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

Будин Д.В.

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.

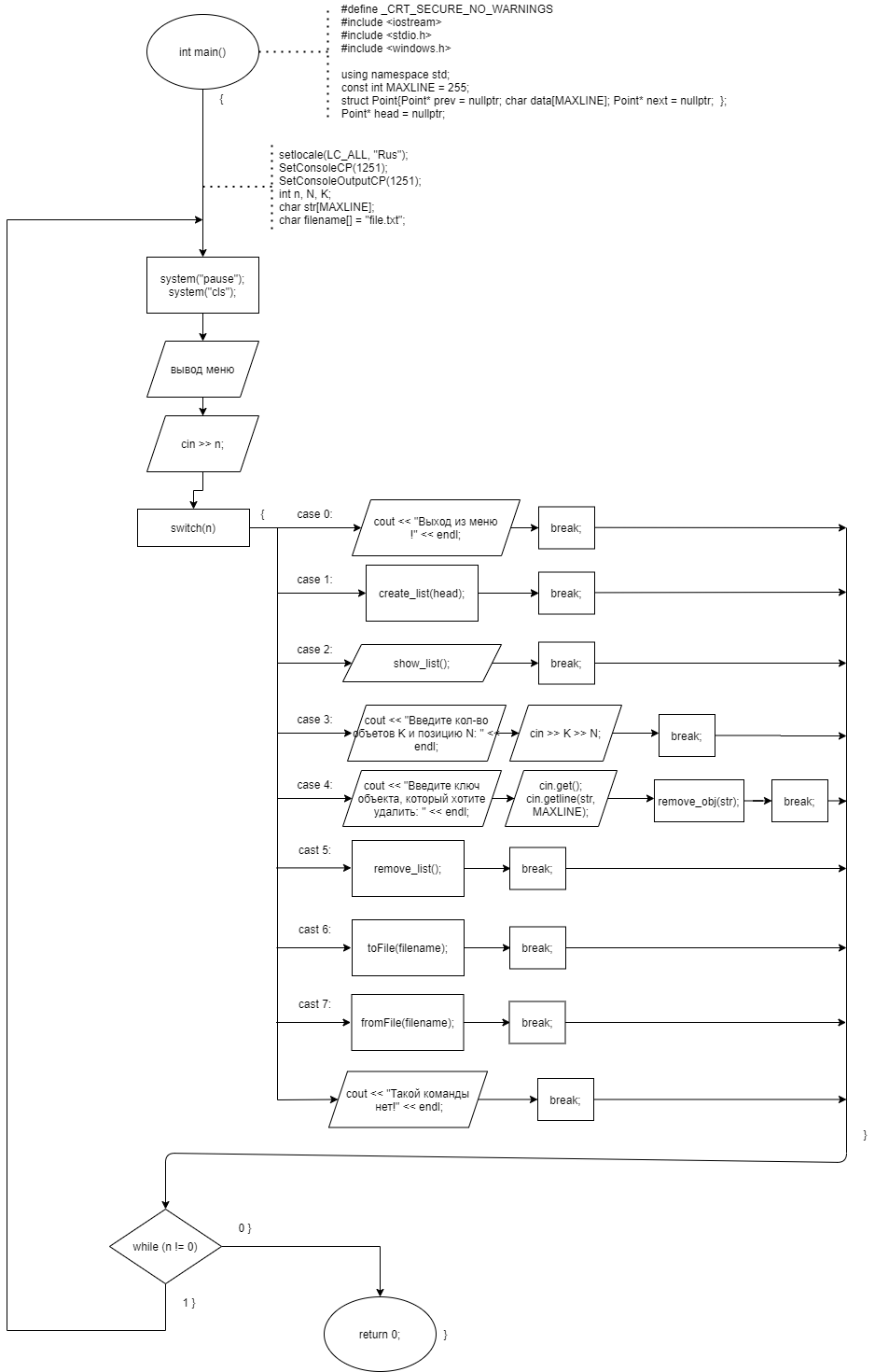
**Для каждого вариант разработать следующие функции:**

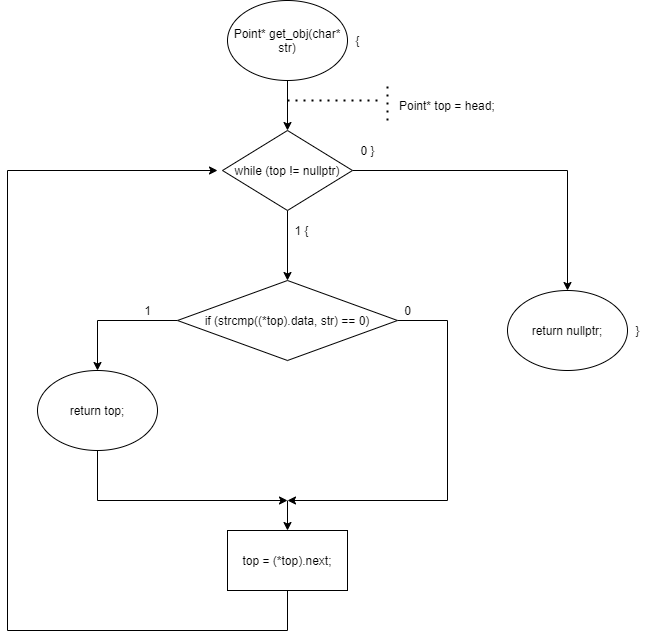
1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

