Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«**Пермский национальный исследовательский политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Информационные динамические структуры»

Семестр 2

Выполнил работу

Студент группы ИВТ-22-2б

Устюгова Полина Дмитриевна

Проверил

Доцент кафедры ИТАС

Полякова Ольга Андреевна

Г. Пермь-2023

**Постановка задачи**

1Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

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.

Восстановление списка из файла.

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

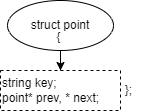
**Алгоритм решения задачи**

1. Создаётся структура struct point с полями string key (информация, хранящаяся в узле), point\* next (указатель на следующий элемент списка) и point\* prev (указатель на предыдущий элемент списка). Данная структура является структурой узла.
2. Создаётся функция point\* make\_list(). Эта функция сначала просит ввести пользователя размер списка, а затем циклично просит ввести информацию, которая будет записываться в узел списка, а после создаёт указатель на следующий и на предыдущий узлы.
3. Создаётся функция void print\_list() с параметрами point\* van (список). Через цикл она выводит значения ключа, а затем переносит указатель на следующий элемент.
4. Создаётся функция point\* del\_point() с параметрами point \* van (список) и int k (номер удаляемого элемента). С помощью цикла находится элемент, предшествующий элементу k. Затем указатель предыдущего элемента переносится с удаляемого на следующий, а указатель со следующего элемента списка переносится на предыдущий выбранному.
5. Создаётся функция point\* add\_point() с параметрами point\* van (список) и int k (номер элемента, справа и слева от которого мы будем добавлять какое-то количество элементов). Через цикл находится элемент, предшествующий выбранному, после чего пользователя просят ввести новый элемент списка столько раз, сколько это попросит сделать пользователь. Затем указатель с предыдущего элемента переносится на новый, с нового на указанный, с указанного снова на новый и с нового на следующий. Аналогично с ссылками на предыдущий элемент списка.
6. Создаётся функция int find\_key\_index() с параметрами point\* van. Здесь пользователя просят ввести ключ (информацию элемента, вокруг которого требуется ввести значения). Затем через цикл сравниваются значения каждого элемента цикла с ключом до тех пор, пока он с ним не совпадёт. Если такой элемент найден, то функция возвращает его индекс, иначе функция выводит надпись о том, что данный ключ не был найден.
7. Создаётся функция void delete\_list() с параметрами point\* van. Через цикл она считает количество элементов в списке, а затем также через цикл удаляет их.
8. Создаётся функция void PrintInFile() с параметрами point\* van и string path (указатель на файл). С помощью библиотеки fstream открывается файл по указанной ссылке, и через цикл в него записывается информация из списка. Если файл не удаётся открыть, функция выводит ошибку.
9. Создаётся функция point\* ReadFromFile() с параметрами string path (ссылка на файл). Создаётся элемент point p (первый элемент списка), а так же указатель на nullptr. Далее создаётся цикл, в котором вся информация записывается в новые узлы, а так же создаются указатели. Цикл завершается, когда заканчивается информация в файле.
10. Создаётся функция void menu() с параметрами int a (номер операции, выбранной пользователем) и point\* first (созданный изначально список).

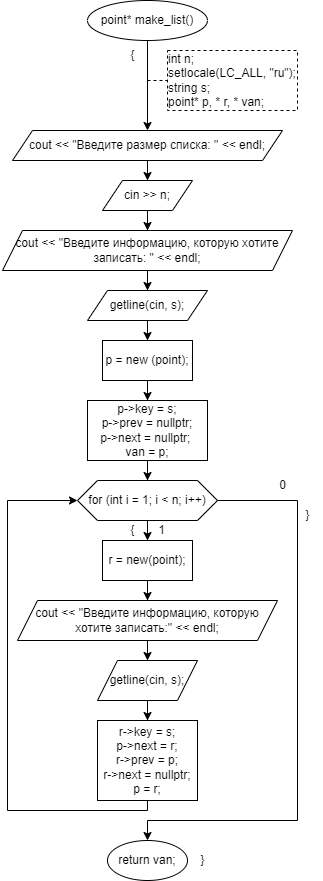
Здесь с помощью свитча выполняются команды, выбранные пользователем. Если пользователь введёт цифру 0 - программа закончит свою работу, а если он введёт какое-то постороннее число, то программа попросит ввести его номер команды заново.

1. Создаётся функция int main(). В ней создаётся список point\* first, который пользователя просят заполнить. Затем выводятся пронумерованные доступные команды, и пользователя просят выбрать нужную, после чего вызывается функция void menu(a, first).

