Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(ПНИПУ)

ОТЧЁТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №9

по теме:

ДИНАМИЧЕСКИЕ СТРУКТУРЫ ДАННЫХ.

СТЕКИ

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**

**Постановка задачи:** Реализовать работу со стеками, а именно:

* Создать стек
* Добавить k элементов в стек
* Удалить k элементов из стека
* Вывод стека в консоль
* Запись стека в файл
* Уничтожение стека
* Восстановление стека из файла

**Цель:** Получить практические навыки работы со стеками.

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

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

# **Анализ задачи**

Так как нужно реализовать программу через структуры, нужно создать структуру, которая будет содержать поля (char data;) для данных и (Stack\* prev;) для адреса следующего элемента.

Создам функцию (Stack\* make\_stack(int n)) для создания стека и его заполнения. В начале проверим с помощью оператора выбора (if) равно ли количество элементов стека 0. Если да, то возвращаем NULL. Иначе выделим память под 1 элемент (p = new Stack;), пользователь записывает данные (cin >> a; p->data = (a);)и адрес на NULL (p->prev = NULL; ) в элемент. Затем ставим указатель на верхний элемент стека (top = p;).

После этого с помощью цикла (for (int i = 2; i <= n; i++)) введём оставшиеся элементы. (пользователь записывает данные и адрес на предыдущий элемент в элемент) В конце возвращаем указатель на верхний элемент.

Создам функцию (void print\_stack(Stack\* top)) для вывода элементов стека. В начале проверим с помощью оператора выбора (if) указывает ли верхний элемент стека (top) на NULL. Если да, то выводим на экран сообщение “стек пуст”. Иначе с помощью (while) выводим элемент и переходим к следующему пока указатель не будет равен NULL.

Создам функцию (char pop(Stack\*& top)) для возвращения первого элемента и его удаления. Для начала считаем количество элементов в стеке (k) с помощью (while). Потом с помощью (if) проверяем равно ли (k) единице. Если да, то обнуляем указатель и возвращаем элемент. Иначе сохраняем значение последнего элемента, делаем второй элемент первым, удаляем последний элемент и возвращаем первый элемент.

Создам функцию (Stack\* push(Stack\*& top, char val)) для добавления элемента в стек. Сначала выделим память под новый элемент. Потом присваиваем значение , которое вводит пользователь, для нового элемента, делаем указатель на нижний элемент и делаем новый элемент первым элементом стека.

Записи в стеке содержат ключевое поле типа char(строка символов). Сформировать стек:

Так как не сказано сколько элементов содержит стек, то пользователь должен ввести количество (n). Потом вызвать ранее написанные функцию для создания стека и его заполнения (Stack\* st = make\_stack(n);) и функцию для вывода стека (print\_stack(st);).

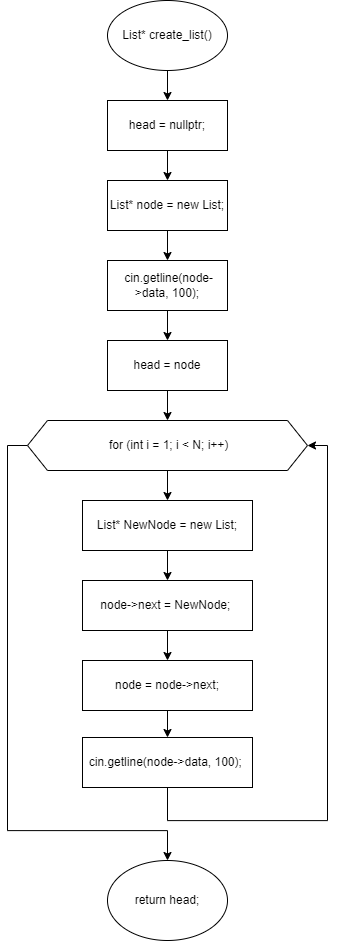
Удалить элемент с заданным ключом:

Чтобы выполнить поставленную задачу создадим второй стек, в который будут переноситься нужные элементы главного стека (Stack\* st2 = make\_stack(0);). Потом, так как ключ не указан, пользователь должен ввести ключ (b) для удаления. После чего с помощью цикла (for (int i = 0; i < n; i++)) и оператора выбора (if) перенесём все нужные элементы во второй стек и посчитаем количество элементов равных ключу (k). Для того, чтобы определить количество элементов после удаления вычтим из n-k. Затем с помощью цикла (for (int i = 0; i < n; i++) ) перенесём элементы из второго стека в исходный. После чего с помощью ранее написанной функции (print) выведем элементы стека на экран.

