Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский государственный технологический университет»

Кафедра информационных систем и программирования

**ТЕСТИРОВАНИЕ И ОТЛАДКА  
ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Отчет по лабораторной работе №6

«Модульное тестирование обьектно-ориентированных программ»

Выполнил:

студент 3 курса

группы 19-КБ-ПР1

Кравцов Олег Юрьевич

Проверил:

ст. преп. каф. ИСП

А. Г. Волик

Краснодар

2021

**1 Цель работы**

Изучить подход к автоматизации процесса модульного тестирования объектно-ориентированных программ.

**2 Задание**

1) Создать класс (в соответствии с вариантом задания из п.5),

реализующий работу с АСД (абстрактная структура данных).

2) Построить диаграмму состояний класса.

3) Составить тестовые требования к методам класса на основе

полученной диаграммы состояний.

4) Определить наборы тестов на основе полученных тестовых

требований.

5) Создать проект для автоматизированного модульного тестирования

на основе тестовых наборов.

6) Запустить тестирование и проверить результаты работы.

7) Составить отчет о результатах проведенного тестирования.

**3 Вариант задания**

Создать класс, реализующий стэк через односвязный список.

**4 Текст программы**

using System;

namespace Лр\_\_\_6\_\_ТОПО\_

{

public interface ISTACK<T>

{

bool IsEmpty { get; }

int Count { get; }

void Push(T item);

T Pop();

T Peek();

string PrintElements();

}

public class Node<T>

{

public Node(T data)

{

Data = data;

}

public T Data { get; set; }

public Node<T> Next { get; set; }

}

public class NodeStack<T>:ISTACK<T>

{

Node<T> head;

int count;

public bool IsEmpty

{

get { return count == 0; }

}

public int Count

{

get { return count; }

}

public void Push(T item)

{

// увеличиваем стек

Node<T> node = new Node<T>(item);

node.Next = head; // переустанавливаем верхушку стека на новый элемент

head = node;

count++;

}

public T Pop()

{

// если стек пуст, выбрасываем исключение

if (IsEmpty)

throw new InvalidOperationException("Стек пуст");

Node<T> temp = head;

head = head.Next; // переустанавливаем верхушку стека на следующий элемент

count--;

return temp.Data;

}

public T Peek()

{

if (IsEmpty)

throw new InvalidOperationException("Стек пуст");

return head.Data;

}

public string PrintElements()

{

string s = "";

Node<T> tempNode = head;

while (tempNode != null)

{

s += tempNode.Data + " ";

tempNode = tempNode.Next;

}

return s;

}

}

public class Test

{

public static void Main(string[]args)

{

}

}

}

**5 Текст модульных тестов**

using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace Lr6\_Tests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void CheckPrintElements()

{

Лр\_\_\_6\_\_ТОПО\_.NodeStack<int> stack = new Лр\_\_\_6\_\_ТОПО\_.NodeStack<int>();

stack.Push(5);

stack.Push(6);

stack.Push(7);

stack.Push(8);

string expect = "8 7 6 5 ";

Assert.AreEqual(expect, stack.PrintElements());

}

[TestMethod]

public void CheckGetFirstElements()

{

Лр\_\_\_6\_\_ТОПО\_.NodeStack<int> stack = new Лр\_\_\_6\_\_ТОПО\_.NodeStack<int>();

stack.Push(5);

stack.Push(6);

stack.Push(7);

stack.Push(8);

int expect = 8;

Assert.AreEqual(expect, stack.Peek());

}

[TestMethod]

public void CheckIsEmpty()

{

Лр\_\_\_6\_\_ТОПО\_.NodeStack<int> stack = new Лр\_\_\_6\_\_ТОПО\_.NodeStack<int>();

bool expect = true;

Assert.AreEqual(expect, stack.IsEmpty);

}

[TestMethod]

public void CheckPop()

{

Лр\_\_\_6\_\_ТОПО\_.NodeStack<int> stack = new Лр\_\_\_6\_\_ТОПО\_.NodeStack<int>();

stack.Push(5);

stack.Push(6);

stack.Push(7);

stack.Push(8);

stack.Pop();

string expect = "7 6 5 ";

Assert.AreEqual(expect, stack.PrintElements());

}

[TestMethod]

public void CheckCount()

{

Лр\_\_\_6\_\_ТОПО\_.NodeStack<int> stack = new Лр\_\_\_6\_\_ТОПО\_.NodeStack<int>();

stack.Push(5);

stack.Push(6);

stack.Push(7);

stack.Push(8);

int expect = 4;

