Министерство высшего образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Пермский национальный исследовательский политехнический университет» (ПНИПУ)**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

ОТЧЁТ

по лабораторной работе №11

Тема: «Информационные динамические структуры»

Выполнил

Студент группы РИС-22-2б

Баяндин К. С.

Проверил доц. Кафедры ИТАС

Полякова Ольга Андреевна

Пермь 2023

**Постановка задачи**

Написать программу, в которой создаются динамические структуры, и выполнить их обработку в соответствии со своим вариантом.

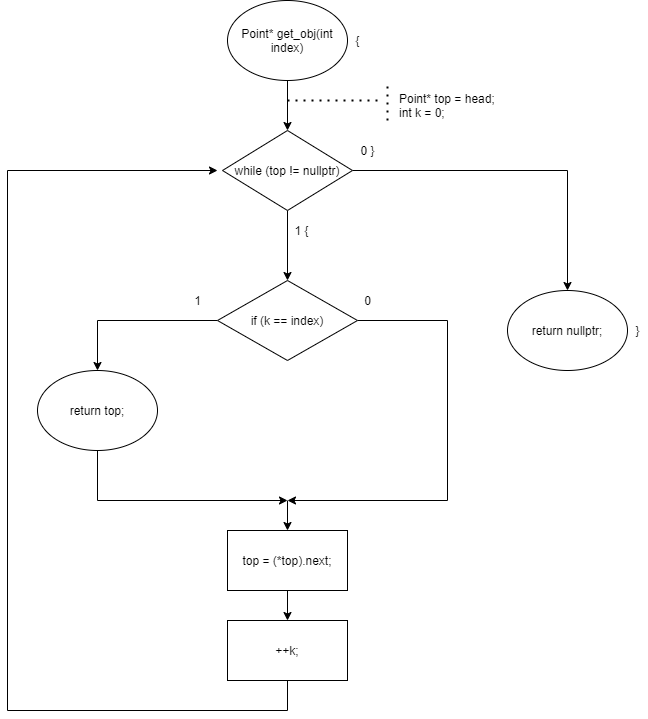
**Для каждого вариант разработать следующие функции:**

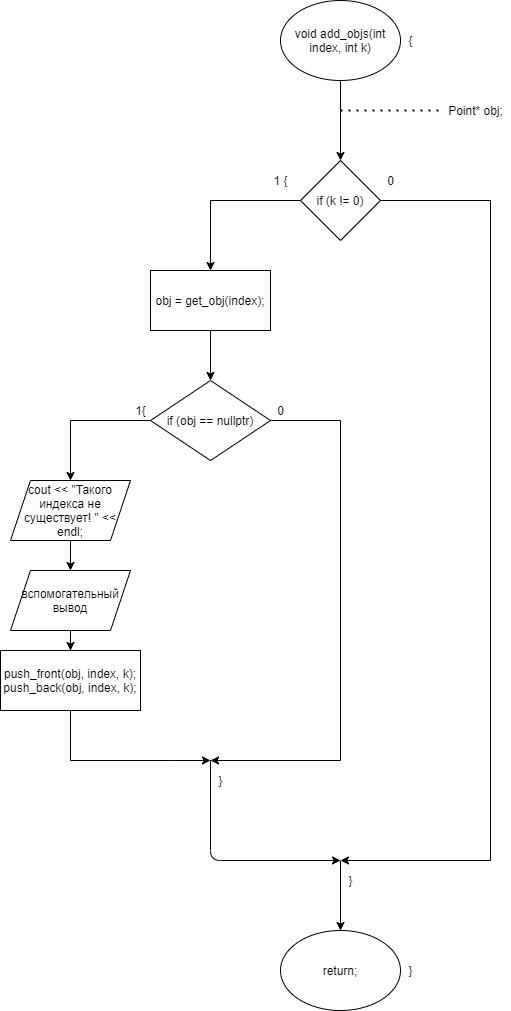
1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

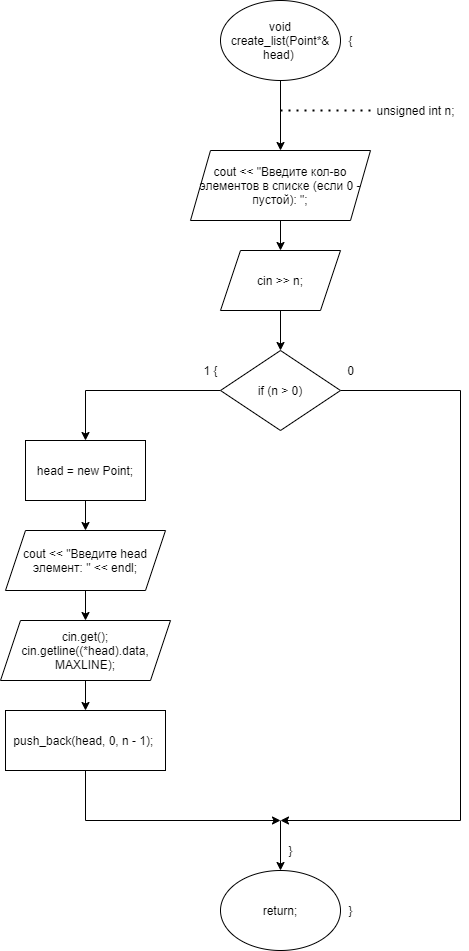
**25 ВАРИАНТ.**

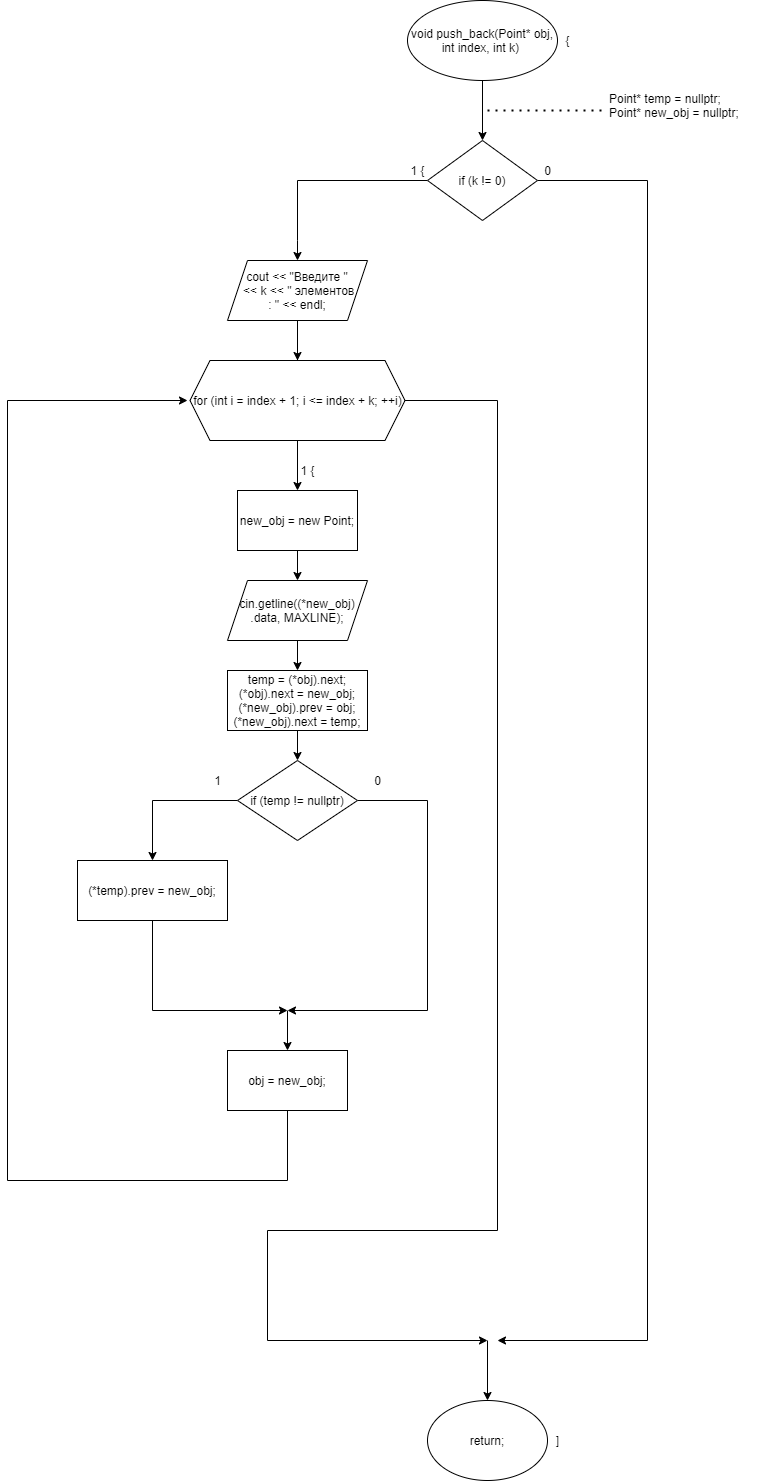
Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить по К элементов перед и после элемента с заданным номером.

