Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №11

по теме:

ОДНОСВЯЗНЫЕ И ДВУСВЯЗНЫЕ СПИСКИ

Выполнила: студентка группы РИС-22-1б

Черкасова А.А\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Полякова О.А\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_/\_\_\_\_\_\_\_/\_\_\_\_\_г.

Пермь 2023

**Содержание**

[**Введение** 3](#_Toc127379672)

[**Анализ задачи** 3](#_Toc127379673)

[**Блок – схема** 4](#_Toc127379674)

[**Приложение А** 7](#_Toc127379675)

[**Приложение Б** 9](#_Toc127379676)

# **Введение**

**Цель:** организация однонаправленных и двунаправленных списков

Для достижения поставленной цели, необходимо решить следующие **задачи**:

* Провести анализ задачи
* Реализовать задачу на языке С++
* Составить блок-схему

**Анализ задачи**

*Однонаправленный список*

1. Каждый узел однонаправленного линейного списка содержит одно поле указателя на следующий узел. Поле указателя последнего узла содержит нулевое значение



Для решения задачи необходимо зарезервировать память под структуру, содержащую информационное поле и адресное поле-указатель на следующий элемент и разработать функцию для формирования однонаправленного списка/ двунаправленного списка через структуру.

2. Функция добавления узла в список принимает два аргумента -указатель на узел, после которого происходит добавление и данные для добавляемого узла.

Процедуру добавления узла можно отобразить следующей схемой:



Добавление узла в односвязный линейный список включает в себя этапы:

- создание добавляемого узла и заполнение его поля данных;

- переустановка указателя узла, предшествующего добавляемому, на добавляемый узел;

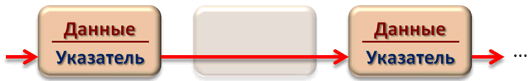
установка указателя добавляемого узла на следующий узел (тот, на который указывал предшествующий узел).

Для добавления k элементов необходимо проделать данную процедуру k раз.

## 3. Удаление узла из односвязного списка.

## В качестве аргументов функции удаления элемента передаются указатель на удаляемый узел, а также указатель на корень списка. Функция возвращает указатель на узел, следующий за удаляемым.

Удаление узла может быть представлено следующей схемой:



Удаление узла включает в себя следующие этапы:

- установка указателя предыдущего узла на узел, следующий за удаляемым;

- освобождение памяти удаляемого узла.

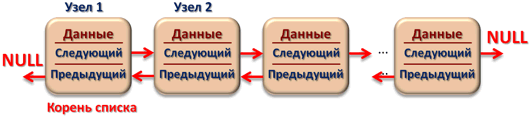
4. Вывод элементов списка. В качестве аргумента в функцию вывода элементов передается указатель на корень списка. Функция осуществляет последовательный обход всех узлов с выводом их значений.

5. Запись однонаправленного списка в файл осуществляется аналогично выводу списка в консоль, одна здесь идёт работа с библиотекой fstream.

6. Функция восстановления списка из файла после удаления. Логика аналогична с тем, как происходит заполнение списка, однако элементы для заполнения берутся из файла.

*Двунаправленный список*

1. Каждый узел двунаправленного линейного списка содержит два поля указателей — на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель на следующий узел последнего узла также содержит нулевое значение.

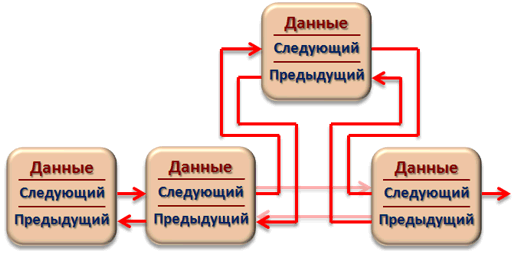


2. Функция добавления узла в список принимает два аргумента:

- Указатель на узел, после которого происходит добавление

- Данные для добавляемого узла.

Процедуру добавления узла можно отобразить следующей схемой:



Добавление узла включает в себя следующие этапы:

- создание узла добавляемого элемента и заполнение его поля данных;

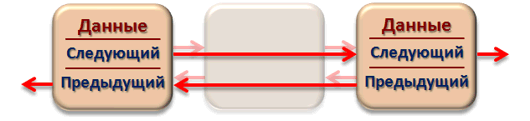
- переустановка указателя «следующий» узла, предшествующего добавляемому, на добавляемый узел;

- переустановка указателя «предыдущий» узла, следующего за добавляемым, на добавляемый узел;

- установка указателя «следующий» добавляемого узла на следующий узел (тот, на который указывал предшествующий узел);

- установка указателя «предыдущий» добавляемого узла на узел, предшествующий добавляемому (узел, переданный в функцию).

В качестве аргументов функции удаления узла ДЛС передается указатель на удаляемый узел. Поскольку узел списка имеет поле указателя на предыдущий узел, нет необходимости передавать указатель на корень списка.  
  
3. Функция возвращает указатель на узел, следующий за удаляемым.  
Удаление узла может быть представлено следующей схемой:



Удаление узла включает в себя следующие этапы:

- установка указателя «следующий» предыдущего узла на узел, следующий за удаляемым;

-установка указателя «предыдущий» следующего узла на узел, предшествующий удаляемому;

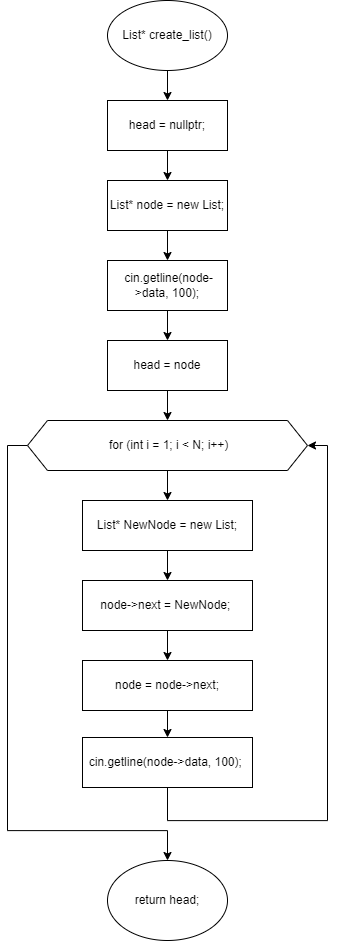
- освобождение памяти удаляемого узла.

