Министерство высшего образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Пермский национальный исследовательский политехнический университет**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЕТ**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: «Создание списка с помощью стэка»

Семестр 2

*Работу выполнил: Учащийся группы ИВТ - 22-2б: Цыбуцынин Фёдор Александрович*

*Работу 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 стэка.

2.Добавление элемента в стэк.

3.Удаление элемента из стэка.

4.Печать стэка.

5.Запись стэка в файл.

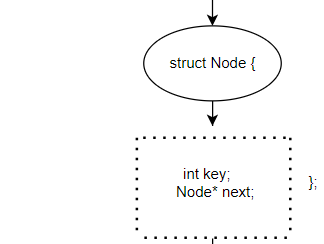
6.Уничтожение стэка.

7. Восстановление стэка из файла.

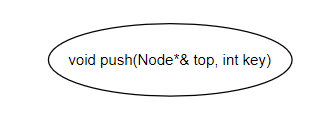
**Алгоритм решения задачи**

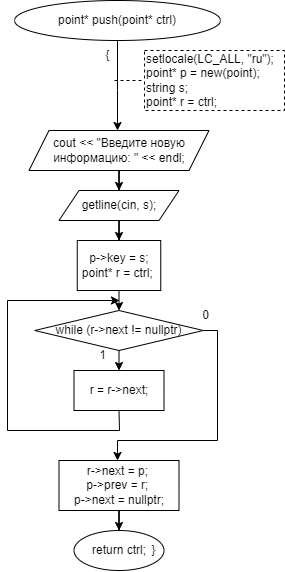
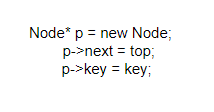
1. Создаётся структура struct Node с полями int key (информация, хранящаяся в узле) и Node\* next (указатель на следующий элемент стэка). Данная структура является структурой узла.
2. Создаётся функция void createStack() с параметром int n (размер). Эта функция циклично просит ввести информацию, которая будет записываться в узел стэка, а после создаёт указатель на следующий узел.
3. Создаётся функция void push() с параметром Node\*& top. Пользователя просят ввести данные нового узла. Затем с помощью цикла находится последний элемент списка, добавляется новый узел, указатели переносятся соответственно.
4. Создаётся функция void pop () с параметрами point Node\*& top. С помощью цикла находит последний элемент списка, затем переносит указатель с предыдущего элемента на nullptr.
5. Создаётся функция void printStack(). Через цикл она выводит значения ключа, а затем переносит указатель на следующий элемент.
6. Создаётся функция void writeStackToFile (). С помощью библиотеки fstream открывается файл по указанной ссылке, и через цикл в него записывается информация из стэка. Если файл не удаётся открыть, функция выводит ошибку.
7. Создаётся функция void destroyStack (). Через цикл она считает количество элементов в стэке, а затем также через цикл удаляет их.
8. Создаётся функция void restoreStackFromFile () с параметрами. Создаётся объект point p (первый элемент стэка), 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 int main(). В ней создаётся список Node\* top, который пользователя просят заполнить.