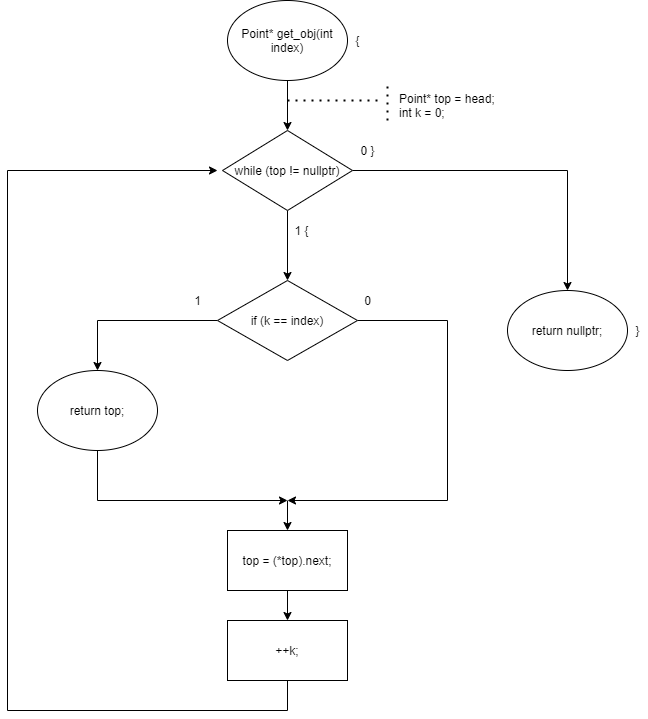
**25 ВАРИАНТ.**

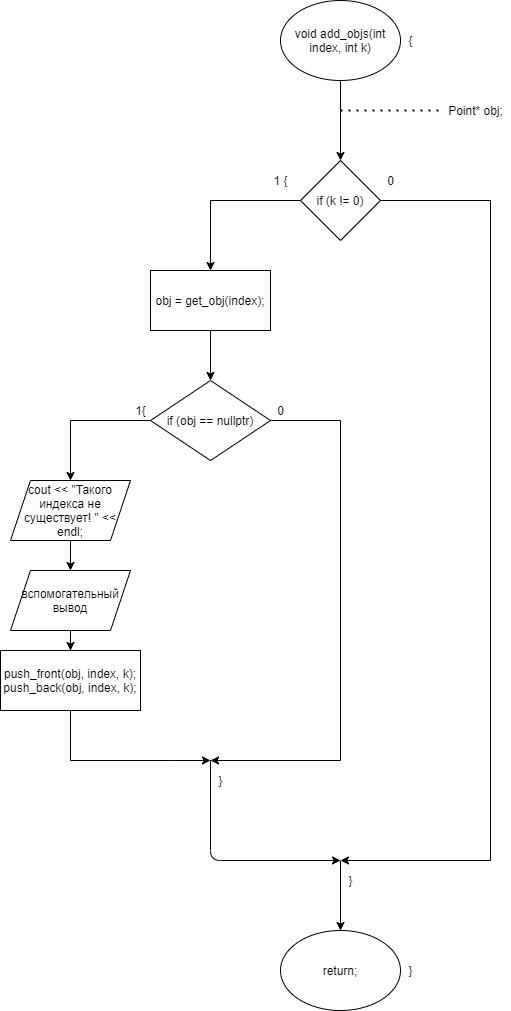
Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить по К элементов перед и после элемента с заданным номером.