4. Функция вывода элементов двусвязного списка аналогична функции для односвязного списка.  
В качестве аргумента в функцию вывода элементов передается указатель на голову списка. Функция осуществляет последовательный обход всех узлов с выводом их значений.

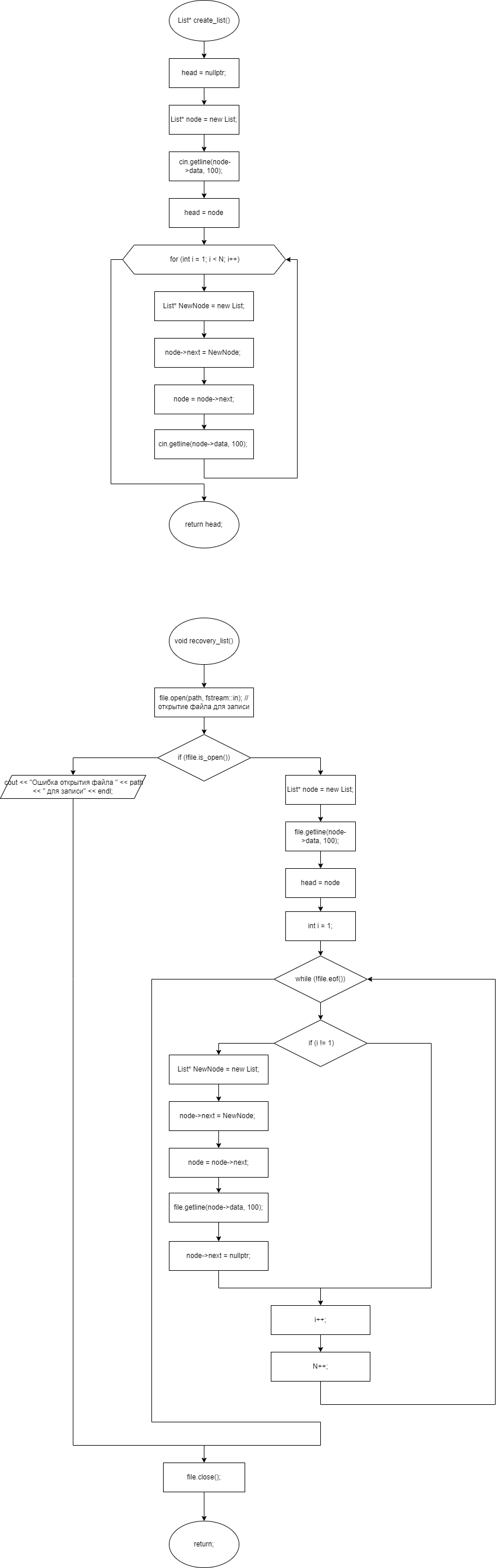
**Блок-схема**

*Однонаправленный список*

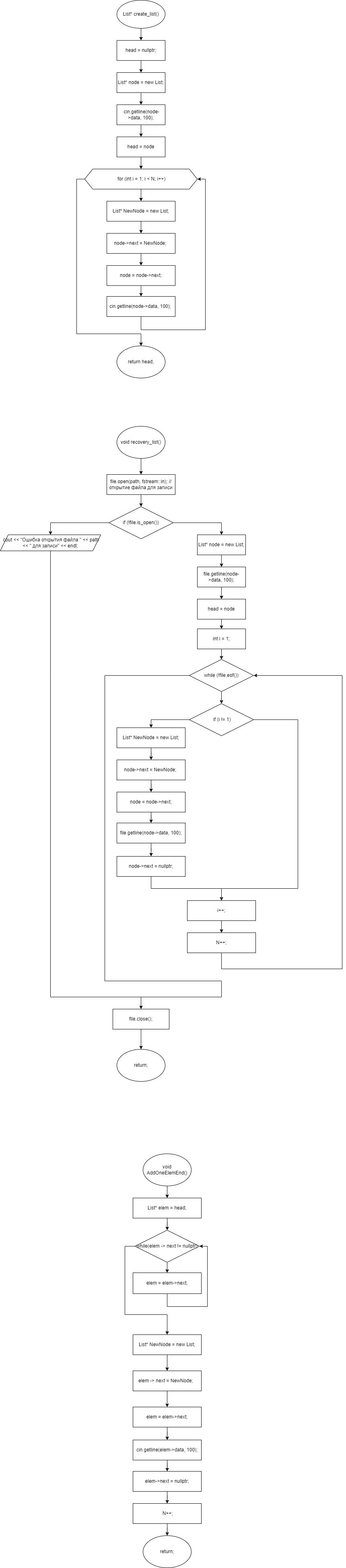
1) Заполнение списка



2) Восстановление списка из файла