**Блок-схема**

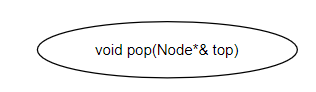
****

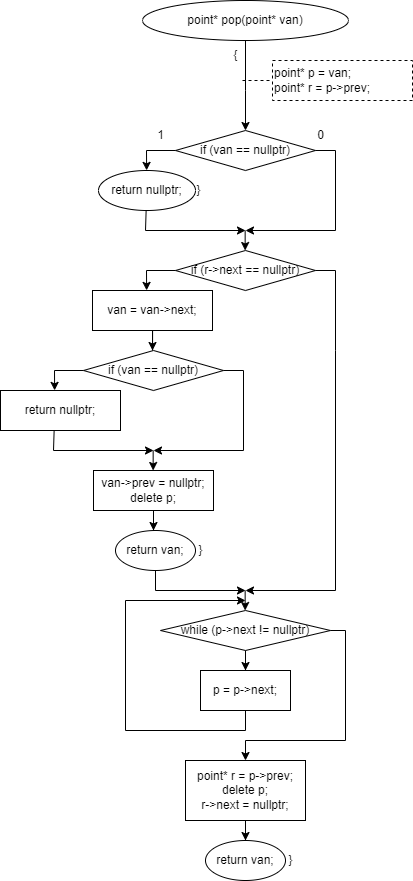
**Рис. 1 – Блок-схема структуры Node**

****

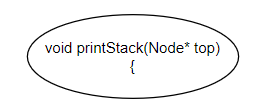


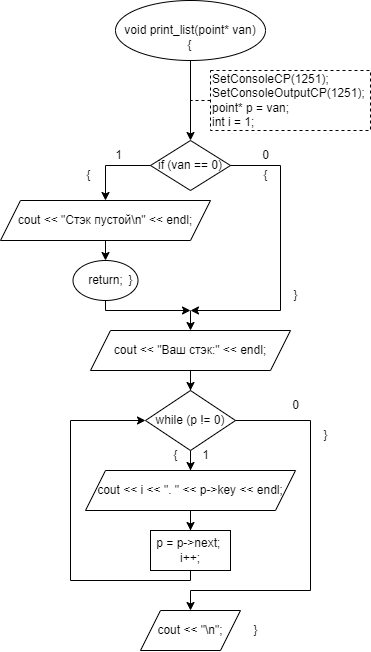
**Рис. 2 – Блок-схема функции void push()**



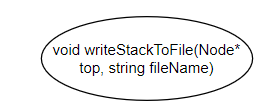
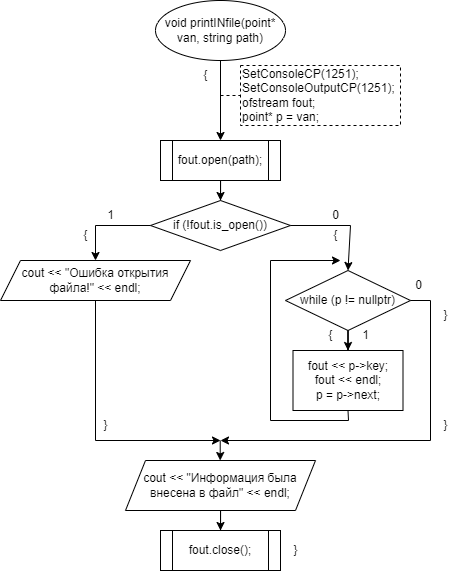


**Рис. 3 – Блок-схема функции void pop()**

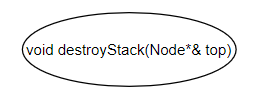


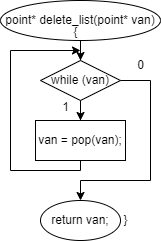
****

**Рис. 4 – Блок-схема функции void print\_list()**

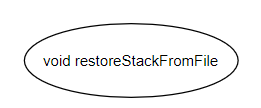
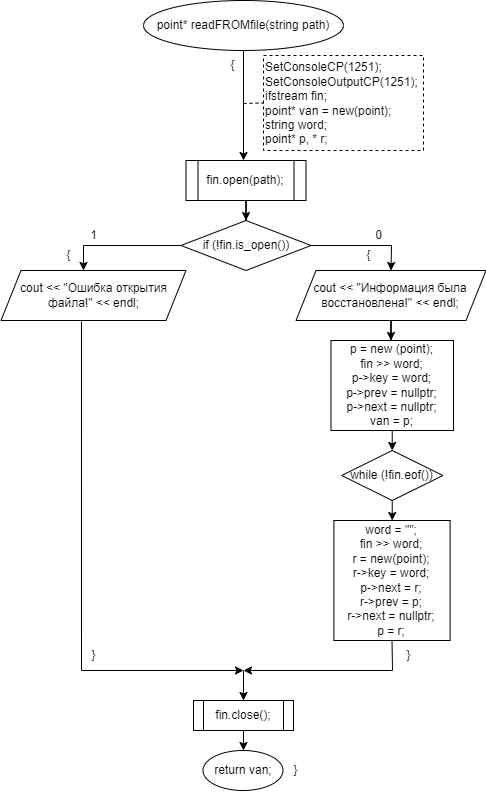
****

**Рис. 5 – Блок-схема функции void writeStackToFile()**

****



**Рис. 6 – Блок-схема функции** **и void destroyStack ()**

****

**Рис. 7– Блок-схема функции restoreStackFromFile()**

**Код программы**

#include <iostream>

#include <fstream>

using namespace std;

struct Node // структура узла стека

{

int key; // ключевое поле

Node\* next; // указатель на следующий элемент

};

void createStack(Node\*& top) // создание пустого стека

{

top = NULL;

