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 РИС-20-2Б

Пономарёв Артём Викторович

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Постановка задачи**

Написать собственный класс, отражающий работу стека.

2

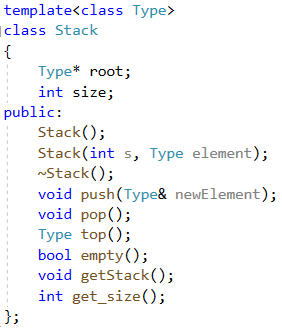
**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать перегруженный класс Stack с полями size типа int, и указателем root типа Type.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Перегруженный класс Stack с полями size типа int, и указателем root типа Type в заголовочном файле Stack.h.



**2.2.** Переменные типа int: size, vvod, element, где size – введённое количество элементов в стеке, vvod – переменная для предоставления пользователю выбора действий в меню, element – переменная, которая хранит элемент, который нужно добавить в стек.

int size;

int vvod=5;

int element;

**2.3.** Объекты класса Stack first, second, third, fourth – массивы типов int, float, double и char соответсвтенно.

Stack<int> first(size, 0);

Stack<float> second(size, 0.5);

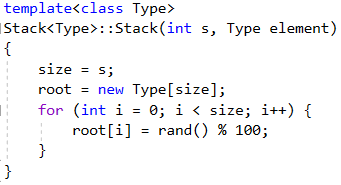
Stack<double> third(size, 0.15625);

Stack<char> fourth(size, 'a');

3

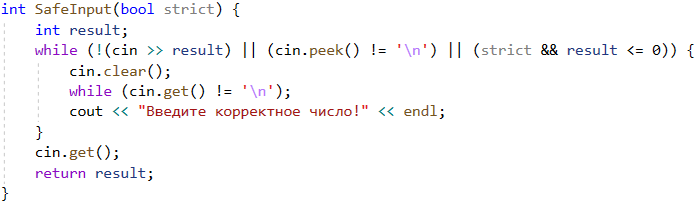
**3.** Для решения задачи данные были представлены в следующем виде:

**3.1.** Стек инициализируется через перегруженный конструктор класса Stack, где в качестве первого параметра отправляется размер стека, а второго параметра отправляется значение, которым можно инициализировать стек, однако для этой задачи используется функция rand().

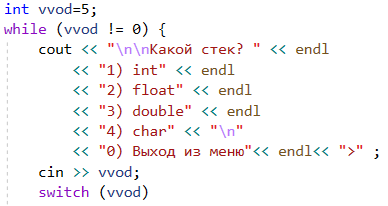


**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Для ввода размера стека используется функция SafeInput(), в которой производится провека на отрицательность и дробность числа. В самой функции число вводится через объект класса ios - cin.



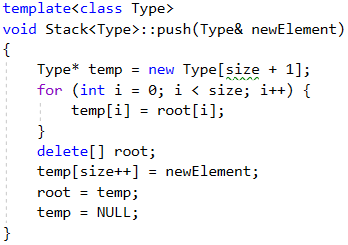
**4.2.** Для выбора пользователем, какого типа необходим стек используется ветвление switch, константа, для которого вводится через объект класса ios - cin.



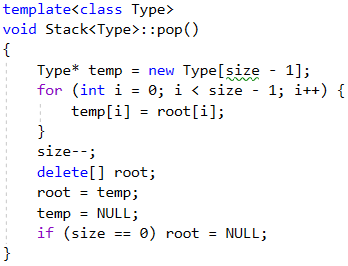
4

**5.** Поставленные задачи будут решены следующими действиями:

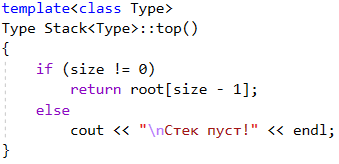
**5.1.** В перегруженном методе push() в качестве параметра передаётся параметр, который необходим добавить в стек. Создаётся указатель temp, указывающий на массив размерностью на 1 большую, чем size. В цикле for массив temp[] заполняется из массива root. Далее в массив temp по индексу size++ добавляется новый элемент. Затем присваивается адрес указателя temp к указателю root и указатель тем присваивается NULL.



**5.2.** В перегруженном методе pop() cоздаётся указатель temp, указывающий на массив размерностью на 1 меньшую, чем size. В цикле for массив temp[] заполняется из массива root до size -1 элемента. Далее необходимо освободить память массива root[] и присвоить память массива temp[] и также присводить NULL указателю temp. Кроме того, добавлена проверка на случай, когда в массиве root нет элементов, тогда необходимо присвоить NULL, дабы предусмотреть утечку памяти.