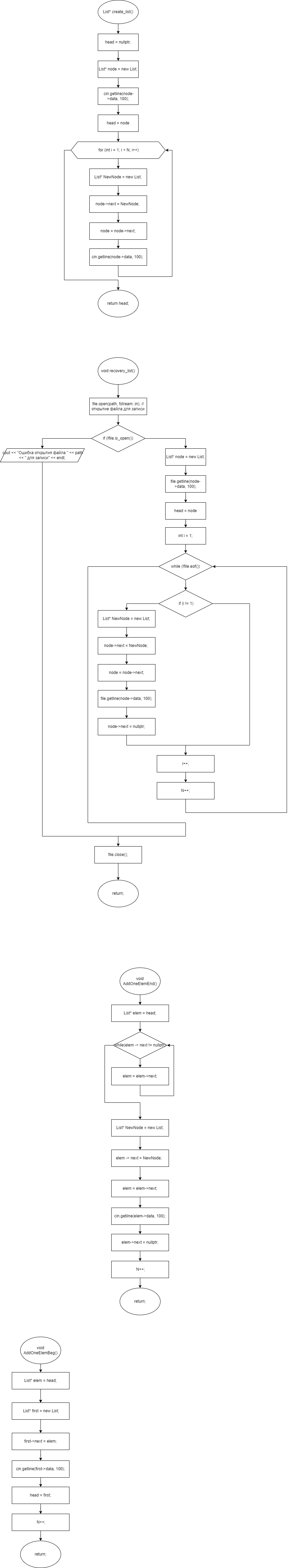


3) Добавление одного элемента в конец

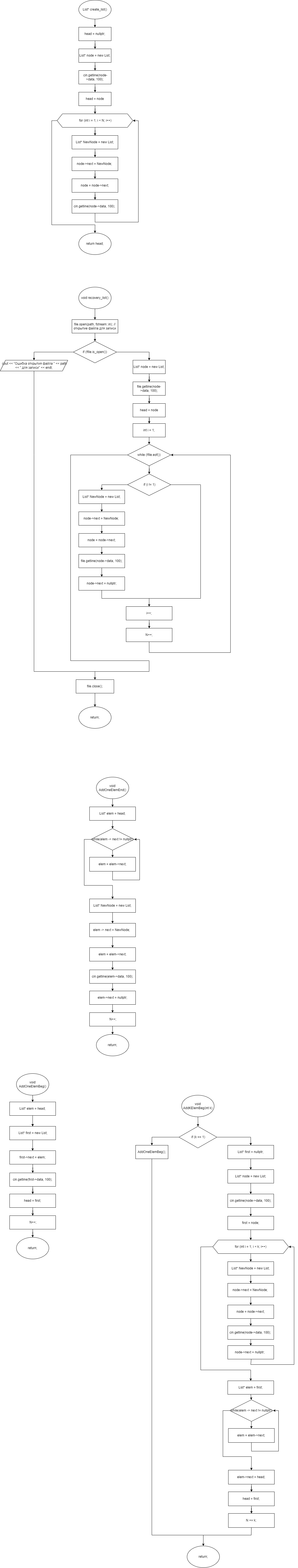


4) Добавление k элементов в конец списка – вызов k раз функции 3).

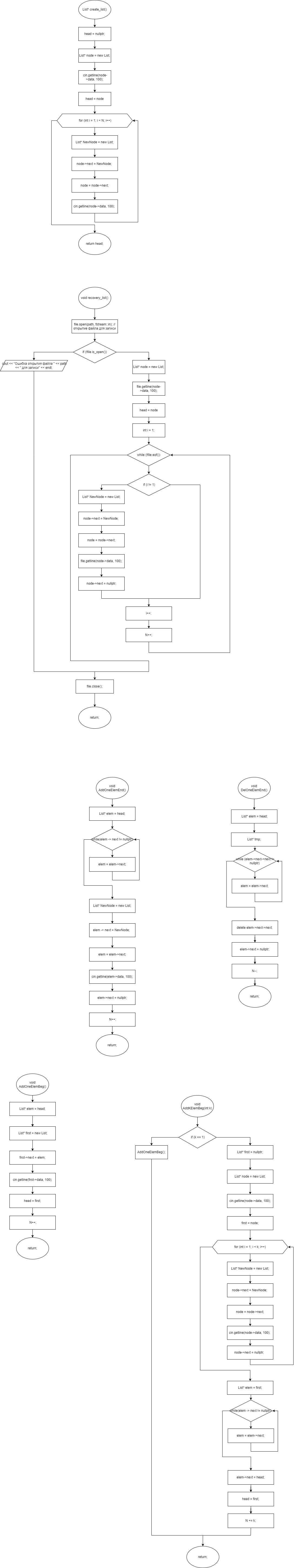
5) Добавление одного элемента в начало



6) Добавление k элементов в начало списка

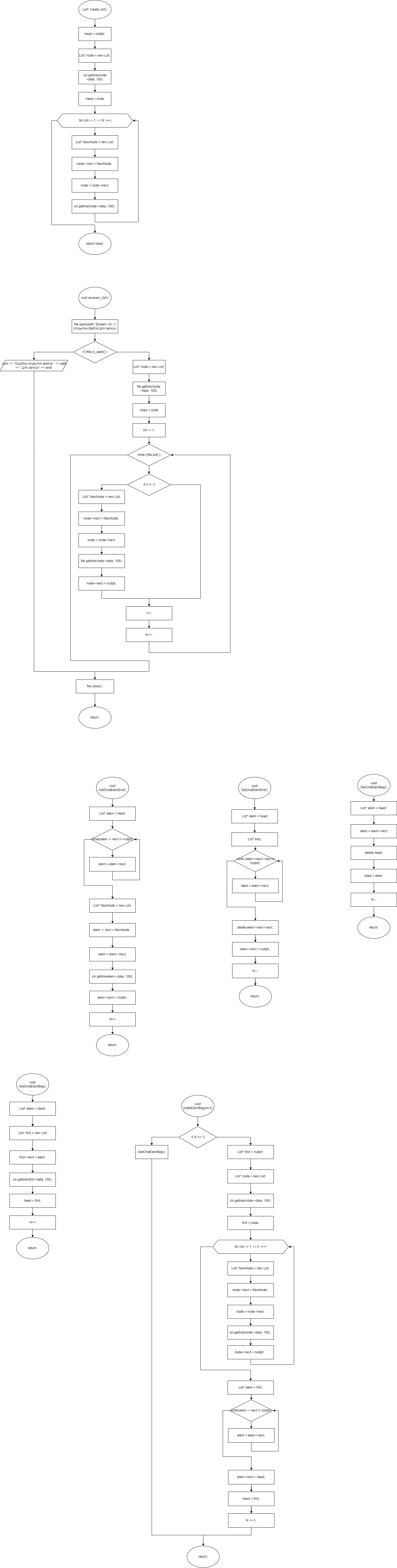


7) Удаление элемента из конца



8) Удаление k элементов из конца – вызов k раз функции 7).

