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

Молоков А.М.

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:**

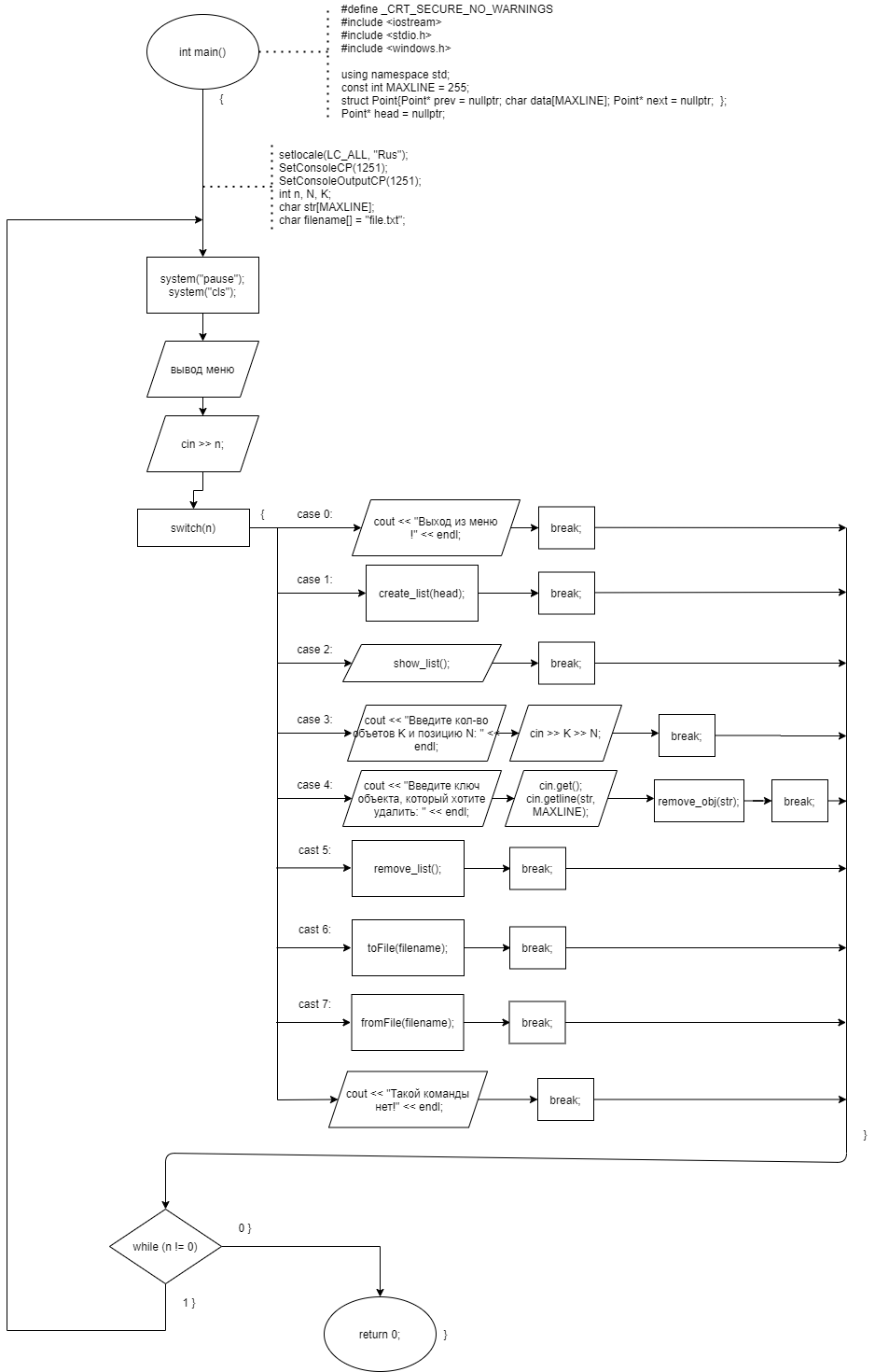
1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

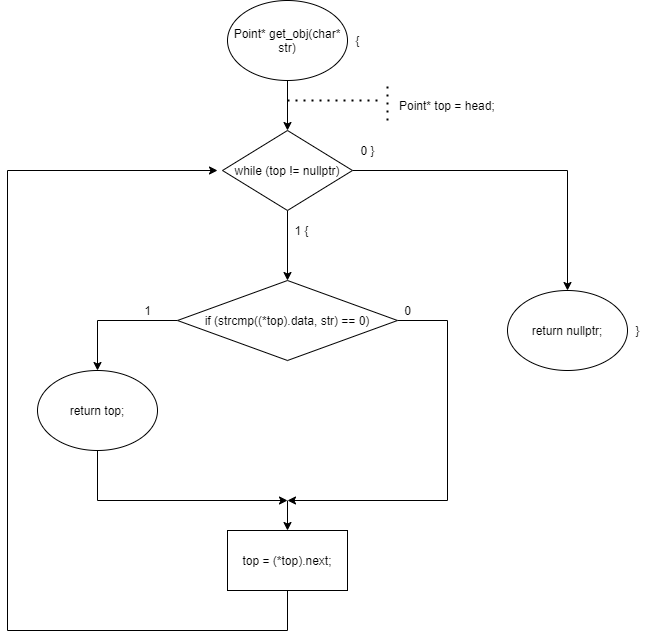
**25 ВАРИАНТ.**

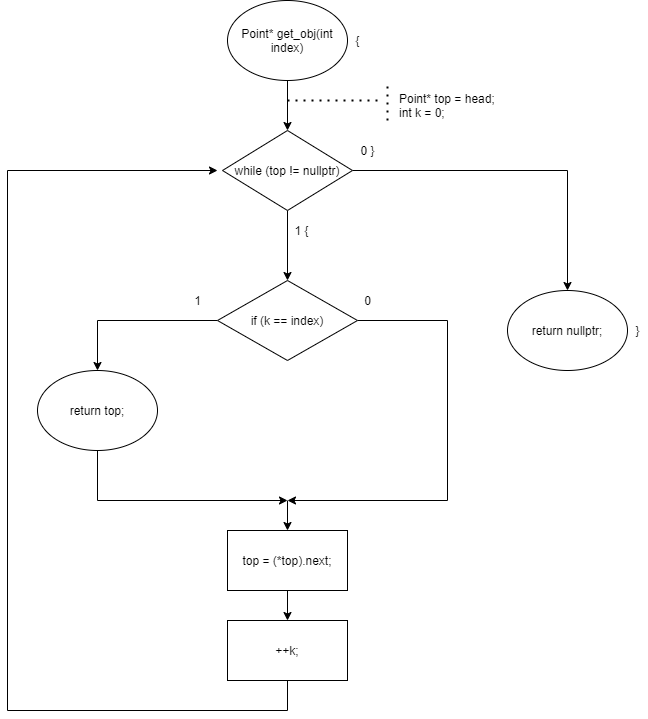
Записи в линейном списке содержат ключевое поле типа \*char (строка символов). Сформировать двунаправленный список. Удалить элемент с заданным ключом. Добавить по К элементов перед и после элемента с заданным номером.

