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**

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

1.Создание очереди.

2.Добавление элемента в очередь.

3.Удаление элемента из очереди.

4.Печать очереди.

5.Запись очереди в файл.

6.Уничтожение очереди.

7. Восстановление очереди из файла.

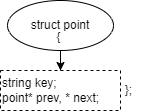
**Алгоритм решения задачи**

1. Создаётся структура struct point с полями string key (информация, хранящаяся в узле) и point\* next (указатель на следующий элемент стэка). Данная структура является структурой узла.
2. Создаётся функция point\* MakeQueue() с параметром int n (размер). Эта функция циклично просит ввести информацию, которая будет записываться в узел очереди, а после создаёт указатель на следующий узел.
3. Создаётся функция void print\_list() с параметрами point\* van (очередь). Через цикл она выводит значения ключа, а затем переносит указатель на следующий элемент.
4. Создаётся функция point\* pop() с параметрами point \* van (очередь). Она переходит на 2 узел очереди, и переносит его указатель prev на nullptr.
5. Создаётся функция point\* push() с параметром point\* van (очередь). Пользователя просят ввести данные нового узла. Затем с помощью цикла находится последний элемент очереди, добавляется новый узел, указатели переносятся соответственно.
6. Создаётся функция void delete\_list() с параметрами point\* van. Через цикл она считает количество элементов в очереди, а затем также через цикл удаляет их.
7. 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 очереди. Если файл не удаётся открыть, функция выводит ошибку.
8. Создаётся функция point\* ReadFromFile() с параметрами string path (ссылка на файл). Создаётся объект point p (первый элемент очереди), 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.
9. Создаётся функция void menu() с параметрами int a (номер операции, выбранной пользователем) и point\* first (созданная изначально p = p->prev;).

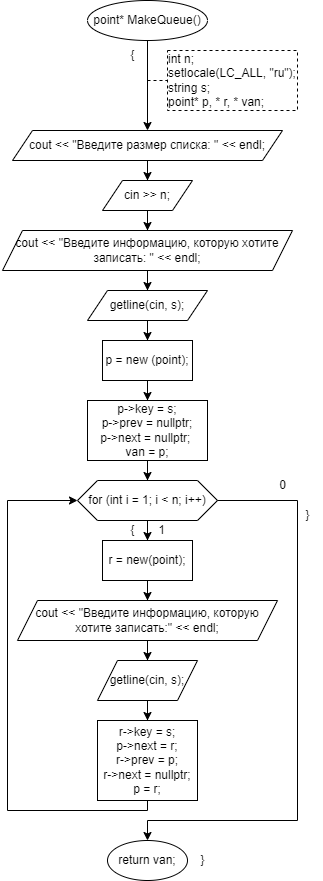
Здесь с помощью свитча выполняются команды, выбранные пользователем. Если пользователь введёт цифру 0 - программа закончит свою работу, а если он введёт какое-то постороннее число, то программа попросит ввести его номер команды заново.

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

**Блок-схема**

****

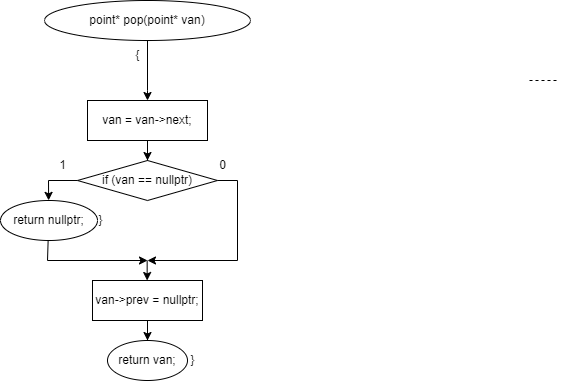
**Рис. 1 – Блок-схема структуры point**

****

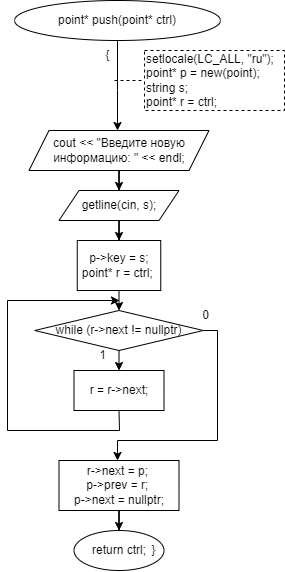
**Рис. 2 – Блок-схема функции point\* MakeQueue()**



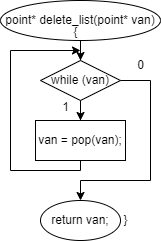
**Рис. 3 – Блок-схема функции void print\_list()**