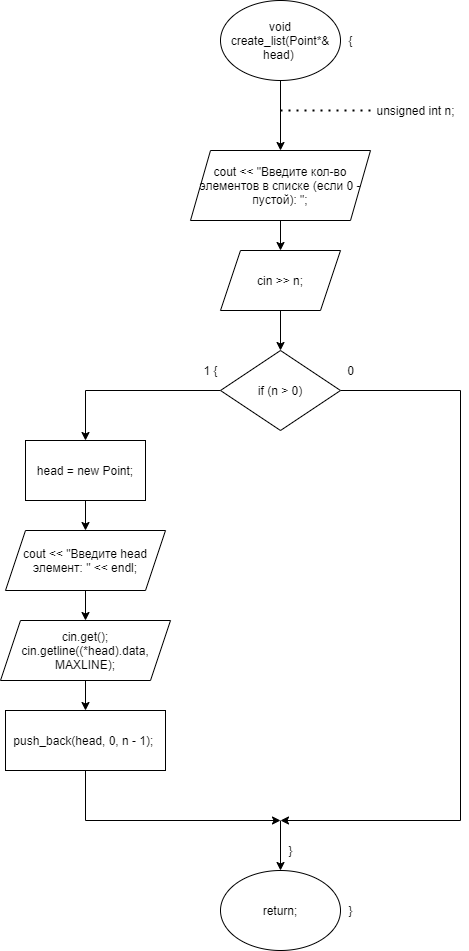
**Алгоритм программы**

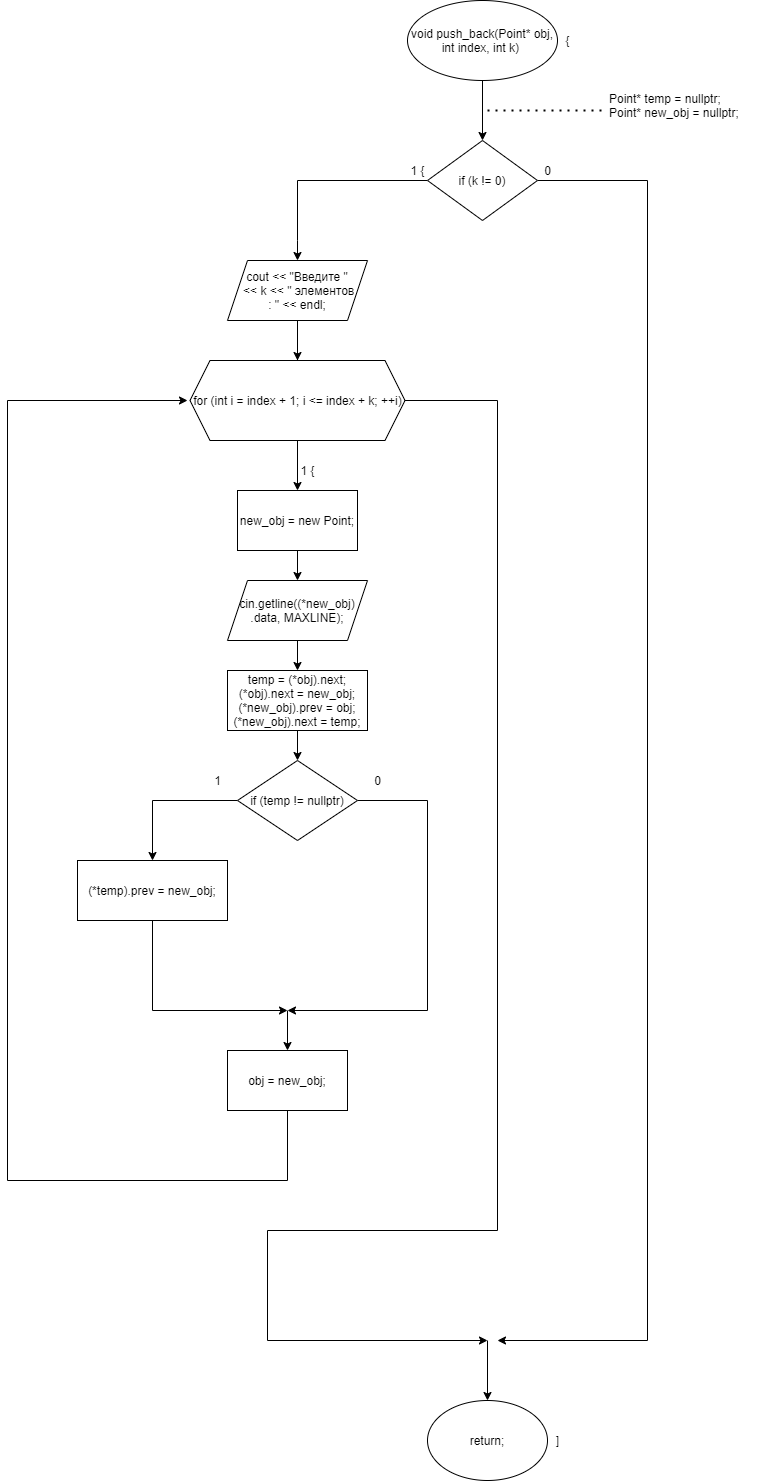


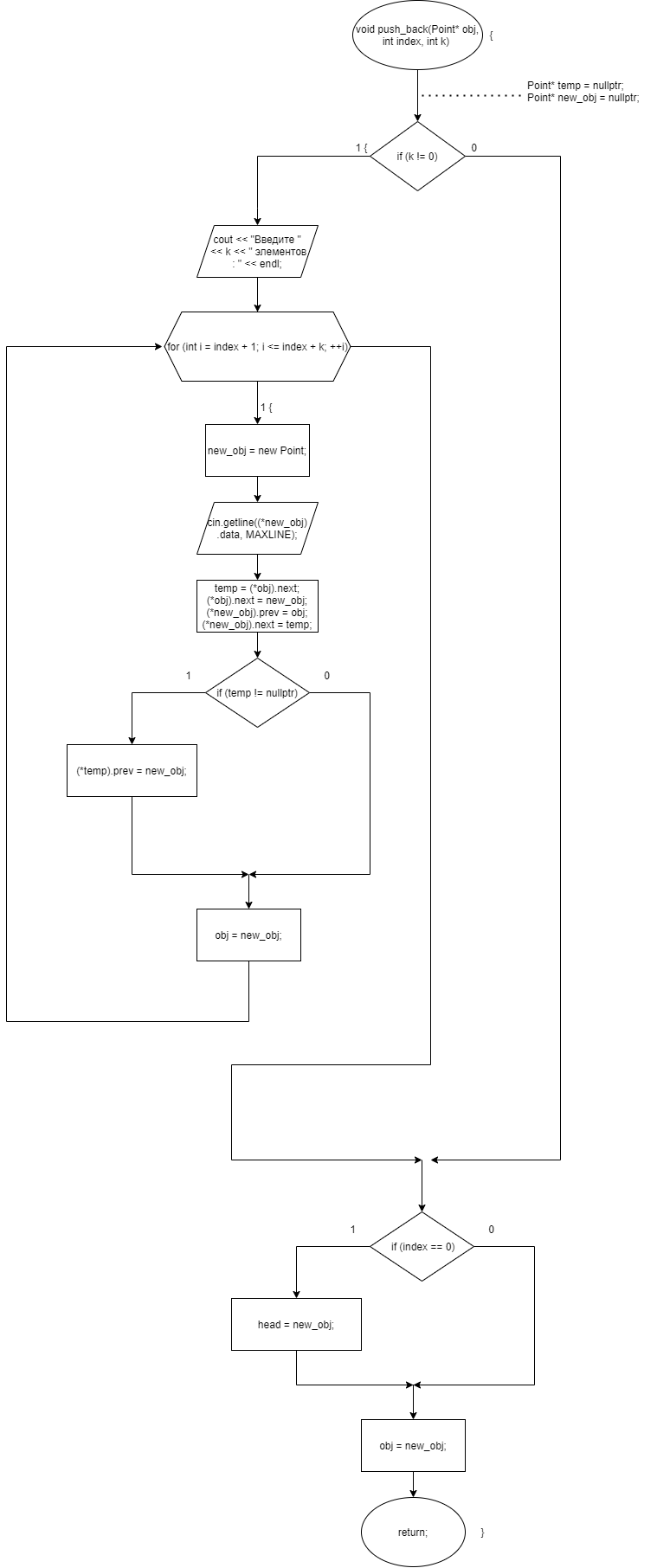


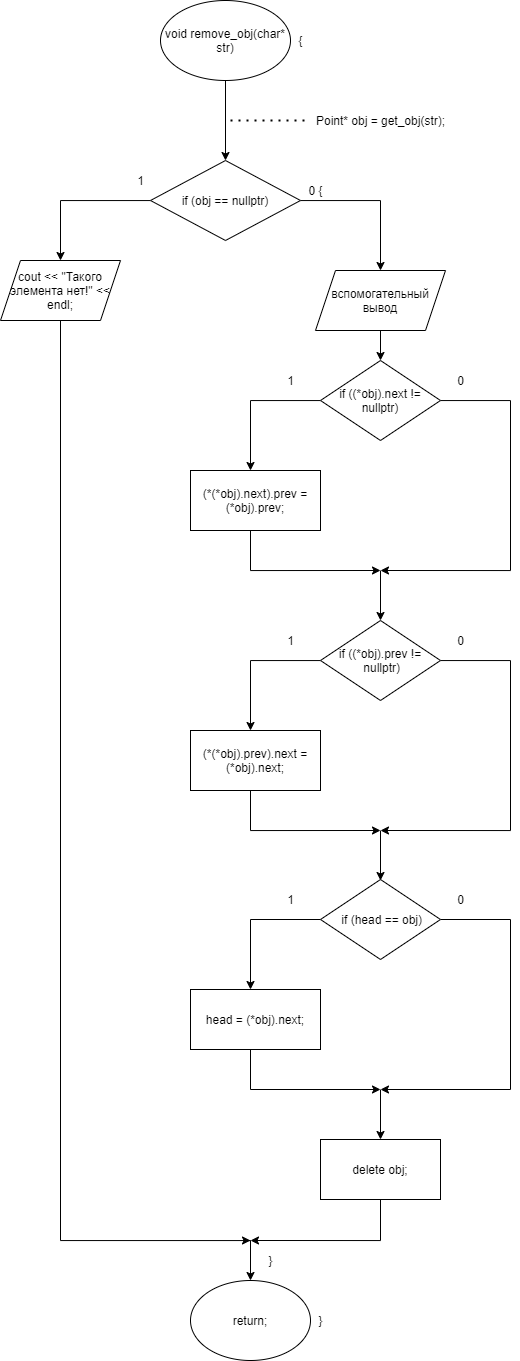


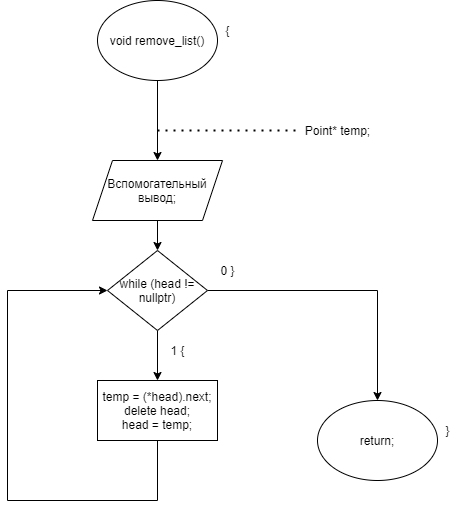