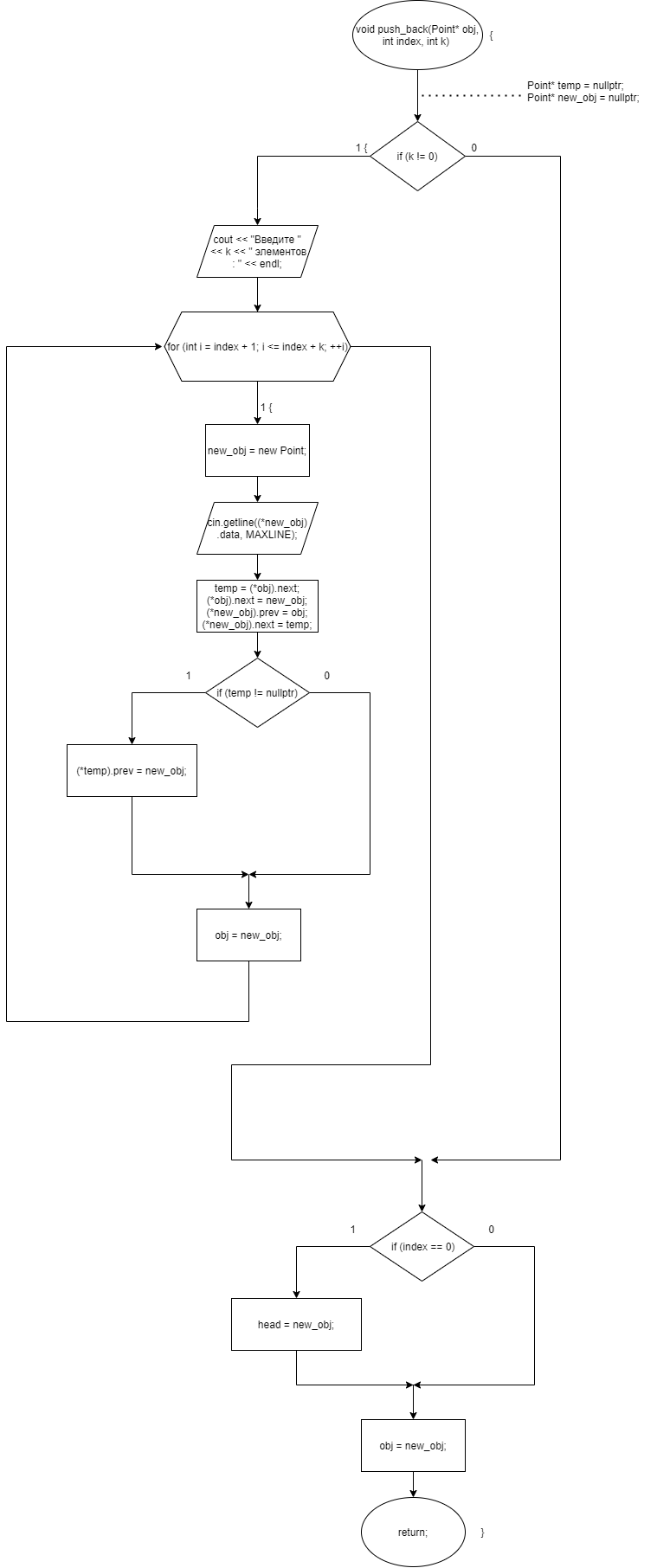
**Блок-схемы:**

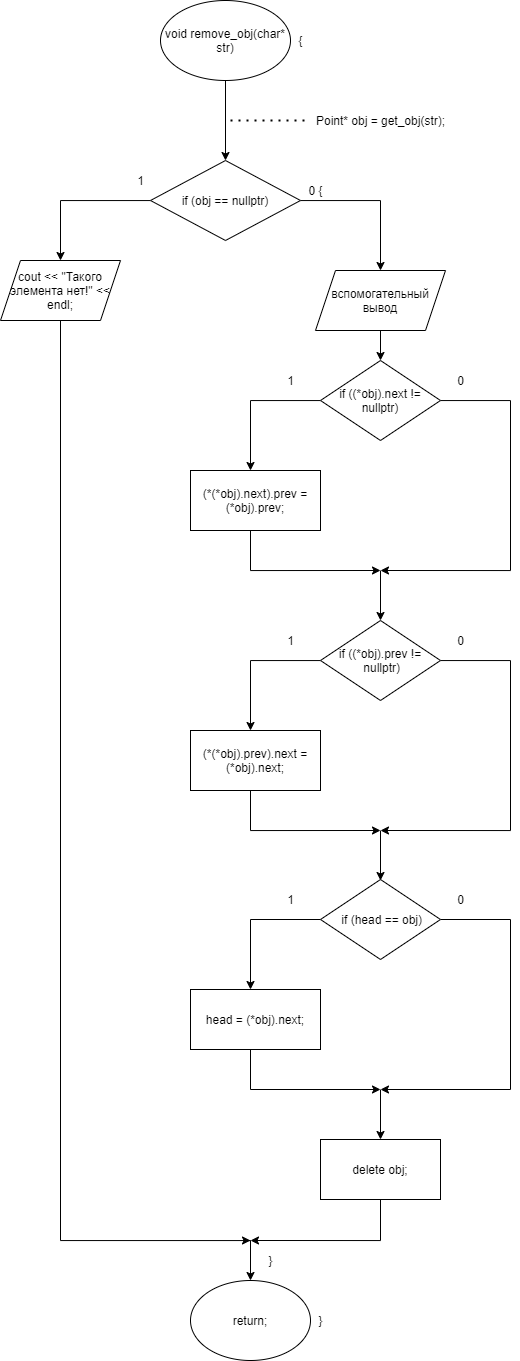


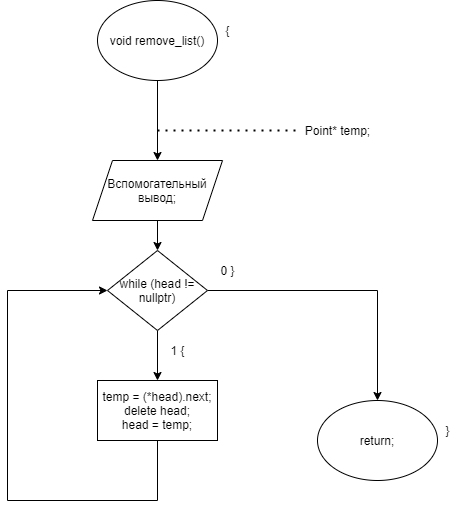


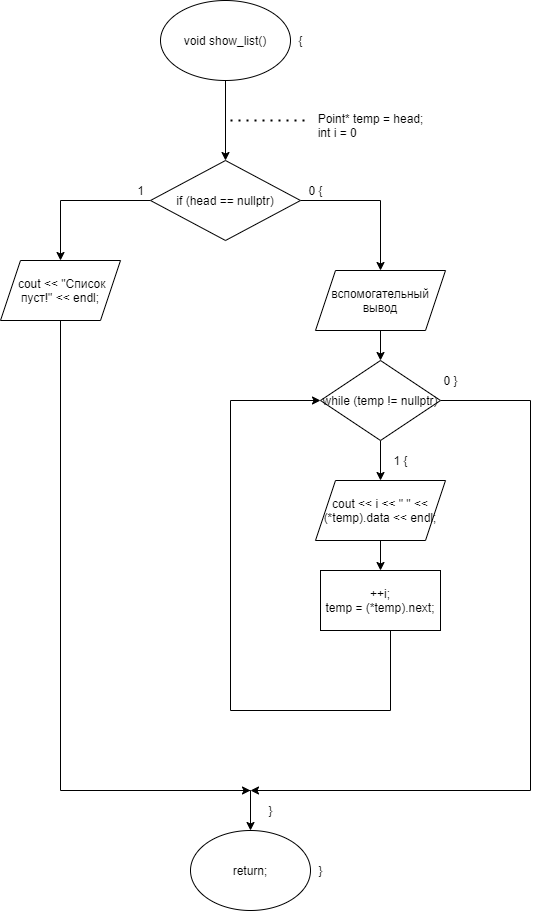


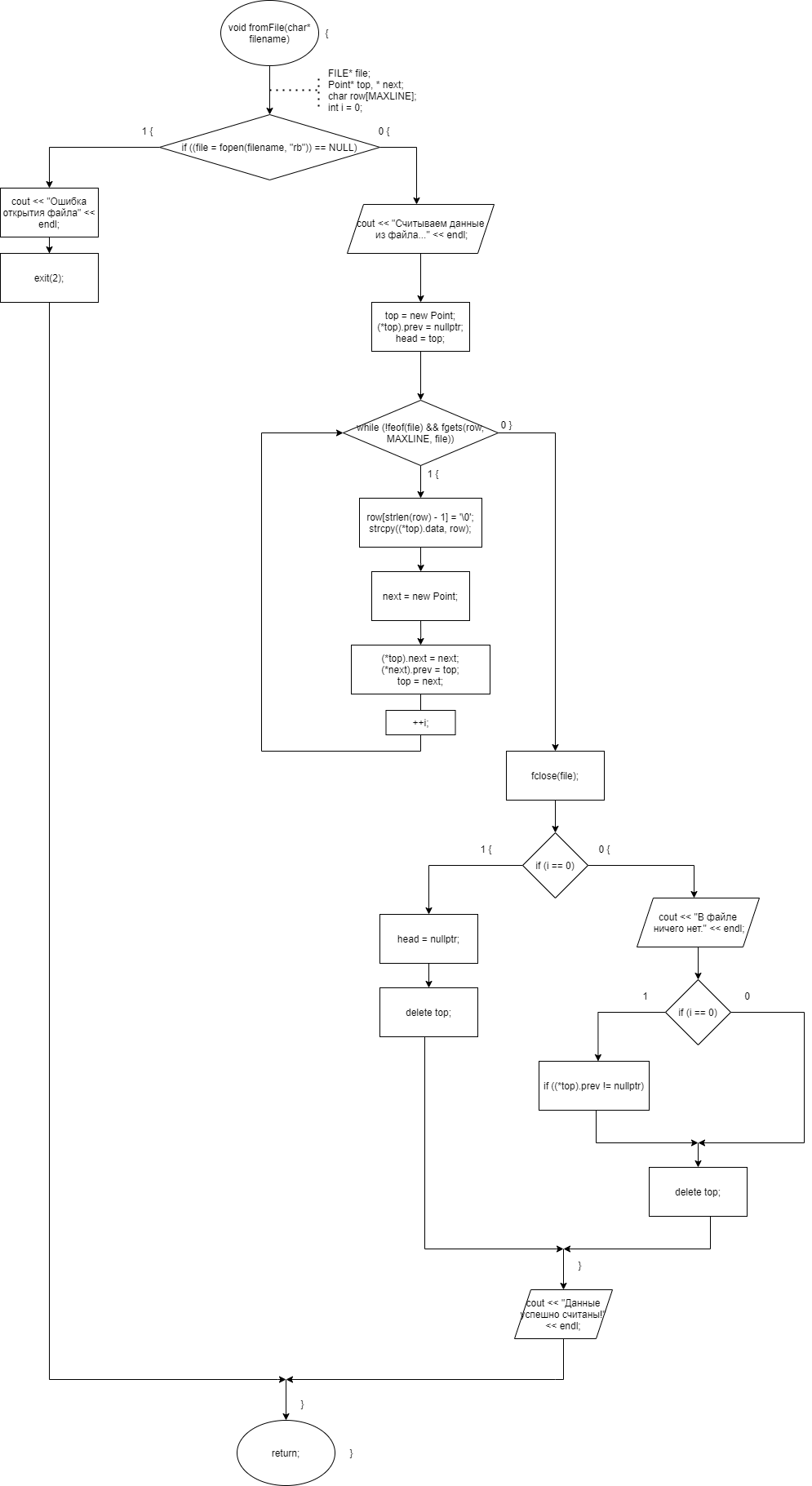


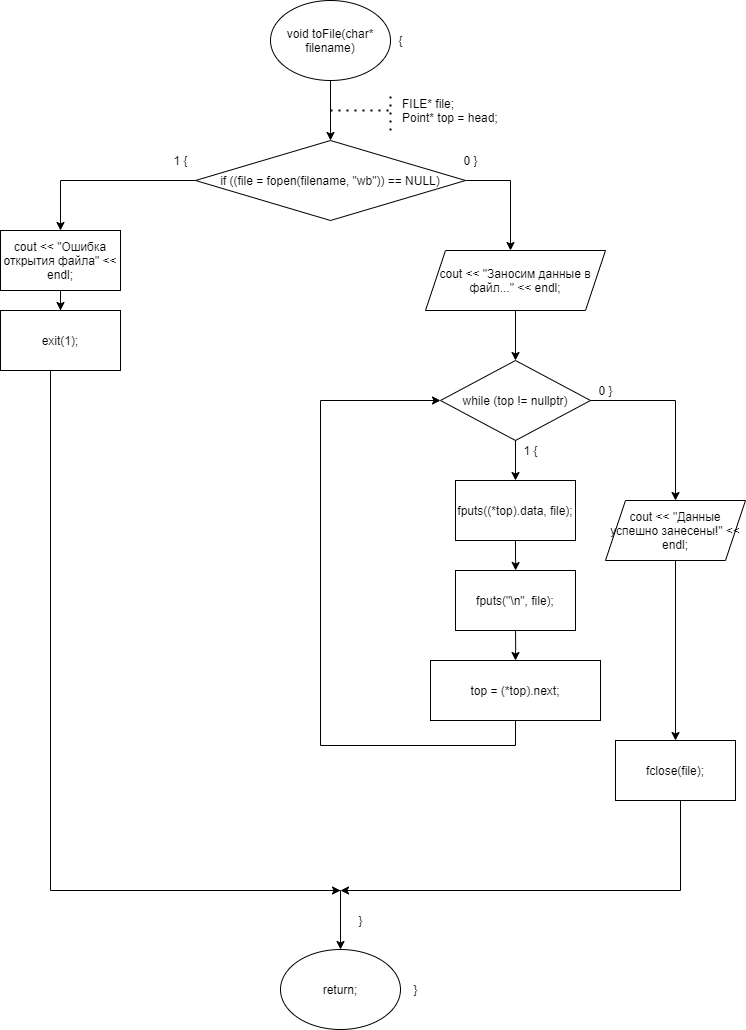


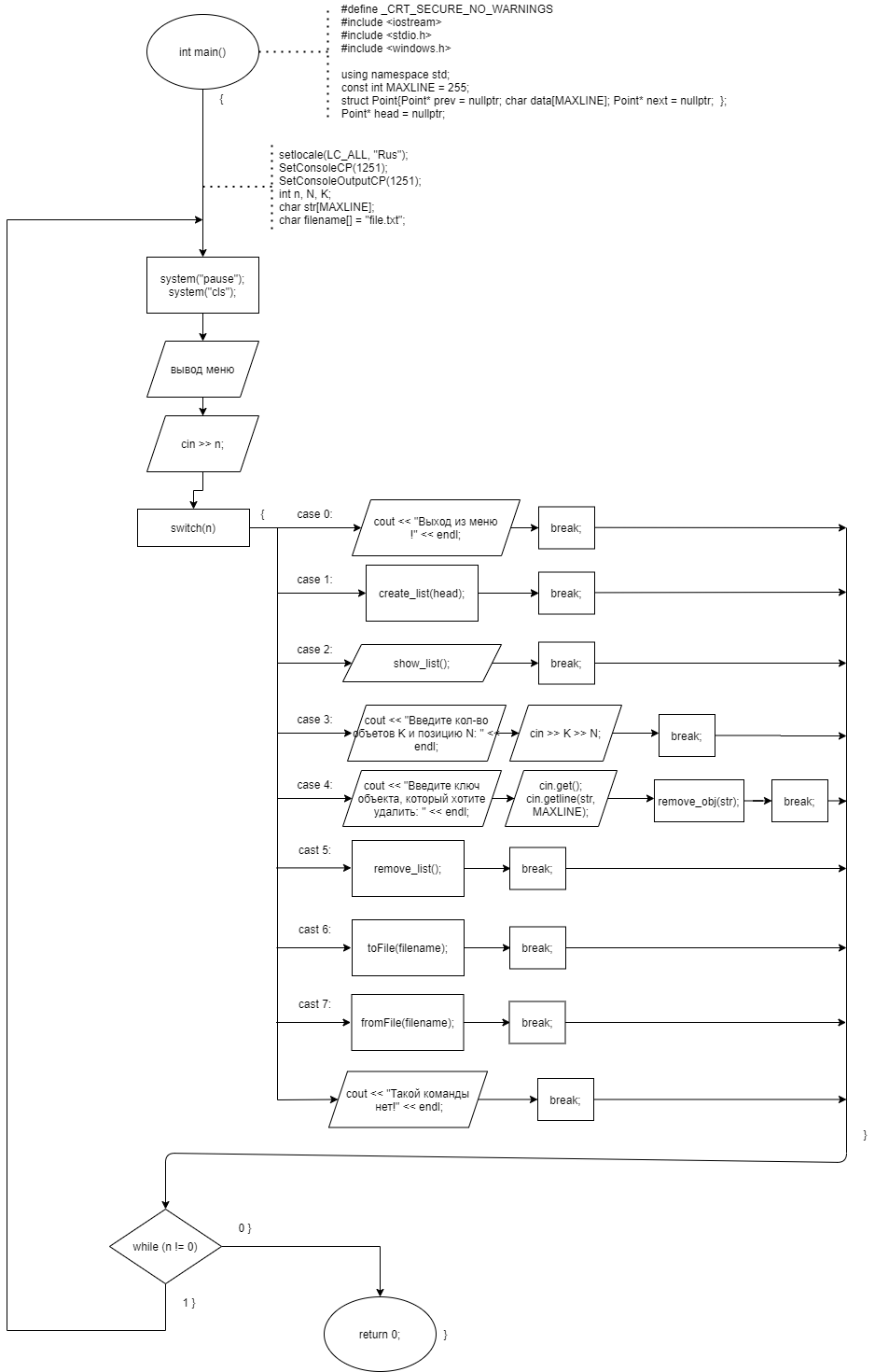




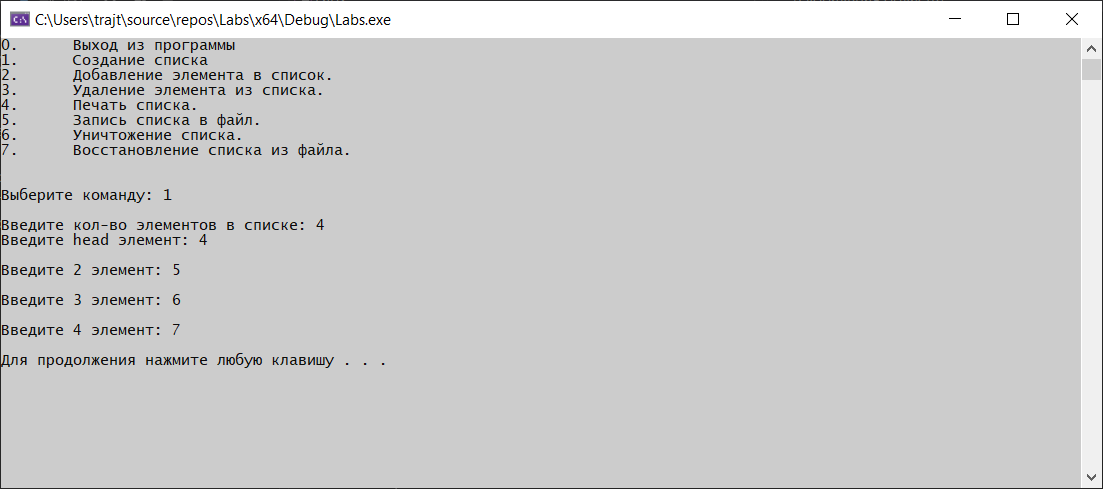


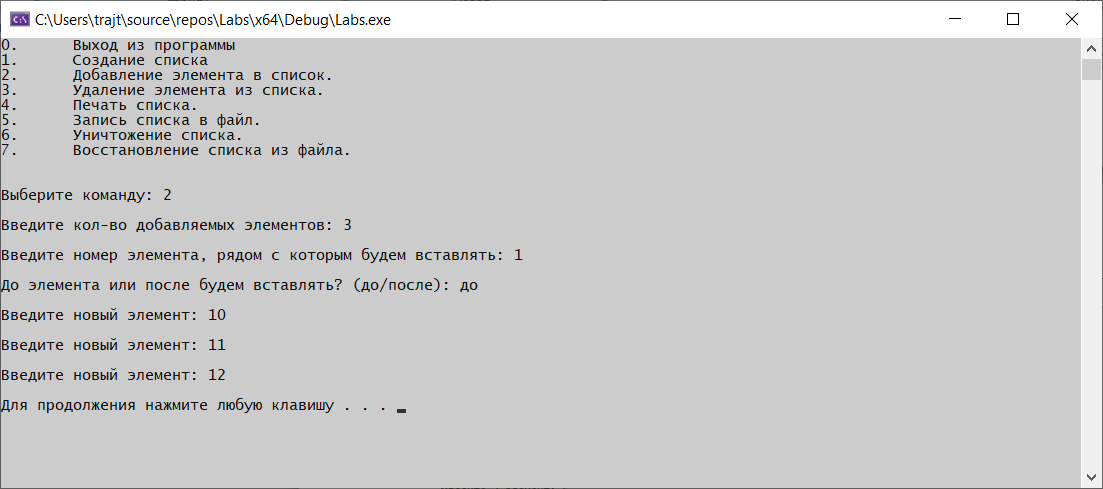


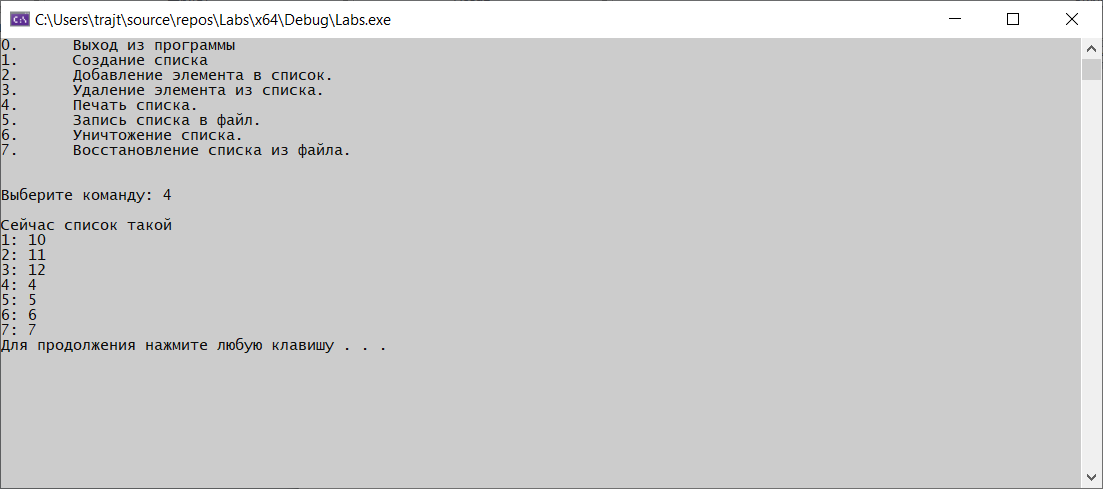


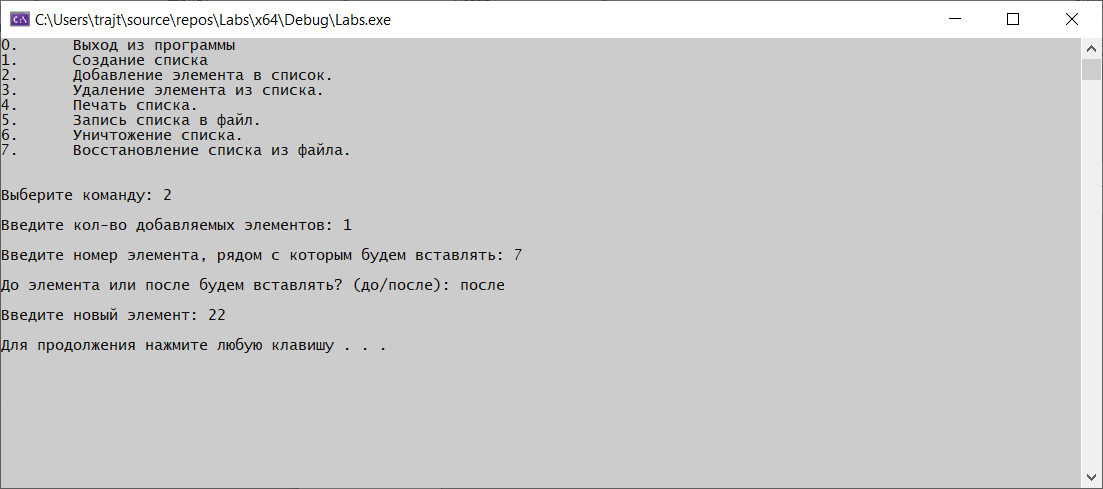


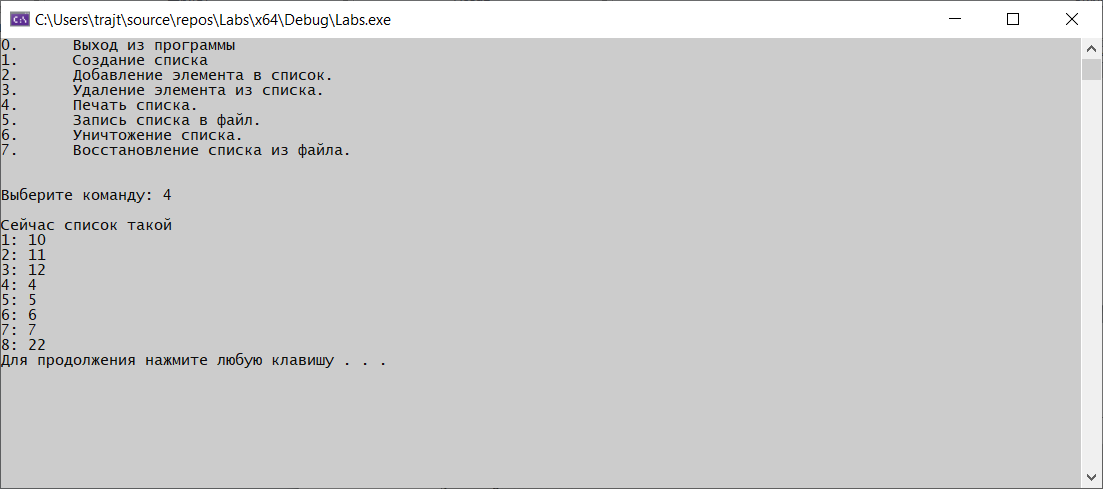
**Результаты работы программы:**





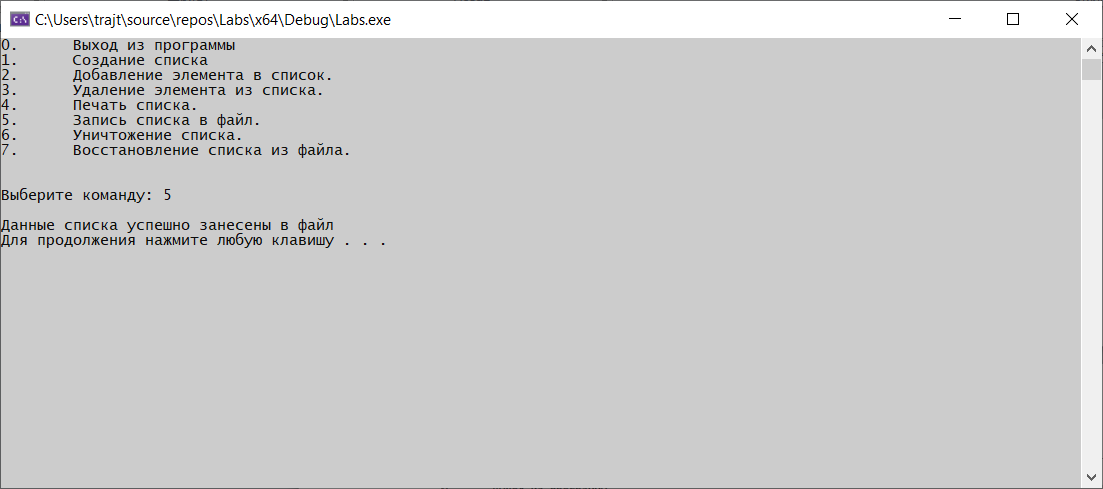




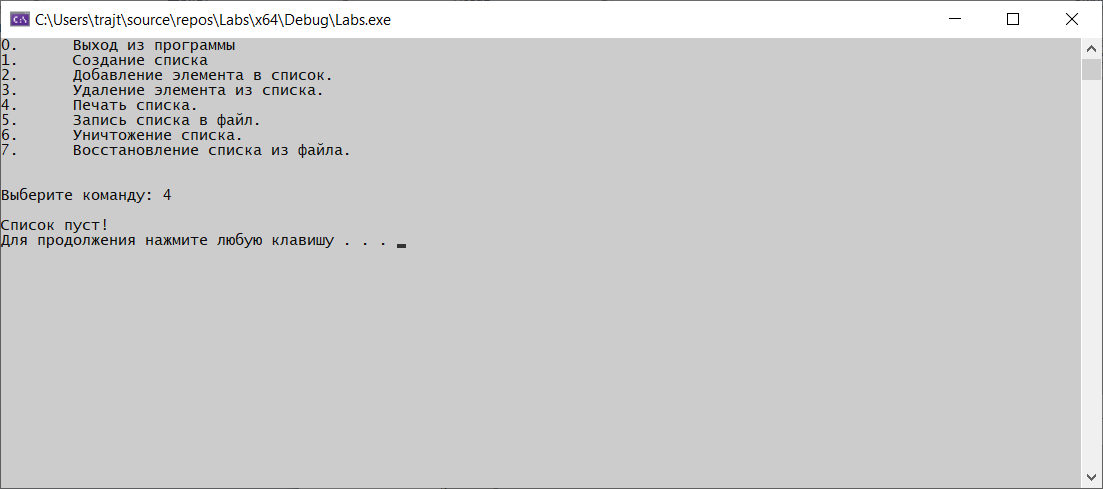


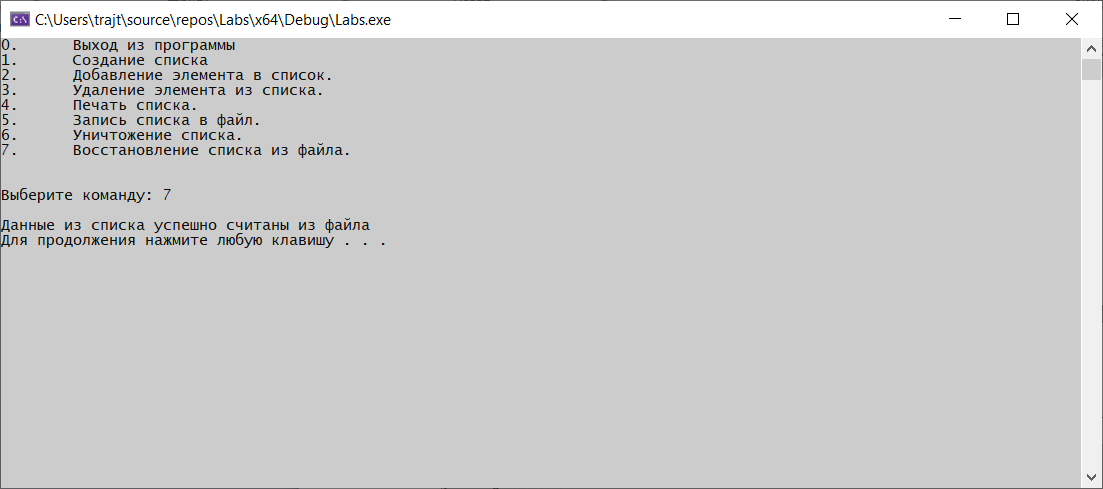














**Односвязный и двусвязный списки**

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <Windows.h>

#include <stdio.h>

using namespace std;

struct Element

{

char data[255]; //поле данных