**Блок-схема**

****

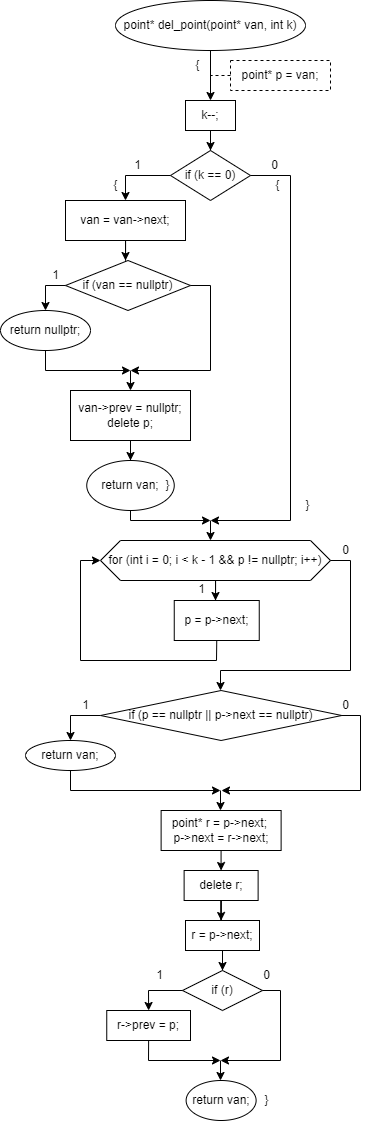
**Рис. 1 – Блок-схема структуры point**

****

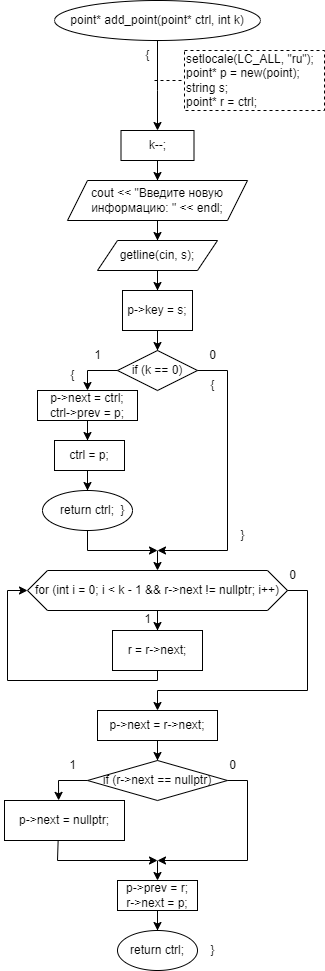
**Рис. 2 – Блок-схема функции point\* make\_list()**



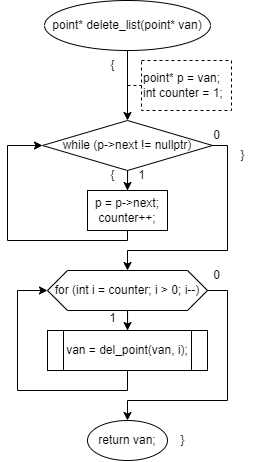
**Рис. 3 – Блок-схема функции void print\_list()**

****

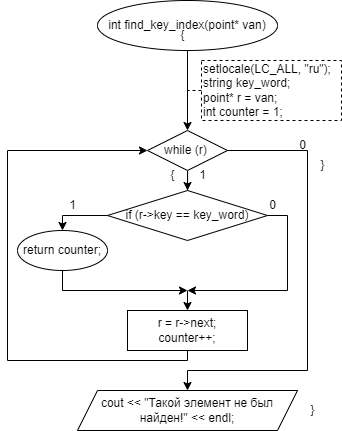
**Рис. 4 – Блок-схема функции point\* del\_point()**



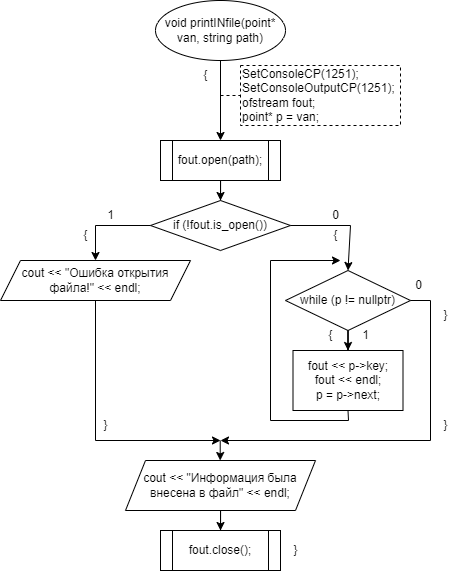
**Рис. 5 – Блок-схема функции point\* add\_point()**

