МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

**«Национальный исследовательский**

**Нижегородский государственный университет им. Н.И. Лобачевского»**

**(ННГУ)**

**Институт информационных технологий, математики и механики**

**Кафедра математического обеспечения и суперкомпьютерных технологий**

**Отчет по учебной практике**

**«Структура хранения данных: мультистек»**

**Выполнил:** студент группы 381706-1

Колесова Кристина Юрьевна

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Оглавление

[1. Введение 3](#_Toc534392616)

[2. Постановка задачи 4](#_Toc534392617)

[3. Руководство пользователя 5](#_Toc534392618)

[4. Руководство программиста 6](#_Toc534392619)

[4.1. Описание структуры программы 6](#_Toc534392620)

[4.2. Описание структур данных 6](#_Toc534392621)

[4.3. Описание алгоритмов 7](#_Toc534392622)

[5. Заключение 8](#_Toc534392623)

[6. Список литературы 9](#_Toc534392624)

# Введение

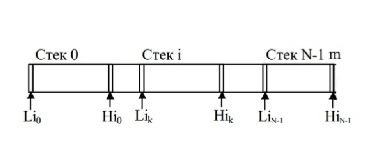
Мультистек — структура данных, представляющая собой упорядоченный набор N стеков, фиксированного размера. Каждый отдельный стек организован по принципу LIFO (англ. last in — first out, «последним пришёл — первым вышел»). Стеки хранятся в памяти друг за другом единым блоком, размер которого m:

Рисунок 1. Структура памяти для хранения мультистека

*Где*  – индекс начала j-го стека, – индекс последнего элемент j-го стека.

Свойства такой структуры памяти:

1. – условие неподвижности первого стека;
2. – условие пустоты;
3. – условие неперекрытия;
4. – условие переполнения;

Цель данной лабораторной работы – реализация структуры данных мультистек на основе массива.

# Постановка задачи

Выполнение работы предполагает решение следующих задач:

1. Разработка и реализация вспомогательного класса стека – TStack
2. Разработка и реализация класса мультистека - TNewStack
3. Реализация класса для обработки исключений – EexceptionLib
4. Реализация тестов на базе Google Test
5. Пример использования класса MStack

# Руководство пользователя

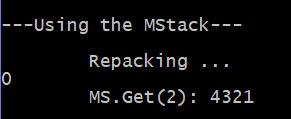
При запуске программы пользователю представляется простой пример использования списков. Создается объект класса TMStack, мультистек. Затем в него кладутся значения и на экран выводятся элементы одного из стеков. На определенном этапе стек, в который пытаемся добавить элемент, переполняется, тогда будет выполнена перепаковка.

Рисунок . Пример работы программы

# Руководство программиста

## Описание структуры программы

В программе содержатся следующие модули:

* Модуль StackLib – статическая библиотека. Содержит файл Stack.h, в котором описан интерфейс и реализация шаблонного класса *TStack*.
* Модуль ExceptionLib – библиотека, позволяющая создавать собственные исключения.
* Модуль MStackLib – содержит файл MStack.h, в котором реализованы классы TMStack и TNewStack
* Модуль MStack – содержит в себе файл реализации примера использования класса TMStack.
* Модуль MStackTest – содержит в себе файл MStackTest.cpp, в котором находится набор тестов, для проверки работоспособности класса TMStack.

## Описание структур данных

* Класс TNewStack

Класс TNewStack является шаблонным.

1. Элементы класса, объявленные со спецификатором public:

TNewStack (int \_size = 0, T\* \_mas = 0) - конструктор

TNewStack(TNewStack <T> &NS) - конструктор копирования

int GetFreeMemory() - получение свободной памяти (кол-во свободных позиций)

int GetSize() - получение размера

int GetTop() - получение первого элемента

void SetMas(int \_size, T\* \_mas) - задать массив размера \_size, содержащий элементы \_mas.

void Put(T \_A) - положить элемент \_A

T Get() - получить значение элемента

~ TNewStack() – деструктор.

* Класс TMStack

1) Элементы класса, объявленные со спецификатором protected:

int size - размер

T\* mas - массив элементов

int n - кол-во стеков

TNewStack<T>\*\* newS - массив указателей на начало каждого стека в мультистеке

int GetFreeMemory() – возвращает число свободных элементов

void Repack(int k) – перепаковка (память k-ого стека увеличивается)

2) Элементы класса, объявленные со спецификатором public:

TMStack(int \_n, int \_size) - конструктор

TMStack(TMStack<T> &A) - конструктор копирования

int GetSize() - возвращает размер

T Get(int \_n) - возвращает значение из n-ого стека

void Set(int \_n, T p) - положить значение в \_n-ый стек

bool IsFull(int \_n) - проверка стека на полноту \_n-ого стека

bool IsEmpty(int \_n) - проверка стека на пустоту \_n-ого стека

~TMStack() - деструктор

## Описание алгоритмов

* Конструктор инициализации

Размер и количество стеков определяются пришедшими значениями. Создается массив (smth), элементы которого, кроме первого, равны целой части от size/ n; первый же элемент больше на остаток от деления на n. Затем мы задаем значения стеков мультистека через конструктор инициализатор. Для первого элемента: newS[0] = new TNewStack<T>(smth[0], &mas[0]); Для всех остальных: int temp = smth[0] + (i - 1)\*smth[i]; newS[i] = new TNewStack<T>(smth[i], &mas[temp]);

* Перепаковка k-ого стека

Запоминаем старые, новые индексы начала, а также новый размер каждого из стеков. Распределяем свободную память поровну между стеками, кроме k-ого, ему выделяется память, размер которой определяется выражением: sizeNewOne[k] += FreeForNow % n, где FreeForNow – число свободных позиций. Индекс начального элемента каждого стека рассчитывается следующим образом: индекс начала предыдущего стека + его новый размер. Не изменяется лишь начало первого стека. Затем выполняем проверку на выполнение одного из следующих условий: а) Если индекс нового начала i-го стека startNewOne[i] не больше, чем индекс старого начала i-го стека startOldOne [i], то копируем элементы по порядку, в котором они хранятся в старом стеке. б) Иначе идем по новым позициям стеков до тех пор, пока не выполняется пункт а). Затем копируем элементы, в котором они хранятся в старом стеке, но в обратном порядке.

# Заключение

В результате выполнения лабораторной работы была разработана библиотека, позволяющая работать с шаблонным классом мультистека. С помощью этой библиотеки можно создать объект класса мультистека и некоторые операции операции над ним.

Были реализованы тесты для проверки работоспособности класса мультистека на базе GoogleTest. А также приведен пример работы стека.

# Список литературы

1. Википедия. Статья «Стек»: [https://ru.wikipedia.org/wiki/Стек].
2. Васильев А.Н. Самоучитель С++ с примерами и задачами. -СПб.: Наука и Техника, 2016. -480с.
3. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.