МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования

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

**Национальный исследовательский университет**

**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

Голубева А. С.

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

аспирант каф. МОСТ ИИТММ

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

Нижний Новгород

2018.

**Содержание**

[1. Введение 3](#_Toc991355)

[2. Постановка задачи 4](#_Toc991356)

[3. Руководство пользователя 5](#_Toc991357)

[4. Руководство программиста 6](#_Toc991358)

[4.1 Описание структуры программы 6](#_Toc991359)

[4.2 Описание структур данных 6](#_Toc991360)

[4.3 Описание алгоритмов 7](#_Toc991361)

[5. Заключение 9](#_Toc991362)

[6. Литература 10](#_Toc991363)

1. **Введение**

Мультистек - структура данных, представляющая собой упорядоченный набор N стеков, фиксированного размера.

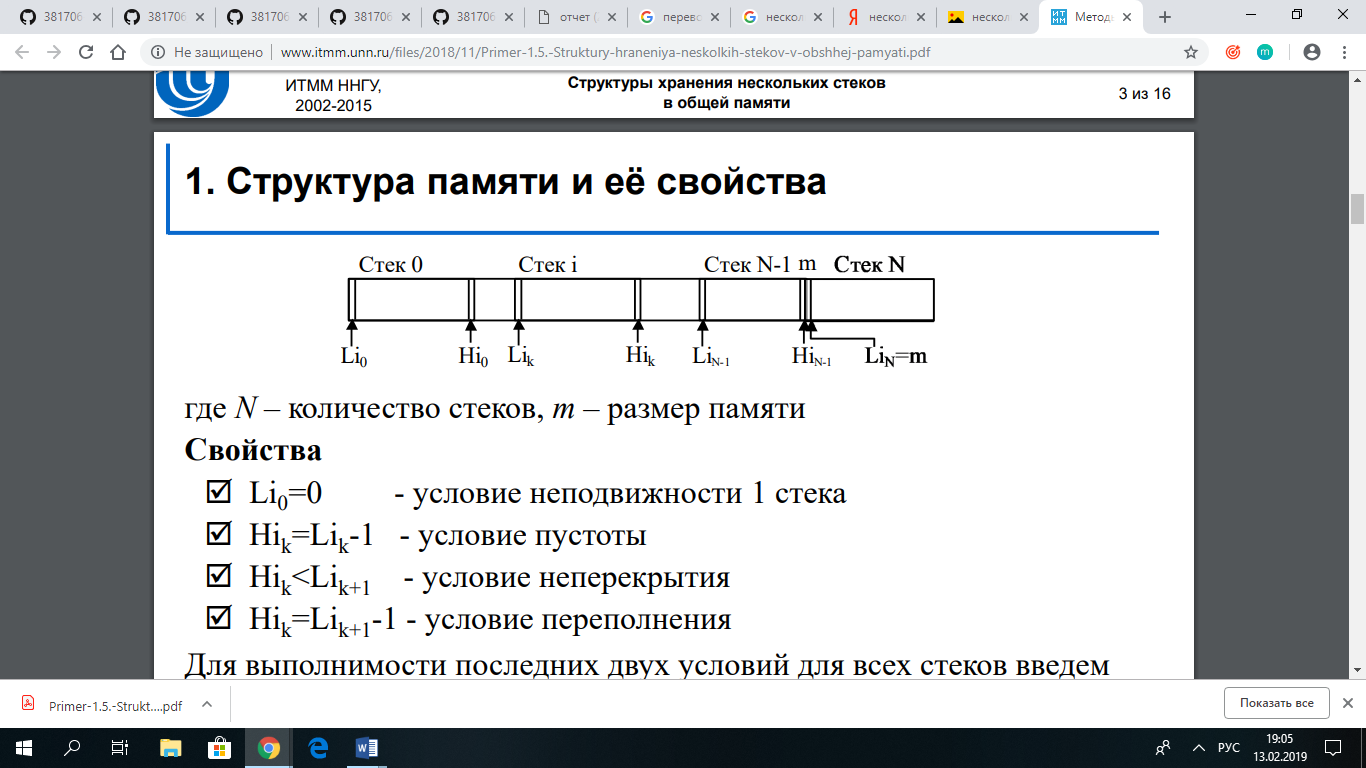


Рисунок 1. Структура хранения мультистека

На рисунке 1 N - количество стеков, m - размер памяти, - индекс начала j-го стека, - индекс последнего элемента j-го стека.