}

void push(Node\*& top, int key) // добавление элемента в стек

{

Node\* p = new Node; // создаем новый элемент

p->next = top;

p->key = key; // присваиваем ему значение ключа

// новый элемент становится первым

top = p; // вершина стека указывает на новый элемент

}

void pop(Node\*& top) // удаление верхнего элемента из стека

{

if (top == NULL) // если стек пустой

{

cout << "Стек пустой!" << endl;

return;

}

Node\* p = top; // сохраняем вершину стека

top = top->next; // вершина стека указывает на следующий элемент

delete p; // удаляем сохраненный элемент

}

void printStack(Node\* top) // печать стека

{

if (top == NULL) // если стек пустой

{

cout << "Стек пустой!" << endl;

return;

}

while (top != NULL) // пока не достигнут конец стека

{

cout << top->key << " "; // печатаем значение ключа

top = top->next; // переходим к следующему элементу

}

cout << endl;

}

void writeStackToFile(Node\* top, string fileName) // запись стека в файл

{

ofstream file(fileName); // создаем файл для записи

if (file.is\_open()) // если файл открыт

{

while (top != NULL) // пока не достигнут конец стека

{

file << top->key << " "; // записываем значение ключа

top = top->next; // переходим к следующему элементу

}

file.close(); // закрываем файл

}

else // если файл не открыт

{

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

}

}

void destroyStack(Node\*& top) // уничтожение стека

{

while (top != NULL) // пока не достигнут конец стека

{

Node\* p = top; // сохраняем вершину стека

top = top->next; // вершина стека указывает на следующий элемент

delete p; // удаляем сохраненный элемент

}

}

void restoreStackFromFile(Node\*& top, string fileName) // восстановление стека из файла

{

destroyStack(top); // уничтожаем существующий стек

ifstream file(fileName); // открываем файл для чтения

if (file.is\_open()) // если файл открыт

{

int key;

while (file >> key) // пока не достигнут конец файла

{

push(top, key); // добавляем элемент в стек

}

file.close(); // закрываем файл

}

else // если файл не открыт

{

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

}

}

int main()

{

setlocale(LC\_ALL, "Russian"); // подключаем русский язык

Node\* top; // указатель на вершину стека

int n, k;

createStack(top); // создаем пустой стек

cout << "Введите количество элементов в стеке: ";

cin >> n;

for (int i = 0; i < n; i++) // заполняем стек

{

int key;

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

cin >> key;

push(top, key); // добавляем элемент в стек

}

cout << "Стек: ";

printStack(top); // печатаем стек

cout << "Введите элемент, который нужно добавить: ";

cin >> k;

push(top, k); // добавляем элемент с заданным номером

cout << "Стек после добавления элемента: ";

printStack(top); // печатаем стек

cout << "Введите количество элементов, которые нужно удалить: ";

cin >> k;

for (int i = 0; i < k; i++) // удаляем элементы из конца стека

{

pop(top);

}

cout << "Стек после удаления элементов: ";

printStack(top); // печатаем стек

writeStackToFile(top, "stack.txt"); // записываем стек в файл

destroyStack(top); // уничтожаем стек

restoreStackFromFile(top, "stack.txt"); // восстанавливаем стек из файла

cout << "Стек после восстановления из файла: ";

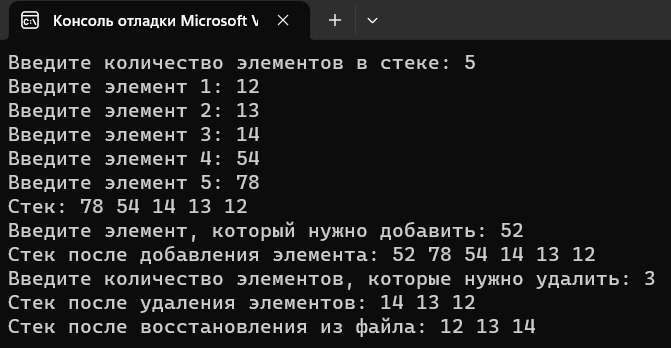
printStack(top); // печатаем стек

destroyStack(top); // уничтожаем стек

return 0;

}

**Результаты работы программы**



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