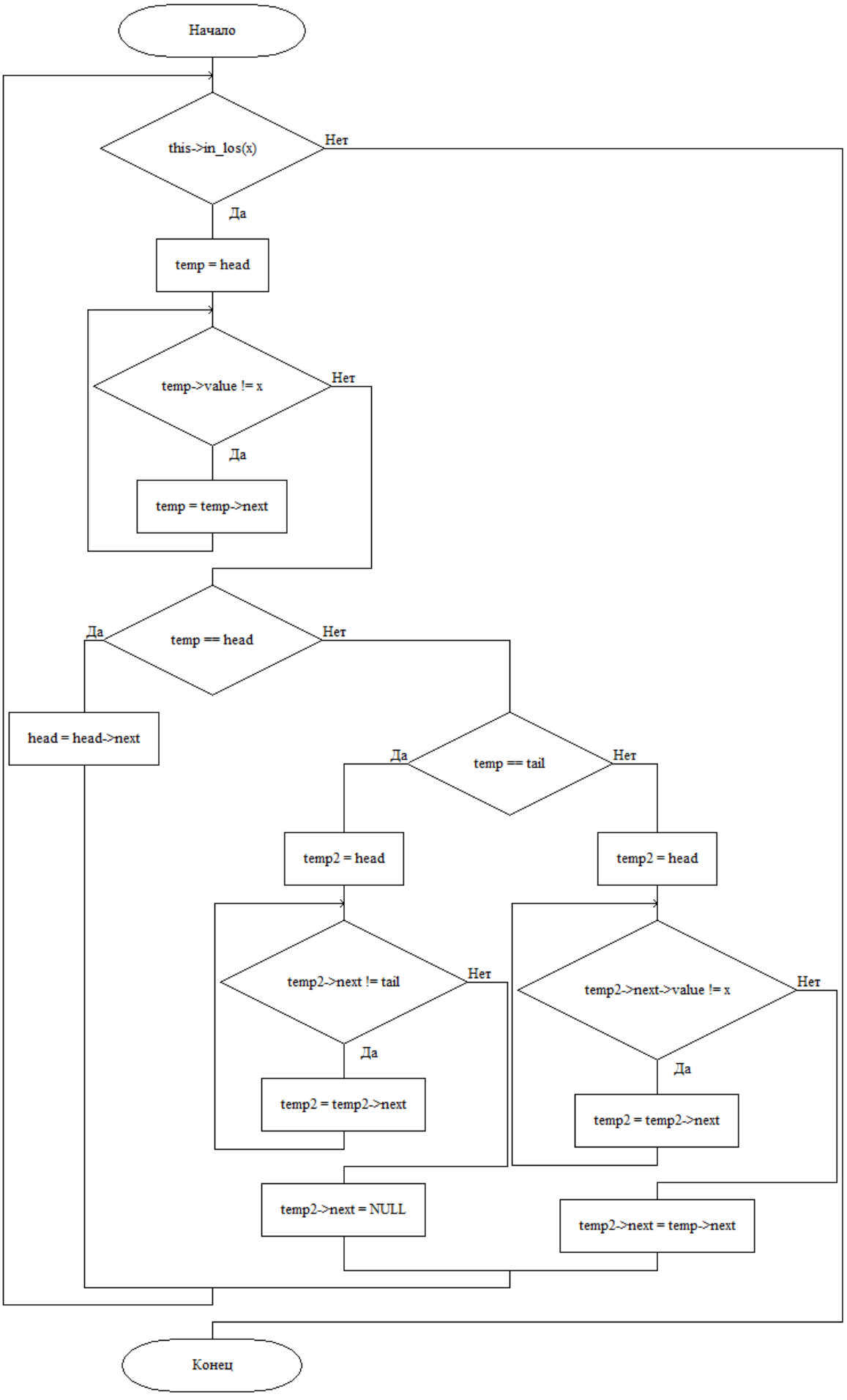
Начало
this->in_los(x)
Нет
Да
temp = head
temp->value != x
Нет
Да
temp = temp->next
Да
temp == head
Нет
head = head->next
Да
temp == tail
Нет
temp2 = head
temp2 = head
temp2->next != tail
Нет
Да
temp2 = temp2->next
temp2->next->value != x
Нет
Да
temp2 = temp2->next
temp2->next = NULL
temp2->next = temp->next
Конец