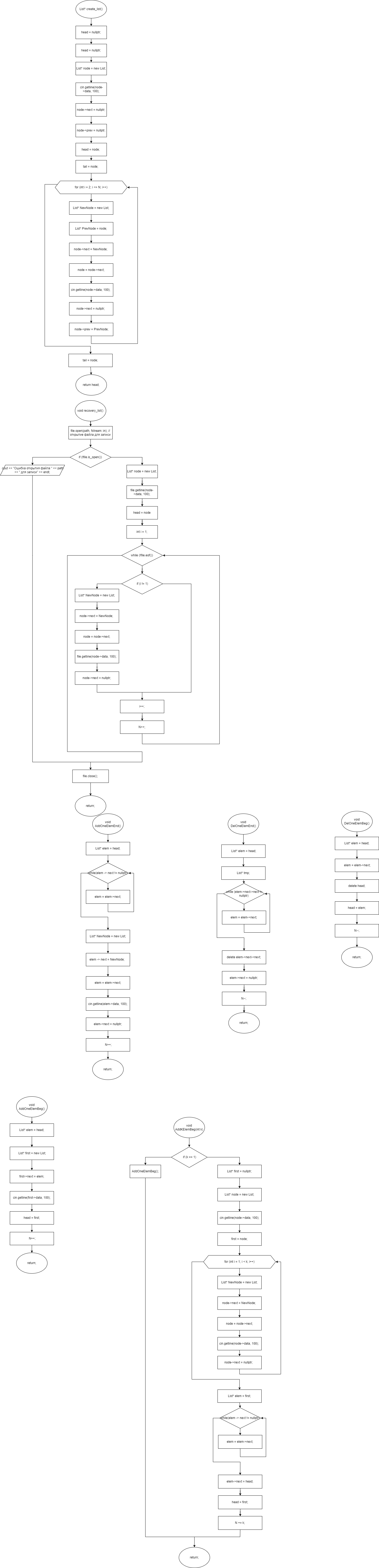
9) Удаление элемента из начала списка



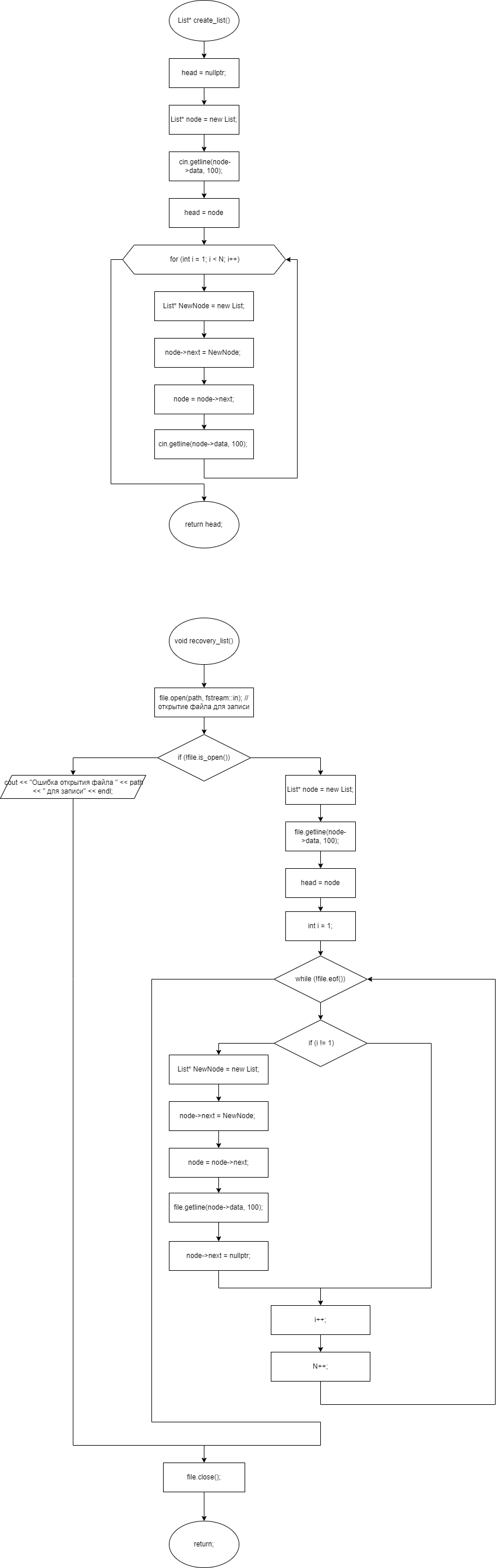
10) Удаление k элементов из начала – вызов k раз функции 9).

