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 ЛАБОРАТОРНОЙ РАБОТЕ**

**«Структура хранения данных: Мультистек»**

**Выполнил:** студент группы 381706-2

Мышкин Андрей Александрович

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Руководитель:**

ассистент кафедры МОСТ ИИТММ,

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018

**Содержание**

1. Введение………………………………………………………………………...3

2. Постановка задачи……………………………………………………...………5

3. Руководство пользователя…………………………………………...………...6

4. Руководство программиста…………………………………………...………..7

4.1. Описание структур данных…..……………………………………….7

4.2. Описание алгоритмов………………………………………………….8

4.3. Описание структуры программы….……………………………..…...8

6. Заключение……………………………………………………………………...9

7. Литература……………….……………………………………………………10

**Введение**

Стек (англ. *stack*— стопка; читается *стэк*) — абстрактный тип данных, представляющий собой список элементов, организованных по принципу *LIFO* (*last in — first out*, «последним пришёл — первым вышел»).

Чаще всего принцип работы стека сравнивают со стопкой тарелок: чтобы взять вторую сверху, нужно снять верхнюю.

В цифровом вычислительном комплексе стек называется магазином — по аналогии с магазином в огнестрельном оружии (стрельба начнётся с патрона, заряженного последним).

В 1946 Алан Тьюринг ввёл понятие стека. А в 1957 году немцы Клаус Самельсон и Фридрих Л. Бауэр запатентовали идею Тьюринга.

В некоторых языках стеком можно назвать любой список, так как для них доступны операции pop и push. В языке C++ стандартная библиотека имеет класс с реализованной структурой и методами.

[](https://commons.wikimedia.org/wiki/File:Lifo_stack.png?uselang=ru)

1.Организация стека в виде одномерного упорядоченного по адресам массива. Показаны операции вталкивания и выталкивания данных из стека операциями *push* и *pop*.

Зачастую стек реализуется в виде однонаправленного списка (каждый элемент в списке содержит помимо хранимой информации в стеке указатель на следующий элемент стека).

При организации стека в виде однонаправленного списка значением переменной стека является указатель на его вершину — адрес вершины. Если стек пуст, то значение указателя равно NULL.

Но также часто стек располагается в одномерном массиве с упорядоченными адресами. Такая организация стека удобна, если элемент информации занимает в памяти фиксированное количество слов, например, 1 слово. При этом отпадает необходимость хранения в элементе стека явного указателя на следующий элемент стека, что экономит память. При этом указатель стека (*Stack Pointer*, — **SP**) обычно является регистром процессора и указывает на адрес головы стека.

В данной лабораторной работе будет реализация мультистека.

Мультистек – это такая структура данных, которая состоит из нескольких упорядоченных N стеков, фиксированного размера и объединенных между собой. С помощью такой программы появляется возможность работы сразу с несколькими стеками, что может быть необходимо в некоторых разработках.

**Постановка задачи**

Основная задача данной работы – это разработка и реализация структуры хранения данных такой, как класс Мультистек, используя уже созданный класс Стек.

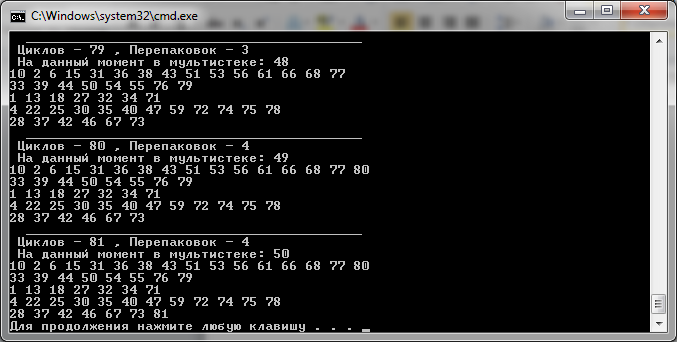
В данной ситуации класс Стек выступает как вспомогательный. Будет создан класс наследник от Стека, который будет являться как единица, то есть один из нескольких стеков класса Мультистек.