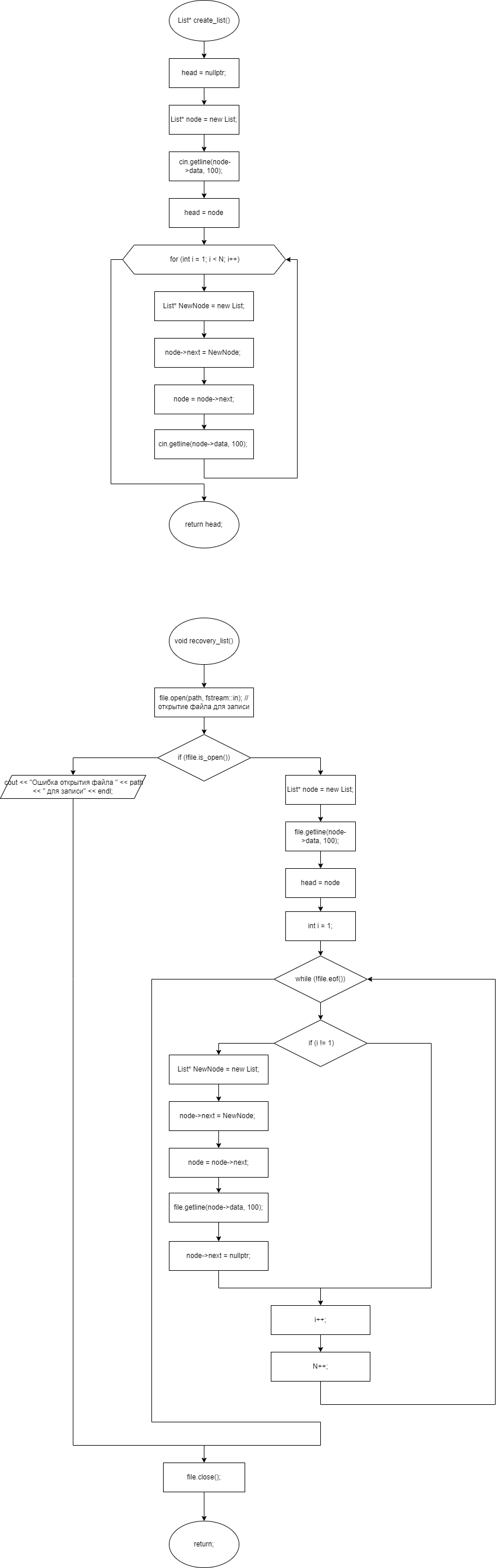
Добавить К элементов перед элементом с заданным номером:

Так как количество элементов для добавления (k) и номер элемента (s), перед которым добавляют элементы,, не указаны, пользователь должен ввести (k) и (s). После этого перенесём с помощью цикла (for (int i = 0; i < n-s+1; i++)) (n-s+1, так как нужно перенести все элементы включая (s) в стек) нужные элементы во второй стек. Затем с помощью цикла (for (int i = 0; i < k; i++)) добавим (k) элементов в исходный стек. После чего с помощью цикла (for (int i = 0; i < n - s + 1; i++)) перенесём элементы из второго стека в исходный и с помощью ранее написанной функции (print) выведем элементы стека на экран.

