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: Стек»**

**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…………………………………………...………..8

4.1. Описание структур данных…..……………………………………….8

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

4.3. Описание структуры программы….……………………………..…...9

6. Заключение…………………………………………………………………….10

7. Литература……………….……………………………………………………11

**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.

**Операции со стеком**

Возможны три операции со стеком: добавление элемента (иначе проталкивание, *push*), удаление элемента (*pop*) и чтение головного элемента (*peek*).

При проталкивании (*push*) добавляется новый элемент, указывающий на элемент, бывший до этого головой. Новый элемент теперь становится головным.

При удалении элемента (*pop*) убирается первый, а головным становится тот, на который был указатель у этого объекта (следующий элемент). При этом значение убранного элемента возвращается.

Программный вид стека используется для обхода структур данных, например, дерево или граф. При использовании рекурсивных функций также будет применяться стек, но аппаратный его вид. Кроме этих назначений, стек используется для организации стековой машины, реализующей вычисления в обратной инверсной записи.

Применение стека упрощает и ускоряет работу программы, так как идет обращение к нескольким данным по одному адресу.

**Постановка задачи**

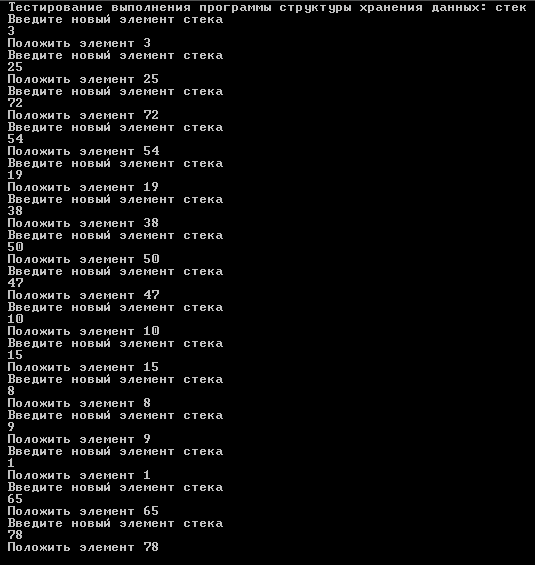
Основная задача – это реализовать структуру хранения данных такой, как класс стек, используя динамический список.

В классе стек должны быть реализованы следующие операции:

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

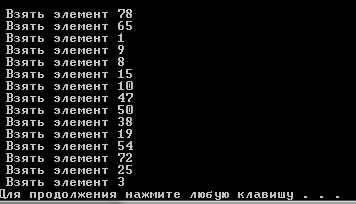
**Руководство пользователя**

С помощью данной программы пользователь может проверить сам, как работает такая структура хранения данных, как стек. При открытии программа сообщает о проведении тестирования на работу стека. После чего пользователю предоставляется возможность заполнить данный стек своими значениями. Для упрощения процесса тестирование проходит на целочисленных данных.



(2)

После заполнения стека элементами программа автоматически производит удаление элементов из стека, при котором происходит одновременное чтение головных элементов стека по всем правилам.



(3)

После всего этого на экране полностью представлена поэтапная работа стека.

**Руководство программиста**

**Описание структур данных.**

Класс TStack.

Поля класса:

ValType \* mem – память для элементов в стеке

int t – индекс последнего занятого в mem

int memSize – размер всей памяти, выделенной под стек

Методы класса:

TStack(int Size = 0) – конструктор

~TStack() – деструктор

int IsEmpty() – метод класса, отвечающий за проверку пустоты

int GetSize() – метод класса, выдающий размер памяти, занимаемой стеком

int GetCount() - метод класса, выходом которого является количество элементов в стеке

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

void Put(const int Val) – метод класса, отвечающий за добавление нового элемента в стек

ValType Get() – метод класса, который удаляет элементы из стека, параллельно выводя их на экран

void Print() – метод класса, который печатает значения из стека

**Описание алгоритмов.**

template<class ValType>

TStack<ValType>::Put(const int Val)

if (IsFull())

throw TExeption(DataFull)

mem[++t] = Val

template<class ValType>

ValType TStack<ValType>::Get()

if (IsEmpty())

throw TExeption(DataEmpty)

return mem[t--]

**Описание структуры программы.**

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

StackLib – модуль реализации класса стек В него входит заголовочный файл - TStack.h

StackTest – тесты класса стек, используя Google C++ Testing Framework, основной файл stack-text.cpp

**Заключение**

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

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

**Литература**

1. <https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA>
2. Гергель В.П. Методические материалы по курсу «Методы программирования 2», Нижний Новгород, 2015