В классе стек должны быть реализованы следующие операции:

* Проверка пустоты мультистека
* Проверка полноты мультистека
* Добавление нового элемента
* Удаление элемента

**Руководство пользователя**

С помощью данной программы пользователь может проверить сам, как работает такая структура хранения данных, как мультистек. При открытии программа сообщает о проведении тестирования на работу мультистека. После чего программа автоматически начнет заполнять 5 стеков случайно сгенерированными значениями, до тех пор, пока их число не станет равное 50. После этого программа остановит свое выполнение, а на консоли во время всего этого процесса будет выводиться количество пройденных циклов, сделанных перепаковок и количество элементов, уже находящихся в мультистеке. Для упрощения процесса тестирование проходит на целочисленных данных.



**Руководство программиста**

**Описание структур данных.**

Класс MStack

Является классом наследником от класса Stack и в свою очередь имеет перегруженный конструктор и деструктор и вдобавок один метод.

MStack(ValType\* ind = NULL, int Size = 0) - конструктор

~MStack() - деструктор

void SetMem(ValType\* ind, int Size) – метод класса, управляющий выделяемой памятью под стек

Класс TMultiStack

Поля:

MStack <ValType> \*\* pStack – массив используемых стеков

int MemSize – общее число выделенной памяти

int CurrentCount – общее число элементов, находящихся в мультистеке в настоящее время

int StackCount – число, описывающее количество стеков

ValType\* StackMem – память, выделенная под один стек

Методы:

TMultiStack(int Size = 50, int Count = 5) – конструктор

TMultiStack(TMultiStack &ms) - конструктор копирования

~TMultiStack() – деструктор

int IsEmpty(int ns) const – метод класса, отвечающий за проверку и контроль пустоты

int IsFull(int ns) const; - метод класса, отвечающий за проверку и контроль перезаполнения

int operator==(const TMultiStack &ms) – метод класса, являющийся оператором сравнения

TMultiStack & operator= (const TMultiStack &ms) – метод класса, являющийся оператором присваивания

void Put(int ns, const ValType &Val) – метод класса, отвечающий за добавления элемента в стек

ValType Get(int ns) – метод класса, отвечающий за взятия элемента из стека и последующее его удаление

void Print() – метода класса, производящий вывод мультистека на консоль

**Описание алгоритмов.**

Выполнение программой такого алгоритма как перепаковка. Перепаковка – перераспределение памяти между стеками.

Данный метод принимает на вход целочисленный параметр, которым является номер переполненного стека

Происходит создание переменных для циклов, делается перерасчет свободной памяти

В случае нахождения свободной памяти, указатели смещаются на начало стеков с учетом уже занятой и незанятой памяти, разделенной на все стеки, нужному стеку присваивается остаток от деления

Происходит смещение памяти, занятой в стеках, в этом случае смещение происходит с конца, чтобы не было «утечек»

Затем делается перераспределение памяти между всеми стеками

Конечный результат перераспределения возвращается

**Описание структуры программы.**

MultiStack – модуль для тестирования выполнения программы структуры хранения данных: мультистек, главный файл которого main.cpp

MultiStackLib – модуль реализации класса стек. В него входит заголовочный файл - TMultiStack.h и пустой файл TMultiStack.cpp

MultiStackTest – тесты класса стек, используя Google C++ Testing Framework, основной файл multistack-text.cpp и файл производящий запуск тестов test\_main.cpp

**Заключение**

В ходе проведение данной лабораторной работы была создана и протестирована такая структура хранения данных как мультистек. В классе мультистек были реализованы такие операции как: проверка полноты и переполнения мультистека, добавление нового элемента, удаление элемента стека. Сам мультистек был реализован при помощи использования существующего класса стек.

Также были освоены инструменты разработки программного обеспечения, как система контроля версий Git и фрэймворк для разработки автоматических тестов Google Test.

**Литература**

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015.
2. Википедия: свободная электронная энциклопедия ‒ Стек: <https://ru.wikipedia.org/wiki/Стек>