МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное учреждения высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**ЛАБОРАТОРНАЯ РАБОТА № 1**

по дисциплине

**«Объектно-ориентированное программирование»**

на тему:

**«Классы и объекты С++»**

*Вариант № 9*

Выполнил:

Студент группы

Ктбо2-8

Жалнин Д. И.

Проверил:

Тарасов С. А.

Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Таганрог 2020

# 1 ФОРМУЛИРОВКА ЗАДАНИЯ

Согласно варианту задания, требуется реализовать класс стек (Stack для какого-либо типа данных с методами push, pop, isEmpty и back (показывается последний элемент без его извлечения), работающими согласно соответствующей дисциплине обслуживания. Размер стека задается при его создании.

При выполнении работы необходимо: − разработать соответствующие классы, конструкторы, поля и методы; − поля класса сделать закрытыми; для чтения и изменения их значений определить открытые методы; – предусмотреть во всех вариантах консольный ввод данных для создания объектов и консольный вывод результатов. – во всех вариантах необходимо использовать хотя бы один раз блоки try catch. Это можно сделать для контроля арифметических ошибок, для проверки существования файлов и т. п. Запрещено пользоваться контейнерами STL, за исключением string.

# 2 Спецификация классов

## 2.1 Класс Stack

class Stack {

private:

int\* \_stackArray; //массив для хранения данных стэка

int \_maxSize; //максимальный размер стека

int \_size = 0; //текущий размер стэка

public:

Stack(int size);/\*конструктор класса с параметром,

устанавливающим максимальный размер стека\*/

int pop();/\*метод, возвращающий и удаляющий «верхний» элемент стека\*/

int back();/\*метод, возвращающий, но не удаляющий «верхний» элемент стека\*/

void push(int element);/\*метод, добавляющий элемент на «вершину» стека\*/

bool isEmpty();/\*метод, возвращающий true (1), если стек пуст и false (0), если в стеке имеются хранимые значения\*/

int getMaxSize();//метод, возвращающий максимальный размер стека

~Stack();/\*деструктор класса, очищающий память, занятую массивом «\_stackarray»\*/

};

## 2.2 Класс StackOverflowException

#include "exception"

/\*класс переопределяет метод «what», который возвращает сообщение об ошибке переполнения стека\*/

class StackOverflowException : public std::exception {

public:

const char \* what() const noexcept override{

return "Stack overflow exception";

}

};

## 2.3 Класс StackUnderflowException

#include "exception"

/\*класс переопределяет метод «what», который возвращает сообщение об ошибке, возникающей при попытке взять элемент у пустого стека\*/

class StackUnderflowException : public std::exception {

public:

const char \* what() const noexcept override{

return "Stack underflow exception";

}

};

## 2.4 Класс StackUserInterface

#include <iostream>

#include "Stack.h"

/\*Класс предназначен для взаимодействия пользователя с объектом класса Stack. Производит вывод информационных сообщений в консоль, а также считывает оттуда данные\*/

class StackUserInterface{

private:

Stack\* \_stack; //ссылка на объект класса Stack

int getNextInt();/\*метод считывает число из консоли, уведомляя об этом пользователя\*/

int getStackSize();/\*метод считывает число (>=0) из консоли, уведомляя пользователя о последующей установке максимального размера стека, равного введенному значению\*/

void printExceptionMessage(std::exception& e);/\*метод выводит в консоль сообщение, полученное методом «what()» из объекта, на который ведет указатель «std::exception& e»\*/

public:

void start();/\*метод, содержащий в себе интерфейс управления стеком для пользователя, циклично выводимый в консоль, после каждой команды\*/

~StackUserInterface();/\*деструктор класса, очищающий память, занятую объектом класса Stack (\_stack)\*/

};

# 3 Диаграмма классов

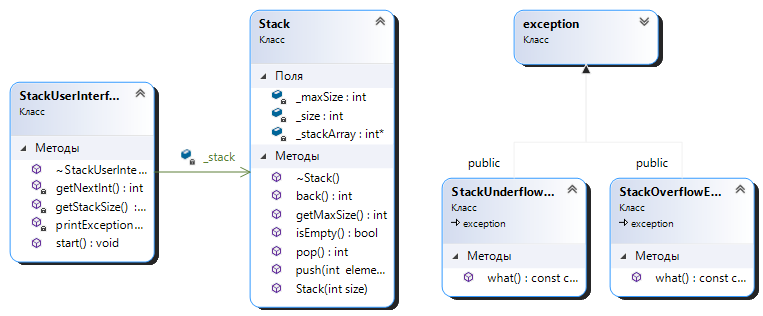


Рисунок 1 – UML диаграмма классов.

