Федеральное агентство по образованию Российской Федерации

Государственное образовательное учреждение

Высшего профессионального образования

Нижегородский государственный университет им. Н.И. Лобачевского

Институт информационных технологий математики механики

Стек

Отчет по лабораторной работе

Выполнил:

студент ИИТММ гр. 381706-2

Антипин А.С.\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил:

ассистент каф. МОСТ, ИИТММ

Лебедев И.Г\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

Нижний Новгород

2018 г.

**Содержание**

[Введение 3](#_Toc533954732)

[Постановка целей и задач 4](#_Toc533954733)

[Руководство пользователя 5](#_Toc533954734)

[Руководство программиста 6](#_Toc533954735)

[Описание структуры программы 6](#_Toc533954736)

[Описание структур данных 7](#_Toc533954737)

[Описание алгоритмов 8](#_Toc533954738)

[Заключение 9](#_Toc533954739)

[Литература 10](#_Toc533954740)

[Приложение 11](#_Toc533954741)

# Введение

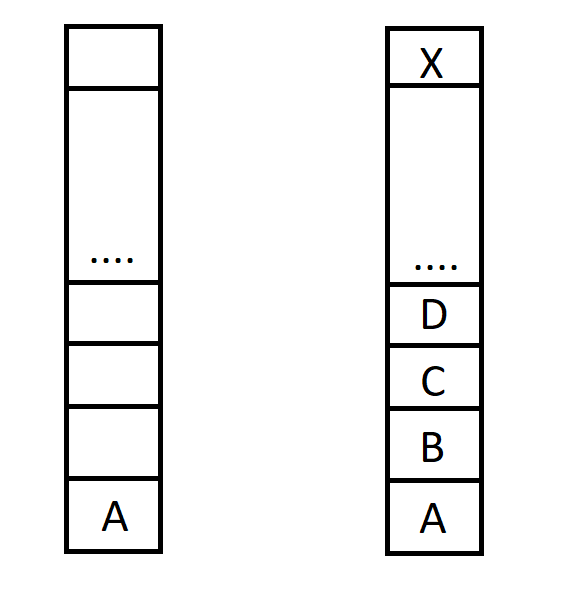
Одной из самых распространенных структур данных является стек. Стек – это структура, при которой элементы хранятся по принципу «первым вошел, последним вышел». Что это значит? Если у нас есть пустой стек, и в него кладется элемент, а за ним еще несколько, то чтобы добраться до нужного нам, придется извлечь все элементы, которые были положены после него (рис. 1).

рис. 1 (представление элементов в стеке. 1-вый столбец – положили один элемент в стек, 2-ой столбец – заполненный стек)

Практическое применение стек нашел в алгоритмах поиска в «глубину» и в «ширину». Таким образом стек - это инструмент решения алгоритмических (и не только) задач.

# Постановка целей и задач

Основной целью лабораторной работы является создание структуры данных «Стек» и реализация таких методов, как доступ к элементам стека:

* Положить элемент в стек;
* Вернуть элемент с удалением;
* Вернуть элемент без удаления;

Для реализации алгоритмов будет использоваться шаблонный класс Stack.

Для проверки правильности работы этого класса будут написаны тесты с использованием фреймворка Google Test, а также тестовый образец программы, которая использует данный класс.

