Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**ОТЧЁТ**

**по лабораторной работе №19**

Дисциплина: «Основы теории алгоритмов и структуры данных»

Тема: “ Основные алгоритмы работы со стеками”

Выполнил:

Студент группы ИВТ-20-2Б Чувашев Максим

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

Реализовать собственный класс, отражающий работу стека.

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать класс Stack c полями которые зранят размер стека и полем типа стринг которое хранит данные

class Stack

{

private:

int st\_size;

string\* st\_stack;

public:

Stack(); // конструктор по умолчанию

Stack(int size); // конструктор с параметрами

Stack(Stack& st); // конструктор копирования

~Stack();

string top();

void push(string elem);

void pop();

bool empty();

int size();

friend istream& operator >> (istream& in, Stack& other);

friend ostream& operator << (ostream& out, Stack& other);

};

* 1. Организовать основные методы для стека

bool Stack::empty()

{

return st\_size == 0;

}

string Stack::top()

{

setlocale(LC\_ALL, "Ru");

if (st\_size != 0)

{

return st\_stack[0];

}

else

{

cout << "Стек пуст!";

return "";

}

}

void Stack::pop()

{

if (st\_size < 1024)

{

for (int i = 0; i < st\_size; i++)

{

st\_stack[i] = st\_stack[i + 1];

}

st\_size--;

}

}

istream& operator>>(istream& in, Stack& other)

{

string value;

cout << "\n\nВведите значение: ";

in >> value;

other.push(value);

return in;

}

ostream& operator<<(ostream& out, Stack& other)

{

cout << "\n\nСтек размера " << other.st\_size << " : ";

for (int i = 0; i < other.st\_size; i++)

{

cout << other.st\_stack[i] << "\t";

}

cout << "\n\n";

return out;

}

1. В ходе работы были использованы следующие типы данных:
   1. Stack st1;
2. Для решения задачи данные были представлены в следующем виде:
   1. Стек заполняется через метод класса, в который передается значение типа t.

void Stack::push(string elem)

{

if (st\_size < 1024)

{

for (int i = st\_size; i > 0; i--) // сдвигаем все элементы на 1

{

st\_stack[i] = st\_stack[i - 1];

}

st\_stack[0] = elem; // элемент становится первым

st\_size++; // размер стека увеличивается на 1

}

}

1. Для операций ввода и вывода использовались следующие операторы и функции:
   1. Для вывода всех элементов стека

ostream& operator<<(ostream& out, Stack& other)

{

cout << "\n\nСтек размера " << other.st\_size << " : ";

for (int i = 0; i < other.st\_size; i++)

{

cout << other.st\_stack[i] << "\t";

}

cout << "\n\n";

return out;

}

* 1. Операторы cin cout

1. Поставленные задачи будут решены следующими действиями:
   1. В методе push в качестве аргумента передаётся параметр, который необходимо добавить в стек. Массив сдвигается, и в нулевой элемент вставляется новый

void Stack::push(string elem)

{

if (st\_size < 1024)

{

for (int i = st\_size; i > 0; i--) // сдвигаем все элементы на 1

{

st\_stack[i] = st\_stack[i - 1];

}

st\_stack[0] = elem; // элемент становится первым

st\_size++; // размер стека увеличивается на 1

}

}

* 1. В методе top возвращается нулевой элемент массива.

string Stack::top()

{

setlocale(LC\_ALL, "Ru");

if (st\_size != 0)

{

return st\_stack[0];

}

else

{

cout << "Стек пуст!";

return "";

}

}

* 1. В методе pop удаляется нулевой элемент и массив сдвигается в другую сторону

void Stack::pop()

{

if (st\_size < 1024)

{

for (int i = 0; i < st\_size; i++)

{

st\_stack[i] = st\_stack[i + 1];

}

st\_size--;

}

}

* 1. Метод empty проверяет пуст ли массив

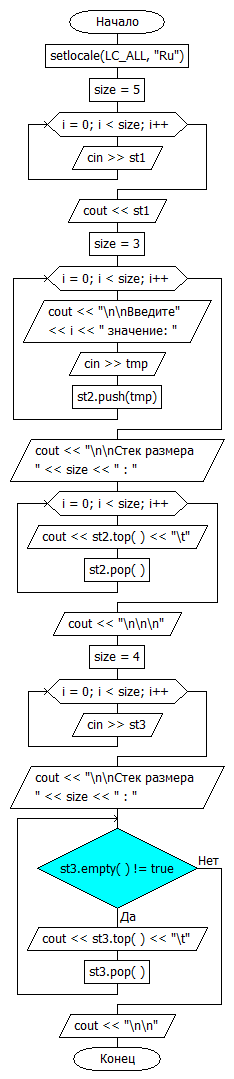
bool Stack::empty()