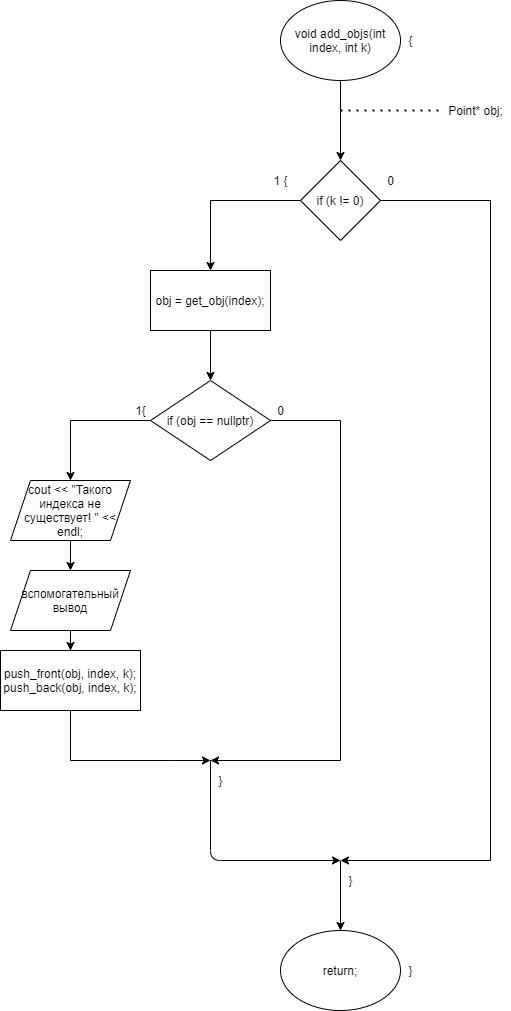
**Алгоритм программы**

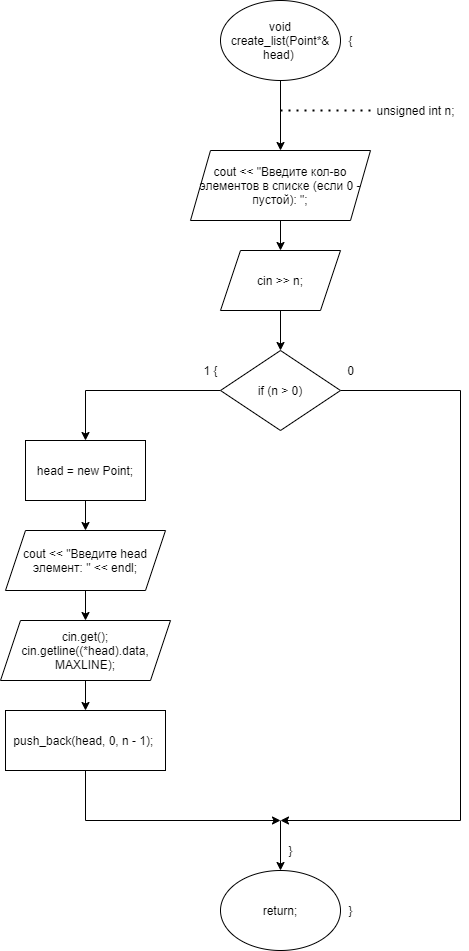
1. **Двусвязный список:**

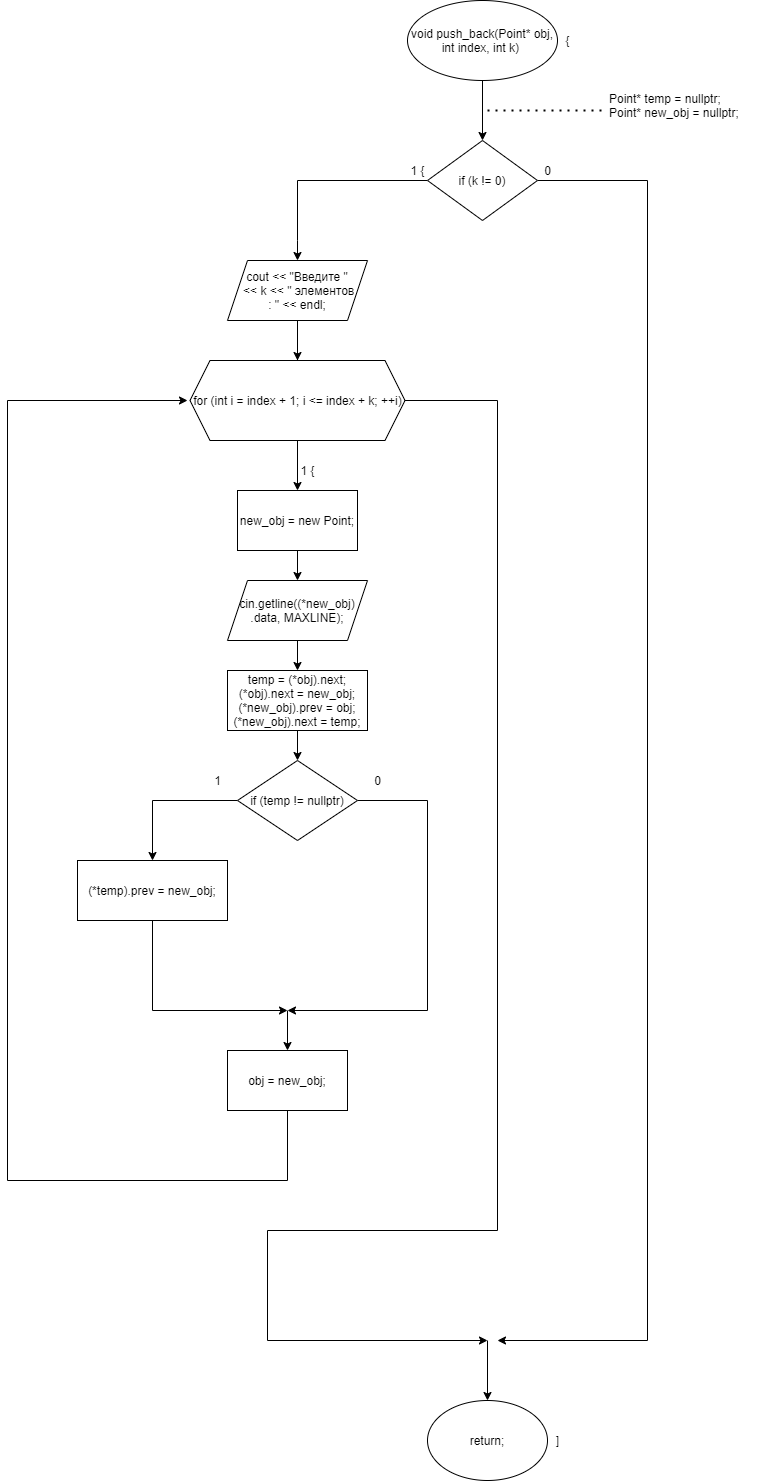