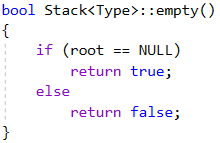


**5.3.** В перегруженном методе top() возвращается элемент по индексу size-1.

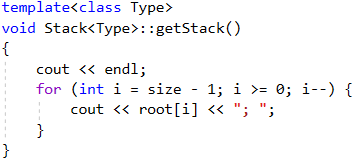


5

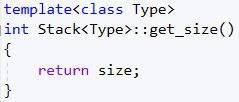
**5.3.** В перегруженном методе empty() возвращается true/false в зависимости от того пуст ли стек: если первый элемент стека – root, указывает на NULL, то стек пуст, иначе заполнен.



**5.4.** В перегруженном методе getStack() в цикле for выводится стек в консоль в порядке от последнего до первого.

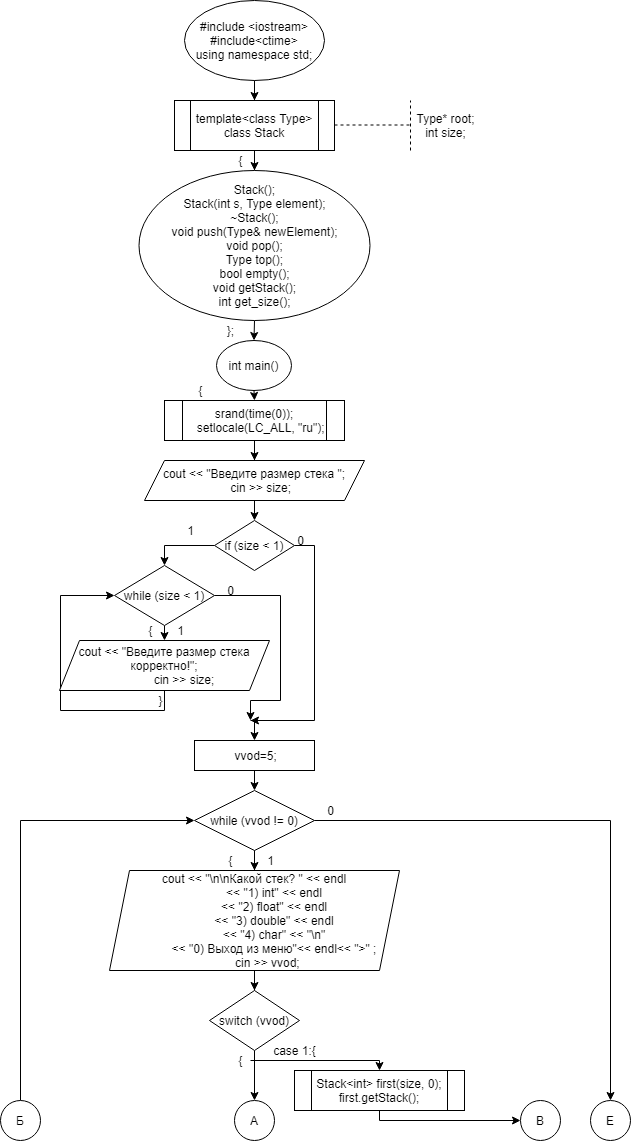


**5.5.** В перегруженном методе get\_size() возвращается значение поля size.

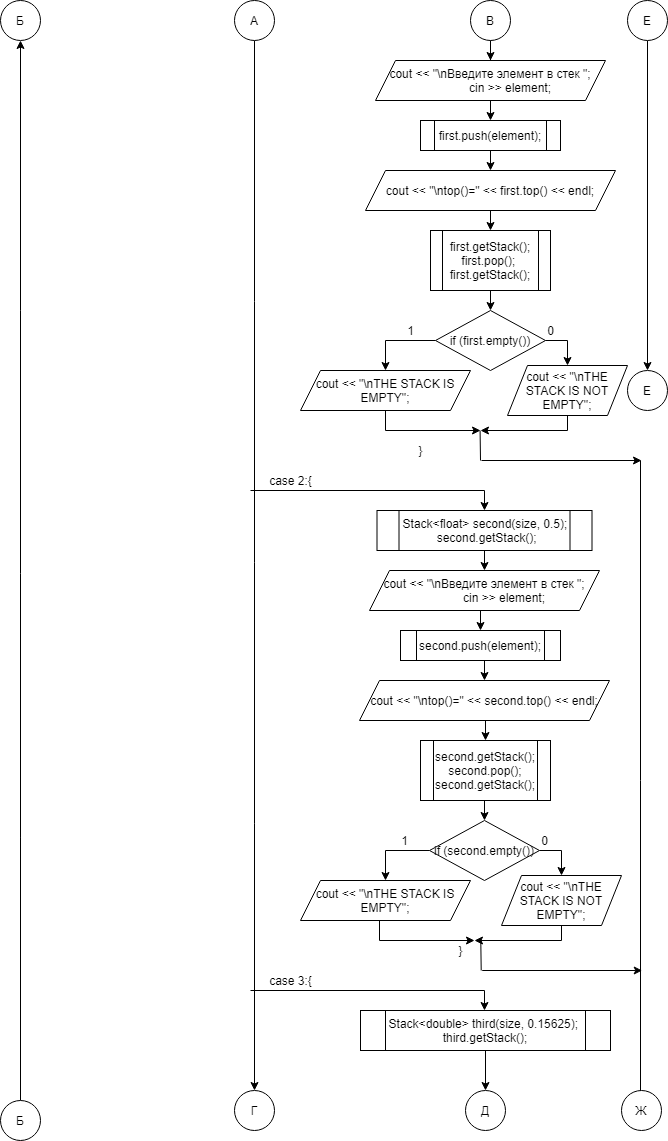


6

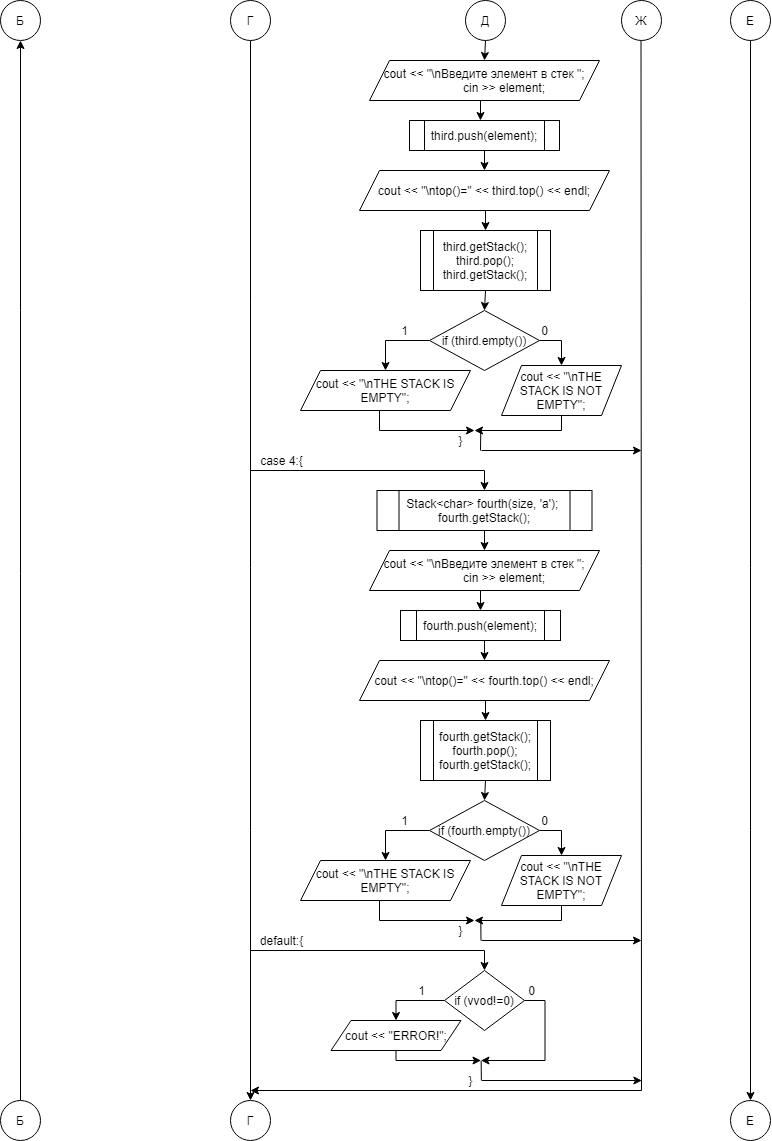
**Блок-схема**



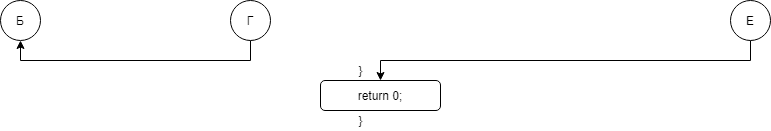
7



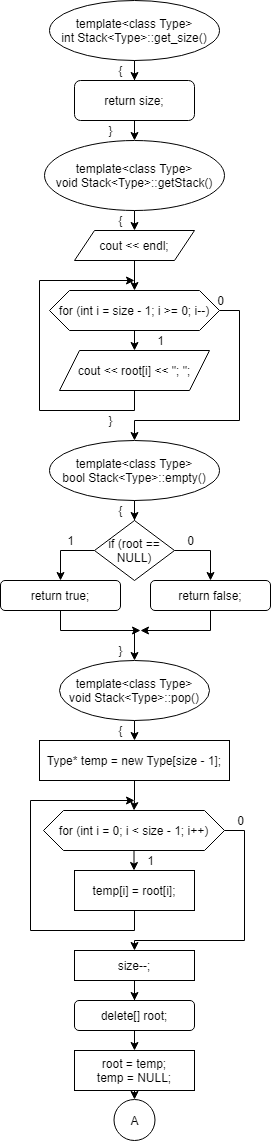
8



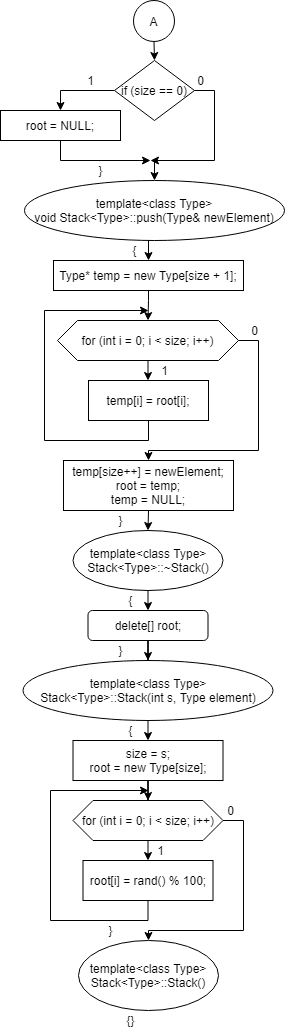
9



10



11



12

**Код**

#include <iostream>

#include "Stack.h"