# Руководство пользователя

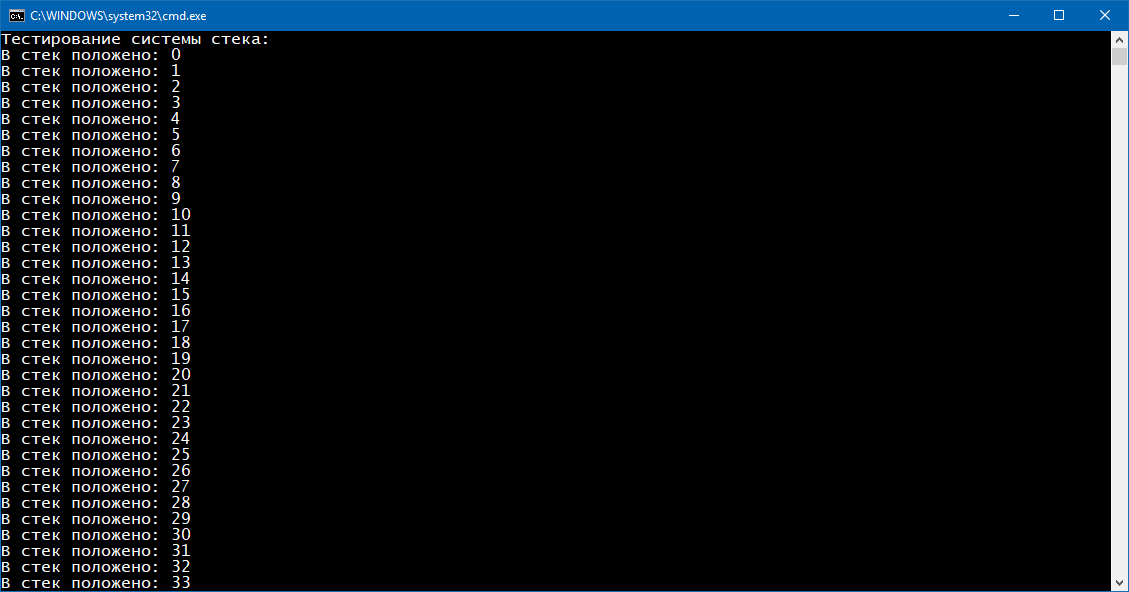
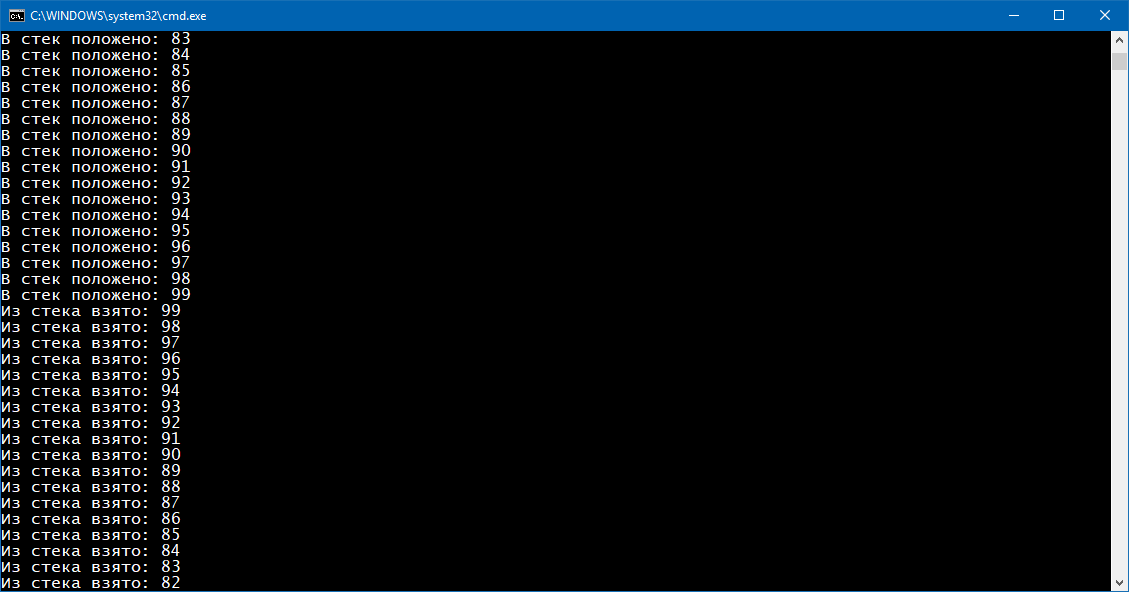
После запуска программы пользователя встречает консольное окно (рис. 2):

рис. 3 (вывод работы стека для пользователя, из стека забираются элементы с 99 до 0)

рис. 2 (вывод работы стека для пользователя, в стек кладутся числа от 0 до 99)

В стек кладутся числа от 0 до 99, а затем из стека забираются эти элементы, причем сначала забирается 99, 98, и т.д., а уже потом 0 (рис. 3).

# Руководство программиста

## Описание структуры программы

Объект типа стек будет выделять необходимое количество памяти под объекты определенного типа, которое будет задаваться при его создании. Т.е. максимальное число объектов в стеке будет ограниченно.

Для реализации алгоритмов будет создан класс Stack, который будет включать в себя все алгоритмы для работы с ним.

А также проект использующий фреймворк Google Test, для проверки правильности работы класса стек, а также тесовый проект, который будет показываться пользователю.

Древо классов

**Класс gtest**

**test\_main.cpp**

**Stack\_test.cpp**

**Проект для пользователя**

**main.cpp**

**stack.h**

**Класс Stack**

**Класс Stack:**

В этом классе реализованы методы работы со стеком, такие как: положить элемент, забрать элемент, вернуть текущее количество элементов в стеке, проверить является ли стек пустым или полным в данный момент.

