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**

оставь надежды всяк сие читающий (карин т.а.)

**Отчет по учебной практике**

**«Структура хранения стека на списках»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc1348912)

[3. Руководство пользователя 5](#_Toc1348913)

[4. Руководство программиста 6](#_Toc1348914)

[4.1. Описание структуры программы 6](#_Toc1348915)

[4.2 Описание структур данных 6](#_Toc1348916)

[4.3 Описание алгоритмов 6](#_Toc1348917)

[5. Заключение 8](#_Toc1348918)

[6. Список литературы 9](#_Toc1348919)

1. Введение

Стек имеет большое значение не только в жизни программистов, но и в жизни повседневной. Самым простым стеком в жизни является стопка книг. Но между стеками на массивах и стеках в жизни есть небольшое различие. А именно: если на большую стопку книг положить ещё одну, то стопка вряд ли упадёт, а если в полный стек положить ещё один элемент – упадёт всё.

Как же выйти из этой ситуации (мы же как-никак ООП изучаем: стопку книг нужно уметь представлять так, чтобы ничего не падало)? Можно вместо обычного стека на массиве использовать мультистек. Но он удобен, когда стопок несколько, а количество книг опять же не выходит за границы массива.

Остаётся вариант, когда необходимо принципиально отказаться от идеи брать в качестве основы массив. Нужно найти такую структуру данных, размер которой увеличивается по мере надобности. Такая структура есть, и она даже нами написана. Называется списком.

Итак, в этой лабораторной работе будет рассмотрен, разработан и описан стек на списках. Размер такого стека хоть и увеличивается, но не может превысить определённого значения, которое зависит от размера общей выделенной памяти. Но мы будем этим пренебрегать и говорить, что он неограниченный.

2. Постановка задачи

Целью данной лабораторной работы является создание библиотеки для работы со стеком неограниченного размера. Для этого необходимо:

* Создать класс, объектами которого будут стеки на списках;
* Написать пример использования программы, в котором пользователь может создавать стек, получать и добавлять элементы; (до тех пор, пока «enter» не сломается)
* Проверить работоспособность библиотеки.

3. Руководство пользователя

Файл *StOnList.cpp,* представляет из себя программу, в которой пользователь может создать свой стек, состоящий из целочисленных значений, добавлять и извлекать из него элементы, делать проверки на пустоту, а также выводить элементы, содержащиеся в стеке на экран.

После создания стека появится меню действий, описанных выше. Пользователю нужно ввести номер команды:

1. Положить число в стек;
2. Получить число из стека;
3. Проверить стек на пустоту;
4. Вывести элементы стека на экран;
5. Выйти из программы.

Для завершения работы программы необходимо выбрать соответствующий пункт меню.

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из трёх модулей:

* *StOnListLib* – библиотека для работы со стеком.
* *StOnList* – пример использования библиотеки *StOnListLib*.
* *StOnListTest* – набор тестов для проверки работоспособности библиотеки *StackLib.*

Также в программе используется класс исключений *TExeption*, содержащийся в файле *Exeption.h* проекта *Exeption*.

4.2 Описание структур данных

Расположен в файле *Stack.h* проекта *StackLib*. Является шаблонным, T – шаблонный тип данных, присваиваемый элементам стека. Класс включает в себя:

Поля со спецификатором доступа protected:

* *Int count* – количество элементов в стеке.
* TList<T>\* *mas –* список элементов стека;

Поля со спецификатором доступа public:

* *TStackList<T>() –* конструктор;
* *TStackList<T>(TStackList<T> &A) –* конструктор копирования;
* *void Put(T A) –* функция добавления элемента в стек;
* *T Get() –* функция взятия элемента из стека;
* *int GetSize() –* возвращает размер стека (количество элементов);
* *bool IsEmpty() –* проверка на пустоту.

4.3 Описание алгоритмов

Добавление элементов в стек

Вызываем метод добавления в начало списка *mas.* Увеличиваем количество элементов на единицу.

Взятие элемента из стека

Проверяем стек на пустоту. Если стек пуст, то выбрасываем исключение. Если в стеке есть элементы, то вызываем метод взятия из начала для списка *mas.* Уменьшаем количество элементов в стеке на единицу и возвращаем значение, полученное из списка *mas.*

5. Заключение

В ходе данной лабораторной работы были получены следующие результаты:

* Создана библиотека, позволяющая работать со стеком на списках;
* Написан пример использования библиотеки;
* Обработаны основные исключительные ситуации;
* Написаны тесты для проверки работоспособности библиотеки.

6. Список литературы

1. Ахо Альфред В, Хопкрофт Джон Э и Ульман Джеффри Д Структуры данных и алгоритмы [Книга]. - [б.м.] : Вильямс, 2003.
2. Лафоре Роберт Структуры данных и алгоритмы в Java [Книга]. - СПб : Питер, 2013. - 2 : стр. 704.
3. Павловская Т. А. C/C++ Программирование на языке высокого уровня [Книга]. - СПб : Питер, 2003.
4. Страуструп Бьерн Язык программирования C++ Бином, 2004.