{

return st\_size == 0;

}

**Блок-схема**









**Код**

#include <iostream>

#include <string>

#include "Stack.h"

using namespace std;

int main()

{

setlocale(LC\_ALL, "Ru");

Stack st1;

int size = 5;

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

{

cin >> st1;

}

cout << st1;

Stack st2;

size = 3;

string tmp;

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

{

cout << "\n\nВведите" << i <<" значение: ";

cin >> tmp;

st2.push(tmp);

}

cout << "\n\nСтек размера " << size << " : ";

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

{

cout << st2.top() << "\t";

st2.pop();

}

cout << "\n\n\n";

Stack st3;

size = 4;

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

{

cin >> st3;

}

cout << "\n\nСтек размера " << size << " : ";

while (st3.empty() != true)

{

cout << st3.top() << "\t";

st3.pop();

}

cout << "\n\n";

}

#include "Stack.h"

using namespace std;

Stack::Stack()

{

st\_size = 0;

st\_stack = new string[1024];

}

Stack::Stack(int size)

{

st\_size = size;

st\_stack = new string[1024];

}

Stack::Stack(Stack& st)

{

st\_size = st.st\_size;

st\_stack = new string[1024];

for (int i = 0; i < st\_size; i++)

{

st\_stack[i] = st.st\_stack[i];

}

}

int Stack::size()

{

return st\_size;

}

Stack::~Stack()

{

delete[] st\_stack;

}

void Stack::push(string elem)

{

if (st\_size < 1024)

{

for (int i = st\_size; i > 0; i--) // сдвигаем все элементы на 1

{

st\_stack[i] = st\_stack[i - 1];

}

st\_stack[0] = elem; // элемент становится первым

st\_size++; // размер стека увеличивается на 1

}

}

bool Stack::empty()

{

return st\_size == 0;

}

string Stack::top()

{

setlocale(LC\_ALL, "Ru");

if (st\_size != 0)

{

return st\_stack[0];

}

else

{

cout << "Стек пуст!";

return "";

}

}

void Stack::pop()

{

if (st\_size < 1024)

{

for (int i = 0; i < st\_size; i++)

{

st\_stack[i] = st\_stack[i + 1];

}

st\_size--;

}

}

istream& operator>>(istream& in, Stack& other)

{

string value;

cout << "\n\nВведите значение: ";

in >> value;

other.push(value);

return in;

}

ostream& operator<<(ostream& out, Stack& other)

{

cout << "\n\nСтек размера " << other.st\_size << " : ";

for (int i = 0; i < other.st\_size; i++)

{

cout << other.st\_stack[i] << "\t";

}

cout << "\n\n";

return out;

}

#pragma once

#include <iostream>

#include <stack>

#include <string>

using namespace std;

class Stack

{

private:

int st\_size;

string\* st\_stack;

public:

Stack(); // конструктор по умолчанию

Stack(int size); // конструктор с параметрами

Stack(Stack& st); // конструктор копирования

~Stack();

string top();

void push(string elem);

void pop();

bool empty();

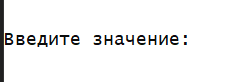
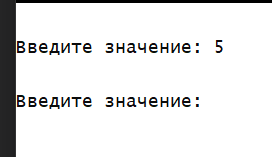
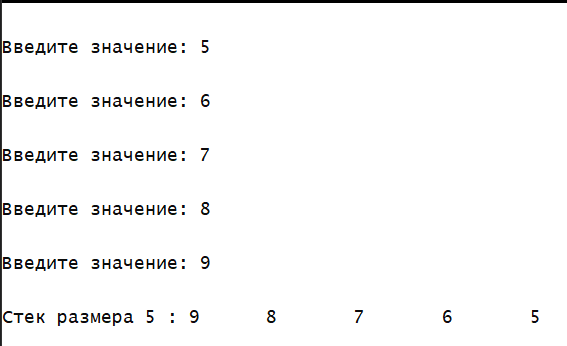
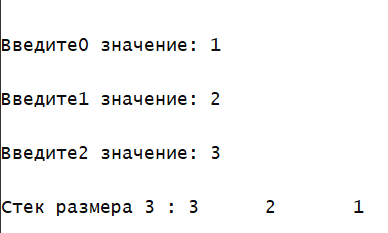
int size();

friend istream& operator >> (istream& in, Stack& other);

friend ostream& operator << (ostream& out, Stack& other);

};

**Скриншоты**

1. 
2. 
3. 
4. 
5. 