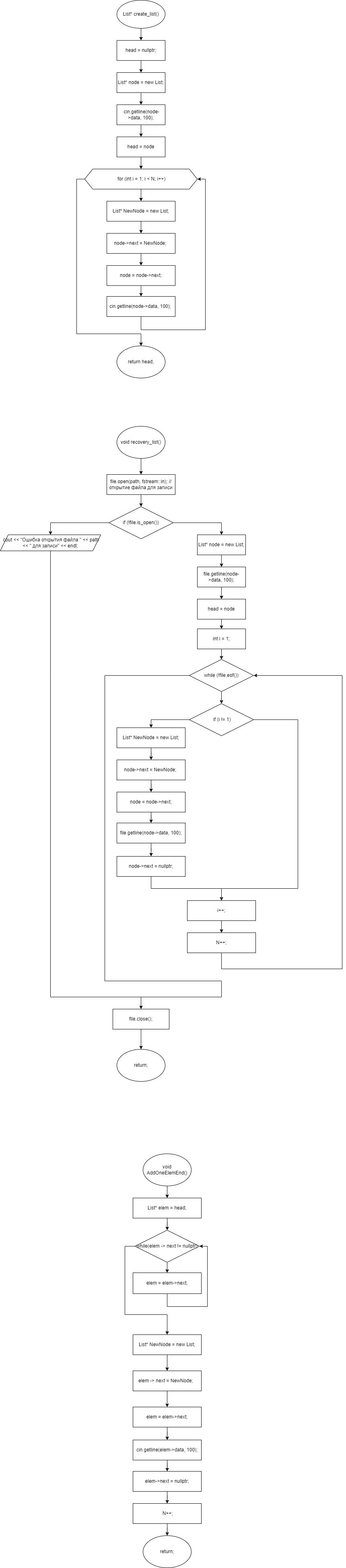
## **Блок – схема**



2) Восстановление из файла

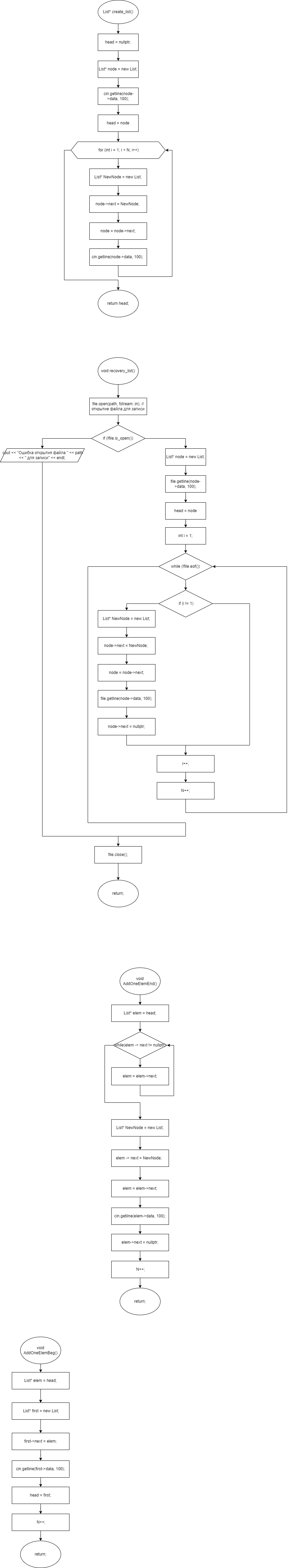


3) Добавление одного элемента в конец

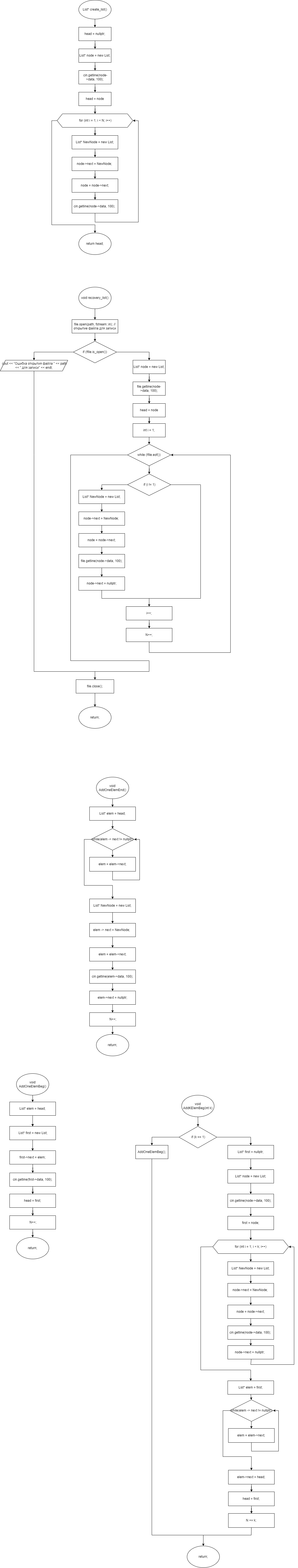


4) Добавление k элементов в конец– вызов k раз функции 3).