*Двунаправленный список*

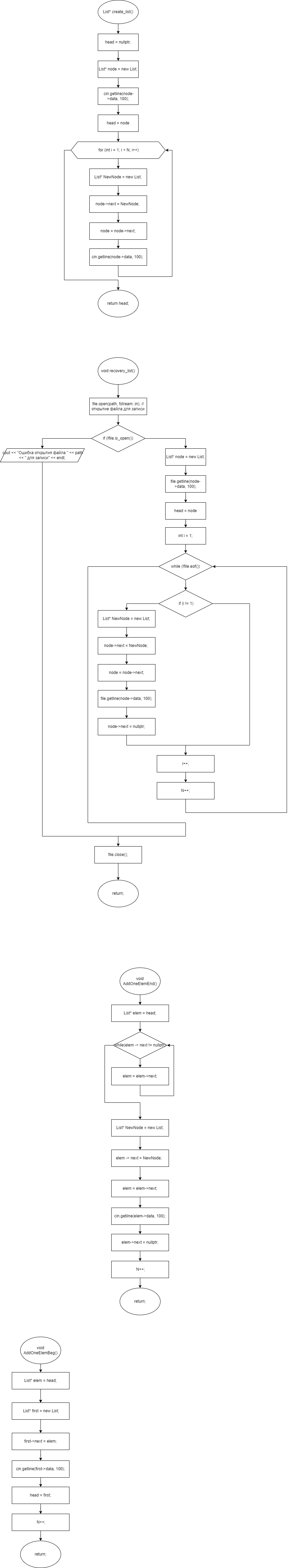
1) Заполнение списка



2) Восстановление списка из файла



3) Добавление одного элемента в начало



**Реализация задачи на языке С++**

Однонаправленный список

/\*18. Записи в линейном списке содержат ключевое поле типа \*char(строка символов) \*/

#include <iostream>

#include <string>

#include <fstream>

#include <math.h>

using namespace std;

struct List // структура односвязного списка

{

char data [100]; //поле, содержащее информацию, хранящуюся в одном узле

List\* next; //поле, хранящее указатель на следующий узел

};

int N;

List\* head;

List\* create\_list()

{

cout << "Метод создания списка вызван" << endl;

head = nullptr; // пока нет элементов, помимо первого, нет и ссылки на следующий элемент

List\* node = new List; // выделение памяти под узел

cout << "Введите элементы списка: ";

cin.getline(node->data, 100);

cin.getline(node->data, 100); // заполнение первого узла данными

head = node; //присваиваем данные первому узлу

for (int i = 1; i < N; i++)

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

node->next = NewNode; // Создание указателя на следующий элемент

node = node->next; // Переход указателем на следующий элемент

cin.getline(node->data, 100); // Запись нового элемента

node->next = nullptr; // Указатель на следующий элемент равен нулю, т. к. был записан последний элемент

}

cout << "Список заполнен" << endl;

return head;

}

void PrintListConsole()

{

List\* current = head; // нужно сделать ссылку на первый элемент

while (current != NULL)

{

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

fstream file;

string path = "OneList.txt";

void PrintListFile()

{

file.open(path, fstream::out); //открытие файла для записи

if (!file.is\_open())

cout << "Ошибка открытия файла " << path << " для записи" << endl;

else

{

List\* current = head; // нужно сделать ссылку на первый элемент

while (current != NULL)

{

file << current->data << endl;

current = current->next;

}

cout << "Данные записаны в файл" << endl;

}

file.close();

}

void AddOneElemEnd()

{

List\* elem = head;

while(elem -> next != nullptr)

{

elem = elem->next;

}

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

elem -> next = NewNode;

elem = elem->next;

cin.getline(elem->data, 100); // Запись нового элемента

elem->next = nullptr;

N++;

}

void AddKElemEnd(int k)

{

int i = 0;

char c [100];

cin.getline(c, 100);

while (i < k)

{

AddOneElemEnd();

i++;

}

}

void AddOneElemBeg()

{

List\* elem = head;

List\* first = new List;

first->next = elem;

cin.getline(first->data, 100);

cin.getline(first->data, 100);

head = first;

N++;

}

void AddKElemBeg(int k)

{

if (k == 1)

{

// Вызов добавления одногом элемента в начало списка

AddOneElemBeg();

}

else

{

List\* first = nullptr; // пока нет элементов, помимо первого, нет и ссылки на следующий элемент

List\* node = new List; // выделение памяти под узел

cin.getline(node->data, 100);

cin.getline(node->data, 100); // заполнение первого узла данными

first = node; //присваиваем данные первому узлу

for (int i = 1; i < k; i++) //создание списка из к элементов, который будет присоединяться к началу существующего списка

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

node->next = NewNode; // Создание указателя на следующий элемент

node = node->next; // Переход указателем на следующий элемент

cin.getline(node->data, 100); // Запись нового элемента

node->next = nullptr; // Указатель на следующий элемент равен нулю, т. к. был записан последний элемент

}

List\* elem = first;

while (elem->next != nullptr)

{

elem = elem->next;

}

elem->next = head;

head = first;

N += k;

}

}

void AddKElemIndex(int pos, int k)

{

if (pos > N)

{

AddKElemEnd(k); // Вызов добавления k элементов в конец списка

}

else if (pos == 0)

{

AddKElemBeg(k); // Вызов добавления k элементов в начало списка

}

else

{

List\* elem = head;

for (int i = 0; i < pos - 1; i++)

{

elem = elem->next; // перемещение по списку до нужного индекса

}

List\* newElem = new List;

List\* nextElem = elem->next; // элемент, стоящий после элемента с заданным индексом до добавления нового эл-та

elem->next = nullptr; // разрыв сарой связи

cin.getline(newElem->data, 100);

delete newElem;

int n = 1; // счётчик добавленных эл-ов

while (n <= k)

{

newElem = new List;

elem->next = newElem;

elem = elem->next;

cin.getline(elem->data, 100);

elem->next = nullptr;

n++;

}

elem->next = nextElem;

N += k;

}

}

void DeleteList()

{

List\* tmp;

while (head != nullptr)

{

tmp = head->next;

delete head;

head = tmp;

N--;

}

}

void DelOneElemEnd()