using namespace std;

int SafeInput(bool strict);

int main()

{

srand(time(0));

setlocale(LC\_ALL, "ru");

int size;

cout << "Введите размер стека ";

size = SafeInput(true);

int vvod=5;

while (vvod != 0) {

cout << "\n\nКакой стек? " << endl

<< "1) int" << endl

<< "2) float" << endl

<< "3) double" << endl

<< "4) char" << "\n"

<< "0) Выход из меню"<< endl<< ">" ;

cin >> vvod;

switch (vvod)

{

case 1:

{

int element;

Stack<int> first(size, 0);

first.getStack();

cout << "\nВведите элемент в стек ";

cin >> element;

first.push(element);

cout << "\ntop()=" << first.top() << endl;

first.getStack();

int length = first.get\_size();

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

first.pop();

first.getStack();

first.empty() ? cout << "\nTHE STACK IS EMPTY" : cout << "\nTHE STACK IS NOT EMPTY";

}

break;

case 2:

{

float element;

Stack<float> second(size, 0.5);

second.getStack();

cout << "\nВведите элемент в стек ";

cin >> element;

second.push(element);

cout << "\ntop()=" << second.top() << endl;

second.getStack();

//int length = second.get\_size();

//for (int i=0; i< length; i++)

second.pop();

second.getStack();

second.empty() ? cout << "\nTHE STACK IS EMPTY" : cout << "\nTHE STACK IS NOT EMPTY";

}

break;

case 3:

{

double element;

13

Stack<double> third(size, 0.15625);

third.getStack();

cout << "\nВведите элемент в стек ";

cin >> element;

third.push(element);

cout << "\ntop()=" << third.top() << endl;

third.getStack();

//int length = third.get\_size();

//for (int i=0; i< length; i++)

third.pop();

third.getStack();

third.empty() ? cout << "\nTHE STACK IS EMPTY" : cout << "\nTHE STACK IS NOT EMPTY";

}

break;

case 4:

{

char element;

Stack<char> fourth(size, 'a');

fourth.getStack();

cout << "\nВведите элемент в стек ";

cin >> element;

fourth.push(element);

cout << "\ntop()=" << fourth.top() << endl;

fourth.getStack();