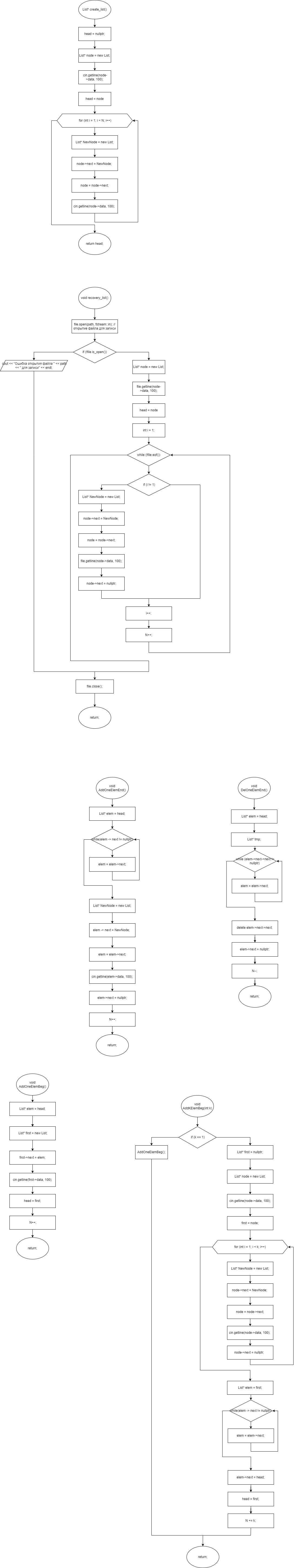
5) Добавление одного элемента в начало



6) Добавление k элементов в начало

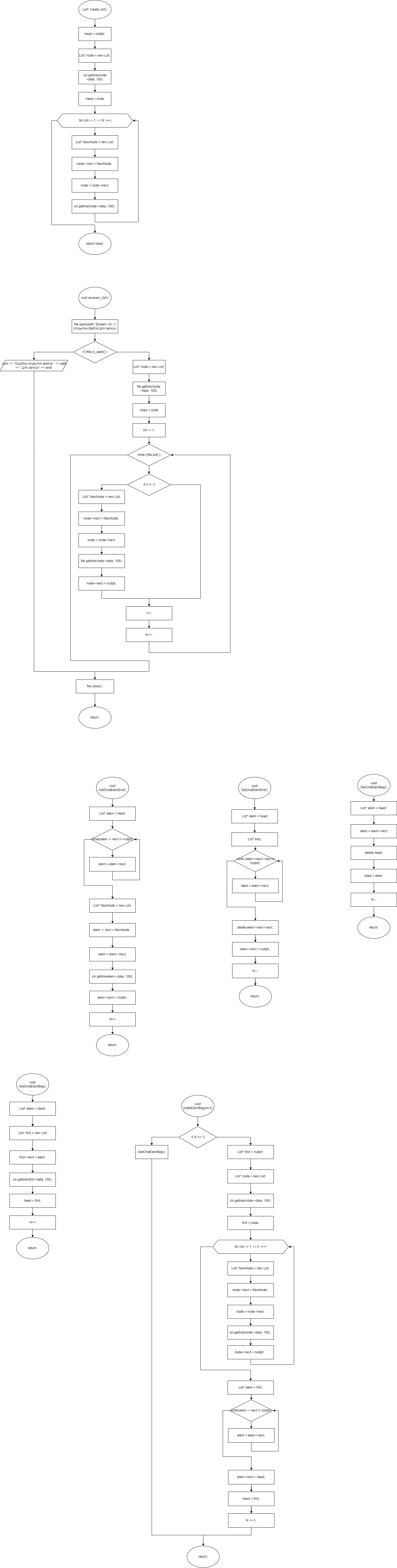


7) Удаление элемента из конца



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

9) Удаление элемента из начала



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

# **Приложение А**

# **Листинг программы**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Stack

{

int data;

Stack\* next;