{

List\* elem = head;

List\* tmp;

while (elem->next->next != nullptr)

{

elem = elem->next;

}

delete elem->next->next;

elem->next = nullptr;

N--;

}

void DelKElemEnd(int k)

{

int i = 0;

while (i < k)

{

DelOneElemEnd();

i++;

}

}

void DelOneElemBeg()

{

List\* elem = head;

elem = elem->next;

delete head;

head = elem;

N--;

}

void DelKElemBeg(int k)

{

int i = 0;

while (i < k)

{

DelOneElemBeg();

i++;

}

}

void DelKElemIndex(int pos, int k) // i > k

{

if (pos > N)

{

DelKElemEnd(k); // Вызов удаления k элементов из конца списка

}

else if (pos < 1)

{

DelKElemBeg(k); // Вызов удаления k элементов из начала списка

}

else

{

List\* elem = head;

int index = abs(pos - k);

for (int i = 0; i < index - 1; i++)

{

elem = elem->next; // перемещение по списку до нужного индекса

}

List\* delElem = elem->next;

List\* nextElem = delElem;

elem->next = nullptr;

int b = 1;

while (b <= k && delElem->next != nullptr)

{

nextElem = delElem->next;

delElem->next = nullptr;

delete delElem;

delElem = nextElem;

b++;

}

elem->next = nextElem;

delElem = nextElem = nullptr;

N -= k;

}

}

void print\_menu() {

system("cls"); // очищаем экран

cout << "ЧТО ВЫ ХОТИТЕ СДЕЛАТЬ?" << endl << endl;

cout << "\t\t\t\tОДНОСВЯЗНЫЙ СПИСОК" << endl << endl;

cout << "1. Создать односвязный список." << endl << endl;

cout << "+ Добавить k элементов в: " << endl << endl;

cout << "\t\t2. конец списка." << endl;

cout << "\t\t3. начало списка." << endl;

cout << "\t\t4. список по индексу." << endl << endl;

cout << "- Удалить k элементов из: " << endl << endl;

cout << "\t\t5. конца списка." << endl;

cout << "\t\t6. начала списка." << endl;

cout << "\t\t7. списка по индексу." << endl << endl;

cout << "8. вывод списка в консоль." << endl;

cout << "9. запись списка в файл." << endl;

cout << "10. уничтожение списка." << endl;

cout << "11. восстановление списка из файла." << endl;

cout << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << endl;

cout << "12. Выход" << endl;

cout << ">> ";

}

int get\_var(int count)

{

int var;

cin >> var;

// пока ввод некорректен, сообщаем об этом и просим повторить его

while (var < 1 or var > count) {

cout << "НЕКОРРЕКТНЫЙ ВВОД! ПОПРОБУЙТЕ СНОВА " << endl; // выводим сообщение об ошибке

system("pause");

cout << ">> ";

cin >> var; // считываем строку повторно

}

return var;

}

void recovery\_list()

{

file.open(path, fstream::in); //открытие файла для записи

if (!file.is\_open())

cout << "Ошибка открытия файла " << path << " для записи" << endl;

else

{

List\* node = new List; // выделение памяти под узел

file.getline(node->data, 100);

head = node; //присваиваем данные первому узлу

int i = 1; // позиция курсора

while (!file.eof())

{

if (i != 1)

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

node->next = NewNode; // Создание указателя на следующий элемент

node = node->next; // Переход указателем на следующий элемент

file.getline(node->data, 100); // Запись нового элемента

node->next = nullptr;

}

i++;

N++;

}

cout << "Список восстановлен!" << endl;

}

file.close();

}

int main()

{

system("chcp 1251>NULL");

print\_menu();

int k, pos;

int var;

do {

print\_menu(); // выводим меню на экран

var = get\_var(12); // получаем номер выбранного пункта меню

switch (var)

{

case 1:

cout << "Введите количество элементов списка: ";

cin >> N;

create\_list();

break;

case 2:

cout << "Введите количество элементов, которое вы хотите добавить в конец списка: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemEnd(k);

break;

case 3:

cout << "Введите количество элементов, которое вы хотите добавить в начало списка: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemBeg(k);

break;

case 4:

cout << "Введите индекс добавления элементов в список: ";

cin >> pos;

cout << "Введите количество элементов, которое вы хотите добавить в список: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemIndex(pos, k);

break;

case 5:

cout << "Введите количество элементов, которое вы хотите удалить из конца списка: ";

cin >> k;

if (k == 1) DelOneElemEnd();

else DelKElemEnd(k);

break;

case 6:

cout << "Введите количество элементов, которое вы хотите удалить из начала списка: ";

cin >> k;

if (k == 1) DelOneElemBeg();

else DelKElemBeg(k);

break;

case 7:

cout << "Введите индекс удаления элементов из списка: ";

cin >> pos;

cout << "Введите количество элементов, которое вы хотите удалить из списка: ";

cin >> k;

DelKElemIndex(pos, k);

break;

case 8:

cout << "Список: ";

PrintListConsole();

break;

case 9:

PrintListFile();

break;

case 10:

DeleteList();

if (N == 0) cout << "Список удалён!"<< endl;

break;

case 11:

recovery\_list();

break;

}

if (var != 12)

system("pause"); // задерживаем выполнение, чтобы пользователь мог увидеть результат выполнения выбранного пункта

} while (var != 12);

return 0;

}

*Двунаправленный список*

/\*18. Записи в линейном списке содержат ключевое поле типа \*char(строка символов) \*/

#include <iostream>

#include <string>

#include <fstream>

#include <math.h>

using namespace std;

struct List // структура односвязного списка

{

char data[100]; //поле, содержащее информацию, хранящуюся в одном узле

List\* next; //поле, хранящее указатель на следующий узел

List\* prev; //поле, хранящее указатель на предыдующий узел

};

int N;

