МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

**«Дальневосточный федеральный университет»**

**ШКОЛА ЕСТЕСТВЕННЫХ НАУК**

**Кафедра информационных систем управления**

**ЛАБОРАТОРНАЯ РАБОТА № 8 ЗАДАЧА № 19**

По дисциплине «Объектно-ориентированное программирование»

На тему «Динамические структуры в языке C++»

|  |
| --- |
| Выполнил студент гр. Б8219 |
| О.В. Константинов |
|  |
| Проверил старший преподаватель |
| Г.Л. Берёзкина |
|  |
| (зачтено/не зачтено) |

г. Владивосток

2016

# Аннотация

Данный отчет подготовлен в рамках задания по дисциплине «Объектно-ориентированное программирование». Отчет призван систематизировать знания об использовании и реализации классов в языке C++. Отчет содержит список выполненных заданий с реализацией на языке C++ в среде программирования MS Visual Studio 2016.

# Задача 19

## Файл *“stack\_lib.h”*

#pragma once

/\*

Задача 19 Описать класс, реализующий стек. Написать программу, использующую этот класс для отыскания прохода по лабиринту.

P.S.: В виду распространенности волнового алгоритма код, реализующий алгоритм, не был включен в данный отчет. В большей степени такое решение вызвано сложностью модификации алгоритма для работы на основе стека.

\*/

#include <cassert>

#include <iostream>

#include <iomanip>

template <typename T>

class Stack {

private:

T \*stackPtr;

const int size;

int top;

public:

Stack(int maxSize = 10) : size(maxSize) { stackPtr = new T[size]; top = 0; };

Stack(const Stack<T> &);

~Stack() { delete[] stackPtr; };

void push(const T &value) { assert(top < size); stackPtr[top++] = value; };

T pop() { assert(top > 0); return stackPtr[--top]; };

void printStack() const;

const T &Peek(int nom) const { assert(nom <= top); return stackPtr[top - nom]; };

int getStackSize() const { return size; };

T \*getPtr() const { return stackPtr; };

int getTop() const { return top; };

};

template <typename T> Stack<T>::Stack(const Stack<T> & otherStack)

: size(otherStack.getStackSize())

{

stackPtr = new T[size];

top = otherStack.getTop();

for (int ix = 0; ix < top; ix++)

stackPtr[ix] = otherStack.getPtr()[ix];

}

template <typename T> void Stack<T>::printStack() const {

for (int ix = top - 1; ix >= 0; ix--)

std::cout << "|" << std::setw(4) << stackPtr[ix] << std::endl;

}