};

int N; //временное количество элементов в стеке

Stack\* peek; // вершина стека

void Push(Stack\* elem) // Добавление элемента в вершину стека

{

if (N != 0)

elem->next = peek;

peek = elem;

N++;

}

void Push\_k(int k)

{

int i = 0;

Stack\* node = new Stack;

while (i < k)

{

node = new Stack;

cin >> node->data; // Запись нового элемента

Push(node);

i++;

}

}

Stack\* Pop() // удаление из вершины стека

{

Stack\* delElem = peek;

peek = peek->next;

delElem->next = nullptr;

N--;

return delElem;

}

void Pop\_k(int k)

{

int i = 0;

while (i < k)

{

Pop();

i++;

}

}

void create()

{

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

peek = nullptr;

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

cout << "Введите элементы стека: ";

cin >> node->data;

node->next = nullptr;

peek = node;

int n = N;

N = 1;

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

{

node = new Stack; // Переход указателем на следующий элемент

cin >> node->data; // Запись нового элемента

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

Push(node);

}

cout << "Стек заполнен" << endl;

}

void PrintConsole()

{

Stack\* first = peek;

while (first != NULL)

{

cout << first->data << " ";

first = first->next;

}

cout << endl;

}

fstream file;

string path = "Stack.txt";

void PrintFile()

{

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

if (!file.is\_open())

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

else

{

Stack\* first = peek;

while (first != NULL)

{

file << first->data << endl;

first = first->next;

}

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

}

file.close();

}

void Delete()

{

while (N != 0)

{

Stack\* tmp = Pop();

delete tmp;

}

}

void Recovery()

{

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

if (!file.is\_open())

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

else

{

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

string tmp;

getline(file,tmp);

node->data = stoi(tmp);

node->next = nullptr;

peek = node;

int i = 1; // позиция курсора

N = 1;

while (!file.eof())

{

if (i != 1)

{

node = new Stack; // Выделение димнамической памяти под новый элемент

getline(file, tmp);

if (tmp != "")

{

node->data = stoi(tmp); // Запись нового элемента

node->next = nullptr;

Push(node);

}

}

i++;

}

cout << "Стек восстановлен!" << endl;

}

file.close();

}

void print\_menu() {

system("cls"); // очищаем экран

cout << "\t\t\t\t~~~~~~~~ СТЕК ~~~~~~~~" << endl << endl;

cout << "\t\t\t\t1. Создать стек" << endl;

cout << "\t\t\t\t2. Добавить k элементов в стек" << endl; // из вершины стека

cout << "\t\t\t\t3. Удалить k элементов из стека" << endl; // из вершины стека

cout << "\t\t\t\t4. Вывод стека в консоль" << endl;

cout << "\t\t\t\t5. Запись стека в файл" << endl;

cout << "\t\t\t\t6. Уничтожение стека" << endl;

cout << "\t\t\t\t7. Восстановление стека из файла" << endl;

cout << "\t\t\t\t8. Выход" << endl;

cout << endl;

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

cout << 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;

}

int main()

{

system("chcp 1251>NULL");

print\_menu();

int k;

int var;

do {

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

var = get\_var(8); // получаем номер выбранного пункта меню

switch (var)

{

case 1:

cout << "Введите количество элементов стека: ";

cin >> N;

create();

break;

case 2:

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

cin >> k;

cout << "Введите элементs: ";

Push\_k(k);

break;

case 3:

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

cin >> k;

Pop\_k(k);

break;

case 4:

if (N == 0)

{

cout << "Элементы в стеке осутствуют!" << endl;

}

cout << "Cтек: ";

PrintConsole();

break;

case 5:

PrintFile();

break;

case 6:

Delete();

if (N == 0) {

cout << "Cтек удален!" << endl;

peek = nullptr;

}

break;

case 7:

Recovery();

break;

}

if (var != 8)

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

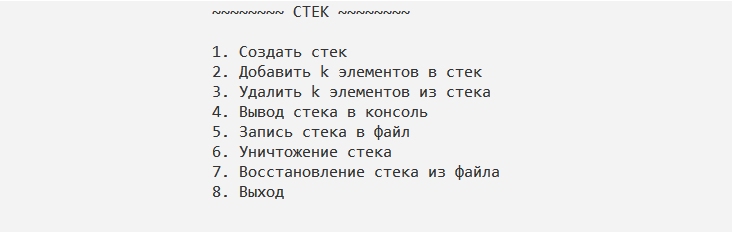
} while (var != 8);