List\* head;

List\* tail;

void create\_list()

{

cout << "Метод создания списка вызван" << endl;

head = nullptr; // пока нет элементов, помимо первого, нет и ссылки на следующий элемент

tail = nullptr;

List\* node = new List; // выделение памяти под узел

cout << "Введите элементы списка: ";

cin.getline(node->data, 100);

cin.getline(node->data, 100); // заполнение первого узла данными

node->next = nullptr;

node->prev = nullptr;

head = node; //присваиваем данные первому узлу

tail = node; // один элемент - и начало и конец списк аодновременно

for (int i = 2; i <= N; i++)

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

List\* PrevNode = node; //Указатель на "предыдущий" элемент(изначально тот, на котором стоит указатель на следующий)

node->next = NewNode;

node = node->next; // Переход указателем на следующий элемент

cin.getline(node->data, 100); // Запись нового элемента

node->next = nullptr; // Указатель на следующий элемент равен нулю, т. к. был записан последний элемент

node->prev = PrevNode;

}

tail = node;

cout << "Список заполнен" << endl;

}

void PrintListConsole()

{

List\* current = head; // нужно сделать ссылку на первый элемент

while (current != NULL)

{

cout << current->data << " ";

current = current->next;

}

cout << endl;

//test

cout << "Вывод списка в обратном порядке для проверки связей: ";

List\* tmp = tail; // нужно сделать ссылку на первый элемент

while (tmp != NULL)

{

cout << tmp->data << " ";

tmp = tmp->prev;

}

cout << endl;

}

fstream file;

string path = "TwoList.txt";

void PrintListFile()

{

file.open(path, fstream::out); //открытие файла для записи

if (!file.is\_open())

cout << "Ошибка открытия файла " << path << " для записи" << endl;

else

{

List\* current = head; // нужно сделать ссылку на первый элемент

while (current != NULL)

{

file << current->data << endl;

current = current->next;

}

cout << "Данные записаны в файл" << endl;

}

file.close();

}

void AddOneElemEnd()

{

List\* elem = tail;

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

elem->next = NewNode;

elem = elem->next;

cin.getline(elem->data, 100); // Запись нового элемента

elem->prev = tail;

tail = elem;

tail->next = nullptr;

N++;

}

void AddKElemEnd(int k)

{

int i = 0;

char c[100];

cin.getline(c, 100);

while (i < k)

{

AddOneElemEnd();

i++;

}

}

void AddOneElemBeg()

{

List\* elem = head;

List\* first = new List;

first->next = elem;

elem->prev = first;

first->prev = nullptr;

cin.getline(first->data, 100);

cin.getline(first->data, 100);

head = first;

N++;

}

void AddKElemBeg(int k)

{

if (k == 1)

{

// Вызов добавления одногом элемента в начало списка

AddOneElemBeg();

}

else

{

List\* first = nullptr; // пока нет элементов, помимо первого, нет и ссылки на следующий элемент

List\* node = new List; // выделение памяти под узел

node->prev = nullptr;

cin.getline(node->data, 100);

cin.getline(node->data, 100); // заполнение первого узла данными

first = node; //присваиваем данные первому узлу

for (int i = 1; i < k; i++) //создание списка из к элементов, который будет присоединяться к началу существующего списка

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

List\* PrevNode = node; //Указатель на "предыдущий" элемент(изначально тот, на котором стоит указатель на следующий)

node->next = NewNode;

node = node->next; // Переход указателем на следующий элемент

cin.getline(node->data, 100); // Запись нового элемента

node->next = nullptr; // Указатель на следующий элемент равен нулю, т. к. был записан последний элемент

node->prev = PrevNode;

}

node->next = head;

head->prev = node;

head = first;

N += k;

}

}

void AddKElemIndex(int pos, int k)

{

if (pos > N)

{

AddKElemEnd(k); // Вызов добавления k элементов в конец списка

}

else if (pos == 0)

{

AddKElemBeg(k); // Вызов добавления k элементов в начало списка

}

else

{

List\* elem;

if (pos < N / 2)

{

elem = head;

for (int i = 0; i < pos - 1; i++)

{

elem = elem->next; // перемещение по списку до нужного индекса

}

}

else

{

elem = tail;

for (int i = 0; i < N - pos; i++)

{

elem = elem->prev; // перемещение по списку до нужного индекса

}

}

List\* newElem = new List;

List\* nextElem = elem->next;//, // элемент, стоящий после элемента с заданным индексом до добавления нового эл-та

//\* prevElem = nextElem->prev;

nextElem->prev = nullptr; // разрыв старой связи

elem->next = nullptr; // разрыв старой связи

cin.getline(newElem->data, 100);

delete newElem;

int n = 1; // счётчик добавленных эл-ов

while (n <= k)

{

newElem = new List;

elem->next = newElem;

newElem->prev = elem;

elem = elem->next;

cin.getline(elem->data, 100);

elem->next = nullptr;

n++;

}

elem->next = nextElem;

nextElem->prev = elem;

N += k;

}

}

void DeleteList()

{

List\* tmp;

while (head->next != nullptr)

{

tmp = head->next;

tmp->prev = nullptr;

head->next = nullptr;

delete head;

head = tmp;

N--;

}

delete head;

N--;

tail = nullptr;

}

void DelOneElemEnd()

{

List\* tmp = tail->prev;

tail->prev = nullptr;

tmp->next = nullptr;

delete tail;

tail = tmp;

N--;

}

void DelKElemEnd(int k)

{

int i = 0;

while (i < k)

{

DelOneElemEnd();

i++;

}

}

void DelOneElemBeg()

{

List\* tmp = head->next;

tmp->prev = nullptr;

head->next = nullptr;

delete head;

head = tmp;

N--;

}

void DelKElemBeg(int k)

