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»**

**Выполнил:**

студент группы 381706­1

Карин Тимофей Андреевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

**Проверил:**

ассистент каф. МОСТ ИИТММ

Лебедев Илья Геннадьевич

\_\_\_\_\_\_\_\_\_\_\_ Подпись

Нижний Новгород

2018.

Содержание

[1. Введение 3](#_Toc534459684)

[2. Постановка задачи 4](#_Toc534459685)

[3. Руководство пользователя 5](#_Toc534459686)

[4. Руководство программиста 6](#_Toc534459687)

[4.1. Описание структуры программы 6](#_Toc534459688)

[4.2 Описание структур данных 6](#_Toc534459689)

[4.3 Описание алгоритмов 7](#_Toc534459690)

[5. Заключение 9](#_Toc534459691)

[6. Список литературы 10](#_Toc534459692)

1. Введение

Структура данных — программная единица, позволяющая хранить и обрабатывать множество однотипных и/или логически связанных [данных](https://ru.wikipedia.org/wiki/%D0%94%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5_(%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%82%D0%B5%D1%85%D0%BD%D0%B8%D0%BA%D0%B0)) в [вычислительной технике](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80). Для добавления, поиска, изменения и удаления данных структура данных предоставляет некоторый набор функций, составляющих её интерфейс.

В данной лабораторной работе будет рассмотрена такая структура данных как стек. Он характерен тем, что в нём доступен только один элемент данных: тот, который был в него вставлен последним. Удалив этот элемент, пользователь получает доступ к предпоследнему элементу и т. д. Такой механизм доступа удобен во многих ситуациях, связанных с программированием. Например, многие микропроцессоры имеют стековую архитектуру. При вызове метода адрес возврата и аргументы заносятся в стек, а при выходе они извлекаются из стека. Операции со стеком встроены в микропроцессор. Стековая архитектура также использовалась в некоторых старых калькуляторах. Вместо того чтобы вводить арифметическое выражение с круглыми скобками, пользователь сохранял промежуточные результаты в стеке.

Зачастую стек реализуется в виде однонаправленного списка (каждый элемент в списке содержит помимо хранимой информации в стеке указатель на следующий элемент стека). Но также часто стек располагается в [одномерном массиве](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%81%D1%81%D0%B8%D0%B2_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) с упорядоченными адресами. В данной работе мы будем рассматривать стек на массиве.

2. Постановка задачи

Целью данной работы является создание библиотеки, предназначенной для хранения и обработки структуры хранения стек. Для этого необходимо:

1. Создать класс для работы со стеком;
2. Реализовать методы:
   * конструктор по умолчанию, конструктор копирования, деструктор;
   * добавления элемента в стек;
   * получения элемента из стека;
   * проверки на пустоту;
   * проверки на полноту;
3. Обработать исключительные ситуации.

3. Руководство пользователя

Библиотека *StackLib* предназначена для программистов и не предусматривает работу с пользователями. По этой причине был создан проект *Stack,* содержащий один из вариантов работы с *StackLib*.

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

После запуска программы пользователю необходимо ввести количество элементов в создаваемом стеке.

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 полноту;
4. Проверить стек на пустоту;
5. Вывести элементы стека на экран;
6. Выйти из программы.

Если в ходе выполнения программы произойдёт исключительная ситуация, на экран будет выведено сообщение об ошибке:

1. Отрицательный размер стека (происходит, когда пользователь пытается создать стек с отрицательным количеством элементов);
2. Полный стек (выбрасывается, когда пользователь пытается положить элемент в уже заполненный стек);
3. Пустой стек (выбрасывается, когда пользователь пытается забрать элемент из пустого стека).

Для завершения работы программы необходимо выбрать соответствующий пункт меню.

4. Руководство программиста

4.1. Описание структуры программы

Программа состоит из трёх проектов:

* *StackLib* – библиотека для работы со стеком. Состоит из файла TStack.h
* *Stack* – пример использования библиотеки *StackLib*. Состоит из файла Stack;
* *StackTest* – 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

**Класс *TStack***

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 *size* – максимальное количество элементов в стеке;
* int *top* – «вершина» стека. Первый пустой элемент в массиве *mas*.
* T\* *mas –* массив элементов стека;

Поля со спецификатором доступа public:

* TStack<T> (int n=0) – конструктор;
* TStack<T> (TStack<T> &A) – конструктор копирования;
* virtual ~TStack<T>() – деструктор;
* void Put (T A) – функция добавления элемента в стек;
* T Get() – функция удаления элемента из стека;
* int GetSize() – функция, возвращающая максимальное число элементов в стеке (размер массива *mas*, то есть значение поля *size*)
* bool IsFull() – функция проверки стека на полноту;
* bool IsEmpty() – функция проверки стека на пустоту.

4.3 Описание алгоритмов

* Добавление элемента в стек

Прежде всего проверяем стек на полноту. Если стек уже заполнен, и места для нового элемента нет, то выбрасывается исключение.

Если же место ещё есть, то в ячейку массива mas с номером *top* помещаем нужный элемент. Затем увеличиваем значение поля *top* на 1.