Рис.7 Схема алгоритма функции dell_ell

2. Реализация алгоритма

Текст исходного кода программы

main.cpp

```
#include "Los.h"

void menu(Los los) {
    bool work = true;
    while (work) {
        int x = 0;
        int a = 0;
        cout <<
"=====================================================================" <<
endl;
        cout << "1 - add element" << endl;
        cout << "2 - add element after another elem." << endl;
        cout << "3 - delete element" << endl;
        cout << "4 - show length of LOS" << endl;
        cout << "5 - show LOS" << endl;
        cout << "6 - end the program" << endl;
        cout << "chose command: ";
        int ch;
        cin >> ch;
        cout << endl;
        cin.ignore();
        switch (ch) {
            case 1:
                cout << "write number: ";
                cin >> x;
                cout << endl;

                los.add_ell(x);
                break;
            case 2:
                cout << "write number: ";
                cin >> x;
                cout << endl;

                cout << "write target number: ";
```

```
                cin >> a;
                cout << endl;

                los.add_ell(x, a);
                break;
            case 3:
                cout << "write number: ";
                cin >> x;
                cout << endl;

                cout << endl;
                los.del_ell(x);
                break;
            case 4:
                cout << los.len_los() << endl;
                break;
            case 5:
                los.out_los();
                break;
            case 6:
                work = false;
                break;
        }
    }
}

int main() {
    Los los;
    menu(los);


}
```

Los.h

```
#ifndef PR_1_LOS_H
#define PR_1_LOS_H

#include <iostream>

using namespace std;

struct Node {
    Node *next = NULL;
    int value = 0;
};

class Los {
private:
    Node *head, *tail;
public:
    Los() : head(NULL), tail(NULL) {};

    void add_ell(int x);

    void add_ell(int x, int a);

    void del_ell(int x);

    int len_los();

    void out_los();

    bool in_los(int x);
```

```
};


#endif
```

Los.cpp

```
#include "Los.h"
void Los::add_ell(int x) {
    Node *temp = new Node;
    temp->value = x;
    temp->next = NULL;
    if (head != NULL) {
        tail->next = temp;
        tail = temp;
    } else head = tail = temp;
};

void Los::add_ell(int x, int a) {
    Node *temp = new Node;
    Node *nx = new Node;
    Node *find = head;
    temp->value = x;
    temp->next = NULL;
    bool tr = false;

    while(find != NULL){
        if (find->value==a){
            nx = find->next;
            find->next = temp;
            temp->next = nx;



            tr = true;
            break;
        }
        find = find->next;
    }
    if(!tr){
        tail->next = temp;
        tail = temp;
    }


};

void Los::out_los() {
    Node *temp = head;
    while(temp!=NULL){
        cout<<temp->value<<endl;
        temp = temp->next;
    }
}

int Los::len_los() {
    int len = 0;
    Node *temp = head;
    while(temp!=NULL){
        len++;
        temp = temp->next;
    }
    return len;
}
```

```
bool Los::in_los(int x) {
    bool t = false;
    Node *temp = head;
    while(temp!=NULL){
        if(temp->value == x)     t = true;
        temp = temp->next;
    }
    return t;
}

void Los::del_ell(int x) {
    while (this->in_los(x)) {
        Node *temp = head;
        while (temp->value != x) {
            temp = temp->next;
        }
        if (temp == head) {
            head = head->next;
        } else if (temp == tail) {
            Node *temp2 = head;
            while (temp2->next != tail) {
                temp2 = temp2->next;
            }
            temp2->next = NULL;
        } else {
            Node *temp2 = head;
            while (temp2->next->value != x) {
                temp2 = temp2->next;
            }
            temp2->next = temp->next;

        }


    }
}
```

1. Тестирование программы

Ниже представлен результат работы программы с введённым ЛОС

```
1 - add element
2 - add element after another elem.
3 - delete element
4 - show length of LOS
5 - show LOS
6 - end the program
chose command:1

write number:23

================================================================================
```

Рис.8 Скриншот добавления в список

```
==========================================================================
1 - add element
2 - add element after another elem.
3 - delete element
4 - show length of LOS
5 - show LOS
6 - end the program
chose command:2

write number:45

write target number:23

==========================================================================
```