Element\* next = nullptr; //указатель на следующий элемент

Element\* prev = nullptr; //указатель на предыдущий элемент

};

Element\* head = nullptr;

//функция создания списка

void createList(Element\*& head, int n)

{

Element\* start = nullptr;

Element\* end = nullptr;

head = new Element;

cout << "Введите head элемент: ";

cin.get();

cin.getline((\*head).data, 255);

cout << endl;

start = head;

end = head;

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

{

cout << "Введите " << i + 1 << " элемент: ";

Element\* elem = new Element;

elem->next = nullptr;

cin.getline((\*elem).data, 255);

cout << endl;

elem->prev = end;

end->next = elem;

end = elem;

}

}

//функция поиска объекта по индексу

Element\* findObj(int index)

{

Element\* temp = head;

int k = 1;

while (temp != nullptr)

{

if (k == index)

return temp;

else {

k++;

temp = temp->next;

}

}

return nullptr;

}

//функция поиска объекта по ключу

Element\* findObj(char\* str)

{

Element\* temp = head;

while (temp != nullptr)

{

if (strcmp((\*temp).data, str) == 0) {

return temp;

}

else {

temp = temp->next;

}

}

return nullptr;

}

//функция добавления элемента после заданного номера

void addElementAfter(int index)

{

Element\* end = findObj(index);

Element\* elemIns = new Element;

cout << "Введите новый элемент: ";

cin.getline((\*elemIns).data, 255);

cout << endl;

//Добавление перед первым элементом

if (index == 1) {

Element\* start = nullptr;

elemIns->next = end;

elemIns->prev = start;

end->prev = elemIns;

head = elemIns;

}

//Добавление в середину перед заданным элементом

else {

Element\* start = end->prev;

elemIns->next = end;

elemIns->prev = start;

start->next = elemIns;

end->prev = elemIns;

}

}

//функция добавления элемента перед заданным номером

void addElementBefore(int index, int size)

{

Element\* start = findObj(index);