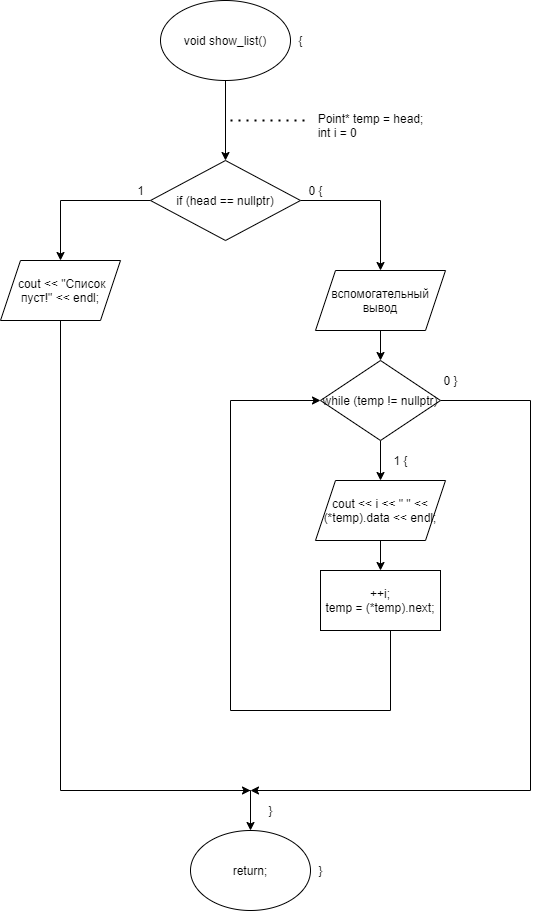


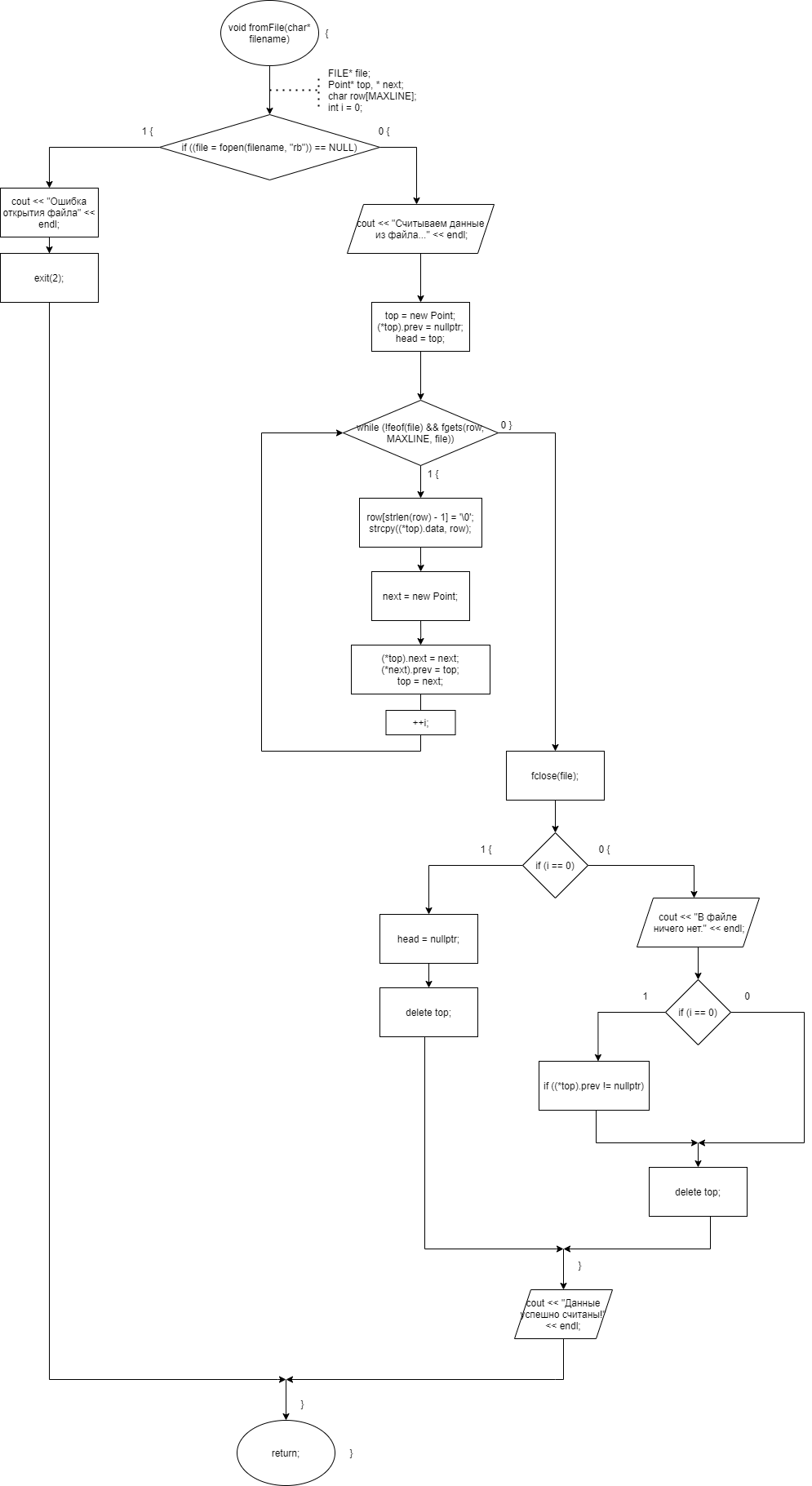


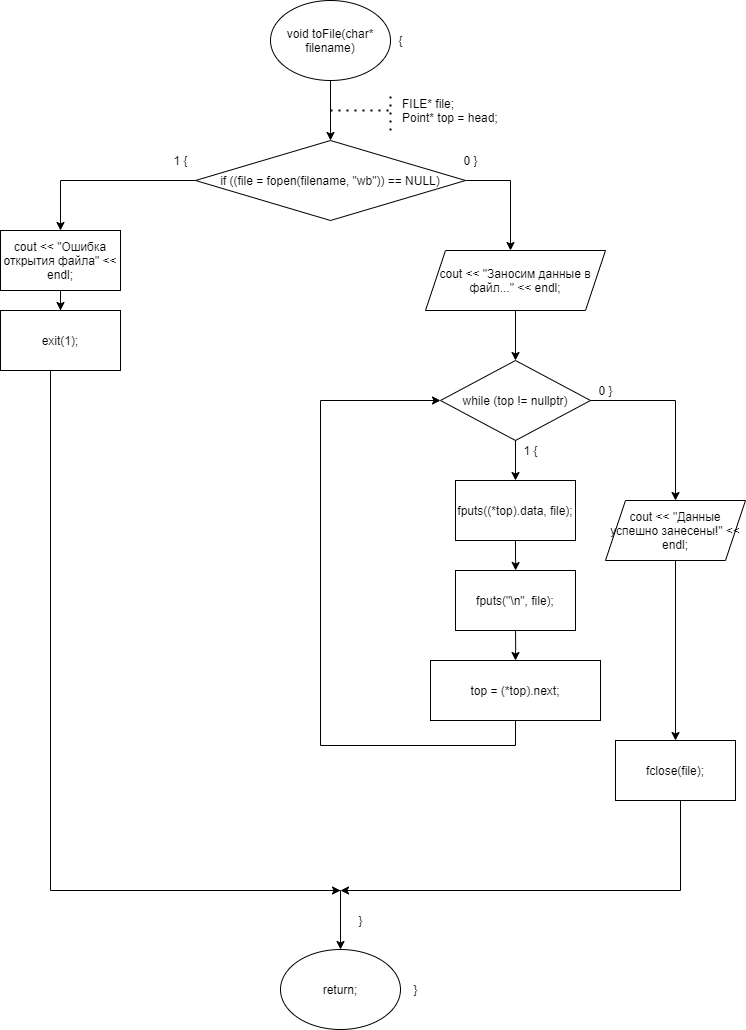


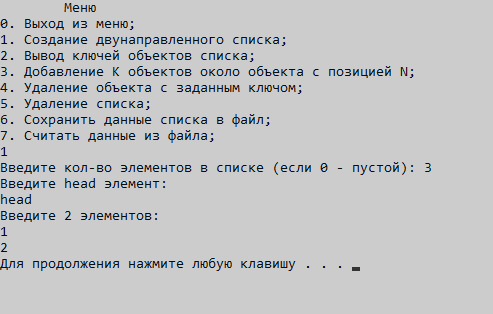


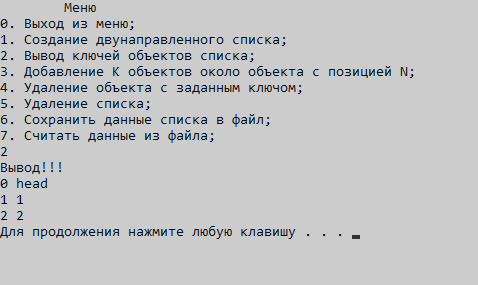