Рис.9 Скриншот добавления в список перед числом 23

```
==========================================================================
1 - add element
2 - add element after another elem.
3 - delete element
4 - show length of LOS
5 - show LOS
6 - end the program
chose command:3

write number:23


==========================================================================
```

Рис.10 Скриншот удаления из списка числа 23

```
==========================================================================
1 - add element
2 - add element after another elem.
3 - delete element
4 - show length of LOS
5 - show LOS
6 - end the program
chose command:5

45
==========================================================================
```

Рис.11 Скриншот вывода списка на экран

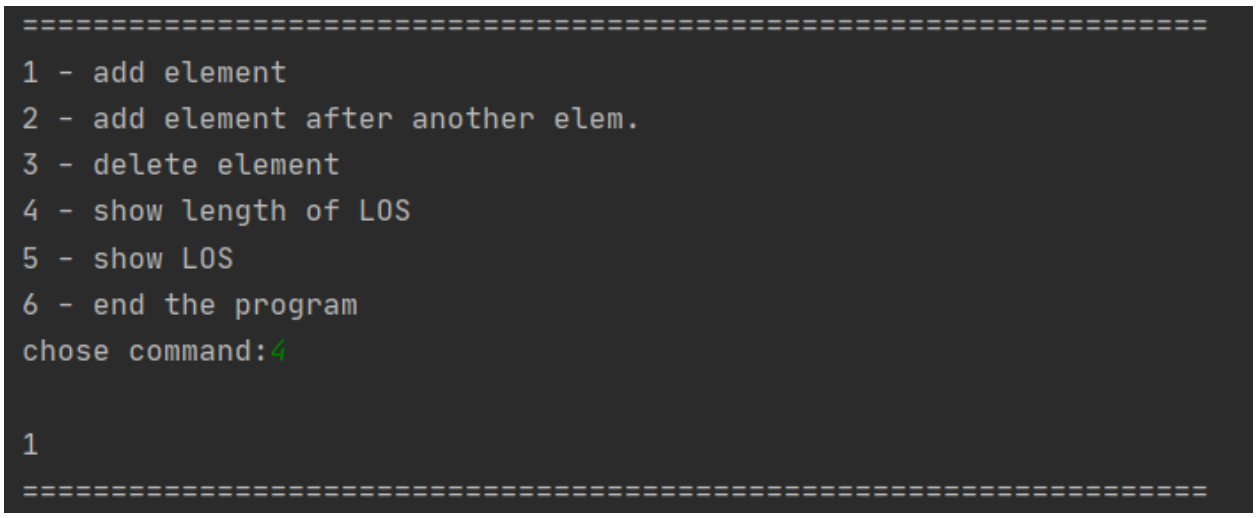

Рис.12 Скриншот вывода длинны списка на экран

1. Выводы

1. В ходе работы был создан односвязный список, ссылки между элементами которого осуществляются при помощи указателей на объекты класса Node.
2. Также были реализованы функции работы с линейным связным списком: добавление элементов, удаление, вывод всех элементов списка, вывод длины.
3. Были изучены положительные и негативные стороны ЛОС:
4. Преимущества: хранение неограниченного количеством и типом данных переменных, неограниченное количество элементов списка, структура ЛОС.
5. Недостатки: не очень удобный поиск конкретного элемента, так как приходится проходить весь ЛОС с крайнего элемента, так как нет способа выбрать конкретный n-ный элемент.
6. Таким образом, была изучена работа линейного связного списка и функций работы над ними и их реализация.

Список используемых информационных источников

1. Сыромятников В.П. Структуры и алгоритмы обработки данных, лекции, РТУ МИРЭА, Москва, 2020/2021 уч./год.
2. Документация по языку программирования C++, интернет-ресурс: https://en.cppreference.com/w/ (Дата обращения – 02.11.2020)
3. Интегрированная среда разработки для языков программирования C и C++, разработанная компанией JetBrains - CLion / Copyright © 2000-2020 JetBrains s.r.o., интернет-ресурс: https://www.jetbrains.com/clion/learning-center/ (Дата обращения – 02.11.2020).
4. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. Интернет-ресурс: http://docs.cntd.ru/document/gost-19-701-90-espd (Дата обращения – 02.11.2020).