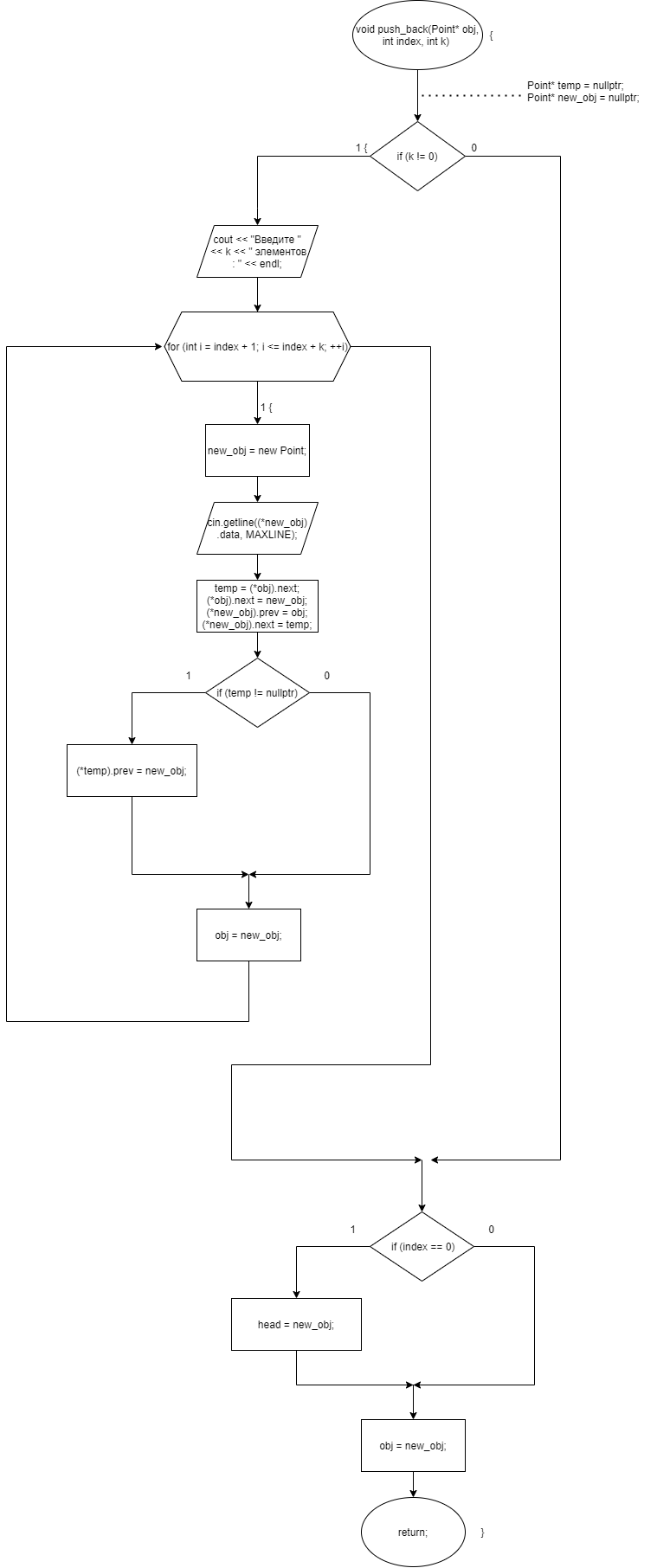


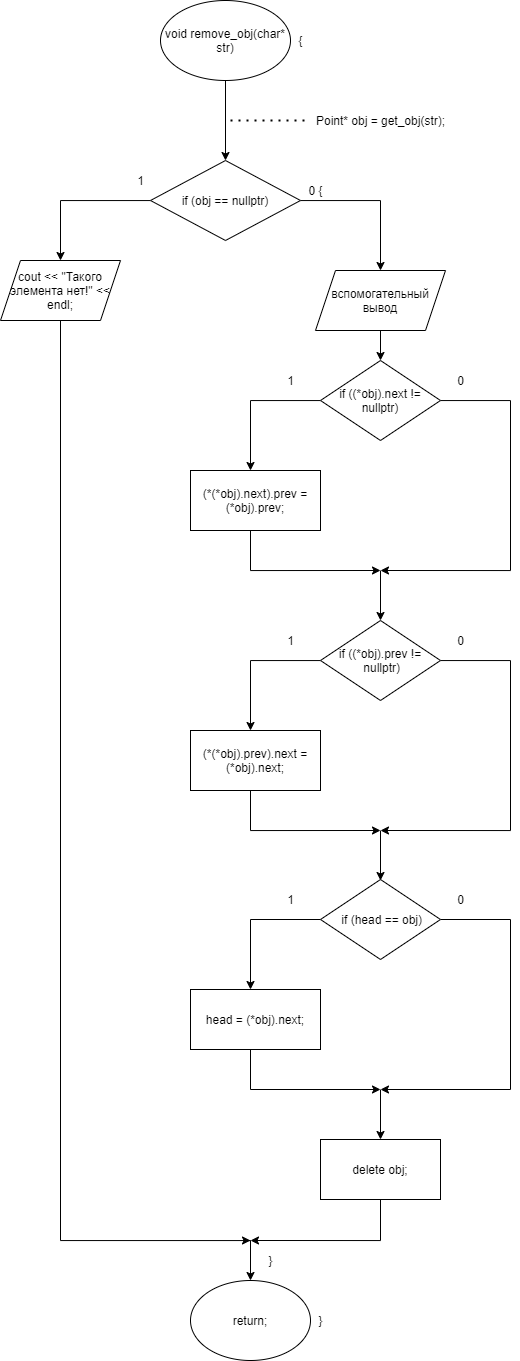


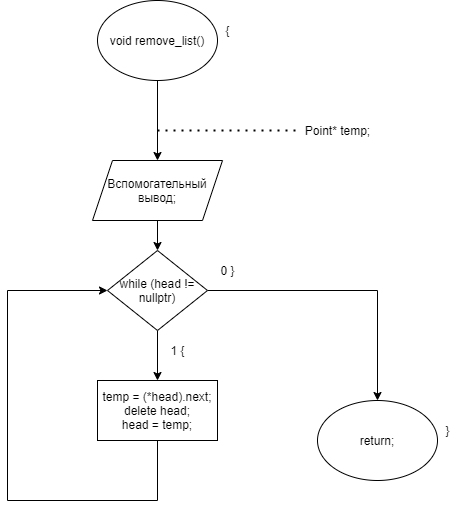


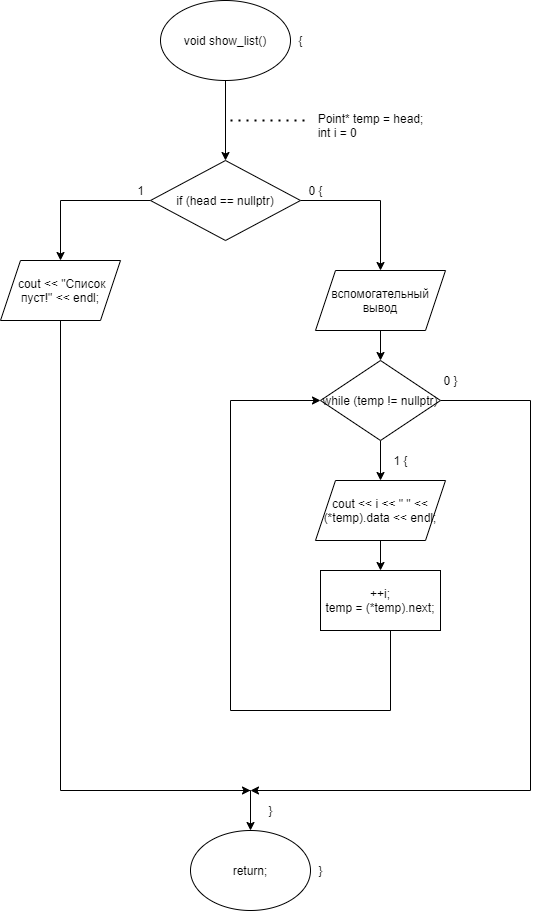


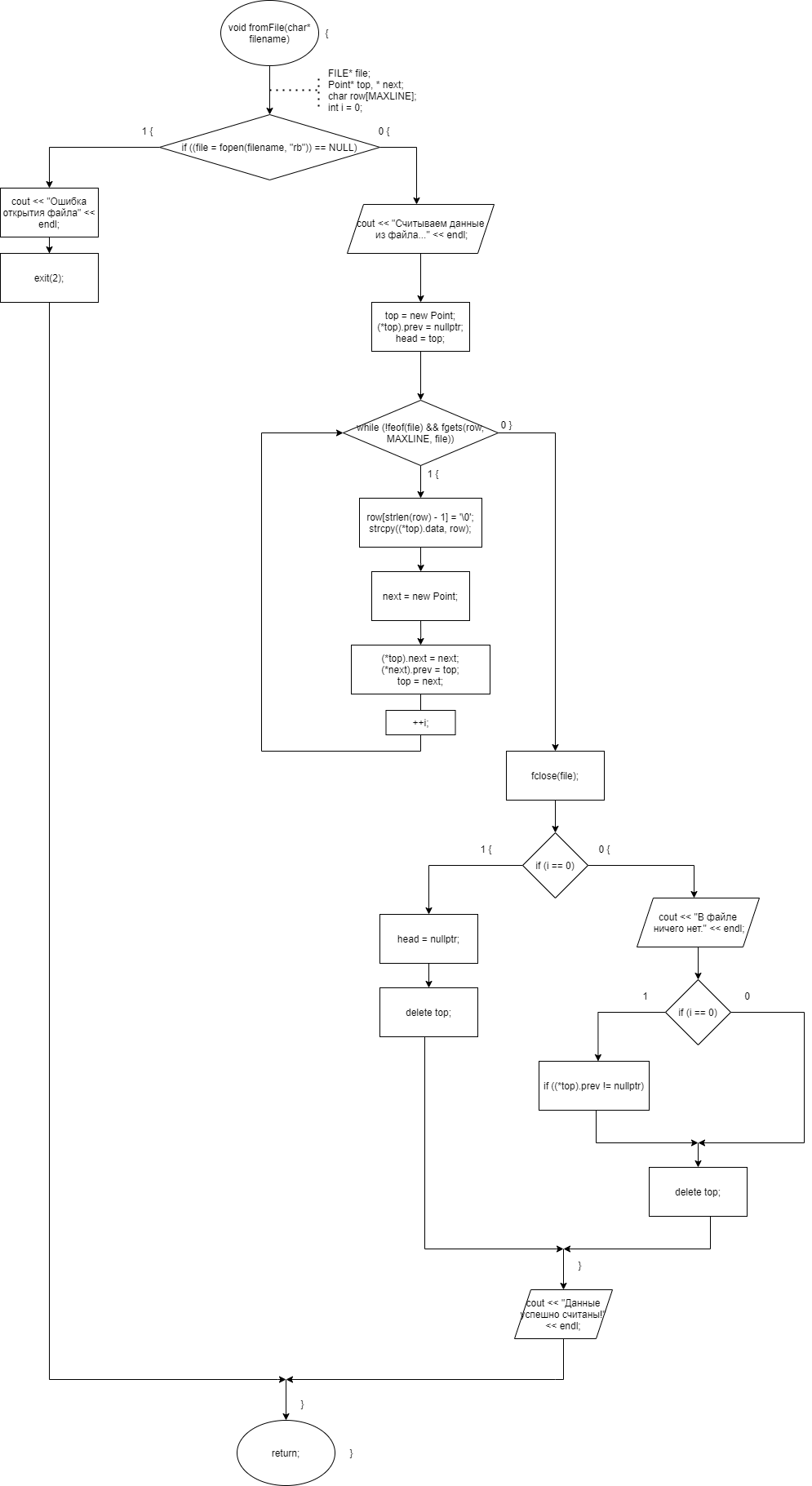