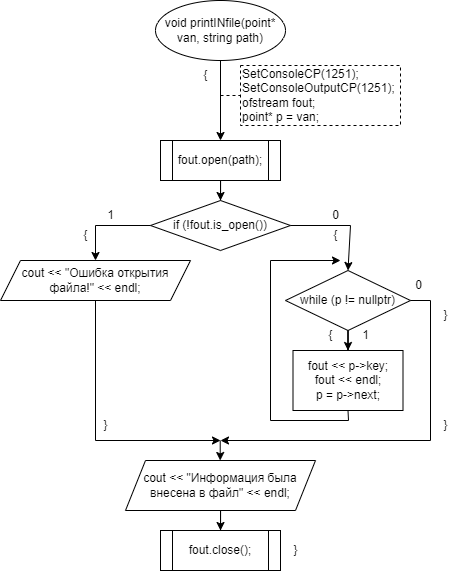
**Рис. 4 – Блок-схема функции point\* pop()**



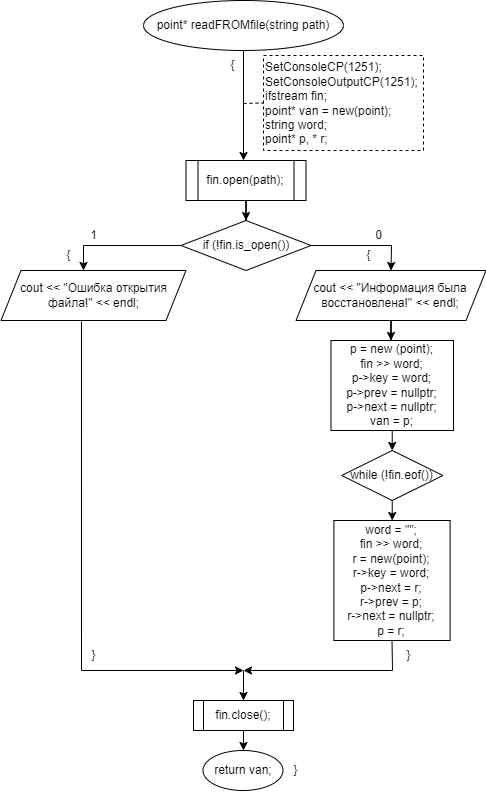
**Рис. 5 – Блок-схема функции point\* push()**



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

****

**Рис. 7 – Блок-схема функции void printINfile()**



**Рис. 8 – 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; // Указатель на следующий элемент

};

point\* MakeQueue(int n) // Создание и заполнение очереди

{

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;

van = p;

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

{

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 очереди на консоль

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

if (van == 0) //если очередь пуста

{

cout << "Очередь пуста\n" << endl;

return;

}

point\* p = van;

int i = 1;

cout << "Ваша очередь:" << endl;

while (p != 0) //пока не конец очереди

{

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

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

i++;

}

cout << "\n";

}

point\* push(point\* ctrl) // Добавляет элемент в конец

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

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;

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

while (r->next != nullptr) r = r->next;

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

p->prev = r; //связать р и r

p->next = nullptr;

return ctrl;

}

point\* pop(point\* van) // Удаляет первый элемент

{

van = van->next;

if (van == nullptr) return nullptr;

van->prev = nullptr;

return van;

}

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

{

while (van) van = pop(van);

return van;

}

void printINfile(point\* van, string path) // Запись очереди 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;

}

}

fin.close();

return van;

}

void menu(int a, point\* first, int n)

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

string path = "myFile.txt";

switch (a)

{

case 0:

exit(777);

case 1:

{

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

cin >> n;

first = MakeQueue(n);

print\_list(first);

break;

}

case 2:

{

first = pop(first);

print\_list(first);

break;

}

case 3:

{

first = push(first);

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;

}

}

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

int a = 1, n;

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

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

cin >> n;

point\* first = MakeQueue(n);

while (true)

{

system("pause");

system("cls");

cout << "0. Выход" << endl;

cout << "1. Создать очередь." << endl;

cout << "2. Удалить элементы из очереди. " << endl;

cout << "3. Добавить элементы в очередь" << endl;

cout << "4. Удалить очередь." << endl;

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

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

cout << "Выберите действие: ";

cin >> a;

system("cls");

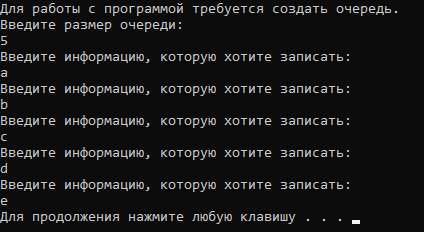
menu(a, first, n);

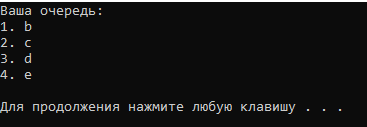
}

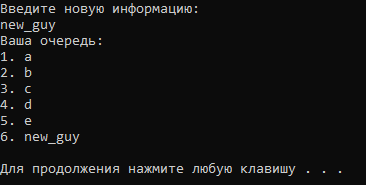
return 0;

}

**Результаты работы программы**

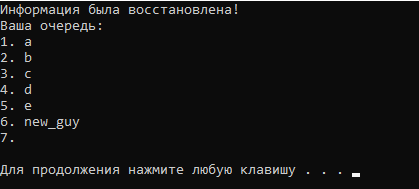












**Рис. 9 – Вывод консоли решения программы**