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**

**«»**

**Выполнил:** студент группы 381706-1

Суслов Егор Игоревич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Научный руководитель:**

ассистент каф. МОСТ ИИТММ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Лебедев И.Г

Нижний Новгород

2019.

**Содержание**

[1. Введение 3](#_Toc533083472)

[2. Постановка задачи 4](#_Toc533083473)

[3. Руководство пользователя 5](#_Toc533083474)

[4. Руководство программиста 6](#_Toc533083475)

[4.1. Описание структуры программы 6](#_Toc533083476)

[4.2. Описание структур данных 6](#_Toc533083477)

[4.3. Описание алгоритмов 7](#_Toc533083478)

[5. Заключение 8](#_Toc533083479)

[6. Литература 9](#_Toc533083480)

1. **Введение**

Таблицы являются одними из наиболее распространенных структур данных, используемых при создании системного и прикладного математического обеспечения. Таблицы широко применяются в трансляторах (таблицы идентификаторов) и операционных системах, могут рассматриваться как программная реализация схемы ассоциативной памяти и т.п.

Под таблицей следует понимать динамическую структуру данных, которая в каждый момент выполнения вычислений состоит из конечного набора элементов (записей); записи таблицы могут подразделяться на несколько полей; при этом количество и тип полей является одинаковыми для всех записей таблицы. Первое поле всех записей таблицы обычно называют ключом, поля записи без ключевого поля образуют тело записи.

Способ построения таблицы должен приводить к быстрому выполнению табличных операций. В зависимости от способов размещения записей в таблице возможно использование различных методов выполнения операций обработки таблиц. Эффективность применяемых способов организации может быть оценена, например, в количестве просматриваемых записей таблицы при выполнении операций поиска, вставки и удаления.

Простейшим способом отыскания нужного элемента является метод полного просмотра (сканирования), когда искомый ключ сравнивается по очереди со всеми ключами таблицы, начиная с первого, вплоть до отыскания совпадающего элемента или до исчерпания записей. Если ключи в таблице расположены в произвольном порядке (неупорядоченная таблица), этот способ является единственно возможным.

Операция вставки в неупорядоченную таблицу может быть выполнена путем добавления новой записи в конец таблицы с корректировкой номера последней занятой строки. Операция удаления строки может быть реализована при помощи переписывания последней записи таблицы на место удаляемой и соответствующей корректировки номера последней строки.

1. **Постановка задачи**

В рамках лабораторной работы ставится задача эффективной реализации структуры данных – просмотровая таблица.

Для работы со просмотровой таблицей необходимо реализовать операции:

* добавления элемента в мультистек,
* извлечения с элемента из мультистека,
* проверка мультистека на полноту/пустоту.

Программное решение будет выглядеть следующим образом:

1. Разработка и реализация вспомогательного класса стека – TNewStack.
2. Разработка и реализация класса мультистека – TMultiStack.
3. Подключение класса для обработки исключений – MyException.
4. Разработка программы, демонстрирующей работу классa TMultiStack (получается при выполнении файла кода «MultiStack\_main.cpp».
5. Реализация набора автоматических тестов с использованием Google C++ Testing Framework.
6. **Руководство пользователя**
7. Создается мультистек состоящий из двух стеков в каждом из которых по 6 элементов.
8. Мультистек с введенными числами выводится на экран
9. Далее берется по 2 элемента из каждого стека
10. Затем добавляется «5» во второй стек
11. Вывод мультистека
12. Закрытие программы
13. **Руководство программиста**
    1. **Описание структуры программы**

Программа состоит из следующих модулей:

* Модуль MultiStack. Содержит пример использования мультистека. Реализация в файле «MultiStack\_main.cpp*».*
* Модуль MultiStackLib – статическая библиотека. Содержит файл MultiStack.h, в котором описан интерфейс и реализация шаблонного класса TMultiStack(2 конструктора и 8 методов), и «TNewStack.h», в котором описан интерфейс и реализация шаблонного класса TNewStack(2 конструктора и 7 методов)
* Модуль MultiStackTest. Содержит 27 тестов, описанных в файле «MultiStackTest.cpp» и разработанных с помощью использования Google C++ Testing Framework.
* Модуль ExceptionLib – библиотека, содержащая класс исключений.
  1. **Описание структур данных**

**Класс TNewStack**

Класс TNewStack является шаблонным классом. Он наследуется от класса TStack со спецификатором public. Внутри класса определены 2 конструктора и 7 методов со спецификатором доступа public:

* TNewStack(int \_size, T\* \_mas) - конструктор по умолчанию.
* TNewStack(TNewStack<T>& A) – конструктор копирования.
* int GetFreeMem() – получить количество свободных ячеек в стеке.
* T Get() – забрать элемент из стека.
* void Push(T \_A) – положить элемент в стек.
* int GetSize() – получить размер стека.
* int GetTop() - получить позицию вершины стека.
* void SetMas(int \_size, T\* \_mas) – преобразование массива mas размером size в стек.
* void PrintNewStack() – вывод стека на консоль.

**Класс TMultiStack**

Класс TMultiStack является шаблонным классом. Внутри класса определены 4 поля и 2 метода со спецификатором доступа protected:

* int size – размер мультистека.
* T\* mas – указатель на область память под хранение мультистека.
* int n – количество стеков в мультистеке.
* TNewStack<T>\*\* stackMas – массив указателей на начало каждого стека в мультистеке.
* int CountFree() – количество свободных элементов в мультистеке.
* void Repack(int \_n) – перепаковка стека, с увеличением свободной памяти в n-м стеке.

В публичной зоне описаны 2 конструктора 6 методов:

* TMStack(int \_n = 1, int \_size = 10) - конструктор по умолчанию.
* TMStack(TMStack &A) - конструктор копирования.
* int GetSize() – возвращает размер мультистека.
* void Set(int \_n, T \_elem); - положить в n-й стек элемент elem.
* T Get(int \_n) – взять элемент из n-го стека.
* bool IsFull(int \_n) – проверка на полноту n-го стека.
* bool IsEmpty(int \_n) – проверка на пустоту n-го стека.
* void PrintMStack() – вывод мультистека на консоль.
  1. **Описание алгоритмов**

**Добавление звена списка в начало и в конец.**

Сначала проверяем не заполнен ли список. Если он заполнен, то бросаем исключение. Если нет, то в очереди свободных позиций freeElem, берем первую свободную ячейку i. По полученному индексу в массив mas записываем значение, которое хотим положить в список. Определяем, что следующим для этого элемента, будет элемент с текущим индексом start, то есть nextInd[i] = start. Если, перед добавлением, список не был пуст, то предыдущим для первого элемента списка делаем только что добавленный элемент, то есть: predInd[start] = i. Если же список был пуст, то определяем, что добавленный элемент является и последним элементом в списке. Затем индекс start переопределяем на только что добавленный элемент: start = i. Увеличиваем количество элементов в списке count++.

Для добавления элемента в конец списка, рассуждения аналогичны, с поправкой на то, что добавляем в конец списка.

1. **Заключение**

Все поставленные задачи были выполнены это подтверждается Google C++ Testing Framework (все 26 тестов, 17 на класс TMultiStack и 9 на вспомогательный класс TNewStack, успешно выполняются) и работой с файлом кода «MultiStack \_main.cpp» использующего наш класс.