**Класс gtest:**

Класс gtest реализует тестирование класса Stack, по средствам фреймворка Google Test. Тесты пишутся для каждого метода классов, каждого ветвления этих методов и для всех возможных исключений этих методов.

**Проект Stack:**

В данном проекте реализован примет использования стека для заполнения элементами и их последующего забора.

## Описание структур данных

**Класс Stack:**

template <class T> - шаблон класса T

int Len; - максимальная длина стека:

T \*Mem; - указатель на первый элемент стека типа Т;

int Index; - индекс текущего элемента в стеке;

Exceptions\_from\_stack\_queue\_multystack except; - переменная для вызова исключений.

**Описание методов:**

|  |  |
| --- | --- |
| Метод: | Описание: |
| Stack<T>::Stack() | Конструктор по умолчанию стека. |
| Stack<T>::Stack(const int \_Len) | Конструктор с параметром, который принимает длину стека. |
| Stack<T>::Stack(const Stack<T>& A) | Конструктор копирования стека, который принимает константную ссылку на другой стек. |
| Stack<T>::~Stack() | Деструктор для класса стек |
| void Stack<T>::SetMem(T\* \_Mem, int \_Len) | Служебный метод, который позволяет изменить указатель на первый элемент стека и длину стека (нужен для реализации мультистека). |
| int Stack<T>::GetIndex() | Метод, который возвращает текущий индекс стека. |
| int Stack<T>::GetLen() | Метод, который возвращает длину стека. |
| void Stack<T>::Put(const T& A) | Метод, который позволяет положить элемент в стек. |
| T Stack<T>::Get() | Возвращение элемента из стека с его удалением. |
| T Stack<T>::GetWithoutDelete() | Возвращение элемента из стека без удаления. |
| bool Stack<T>::IsFull() | Проверка стека на полноту. |
| bool Stack<T>::IsEmpty() | Проверка стека на пустопу. |

## Описание алгоритмов

**Подробное описание некоторых методов**

Добавление элемента в стек:

* Проверка на полноту, если стек полон, то выдается ошибка о полноте стека;
* Если в стеке есть свободное место, то элемент кладется в стек, а индекс увеличивается на 1.

Возврат элемента из стека:

* Проверка на пустоту, если стек пустой, то выдается ошибка о пустоте стека;
* Если в стеке есть элементы, то элемент возвращается, а индекс уменьшается на 1.

# Заключение

Подводя итоги можно сказать, что стек имеет огромное практическое применение, а его реализация не составляет труда. Также нельзя не отметить скорость работы стека, т.к. в алгоритмах его реализации не присутствует циклов, а значит сложность O(1). Все эти неоспоримые преимущества показывает доказывают эффективность применения структуры данных «стек» в практических задачах.

# Литература

* Учебные материалы к учебному курсу «Методы программирования» - Гергель В.П.
* http://www.cyberforum.ru/cpp-beginners/thread587712.html

# Приложение

**stack.h:**

#pragma once

#include "Exception\_class.h"

#include <iostream>

template<class T>

class Stack

{

public:

int Len;

T \*Mem;

int Index;

Exceptions\_from\_stack\_queue\_multystack except;

void SetMem(T\* \_Mem, int \_Len);

public:

Stack();

Stack(const int \_Len);

Stack(const Stack<T>& A);

~Stack();

int GetIndex();

int GetLen();

void Put(const T& A);

T Get();

T GetWithoutDelete();

bool IsFull();

bool IsEmpty();

};

template<class T>

Stack<T>::Stack()

{

Len = 0;

Mem = NULL;

Index = 0;

}

template<class T>

Stack<T>::Stack(const int \_Len)

{

if (\_Len < 0)

except.except\_throw(101);

else if (\_Len == 0)

{

Len = 0;

Mem = NULL;

Index = 0;

}

else

{

Len = \_Len;

Mem = new T[Len];

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

Mem[i] = 0;

Index = 0;

}

}

template<class T>

Stack<T>::Stack(const Stack<T>& A)

{

Len = A.Len;

Mem = new T[Len];

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

Mem[i] = A.Mem[i];

Index = A.Index;

}

template<class T>

Stack<T>::~Stack()

{

delete[] Mem;

}

template <class T>

void Stack<T>::SetMem(T\* \_Mem, int \_Len)

{

Mem = \_Mem;

Len = \_Len;