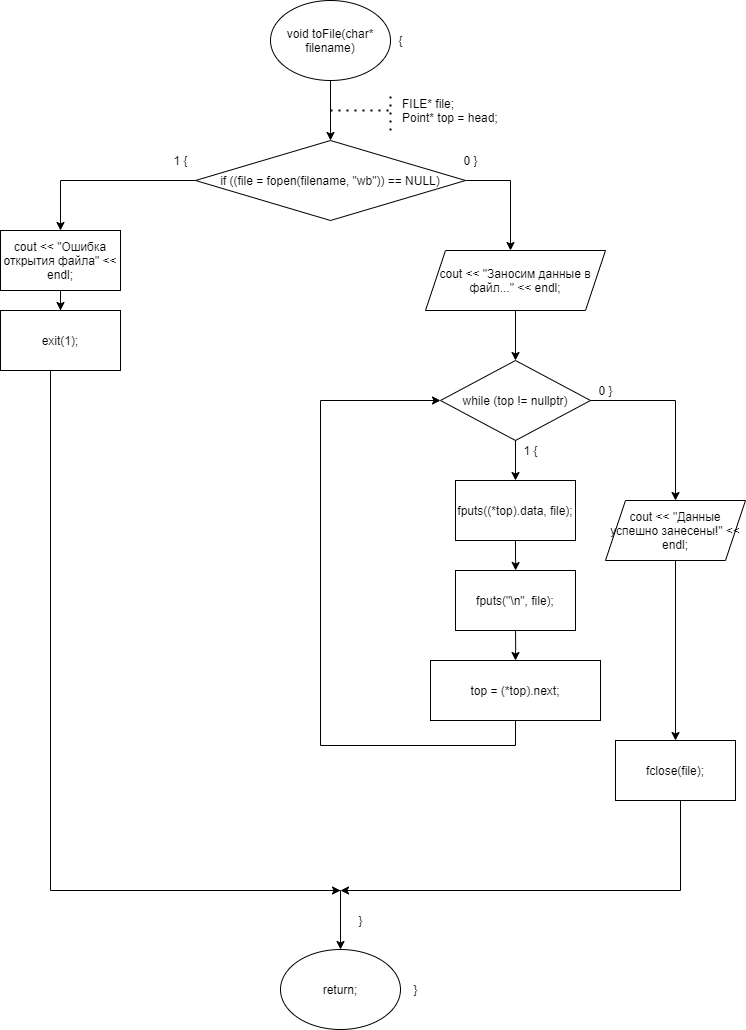




****

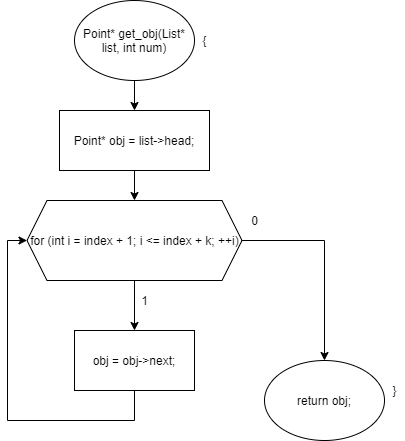
****

****

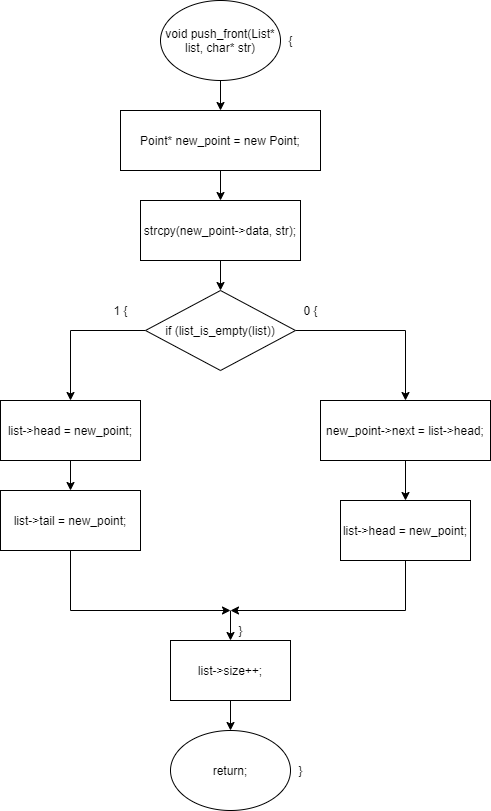
****

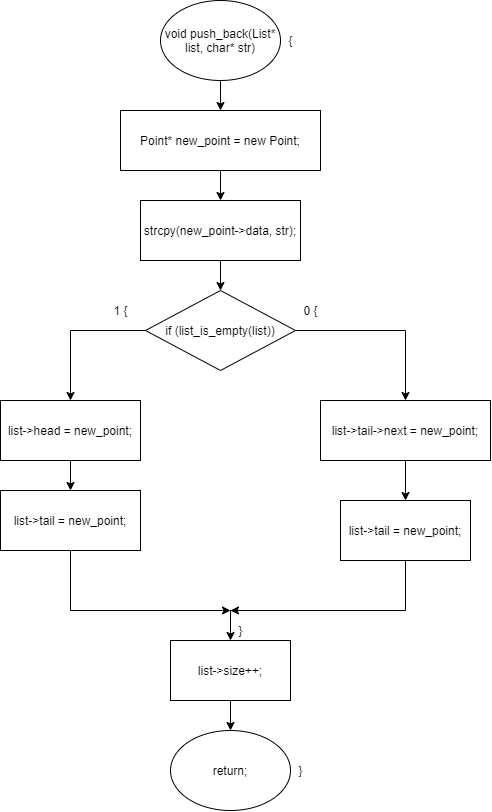