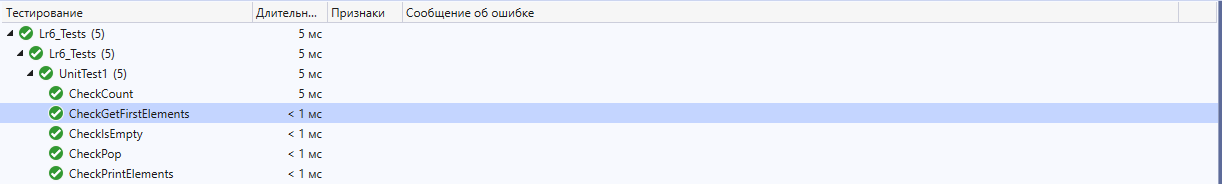
Assert.AreEqual(expect, stack.Count);

}

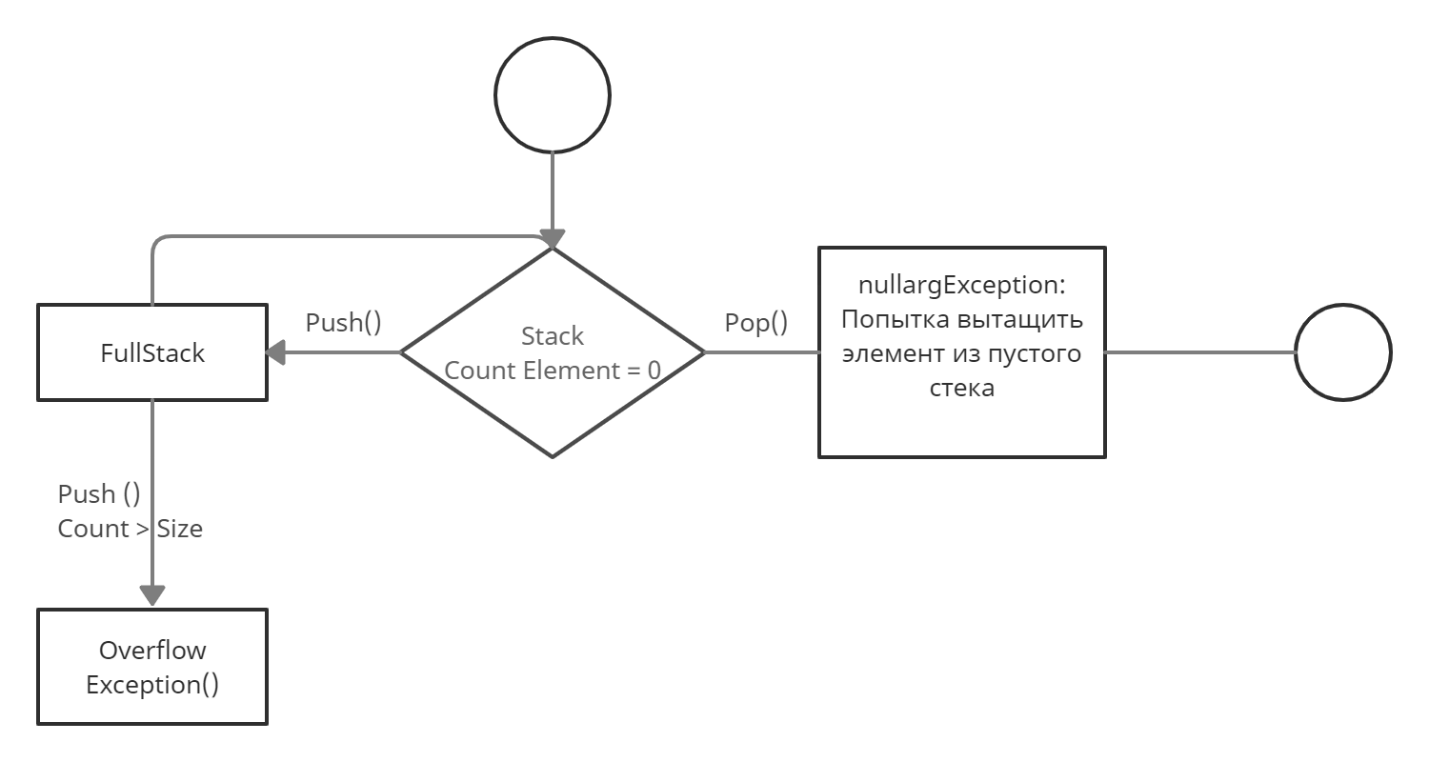
}

}

**6 Результаты тестирования программы**



9



**7 Выводы**

В результате выполнения лабораторной работы был изучен подход к автоматизации процесса модульного тестирования объектно-ориентированных программ. Так, был создан интерфейс стэка и класс, который его реализует, а также написан набор тестов для тестирования стэка.