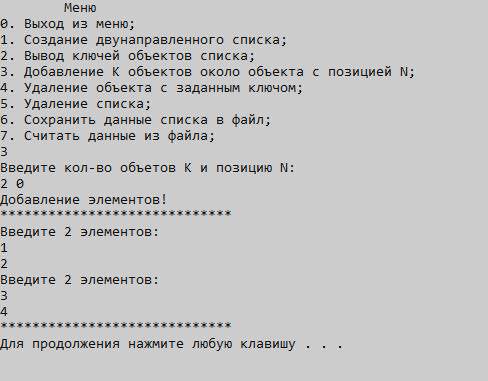
****

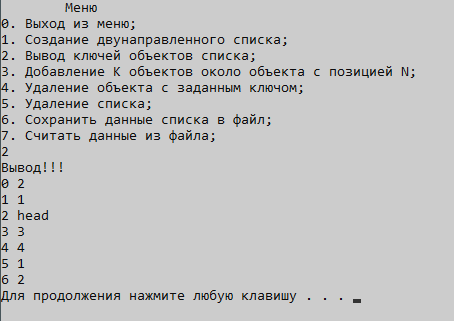
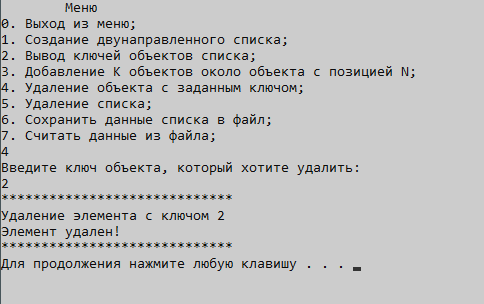
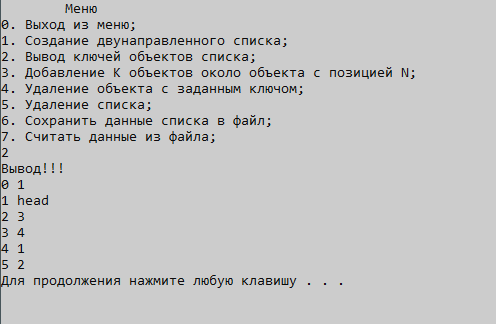
****

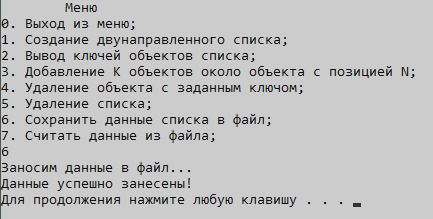