Element\* elemIns = new Element;

cout << "Введите новый элемент: ";

cin.getline((\*elemIns).data, 255);

cout << endl;

//добавление в конец после заданного номера

if (index == size) {

Element\* end = nullptr;

elemIns->next = end;

elemIns->prev = start;

start->next = elemIns;

}

//добавление в середину после заданного номера

else {

Element\* end = start->next;

elemIns->next = end;

elemIns->prev = start;

start->next = elemIns;

end->prev = elemIns;

}

}

//функция удаления элемента

void removeElementList(char\* s)

{

Element\* elemIns = findObj(s);

if (elemIns == nullptr)

cout << "Ошибка! Такого элемента нет!" << endl;

else {

if ((\*elemIns).prev != nullptr)

(\*elemIns).prev->next = elemIns->next;

else

head = elemIns->next;

if ((\*elemIns).next != nullptr)

(\*elemIns).next->prev = elemIns->prev;

delete elemIns;

}

}

//функция печати списка

void printList()

{

Element\* elem = head;

for (int i = 1; elem != nullptr; i++)

{

cout << i << ": " << (\*elem).data << endl;

elem = elem->next;

}

}

//запись списка в файл

void toFile(char\* filename)

{

FILE\* file;

if ((file = fopen(filename, "wb")) == NULL)

{

cout << "Ошибка открытия файла" << endl;

return;

}

Element\* temp = head;

while (temp != nullptr)

{

fputs(temp->data, file);

fputs("\n", file);

temp = temp->next;

}

cout << "Данные списка успешно занесены в файл" << endl;

fclose(file);

}

//функция удаления списка

void deleteList()

{

Element\* elem = head;

while (head != nullptr)

{

elem = head->next;

delete head;

head = elem;

}

}

//функция восстановления списка из файла

int fromFile(char\* filename)

{

FILE\* file;

char row[255];

int i = 0;

if ((file = fopen(filename, "rb")) == NULL)

{

cout << "Ошибка открытия файла" << endl;

return i;

}

Element\* start = new Element;

Element\* end;

start->prev = nullptr;

head = start;