1. **Односвязный список**

**Ф-ия получения объекта по ключу**

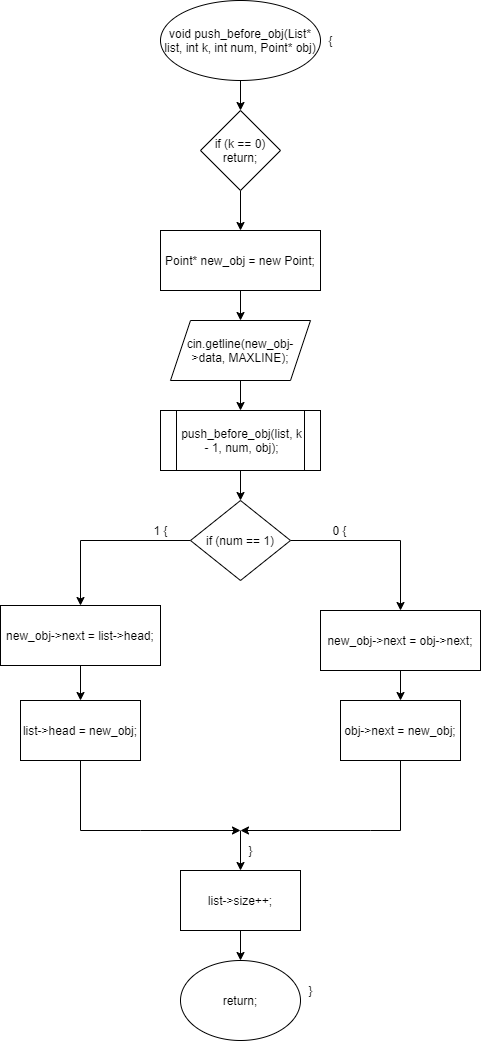
****

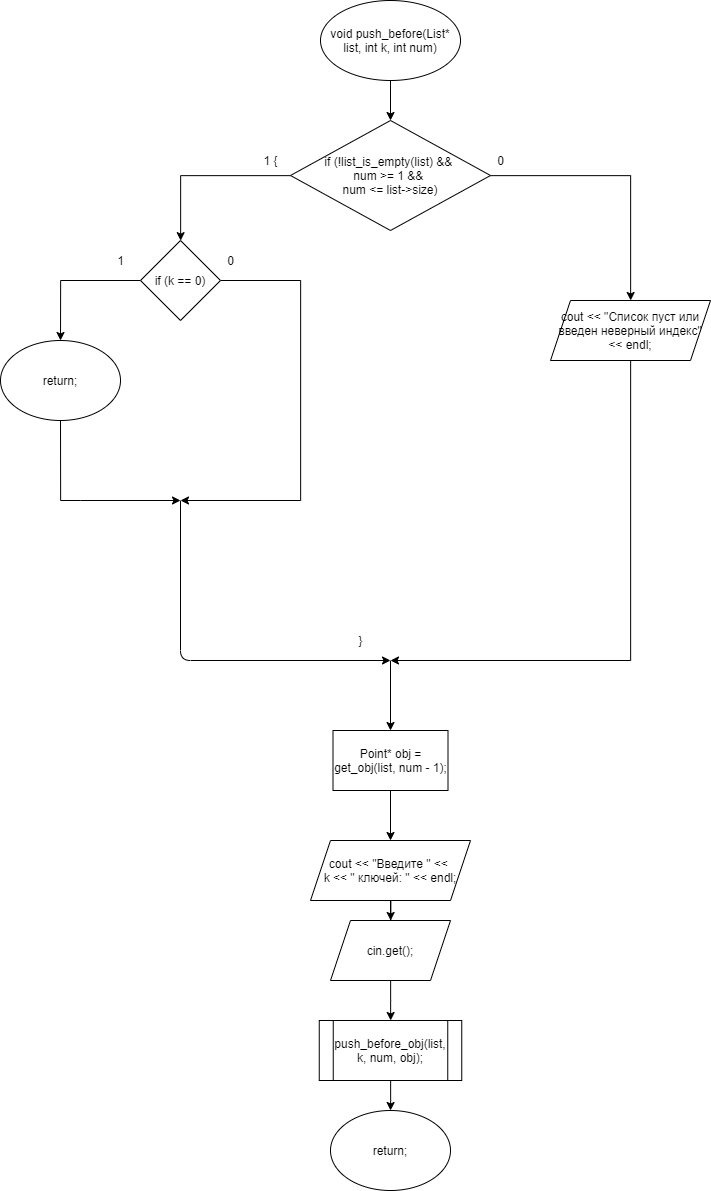
**Ф-ия добавления объекта в конец и начало списка:**