****

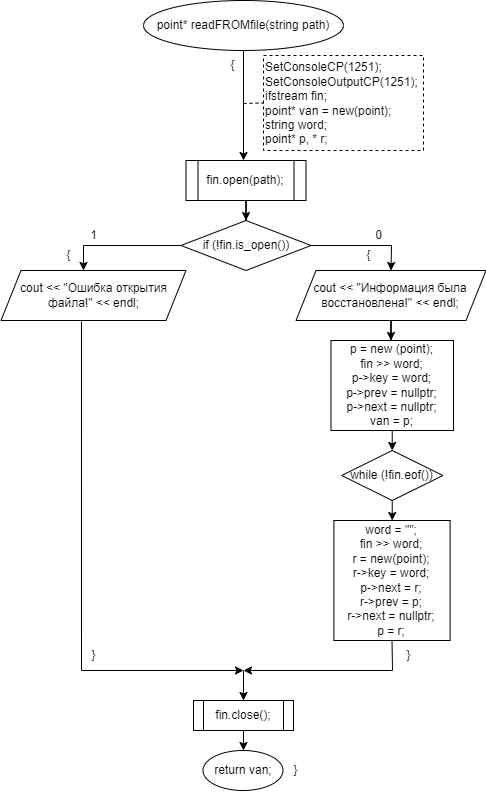
**Рис. 6 – Блок-схема функции point\* delete\_list()**

****

**Рис. 7 – Блок-схема функции int find\_key\_index()**

****

**Рис. 8 – Блок-схема функции void printINfile()**

****

**Рис. 9 – Блок-схема функции void readFROMfile()**

**Код программы**

#include <iostream>

#include <fstream>

#include <string>

#include <Windows.h>

using namespace std;

struct point // узел

{

string key; // Информация в узле

point\* prev, \* next; // Указатели на предыдущий и на следующий элементы

};

point\* make\_list() // Создание и заполнение списка

{

int n;

setlocale(LC\_ALL, "ru");

cout << "Введите размер списка: " << endl;

cin >> n;

cout << "Введите информацию, которую хотите записать: " << endl;

string s;

getline(cin, s);

getline(cin, s);

point\* p, \* r, \* van;

p = new (point);

p->key = s;

p->prev = nullptr;

p->next = nullptr; // Делает введённый пользователем элемент последним

van = p;

for (int i = 1; i < n; i++) //добавить элементы в конец списка

{

r = new(point);

cout << "Введите информацию, которую хотите записать:" << endl;

string s;

getline(cin, s);

r->key = s;

p->next = r;

r->prev = p;

r->next = nullptr;

p = r;

}

return van;

}

void print\_list(point\* van) // Вывод списка на консоль

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

if (van == 0) //если список пустой

{

cout << "Список пустой\n" << endl;

return;

}

point\* p = van;

int i = 1;

cout << "Ваш список:" << endl;

while (p != nullptr) //пока не конец списка

{

cout << i << ". " << p->key << endl; // Вывод информации, содержащейся в узле

p = p->next; //перейти на следующий элемент

i++;

}

cout << "\n";

}

point\* del\_point(point\* van, int k) // Удаление узлов (элементов списка)

{

k--;

point\* p = van;

if (k == 0) {

van = van->next;

if (van == nullptr) return nullptr;

van->prev = nullptr;

delete p;

return van;

}

for (int i = 0; i < k - 1 && p != nullptr; i++, p = p->next);

if (p == nullptr || p->next == nullptr) return van;

point\* r = p->next; //встать на удаляемый элемент

p->next = r->next; //изменить указатель

delete r;

r = p->next; //встать на следующий

if (r) r->prev = p;

return van;

}

point\* add\_point(point\* ctrl, int k) // Добавить узел

{

setlocale(LC\_ALL, "ru");

k--;

point\* p;

p = new(point);

cout << "Введите новую информацию: " << endl; // Заполнение нового нового узла информацией

string s;

getline(cin, s);

getline(cin, s);

p->key = s;

if (k == 0)

{

p->next = ctrl;

p->prev = nullptr; //обнулить адрес предыдущего

//связать список с добавленным элементом

ctrl->prev = p;

ctrl = p;

return ctrl;

}

point\* r = ctrl; //встать на начало списка

for (int i = 0; i < k - 1 && r->next != nullptr; i++, r = r->next);

p->next = r->next; //связать р с концом списка

//если элемент не последний, то связать конец списка с р

if (r->next == nullptr) r->next->prev = p;

p->prev = r; //связать р и r

r->next = p;

return ctrl;

}

int find\_key\_index(point\* van) // Находит индекс элемента по информации, хранящейся в нём

{

setlocale(LC\_ALL, "ru");

cout << "Введите ключ: " << endl;

string key\_word;

getline(cin, key\_word);

getline(cin, key\_word);

point\* r = van;

int counter = 1;

while (r)