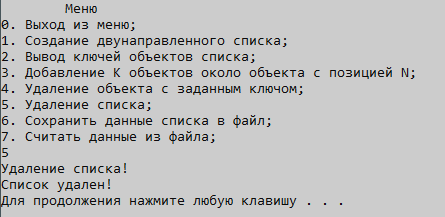
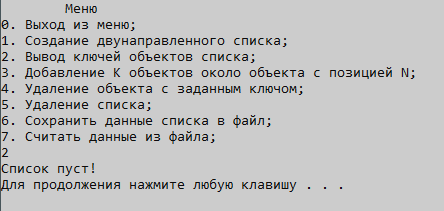
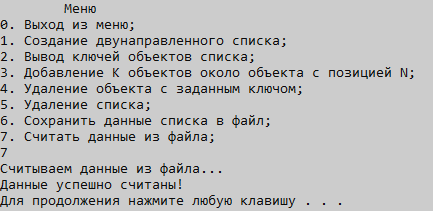
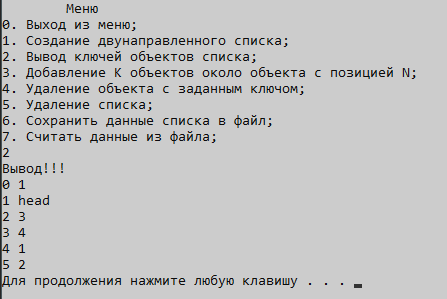
****

****

**Скриншоты работы программы**







**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

{

Point\* prev = nullptr; //предыдущий элемент списка

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

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

};