}

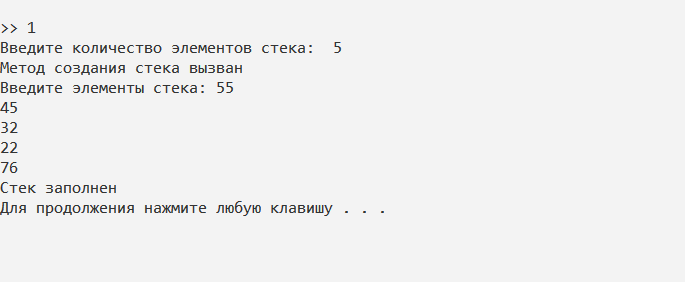
# **Приложение Б**

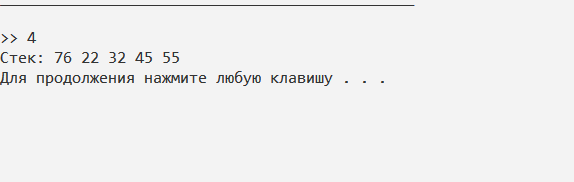
# **Результаты выполнения программы**

Меню

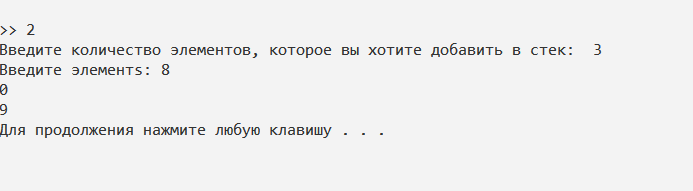


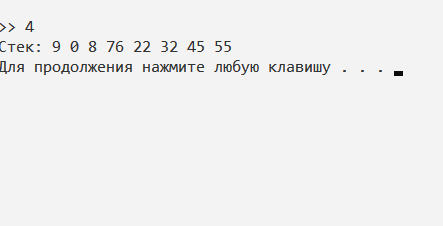
Создание стека



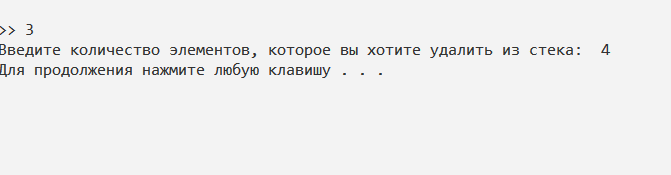


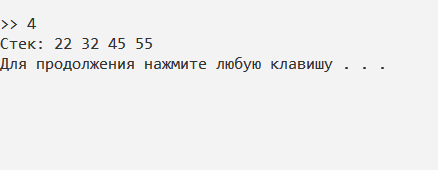
Добавить k элементов в стек



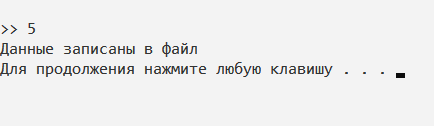


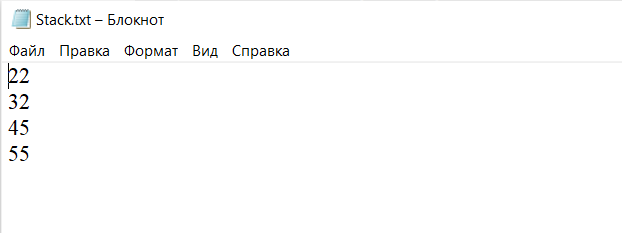
Удалить k элементов из стека



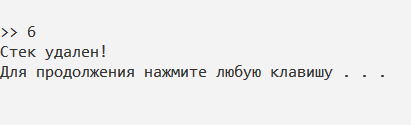


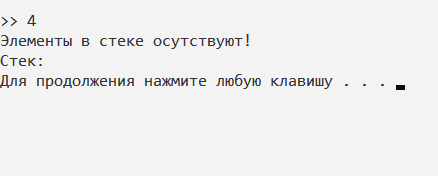
Запись стека в файл





Уничтожение стека





Восстановление стека из файла