//пока файл не пустой, считываем элементы

while (!feof(file) && fgets(row, 255, file))

{

row[strlen(row) - 1] = '\0';

strcpy(start->data, row);

end = new Element;

start->next = end;

end->prev = start;

start = end;

++i;

}

fclose(file);

if (i == 0) {

cout << "В файле ничего нет." << endl;

head = nullptr;

delete start;

}

else {

if (start->prev != nullptr)

start->prev->next = nullptr;

delete start;

}

cout << "Данные из списка успешно считаны из файла" << endl;

return i;

}

//меню команд

const char menu[] =

"0. Выход из программы\n"

"1. Создание списка\n"

"2. Добавление элемента в список.\n"

"3. Удаление элемента из списка.\n"

"4. Печать списка.\n"

"5. Запись списка в файл.\n"

"6. Уничтожение списка.\n"

"7. Восстановление списка из файла.\n";

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int cmd, k, num;

int size = 0;

char keys[255];

char orientation[255];

char filename[] = "list.txt";

for (;;)

{

cout << menu << endl;

cout << "\nВыберите команду: ";

cin >> cmd;

cout << endl;

switch (cmd)

{

case 0: //выход из программы

cout << "Спасибо что выбрали данную программу!" << endl;

return 0;

case 1: //создание списка

cout << "Введите кол-во элементов в списке: ";

cin >> size;

createList(head, size);

break;

case 2: //добавление элементов в список

cout << "Введите кол-во добавляемых элементов: ";

cin >> k;

cout << "\nВведите номер элемента, рядом с которым будем вставлять: ";

cin >> num;

cout << "\nДо элемента или после будем вставлять? (до/после): ";

cin.get();

cin.getline(orientation, 255);

if (num >= 1 && num <= size) {

if (strcmp(orientation, "до"))

{

cin.get();

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

{

addElementBefore(num, size);

size++;

num++;

}

}

else

{

cin.get();

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

{

addElementAfter(num);

size++;

num++;

}

}

}

else

cout << endl << "Ошибка! Введите корректный номер элемента!" << endl;

break;

case 3: //удаление элемента из списка по ключу

cout << "Введите ключ удаляемого элемента: ";

cin.get();

cin.getline(keys, 255);

removeElementList(keys);

size--;

break;

case 4: //печать списка

if (size <= 0) {

cout << "Список пуст!" << endl;

}

else {

cout << "Сейчас список такой" << endl;

printList();

}

break;

case 5: //запись в файл

toFile(filename);

break;

case 6: //удаление списка

if (size == 0) {

cout << "Список пуст, удалять нечего" << endl;

}

else {

deleteList();

cout << "Список успешно удален" << endl;

size = 0;

}

break;

case 7: //считывание с файла

size = fromFile(filename);

break;

default:

cout << "Неправильная команда, попробуйте другую" << endl;

}

system("pause");

system("cls");

}

}

**Стек**

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdio.h>

#include <Windows.h>

using namespace std;

struct Point

{

char data[255]; //поле данных

Point\* next = nullptr; //указатель на следующий элемент стека

};

struct Stack

{

Point\* head = nullptr; //указатель на последний элемент стека

size\_t stack\_size = 0;

};

bool stack\_is\_empty(Stack& stack); //проверка на пустой стек

void output(Point\* obj); //рекурсивная функция для обратного вывода 3 2 1 -> 1 2 3 (ввод был 1 2 3)

void show\_stack(Stack& stack); //вывод стека

void push(Stack& stack, char\* str); //добавить новый элемент сверху

Stack\* create\_stack(int size); //создание стека из n элементов

void remove\_obj(Stack& stack, char\* str); //удалить объект с заданным ключом

void push\_before(Stack& stack, unsigned index, unsigned k); //вставить перед элементом

void remove\_stack(Stack& stack); //удаление списка

void toFile\_obj(Point\* obj, FILE\* file); //рекурсия ф-ия для обратного порядка заноса в файл

void toFile(Stack& stack, char\* filename); //загрузить в файл ключи стека

void fromFile(Stack& stack, char\* filename); //выгрузить из файла ключи стека

bool stack\_is\_empty(Stack& stack) //проверка на пустой стек

{

return stack.head == nullptr; //stack\_size = 0;

}

void output(Point\* obj) //рекурсивная функция для обратного вывода 3 2 1 -> 1 2 3 (ввод был 1 2 3)

{

if (obj == nullptr) return;

output(obj->next);

cout << obj->data << endl;

}

void show\_stack(Stack& stack) //вывод стека

{

if (stack\_is\_empty(stack)) cout << "Стек пуст" << endl;

else output(stack.head);

}

void push(Stack& stack, char\* str) //добавить новый элемент сверху

{

stack.stack\_size++;

Point\* obj = new Point;

strcpy(obj->data, str);

obj->next = stack.head;

stack.head = obj;

}

void pop(Stack& stack)

{

if (stack.head != nullptr)

{

Point\* temp = stack.head;

stack.head = stack.head->next;

delete temp;

stack.stack\_size--;

}

}

Stack\* create\_stack(int size) //создание стека из n элементов

{

Stack\* stack = new Stack;

if (size > 0 && stack\_is\_empty(\*stack))

{

char str[255];

cout << "Введите ключи " << size << " элементов стека: " << endl;

cin.get();

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

{

cout << i << ". ";

gets\_s(str, 255);

push(\*stack, str);

}

}

return stack;

}

void push\_before(Stack& stack, unsigned index, unsigned k)

{

int new\_index = stack.stack\_size - index + 1;

if (new\_index < 0 || stack\_is\_empty(stack)) cout << "Введен неверный индекс/стек пуст" << endl;

else

{

char str[255];

Stack\* stack2 = new Stack;

for (int i = 1; i <= new\_index; i++)

{

push(\*stack2, stack.head->data);

pop(stack);

}

cout << "Введите " << k << " элементов: " << endl;

cin.get();

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

{

cin.getline(str, 255);

push(stack, str);

}

for (int i = 0; i < new\_index; i++)

{

push(stack, stack2->head->data);

pop(\*stack2);

}

}

}

void remove\_obj(Stack& stack, char\* str)

{

int stack\_size = stack.stack\_size;

Stack\* stack2 = create\_stack(0);

for (int i = 1; i <= stack\_size; i++)

{

if (strcmp(stack.head->data, str) != 0)

{

push(\*stack2, stack.head->data);

}

pop(stack);

}

cout << endl;

int stack\_size2 = stack2->stack\_size;

for (int i = 1; i <= stack\_size2; i++)

{

push(stack, stack2->head->data);

pop(\*stack2);

}

}

void remove\_stack(Stack& stack) //удаление списка

{

Point\* temp;

while (stack.head != nullptr)

{

temp = stack.head->next;

delete stack.head;

stack.head = temp;

}

stack.stack\_size = 0;

cout << "Список удален!" << endl;

}

void toFile\_obj(Point\* obj, FILE\* file)

{

if (obj == nullptr) return;

toFile\_obj(obj->next, file);

fputs(obj->data, file);

fputs("\n", file);

}

void toFile(Stack& stack, char\* filename) //загрузить в файл ключи стека

{

FILE\* file;

if ((file = fopen(filename, "wb")) == NULL) //ошибка открытия файла

{

cout << "Ошибка открытия файла" << endl;

exit(1);

}

cout << "Заносим данные в файл..." << endl;

toFile\_obj(stack.head, file);

cout << "Данные успешно занесены!" << endl;

fclose(file);

}

void fromFile(Stack& stack, char\* filename) //выгрузить из файла ключи списка

{

FILE\* file;

char row[255];

if ((file = fopen(filename, "rb")) == NULL) //ошибка открытия файла

{

cout << "Ошибка открытия файла" << endl;

exit(2);