{

if (r->key == key\_word) return counter;

r = r->next;

counter++;

}

cout << "Такой элемент не был найден!" << endl;

return 0;

}

point\* delete\_list(point\* van) // Удалить список

{

point\* p = van;

int counter = 1; // Количество элементов в списке

while (p->next != nullptr) {

p = p->next;

counter++;

}

for (int i = counter; i > 0; i--) {

van = del\_point(van, i);

}

return van;

}

void printINfile(point\* van, string path) // Запись списка в файл

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ofstream fout;

fout.open(path);

if (!fout.is\_open()) cout << "Ошибка открытия файла!" << endl;

else {

point\* p = van;

while (p != nullptr) {

fout << p->key;

fout << endl;

p = p->next;

}

}

cout << "Информация была внесена в файл" << endl;

fout.close();

}

point\* ReadFromFile(string path) // Восстановление информации из файла

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

ifstream fin;

fin.open(path);

point\* van = new(point);

if (!fin.is\_open()) { cout << "Ошибка открытия файла!" << endl; }

else {

cout << "Информация была восстановлена!" << endl;

string word;

point\* p, \* r;

p = new (point);

fin >> word;

p->key = word;

p->prev = nullptr;

p->next = nullptr;

van = p;

while (!fin.eof())

{

word = "";

fin >> word;

r = new(point);

r->key = word;

p->prev = nullptr;

p->next = r;

r->prev = p;

r->next = nullptr;

p = r;

}

p = p->prev;

p->next = nullptr;

}

fin.close();

return van;

}

void menu(int a, point\* first)

{

setlocale(LC\_ALL, "ru");

int num;

string path = "myFile.txt";

point\* second;

switch (a)

{

case 0:

exit(777);

case 1:

first = make\_list();

print\_list(first);

break;

case 2:

cout << "Введите позицию элемента, который вы хотите удалить: " << endl;

int n;

cin >> n;

first = del\_point(first, n);

print\_list(first);

break;

case 3:

num = find\_key\_index(first);

if (num != 0)

{

cout << "Введите количество элементов k, которое хотите добавить" << endl;

int k;

cin >> k;

for (int i = 0; i < k; i++) {

first = add\_point(first, num + i);

}

num += k;

for (int i = 0; i < k; i++)

{

first = add\_point(first, num + i + 1);

}

print\_list(first);

}

break;

case 4:

first = delete\_list(first);

print\_list(first);

break;

case 5:

printINfile(first, path);

break;

case 6:

first = ReadFromFile(path);

print\_list(first);

break;

default:

cout << "Пожалуйста, введите номер операции верно" << endl;

break;

}

system("pause");

system("cls");

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a = 1;

cout << "Для работы с программой требуется создать список." << endl;

point\* first = make\_list();

while (true)

{

system("pause");

system("cls");

cout << "0. Выход" << endl;

cout << "1. Создать список." << endl;

cout << "2. Удалить элемент списка. " << endl;

cout << "3. Добавить k элементов справа и слева от выбранного элемента." << endl;

cout << "4. Удалить список." << endl;

cout << "5. Записать список в файл." << endl;

cout << "6. Восстановить список." << endl;

cout << "Выберите действие: ";

cin >> a;

system("cls");

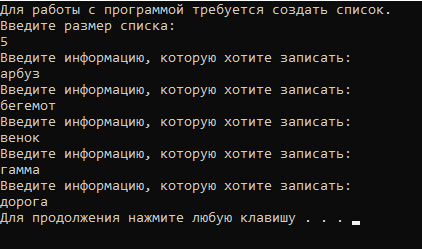
menu(a, first);

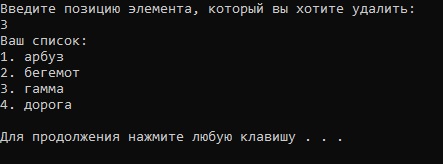
}

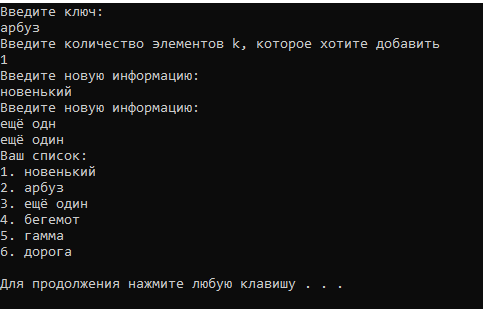
return 0;

}

**Результаты работы программы**

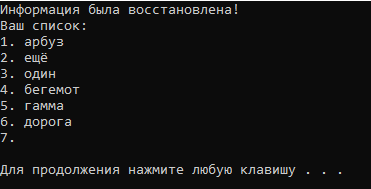












**Рис. 10 – Вывод консоли решения программы**