Таким образом, элемент помещён в массив *mas* в первую свободную ячейку. А *top* всё так же хранит номер первой пустой ячейки массива *mas* (или же равен размеру *mas*, в случае, когда в массиве не осталось свободных ячеек).

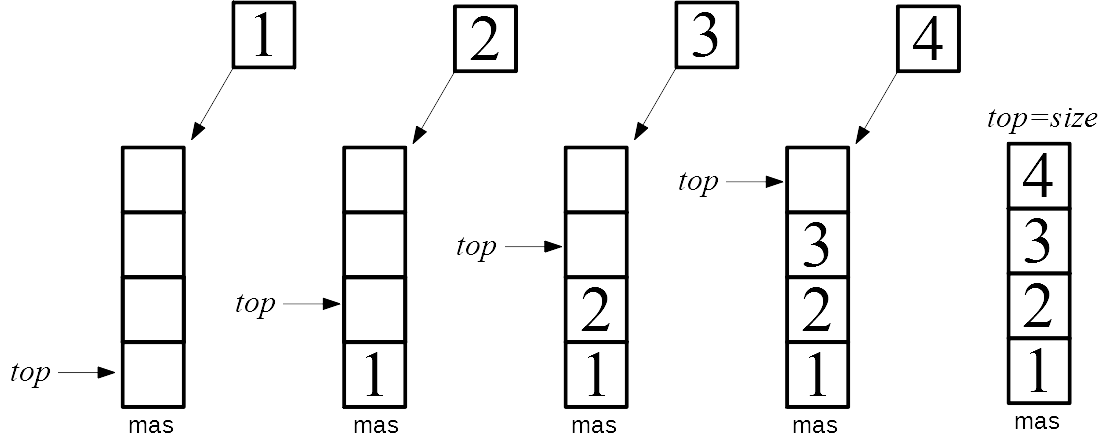
Пример заполнения стека элементами представлен на рисунке 1.

Рисунок 1. Добавление элементов в стек:

* Получение элемента из стека

Проверяем стек на пустоту. Если в стеке нет элементов, то вызываем исключение.

Если элементы в массиве *mas* есть, то уменьшаем значение поля *top* на 1 (теперь *top* хранит номер последней непустой ячейки массива *mas*).

Возвращаем значение элемента, хранящегося в ячейке под номером *top* (то есть элемент, который был добавлен позднее остальных).

Таким образом, был возвращён и удалён из стека элемент, который был добавлен в стек позднее оставшихся. Поле *top* по-прежнему хранит номер первой пустой ячейки массива *mas*.

Пример извлечения элементов из стека представлен на рисунке 2.

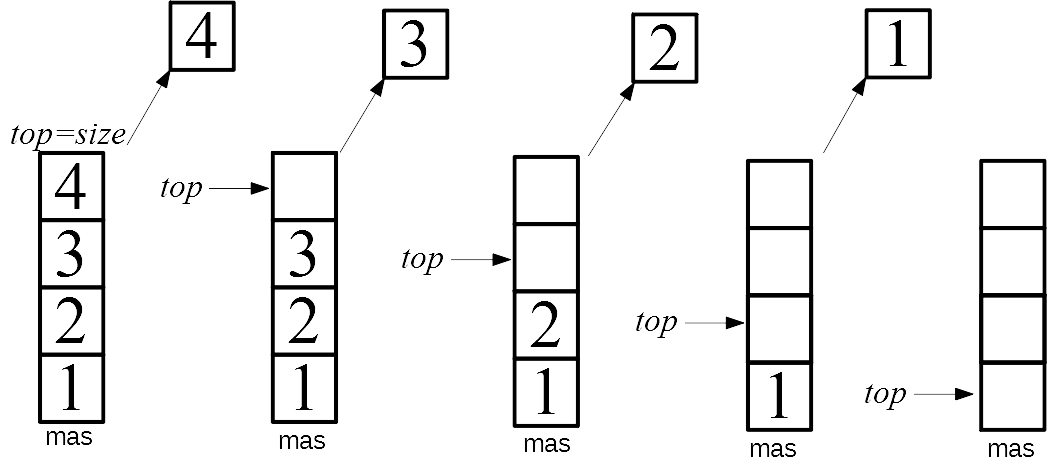


Рисунок 2. Удаление элементов из стека

5. Заключение

В ходе работы были достигнуты следующие результаты:

* Создан класс *TStack*, предназначенный для работы со стеком;
* Обработаны следующие исключительные ситуации: создание стека с некорректным размером, переполнение стека, получение несуществующего элемента;
* Написана программа, демонстрирующая один из вариантов использования стека.
* Проверены все методы класса *TStack* с помощью Google Test Framework.

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. Стек - Википедия [В Интернете] // Wikipedia. - Wikimedia Foundation, Inc, 2 ноябрь 2018 г.. - 23 декабрь 2018 г.. - https://ru.wikipedia.org/wiki/Стек.
5. Страуструп Бьерн Язык программирования C++ [Книга]. - [б.м.] : Бином, 2004.
6. Структура данных - Википедия [В Интернете] // Wikipedia. - Wikimedia Foundation, Inc, 11 декабрь 2018 г.. - 23 декабрь 2018 г.. - https://ru.wikipedia.org/wiki/Структура\_данных.