# 4 Листинг программы

## 4.1 Main.cpp

#include "StackUserInterface.h"

#include <locale>

#include <vld.h>

int main(){

setlocale(LC\_ALL, "Russian");

StackUserInterface\* stackInterface = new StackUserInterface();

stackInterface->start();

delete (stackInterface);

system("pause");

return 0;

}

## 4.2 Stack.h

#pragma once

class Stack {

private:

int\* \_stackArray;

int \_maxSize;

int \_size = 0;

public:

Stack(int size);

int pop();

int back();

void push(int element);

bool isEmpty();

int getMaxSize();

~Stack();

};

## 4.3 Stack.cpp

#include "Stack.h"

#include "MyExceptions.h"

#include <string.h>

Stack::Stack(int size) {

\_stackArray = new int[size];

\_maxSize = size;

}

Stack::~Stack() {

delete(\_stackArray);

}

bool Stack::isEmpty(){

return (\_size == 0);

}

void Stack::push(int element) {

if (\_size >= \_maxSize) {

throw StackOverflowException();

}

else {

Stack::\_stackArray[\_size++] = element;

}

}

int Stack::pop() {

if (\_size <= 0) {

throw StackUnderflowException();

}

else {

return \_stackArray[--\_size];

}

}

int Stack::back() {

if (\_size <= 0) {

throw StackUnderflowException();

}

else {

return \_stackArray[\_size-1];

}

}

int Stack::getMaxSize() {

return \_maxSize;

}

## 4.4 StackUserInterface.h

#pragma once

#include <iostream>

#include "Stack.h"

class StackUserInterface{

private:

Stack\* \_stack;

int getNextInt();

int getStackSize();

void printExceptionMessage(std::exception& e);

public:

void start();

~StackUserInterface();

};

## 4.5 StackUserInterface.cpp

#include "Stack.h"

#include <iostream>

#include "MyExceptions.h"

#include "StackUserInterface.h"

int StackUserInterface::getNextInt() {

int t;

std::cout << "Введите значение, помещаемое в стэк> ";

std::cin >> t;

return t;

}

void StackUserInterface::printExceptionMessage(std::exception &e) {

std::cout << "Поймано исключение [" << e.what() << "]" << std::endl;

}

int StackUserInterface::getStackSize() {

int n;

std::cout << "Введите размер стэка> ";

std::cin >> n;

if (n < 0) {

std::cout << "Размер должен быть >= 0!" << std::endl;

return getStackSize();

}

return n;

}

void StackUserInterface::start() {

\_stack = new Stack(getStackSize());

bool running = true;

int choice;

while (running) {

std::cout << "1 - pop\n2 - push\n3 - back\n4 - isEmpty()?\n5 - exit\n";

std::cin >> choice;

try {

switch (choice) {

case 1:

std::cout << \_stack->pop() << std::endl;

break;

case 2:

\_stack->push(getNextInt());

break;

case 3:

std::cout << \_stack->back() << std::endl;

break;

case 4:

std::cout << (\_stack->isEmpty() ? "true" : "false") << std::endl;

break;

case 5:

running = false;

break;

default:

continue;

}

} catch (StackOverflowException &e0) {

printExceptionMessage(e0);

std::cout << "Попытка переполнения стэка с размером - " << \_stack->getMaxSize() << std::endl;

} catch (StackUnderflowException &e1) {

printExceptionMessage(e1);

std::cout << "Попытка взять элемент из пустого стэка" << std::endl;

} catch (std::exception &e2) {

printExceptionMessage(e2);

}

}

}

StackUserInterface::~StackUserInterface() {

delete (\_stack);

}

## 4.6 MyExceptions.h

#include "exception"

class StackOverflowException : public std::exception {

public:

const char \* what() const noexcept override{

return "Stack overflow exception";

}

};

class StackUnderflowException : public std::exception {

public:

const char \* what() const noexcept override{

return "Stack underflow exception";

}

};