}

cout << "Считываем данные из файла..." << endl;

while (!feof(file) && fgets(row, 255, file))

{

row[strlen(row) - 1] = '\0'; //заносили с \n, нужно его убрать.

push(stack, row);

}

fclose(file);

cout << "Данные успешно считаны!" << endl;

}

//меню команд

const char menu[] =

"0. Выход из программы\n"

"1. Создание стека\n"

"2. Добавление элемента в стек.\n"

"3. Удаление элемента из стека.\n"

"4. Печать стека.\n"

"5. Запись стека в файл.\n"

"6. Уничтожение стека.\n"

"7. Восстановление стека из файла.\n";

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Stack\* stack = new Stack;

int cmd, k, n;

int size = 0;

char keys[255];

char filename[] = "list.txt";

for (;;)

{

cout << menu << endl;

cout << "\nВыберите команду: ";

cin >> cmd;

cout << endl;

switch (cmd)

{

case 0: //выход из программы

cout << "Спасибо что выбрали данную программу!" << endl;

return 0;

case 1: //создание стека

cout << "Введите кол-во элементов в списке: ";

cin >> size;

stack = create\_stack(size);

break;

case 2: //добавление элементов в стека

cout << "Введите кол-во объетов и позицию: " << endl;

cin >> k >> n;

push\_before(\*stack, k, n);

break;

case 3: //удаление элемента из стека по ключу

cout << "Введите ключ объекта, который хотите удалить: " << endl;

cin.get();

cin.getline(keys, 255);

remove\_obj(\*stack, keys);

break;

case 4: //печать стека

if (size <= 0) {

cout << "Стек пуст!" << endl;

}

else {

cout << "Сейчас стек такой" << endl;

show\_stack(\*stack);

}

break;

case 5: //запись в файл

toFile(\*stack, filename);

break;

case 6: //удаление стека

if (size == 0) {

cout << "Стек пуст, удалять нечего" << endl;

}

else {

remove\_stack(\*stack);

cout << "Стека успешно удален" << endl;

size = 0;

}

break;

case 7: //считывание с файла

fromFile(\*stack, filename);

break;

default:

cout << "Неправильная команда, попробуйте другую" << endl;

}

system("pause");

system("cls");

}

}

**Очереди**

**Код программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS //чтобы не жаловался

#include <iostream>

#include <stdio.h> //для работы с файлами

#include <windows.h> //для русского ввода-вывода штучки

using namespace std;

const int MAXLINE = 255; //макс длина строки

struct Point

{

char data[MAXLINE]; //строка, название и тд элемента очереди

Point\* next = nullptr; //следующий элемент очереди

};

Point\* head = nullptr; //указатель на первый элемент очереди

Point\* top = nullptr; //указатель на последний элемент очереди

size\_t size\_queue = 0;

void push(char\* str); //добавить новый элемент сверху

bool queue\_is\_empty(); //проверка на пустой стек

void create\_queue(int size)

{

if (size > 0 && queue\_is\_empty())

{

char str[MAXLINE];

cout << "Введите ключи " << size << " элементов очереди: " << endl;

cin.get();

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

{

cout << i << ". ";

gets\_s(str, MAXLINE);

push(str);

}

}

}

void push(char\* str) //добавить новый элемент сверху

{

if (queue\_is\_empty())

{

head = new Point;

strcpy(head->data, str);

top = head;

}

else

{

Point\* obj = new Point;

strcpy(obj->data, str);

top->next = obj;

top = obj;

}

size\_queue++;

}

void pop()

{

Point\* temp = head;

head = head->next;

size\_queue--;

delete temp;

}

bool queue\_is\_empty() //проверка на пустой стек

{

return size\_queue == 0;

}

void push\_before(int index, int k)

{

if (!queue\_is\_empty() && index >= 1 && index <= size\_queue && k > 0)

{

int i = 1;

while (i < index)

{

push(head->data);

pop();

i++;

}

char str[MAXLINE];

cout << "Введите элементы для вставки: " << endl;

cin.get();

for (int j = 1; j <= k; j++)

{

cin.getline(str, MAXLINE);

push(str);

}

for (i; i <= size\_queue - k; i++)

{

push(head->data);

pop();

}

}

else cout << "Очередь пуста или введен неверный индекс" << endl;

}

void output(Point\* obj)

{

cout << obj->data << endl;

if (obj == top) return;

output(obj->next);

}

void show\_queue() //вывод стека

{

if (queue\_is\_empty()) cout << "Очередь пуста" << endl;

else output(head);

}

void remove\_obj(char\* str)

{

if (queue\_is\_empty()) cout << "Очередь пуста, нечего удалять!" << endl;

int i = 1;

while (i <= size\_queue)

{

if (strcmp(head->data, str) == 0)

{

pop();

}

else

{

push(head->data);

pop();

i++;

}

}

}

void remove\_queue()

{

while (head != nullptr)

{

pop();

}

cout << size\_queue << " Очередь удалена!" << endl;

}

void toFile\_obj(Point\* obj, FILE\* file)

{

if (obj == nullptr) return;

fputs(obj->data, file);

fputs("\n", file);

toFile\_obj(obj->next, file);

}

void toFile(char\* filename) //загрузить в файл ключи стека

{

FILE\* file;

if ((file = fopen(filename, "wb")) == NULL) //ошибка открытия файла

{

cout << "Ошибка открытия файла" << endl;

exit(1);

}

cout << "Заносим данные в файл..." << endl;

toFile\_obj(head, file);

cout << "Данные успешно занесены!" << endl;

fclose(file);

}

void fromFile(char\* filename) //выгрузить из файла ключи списка

{

FILE\* file;

char row[MAXLINE];

if ((file = fopen(filename, "rb")) == NULL) //ошибка открытия файла

{

cout << "Ошибка открытия файла" << endl;

exit(2);

}

cout << "Считываем данные из файла..." << endl;

while (!feof(file) && fgets(row, MAXLINE, file))

{

row[strlen(row) - 1] = '\0'; //заносили с \n, нужно его убрать.

push(row);

}

fclose(file);

cout << "Данные успешно считаны!" << endl;

}

void show\_menu()

{

cout << "\tМеню\t" << endl;

cout << "0. Выход из меню;" << endl;

cout << "1. Создание очереди;" << endl;

cout << "2. Вывод ключей объектов очереди;" << endl;

cout << "3. Добавление K объектов перед объектом с позицией N;" << endl;

cout << "4. Удаление объекта с заданным ключом;" << endl;

cout << "5. Удаление очереди;" << endl;

cout << "6. Сохранить данные очереди в файл;" << endl;

cout << "7. Считать данные из файла в очередь;" << endl;

}

void menu()

{

int n, N, K;

int size;

char str[MAXLINE];

char filename[] = "file.txt";

do

{

system("pause");

system("cls");

show\_menu();

cin >> n;

cout << "---" << endl;

switch (n)

{

case 0: {

cout << "Выход из меню!" << endl;

break;

}

case 1: {

cout << "Введите кол-во элементов: ";

cin >> size;

create\_queue(size);

break; }

case 2: {

show\_queue();

break; }

case 3: {

cout << "Введите кол-во объетов K и позицию N: " << endl;

cin >> K >> N;

push\_before(N, K);

break;

}

case 4: {

cout << "Введите ключ объекта, который хотите удалить: " << endl;

cin.get();

cin.getline(str, MAXLINE);

remove\_obj(str);

break;

}

case 5: {remove\_queue(); break; }

case 6: {toFile(filename); break; }

case 7: {fromFile(filename); break; }

default: { cout << "Такой команды нет!" << endl; break; }

}

} while (n != 0);

}

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

return 0;

}