Point\* head = nullptr; //указатель на первый обьект списка

void add\_objs(int index, int k);

void push\_back(Point\* obj, int index, int k);

void push\_front(Point\* obj, int index, int k);

Point\* get\_obj(int index) //получение обьекта списка по индексу

{

Point\* top = head;

int k = 0;

while (top != nullptr)

{

if (k == index)

return top;

top = (\*top).next;

++k;

}

return nullptr;

}

Point\* get\_obj(char\* str) //получение обьекта списка по ключу

{

Point\* top = head;

while (top != nullptr)

{

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

return top;

top = (\*top).next;

}

return nullptr;

}

void create\_list(Point\*& head) //создание списка

{

unsigned int n; //кол-во элементов в списке

cout << "Введите кол-во элементов в списке (если 0 - пустой): ";

cin >> n;

if (n > 0)

{

head = new Point;

cout << "Введите head элемент: " << endl;

cin.get();

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

push\_back(head, 0, n - 1); //n-1, т.к. создали указатель на первый.

}

}

void add\_objs(int index, int k) //добавление элементов

{

if (k == 0) return;

Point\* obj;

obj = get\_obj(index);

if (obj == nullptr) { cout << "Такого индекса не существует! " << endl; return; }

cout << "Добавление элементов!" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

push\_front(obj, index, k);

push\_back(obj, index, k);

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void push\_back(Point\* obj, int index, int k) //добавление в конец списка

{

if (k == 0) return;

Point\* temp = nullptr;

Point\* new\_obj = nullptr;

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

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

{

new\_obj = new Point;

cin.getline((\*new\_obj).data, MAXLINE);

temp = (\*obj).next;

(\*obj).next = new\_obj;

(\*new\_obj).prev = obj;

(\*new\_obj).next = temp;

if (temp != nullptr)

{

(\*temp).prev = new\_obj;

}

obj = new\_obj;

}

}

void push\_front(Point\* obj, int index, int k) //добавление в начало списка

{

if (k == 0) return;

Point\* temp = nullptr;

Point\* new\_obj = nullptr;

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

cin.get();

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

{

new\_obj = new Point;

cin.getline((\*new\_obj).data, MAXLINE);

temp = (\*obj).prev;

(\*obj).prev = new\_obj;

(\*new\_obj).next = obj;

(\*new\_obj).prev = temp;

if (temp != nullptr)

{

(\*temp).next = new\_obj;

}

obj = new\_obj;

}

if (index == 0) head = new\_obj;

}

void remove\_obj(char\* str) //удаление обьекта списка по ключу

{

Point\* obj = get\_obj(str);

if (obj == nullptr) { cout << "Такого элемента нет!" << endl; return; }

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "Удаление элемента с ключом " << str << endl;

if ((\*obj).next != nullptr) (\*(\*obj).next).prev = (\*obj).prev;

if ((\*obj).prev != nullptr) (\*(\*obj).prev).next = (\*obj).next;

if (head == obj) head = (\*obj).next;

delete obj;

cout << "Элемент удален!" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

}

void show\_list() //показать список

{

if (head == nullptr) { cout << "Список пуст!" << endl; return; }

else cout << "Вывод!!! " << endl;

Point\* temp = head;

int i = 0;

while (temp != nullptr)

{

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

++i;

temp = (\*temp).next;

}

}

void remove\_list() //удаление списка

{

Point\* temp;

cout << "Удаление списка!" << endl;

while (head != nullptr)

{

temp = (\*head).next;

delete head;

head = temp;

}

cout << "Список удален!" << endl;

}

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

{

FILE\* file;

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

{

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

exit(2);

}

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

Point\* top = head;

while (top != nullptr)

{

fputs((\*top).data, file);

fputs("\n", file);

top = (\*top).next;

}

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;

Point\* top, \* next;

top = new Point;

(\*top).prev = nullptr;

head = top;

int i = 0;

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

{

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

strcpy((\*top).data, row);

next = new Point;

(\*top).next = next;

(\*next).prev = top;

top = next;

++i;

}

fclose(file);

if (i == 0) { head = nullptr; delete top; }

else {

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

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

(\*(\*top).prev).next = nullptr;

delete top;

}

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

}

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int n, N, K;

char str[MAXLINE];

char filename[] = "file.txt";

do

{

system("pause");

system("cls");

cout << "\tМеню\t" << endl;

cout << "0. Выход из меню;" << endl;

cout << "1. Создание двунаправленного списка;" << endl;

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

cout << "3. Добавление K объектов около объекта с позицией N;" << endl;

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

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

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

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

cin >> n;

switch (n)

{

case 0: {

cout << "Выход из меню!" << endl;

break;

}

case 1: {create\_list(head); break; }

case 2: {show\_list(); break; }

case 3: {

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

cin >> K >> N;

add\_objs(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\_list(); break; }

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

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

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

}

} while (n != 0);

return 0;

}