Целью данной лабораторной работы является реализация структуры хранения мультистека.

1. **Постановка задачи**

Выполнение работы предполагает решение следующих задач:

1. Реализация класса вспомогательного класса TNewStack.
2. Реализация класса мультистека TMStack.
3. Реализация класса TException для обработки исключений
4. Реализация тестов и обеспечение их работоспособности.
5. Пример использования и обеспечение его работоспособности.
6. **Руководство пользователя**

Рассмотрим пример работы программы.

Создаем мультистек и добавляем в него элементы. При очередной операции добавления элемента один из стеков переполняется и выполняется перепаковка мультистека. Затем на экран выводятся элементы этого стека.

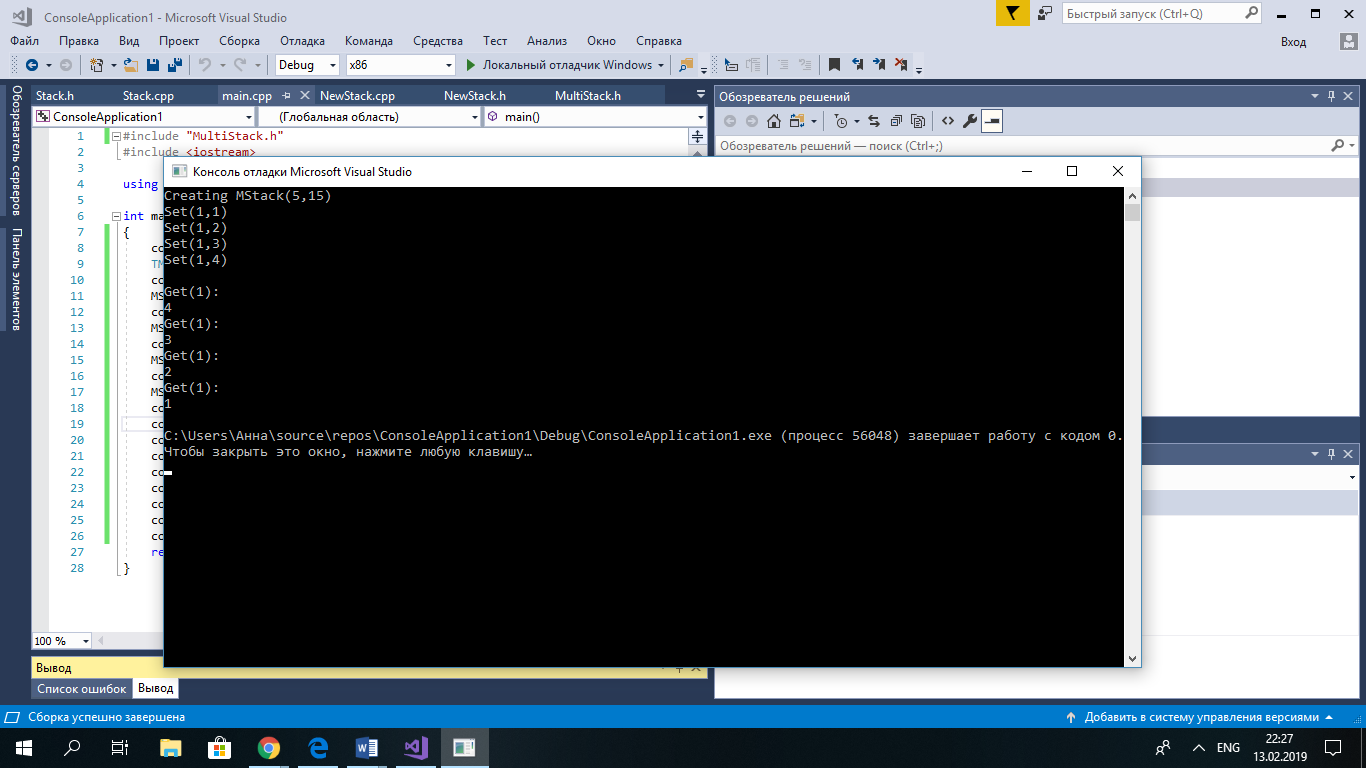


Рисунок 2. Пример работы программы

1. **Руководство программиста**
   1. **Описание структуры программы**

Программа состоит из следующих модулей:

* Модуль StackLib – содержит заголовочный файл Stack.h, в котором определен интерфейс класса TStack и реализованы его методы. (Класс TStack шаблонный), и файл Stack.cpp;
* Модуль MultiStack – содержит файл main.cpp, в котором реализован пример использования класса TMStack;
* Модуль MultiStackLib - содержит заголовочный файлы NewStack.h и MultiStack.h, в которых реализованы классы TNewStack и TMStack, также содержит файлы MultiStack.cpp и NewStack.cpp;
* Модуль MultiStackTest – содержит файл MultiStackTest.cpp, в котором реализованы тесты для класса TMStack;
* Модуль Exception – содержит заголовочный файл Exception.h, в котором реализован класс исключений, и файл Exception.cpp.
  1. **Описание структур данных**

**Класс TNewStack**

Класс TNewStack – шаблонный класс, является наследником класса TStack.

*Public:*

TNewStack(int \_size , T\* \_mas ) – конструктор.

TNewStack(TNewStack <T> & A) - конструктор копирования.

int GetFree() – метод, позволяющий получить количество свободных ячеек в стеке.

int GetSize() – метод, позволяющий получить размер стека.

int GetTop() – метод, позволяющий получить позицию первого элемента стека.

void SetMas (int \_size, T\* \_mas) – метод, позволяющий задать стек размера \_size, содержащий элементы \_mas.

void Put(T \_A) – метод, позволяющий положить элемент в стек.

T Get() –метод, позволяющий забрать элемент из стека.

**Класс TMStack**

Класс TMStack **-** шаблонный, наследуется от класса TNewStack.

*Protected:*

int size – размер мультистека.

T\* mas - массив элементов.

int n - количество стеков.

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

int GetFreeMem() – количество свободных элементов в мультистеке.

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

*Public:*

TMStack(int \_n, int \_size) – конструктор.

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

int GetSize() – метод, возвращающий размер мультистека.

T Get(int \_n) – забрать элемент из n-ого стека.

void Set(int \_n, T \_elem) - положить значение в \_n-й стек.

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

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

* 1. **Описание алгоритмов**

**Создание мультистека**

Присваиваем соответствующим полям количество стеков и их размер. Создаём массив объектов класса TNewStackразмера равного количеству стеков и массив элементов мультистека. Далее создаем массив A, элементы которого равны целой части от size/ n, а элемент A[0] += (size % n) + int(size/n). Затем для каждого стека определяем указатель на его первый элемент: для первого стека h[0] = new TNewStack(A[0], &mas[0]), для остальных стеков h[i] = new TNewStack<T>(A[i], &mas[A[0]] + ((i - 1)\*A[i])).

**Перепаковка мультистека**

Перепаковка выполняется, когда требуется добавить элемент в один из стеков, а он переполнен. При этом в других стеках ещё есть свободные ячейки.

Перепаковка k-го стека:

Находим количество свободных ячеек, которые можно добавить в каждый стек add\_ev находим путем деления количества свободных ячеек во всем мультистеке freeMem на количество стеков n.Затем увеличиваем старые размеры стеков на add\_ev, получаем новые размеры стеков new\_size[i].Если количество свободных ячеек во всем мультистеке freeMem не кратно количеству стеков n, то оставшиеся свободные ячейки добавляем в k-й стек. Определяем новый индекс начала каждого стека. Индекс начального элемента каждого стека находим следующим образом: к индексу начала предыдущего стека добавляем его новый размер. Начало первого стека не изменяется. Далее:

а) Если индекс нового начала i-го стека new\_start[i] не больше, чем индекс старого начала i-го стека old\_start[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.

1. **Заключение**

В результате лабораторной работы реализована структура хранения мультистека, также были реализованы операции для работы с мультистеком.

Успешно выполнены тесты, проверяющие работоспособность методов класса TMultiStack. Приведен пример, демонстрирующий работу с мультистеком.

1. **Литература**
   1. Википедия: свободная электронная энциклопедия: на русском языке: URL: https://ru.wikipedia.org/wiki/Стек

(Дата обращения: 13.02.2019)

* 1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015. URL: http://www.itmm.unn.ru/files/2018/11/Primer-1.5.-Struktury-hraneniya-neskolkih-stekov-v-obshhej-pamyati.pdf