}

template<class T>

int Stack<T>::GetIndex()

{

return Index;

}

template<class T>

int Stack<T>::GetLen()

{

return Len;

}

template<class T>

void Stack<T>::Put(const T& A)

{

if (IsFull() == false)

{

Mem[Index] = A;

Index++;

}

else

except.except\_throw(102);

}

template<class T>

T Stack<T>::Get()

{

if (IsEmpty() == false)

{

Index--;

return Mem[Index];

}

else

except.except\_throw(103);

}

template<class T>

T Stack<T>::GetWithoutDelete()

{

if (IsEmpty() == false)

return Mem[Index];

else

except.except\_throw(103);

}

template<class T>

bool Stack<T>::IsFull()

{

if (Index == Len)

return true;

else if (Index > Len || Index < 0)

except.except\_throw(104);

else

return false;

}

template<class T>

bool Stack<T>::IsEmpty()

{

if (Index == 0)

return true;

else if (Index > Len || Index < 0)

except.except\_throw(105);

if (Index > 0)

return false;

}

**Stack\_test.cpp:**

#include <gtest.h>

#include "Stack.h"

TEST(Stack, can\_create\_stack\_with\_positive\_len)

{

ASSERT\_NO\_THROW(Stack<int> A(5));

}

TEST(Stack, can\_create\_stack\_with\_negative\_len)

{

ASSERT\_ANY\_THROW(Stack<int> A(-2));

}

#define SETMEMORY

#ifdef SETMEMORY

TEST(Stack, can\_set\_memory)

{

Stack<int> A(3);

int\* B = new int[10];

A.SetMem(B, 10);

EXPECT\_EQ(10, A.GetLen());

EXPECT\_EQ(B, A.Mem);

}

#endif

TEST(Stack, can\_return\_index)

{

Stack<int> A(3);

A.Put(10);

EXPECT\_EQ(1, A.GetIndex());

}

TEST(Stack, can\_return\_len)

{

Stack<int> A(3);

EXPECT\_EQ(3, A.GetLen());

}

TEST(Stack, can\_put\_item)

{

Stack<int> A(3);

A.Put(10);

EXPECT\_EQ(10, A.Get());

}

TEST(Stack, can\_throw\_when\_put\_item\_in\_full\_stack)

{

Stack<int> A(3);

A.Put(10);

A.Put(5);

A.Put(1);

ASSERT\_ANY\_THROW(A.Put(3));

}

TEST(Stack, can\_get\_item)

{

Stack<int> A(3);

A.Put(10);

A.Put(5);

A.Put(2);

EXPECT\_EQ(2, A.Get());

}

TEST(Stack, can\_throw\_when\_get\_item\_in\_empty\_stack)

{

Stack<int> A(3);

ASSERT\_ANY\_THROW(A.Get());

}

TEST(Stack, can\_get\_item\_without\_delete)

{

Stack<int> A(3);

A.Put(10);

A.Put(5);

A.Put(2);

EXPECT\_EQ(2, A.GetWithoutDelete());

EXPECT\_EQ(2, A.GetWithoutDelete());

}

TEST(Stack, can\_throw\_when\_get\_item\_in\_empty\_stack)

{

Stack<int> A(3);

ASSERT\_ANY\_THROW(A.GetWithoutDelete());

}

TEST(Stack, can\_report\_if\_stack\_is\_full)

{

Stack<int> A(3);

A.Put(10);

A.Put(5);

A.Put(2);

EXPECT\_EQ(1, A.IsFull());

}

TEST(Stack, can\_report\_if\_stack\_is\_not\_full)

{

Stack<int> A(3);

A.Put(10);

EXPECT\_EQ(0, A.IsFull());

}

TEST(Stack, can\_report\_if\_stack\_is\_empty)

{

Stack<int> A(3);

EXPECT\_EQ(1, A.IsEmpty());

}

TEST(Stack, can\_report\_if\_stack\_is\_not\_empty)

{

Stack<int> A(3);

A.Put(10);

EXPECT\_EQ(0, A.IsEmpty());

}

**main.cpp:**

#include "Stack.h"

#include <iostream>

void main()

{

setlocale(LC\_ALL, "RUS");

#ifdef stack

Stack<int> st(100);

std::cout << "Тестирование системы стека:" << std::endl;

for (int i = 0; i < 100; i++)

{

st.Put(i);

std::cout << "В стек положено: " << i << std::endl;

}

for (int i = 0; i < 100; i++)

std::cout << "Из стека взято: " << st.Get() << std::endl;

}