****

****

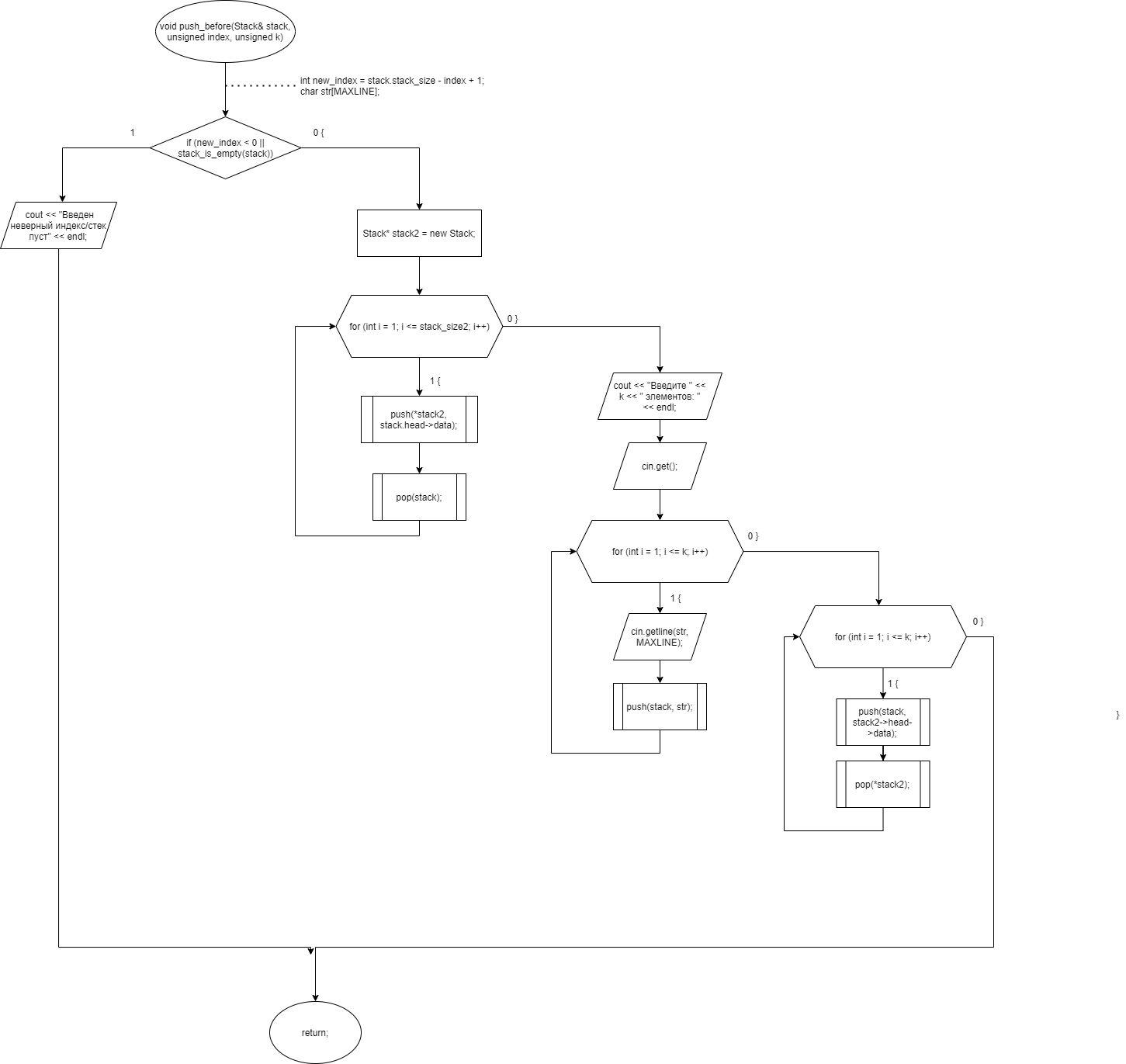
**Рекурсивная ф-ия добавления обхекта перед заданным (по позиции)**