{

int i = 0;

while (i < k)

{

DelOneElemBeg();

i++;

}

}

void DelKElemIndex(int pos, int k) // i > k

{

int index = abs(pos - k);

if (pos > N)

{

DelKElemEnd(k); // Вызов удаления k элементов из конца списка

}

else if (pos <= 1 || index < 1)

{

DelKElemBeg(k); // Вызов удаления k элементов из начала списка

}

else

{

List\* elem;

if (pos < N / 2)

{

elem = head;

for (int i = 0; i < index - 1; i++)

{

elem = elem->next; // перемещение по списку до нужного индекса

}

}

else

{

elem = tail;

for (int i = 0; i < N - index; i++)

{

elem = elem->prev; // перемещение по списку до нужного индекса

}

}

List\* delElem = elem->next;

List\* nextElem = delElem;

//nextElem->prev = nullptr;

elem->next = nullptr;

int b = 1;

while (b <= k && delElem->next != nullptr)

{

nextElem = delElem->next;

delElem->next = nullptr;

delElem->prev = nullptr;

delete delElem;

delElem = nextElem;

b++;

}

elem->next = nextElem;

nextElem->prev = elem;

delElem = nextElem = nullptr;

N -= k;

}

}

void print\_menu() {

system("cls"); // очищаем экран

cout << "ЧТО ВЫ ХОТИТЕ СДЕЛАТЬ?" << endl << endl;

cout << "\t\t\t\tДВУСВЯЗНЫЙ СПИСОК" << endl << endl;

cout << "1. Создать двусвязный список." << endl << endl;

cout << "+ Добавить k элементов в: " << endl << endl;

cout << "\t\t2. конец списка." << endl;

cout << "\t\t3. начало списка." << endl;

cout << "\t\t4. список по индексу." << endl << endl;

cout << "- Удалить k элементов из: " << endl << endl;

cout << "\t\t5. конца списка." << endl;

cout << "\t\t6. начала списка." << endl;

cout << "\t\t7. списка по индексу." << endl << endl;

cout << "8. вывод списка в консоль." << endl;

cout << "9. запись списка в файл." << endl;

cout << "10. уничтожение списка." << endl;

cout << "11. восстановление списка из файла." << endl;

cout << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << endl;

cout << "12. Выход" << endl;

cout << ">> ";

}

int get\_var(int count)

{

int var;

cin >> var;

// пока ввод некорректен, сообщаем об этом и просим повторить его

while (var < 1 or var > count) {

cout << "НЕКОРРЕКТНЫЙ ВВОД! ПОПРОБУЙТЕ СНОВА " << endl; // выводим сообщение об ошибке

system("pause");

cout << ">> ";

cin >> var; // считываем строку повторно

}

return var;

}

void recovery\_list()

{

file.open(path, fstream::in); //открытие файла для записи

if (!file.is\_open())

cout << "Ошибка открытия файла " << path << " для записи" << endl;

else

{

List\* node = new List; // выделение памяти под узел

file.getline(node->data, 100);

node->next = nullptr;

node->prev = nullptr;

head = node; //присваиваем данные первому узлу

tail = node; // один элемент - и начало и конец списк аодновременно

int i = 1; // позиция курсора

while (!file.eof())

{

if (i != 1)

{

List\* NewNode = new List; // Выделение димнамической памяти под новый элемент

List\* PrevNode = node; //Указатель на "предыдущий" элемент(изначально тот, на котором стоит указатель на следующий)

node->next = NewNode;

node = node->next; // Переход указателем на следующий элемент

file.getline(node->data, 100); // Запись нового элемента

node->next = nullptr; // Указатель на следующий элемент равен нулю, т. к. был записан последний элемент

node->prev = PrevNode;

}

i++;

N++;

}

tail = node;

cout << "Список восстановлен!" << endl;

}

file.close();

}

int main()

{

system("chcp 1251>NULL");

print\_menu();

int k, pos;

int var;

do {

print\_menu(); // выводим меню на экран

var = get\_var(12); // получаем номер выбранного пункта меню

switch (var)

{

case 1:

cout << "Введите количество элементов списка: ";

cin >> N;

create\_list();

break;

case 2:

cout << "Введите количество элементов, которое вы хотите добавить в конец списка: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemEnd(k);

break;

case 3:

cout << "Введите количество элементов, которое вы хотите добавить в начало списка: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemBeg(k);

break;

case 4:

cout << "Введите индекс добавления элементов в список: ";

cin >> pos;

cout << "Введите количество элементов, которое вы хотите добавить в список: ";

cin >> k;

cout << "Введите элементs: ";

AddKElemIndex(pos, k);

break;

case 5:

cout << "Введите количество элементов, которое вы хотите удалить из конца списка: ";

cin >> k;

if (k == 1) DelOneElemEnd();

else DelKElemEnd(k);

break;

case 6:

cout << "Введите количество элементов, которое вы хотите удалить из начала списка: ";

cin >> k;

if (k == 1) DelOneElemBeg();

else DelKElemBeg(k);

break;

case 7:

cout << "Введите индекс удаления элементов из списка: ";

cin >> pos;

cout << "Введите количество элементов, которое вы хотите удалить из списка: ";

cin >> k;

DelKElemIndex(pos, k);

break;

case 8:

if (N == 0)

{

cout << "Элементы в списке отсутствуют!" << endl;

}

else

{

cout << "Список: ";

PrintListConsole();

}

break;

case 9:

PrintListFile();

break;

case 10:

DeleteList();

if (N == 0) cout << "Список удалён!" << endl;

break;

case 11:

recovery\_list();

break;

}

if (var != 12)

system("pause"); // задерживаем выполнение, чтобы пользователь мог увидеть результат выполнения выбранного пункта

} while (var != 12);

return 0;

}