//int length = fourth.get\_size();

//for (int i=0; i< length; i++)

fourth.pop();

fourth.getStack();

fourth.empty() ? cout << "\nEMPTY" : cout << "\nNOT EMPTY";

}

break;

default:

if (vvod!=0)

cout << "ERROR!";

break;

}

}

return 0;

}

int SafeInput(bool strict) {

int result;

while (!(cin >> result) || (cin.peek() != '\n') || (strict && result <= 0)) {

cin.clear();

while (cin.get() != '\n');

cout << "Введите корректное число!" << endl;

}

cin.get();

return result;

}

14

#pragma once

#include<iostream>

#include<ctime>

using namespace std;

template<class Type>

class Stack

{

Type\* root;

int size;

public:

Stack();

Stack(int s, Type element);

~Stack();

void push(Type& newElement);

void pop();

Type top();

bool empty();

void getStack();

int get\_size();

};

#include "Stack.cpp"

15

#ifndef \_STACK\_CPP\_

#define \_STACK\_CPP\_

#include "Stack.h"

#include <assert.h>

template<class Type>

Stack<Type>::Stack()

{

}

template<class Type>

Stack<Type>::Stack(int s, Type element)

{

size = s;

root = new Type[size];

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

root[i] = rand() % 100;

}

}

template<class Type>

Stack<Type>::~Stack()

{

delete[] root;

}

template<class Type>

void Stack<Type>::push(Type& newElement)

{

Type\* temp = new Type[size + 1];

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

temp[i] = root[i];

}

delete[] root;

temp[size++] = newElement;

root = temp;

temp = NULL;

}

template<class Type>

void Stack<Type>::pop()

{

Type\* temp = new Type[size - 1];

for (int i = 0; i < size - 1; i++) {

temp[i] = root[i];

}

size--;

delete[] root;

root = temp;

temp = NULL;

if (size == 0) root = NULL;

}

template<class Type>

bool Stack<Type>::empty()

{

if (root == NULL)

return true;

else

return false;

}

template<class Type>

void Stack<Type>::getStack()

{

cout << endl;

for (int i = size - 1; i >= 0; i--) {

cout << root[i] << "; ";

}

}

template<class Type>

16

int Stack<Type>::get\_size()

{

return size;

}

template<class Type>

Type Stack<Type>::top()

{

if (size != 0)

return root[size - 1];

else

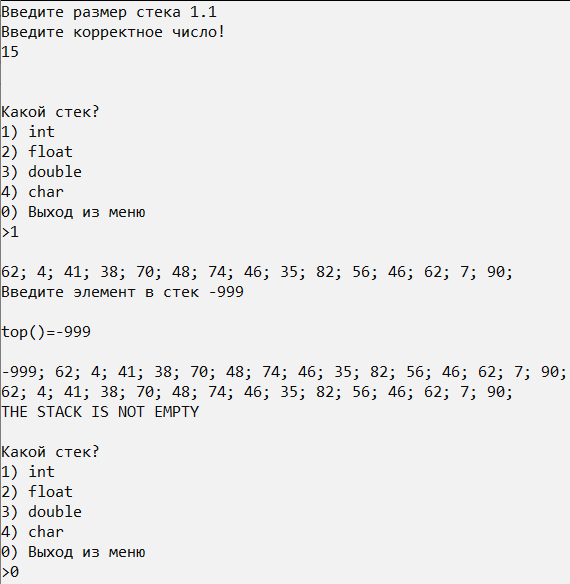
cout << "\nСтек пуст!" << endl;

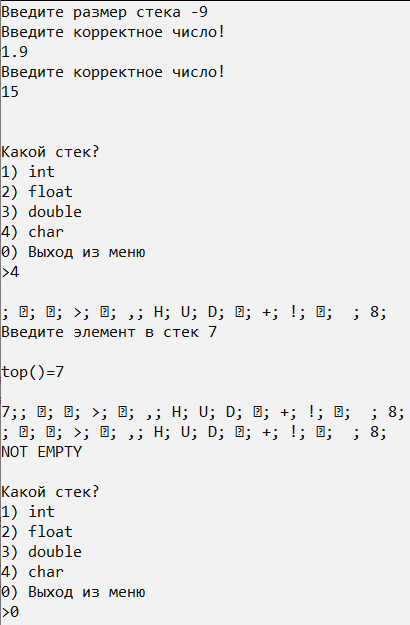
}

#endif

17

**Скриншоты**





18