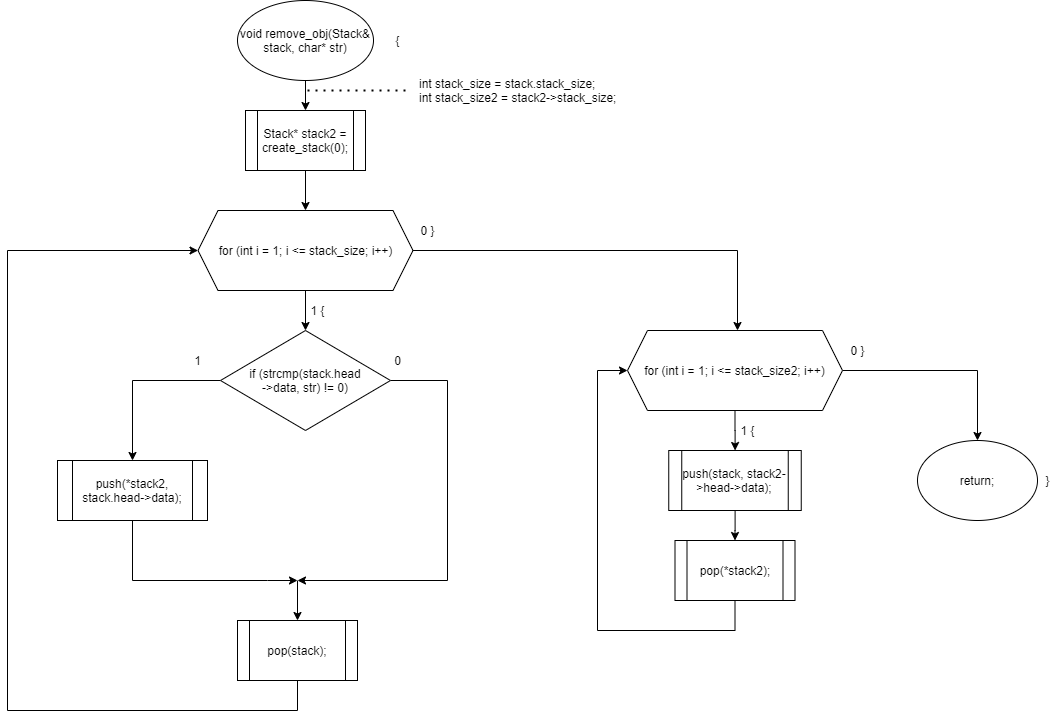
****

****

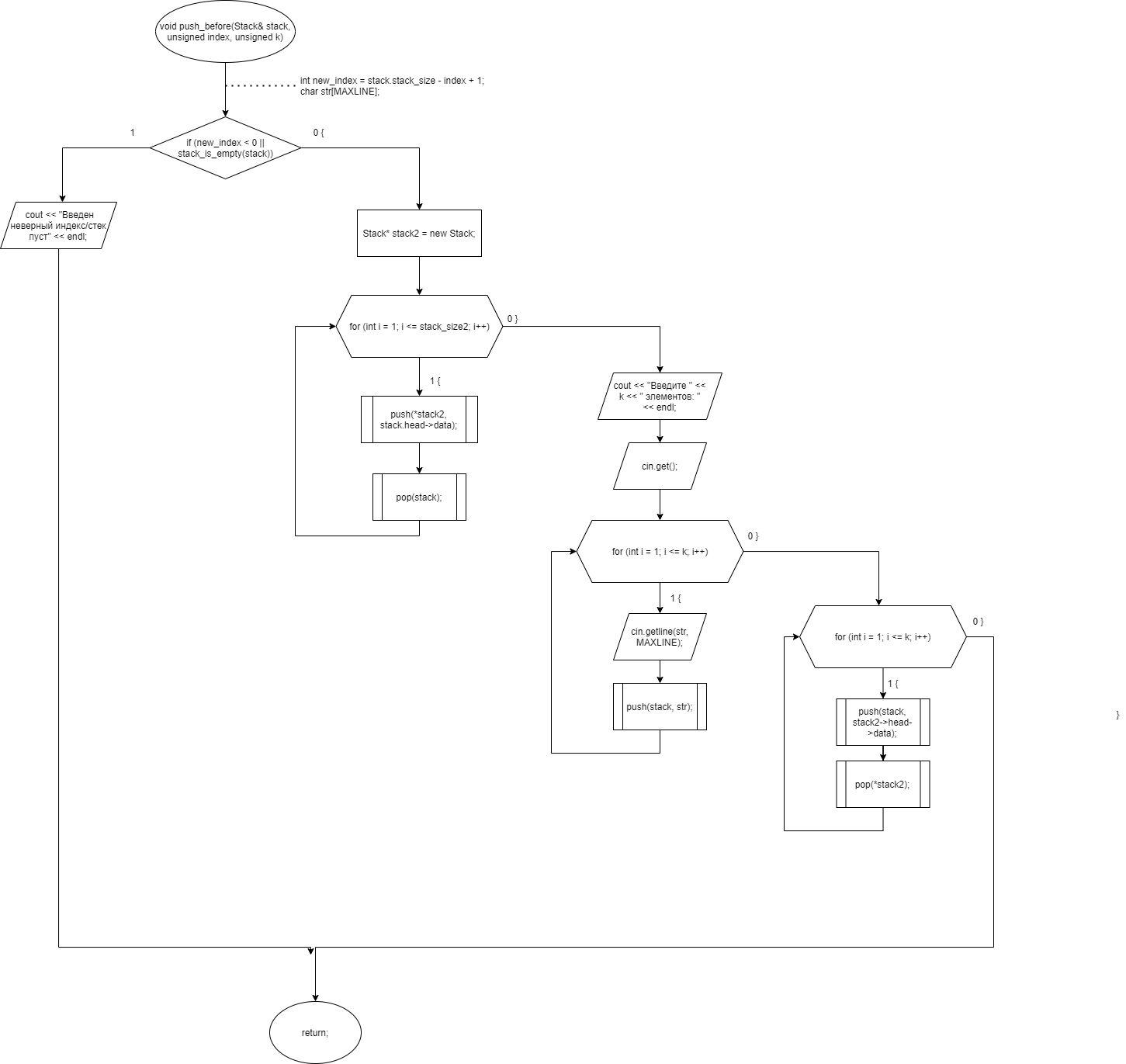
1. **Очередь**

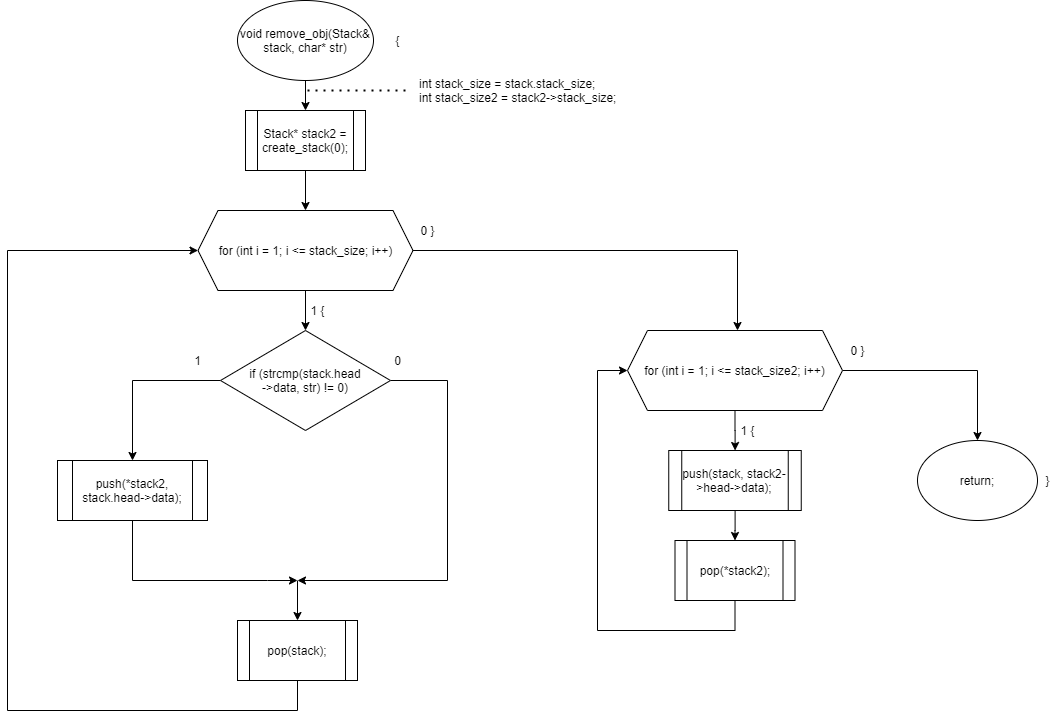
**Ф-ия push\_before**

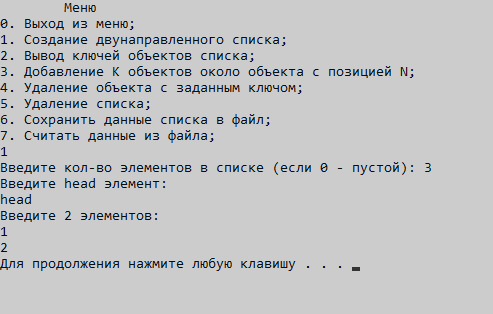
****

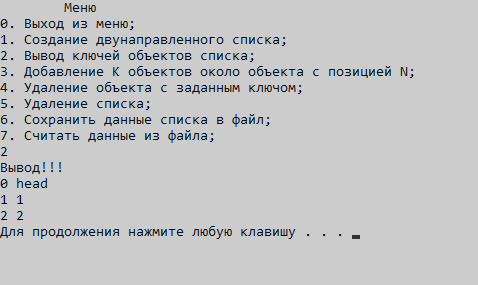
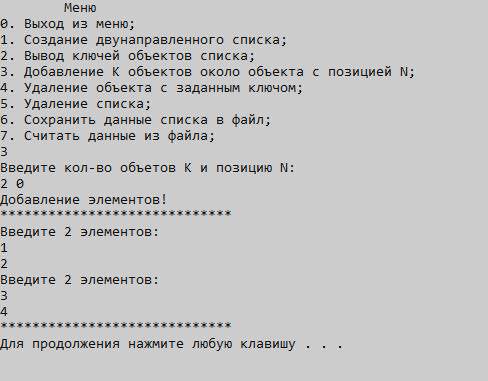
****

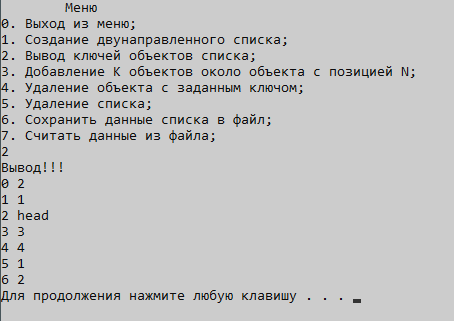
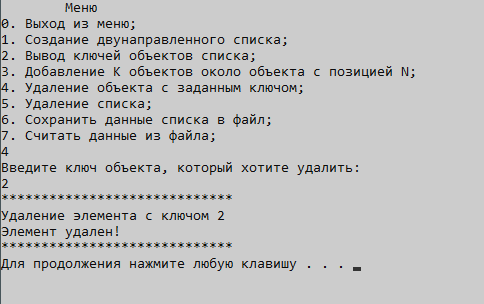
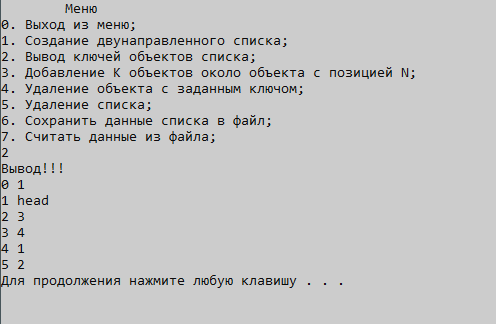
1. СТЕК

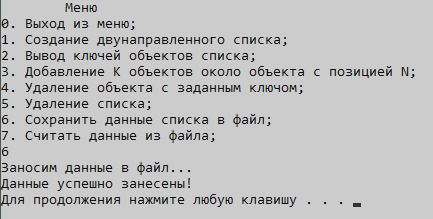
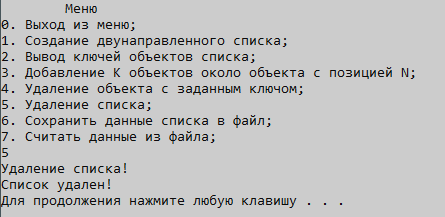
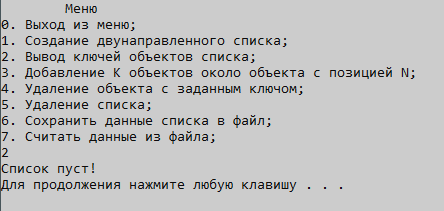
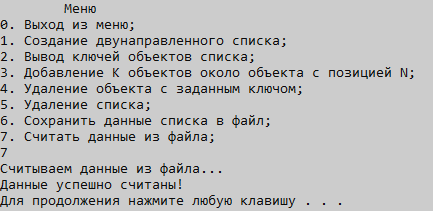
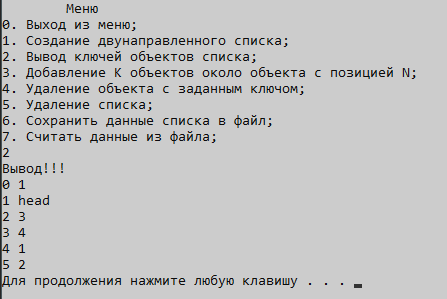


****

**Скриншоты работы программы**







**КОД ПРОГРАММЫ**

**Двусвязный список**

#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;

}

Стек

#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 стека

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

};

struct Stack

{

Point\* head = nullptr; //указатель на последний входящий 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

void show\_menu(); //показать пункты менюшки

void menu(); //сама менюшка

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[MAXLINE];

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

cin.get();

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

{

cout << i << ". ";

gets\_s(str, MAXLINE);

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[MAXLINE];

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, MAXLINE);

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[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(stack, row);

}

fclose(file);

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

}

void show\_menu()

{

cout << "\tМеню\t" << endl;

cout << "0. Выход из меню;" << endl;

cout << "1. Создание стека;" << endl;

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

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

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

cout << "5. Удаление стека;" << endl;

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

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

}

void menu()

{

Stack\* stack = new Stack;

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;

stack = create\_stack(size);

break; }

case 2: {

show\_stack(\*stack);

break; }

case 3: {

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

cin >> K >> N;

push\_before(\*stack, N, K);

break;

}

case 4: {

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

cin.get();

cin.getline(str, MAXLINE);

remove\_obj(\*stack, str);

break;

}

case 5: {remove\_stack(\*stack); break; }

case 6: {toFile(\*stack, filename); break; }

case 7: {fromFile(\*stack, 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;

}

ОЧЕРЕДЬ

#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;

}

Односвязный список

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

#include <iostream> // cout - cin

#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 списка

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

};

struct List

{

size\_t size = 0;

Point\* head = nullptr;

Point\* tail = nullptr;

};

void show\_menu(); // визуал меню

void menu(); // меню из switch

List\* create\_list(int size); // создание списка из size элементов

bool list\_is\_empty(List\* list); // проверка на пустоту списка

void push\_back(List\* list, char\* str); // добавление элемента в конец списка

void push\_front(List\* list, char\* str); // добавление элемента в начало списка

void push\_before(List\* list, int k, int num); // добавление K объектов перед заданным номером

void push\_before\_obj(List\* list, int k, int num, Point\* obj); // добавление обьекта перед заданным рекурсивно

Point\* get\_obj(List\* list, int num); // получение объекта с заданным номером

void remove\_obj(List\* list, char\* str); // удаление объекта с заданным ключом

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

{

if (obj == nullptr) return;

cout << obj->data << endl;

output(obj->next);

}

void show\_list(List\* list)

{

if (list\_is\_empty(list)) cout << "Список пуст" << endl;

else output(list->head);

}

bool list\_is\_empty(List\* list) //проверка на пустой список

{

return list == nullptr || list->size == 0;

}

List\* create\_list(int size)

{

List\* list = new List;

char str[MAXLINE];

if (size > 0)

{

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

cin.get();

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

{

cout << i << ". ";

gets\_s(str, MAXLINE);

push\_back(list, str);

}

}

return list;

}

void push\_back(List\* list, char\* str) //добавить новый элемент сверху

{

Point\* new\_point = new Point;

strcpy(new\_point->data, str);

if (list\_is\_empty(list))

{

list->head = new\_point;

list->tail = new\_point;

} else {

list->tail->next = new\_point;

list->tail = new\_point;

}

list->size++;

}

void push\_front(List\* list, char\* str)

{

Point\* new\_point = new Point;

strcpy(new\_point->data, str);

if (list\_is\_empty(list))

{

list->head = new\_point;

list->tail = new\_point;

}

else

{

new\_point->next = list->head;

list->head = new\_point;

}

list->size++;

}

Point\* get\_obj(List\* list, int num)

{

Point\* obj = list->head;

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

{

obj = obj->next;

}

return obj;

}

void remove\_obj(List\* list, char\* str)

{

if (list\_is\_empty(list)) cout << "Список пуст, нечего удалять!" << endl;

else

{

Point\* temp = list->head;

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

{

delete list->head;

list->head = nullptr;

list->tail = nullptr;

list->size--;

}

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

{

temp = list->head;

list->head = list->head->next;

delete temp;

list->size--;

}

else

{

while (temp->next != nullptr)

{

if (strcmp((temp->next)->data, str) == 0)

{

if (temp->next == list->tail)

{

delete list->tail;

list->tail = temp;

list->tail->next = nullptr;

}

else

{

Point\* temp2 = (temp->next)->next;

delete temp->next;

temp->next = temp2;

}

list->size--;

return;

}

temp = temp->next;

}

cout << "Элемент не найден!" << endl;

}

}

}

void remove\_list(List\* list)

{

Point\* temp;

while (list->head != nullptr)

{

temp = list->head->next;

delete list->head;

list->head = temp;

}

list->size = 0;

list->tail = nullptr;

cout << "Список удален!" << endl;

}

void push\_before\_obj(List\* list, int k, int num, Point\* obj)

{

if (k == 0) return;

Point\* new\_obj = new Point;

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

push\_before\_obj(list, k - 1, num, obj);

if (num == 1)

{

new\_obj->next = list->head;

list->head = new\_obj;

}

else {

new\_obj->next = obj->next;

obj->next = new\_obj;

}

list->size++;

}

void push\_before(List\* list, int k, int num)

{

if (!list\_is\_empty(list) && num >= 1 && num <= list->size)

{

if (k == 0) return;

Point\* obj = get\_obj(list, num - 1);

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

cin.get();

push\_before\_obj(list, k, num, obj);

}

else cout << "Список пуст или введен неверный индекс" << 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(List\* list, 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(list->head, file);

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

fclose(file);

}

void fromFile(List\* list, 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\_back(list, row);

}

fclose(file);

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

}

void show\_menu()

{

cout << "\tМеню\t" << endl;

cout << "0. Выход из меню;" << endl;

cout << "1. Создание стека;" << endl;

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

cout << "3. Добавление объекта в начало списка;" << endl;

cout << "4. Добавление объекта в конец списка;" << endl;

cout << "5. Добавление K объектов перед объектом с позицией N;" << endl;

cout << "6. Удаление объекта с заданным ключом;" << endl;

cout << "7. Удаление списка;" << endl;

cout << "8. Сохранить данные списка в файл;" << endl;

cout << "9. Считать данные из файла в список;" << endl;

}

void menu()

{

List\* list = new List;

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;

list = create\_list(size);

break; }

case 2: {

show\_list(list);

break; }

case 3: {

cout << "Введите ключ head элемента:" << endl;

cin.get();

gets\_s(str, MAXLINE);

push\_front(list, str);

break;

}

case 4: {

cout << "Введите ключ tail элемента:" << endl;

cin.get();

gets\_s(str, MAXLINE);

push\_back(list, str);

break;

}

case 5: {

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

cin >> K >> N;

push\_before(list, K, N);

break;

}

case 6: {

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

cin.get();

cin.getline(str, MAXLINE);

remove\_obj(list, str);

break;

}

case 7: {remove\_list(list); break; }

case 8: {toFile(list, filename); break; }

case 9: {fromFile(list, 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;

}