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 на списке»**

**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….……………………………..…...8

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)

Рисунок 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.

Связный список — базовая динамическая структура данных в информатике, состоящая из узлов, каждый из которых содержит как собственно данные, так и одну или две ссылки («связки») на следующий и/или предыдущий узел списка. Принципиальным преимуществом перед массивом является структурная гибкость: порядок элементов связного списка может не совпадать с порядком расположения элементов данных в памяти компьютера, а порядок обхода списка всегда явно задаётся его внутренними связями.

Достоинства:

* эффективное (за константное время) добавление и удаление элементов
* размер ограничен только объёмом памяти [компьютера](https://ru.wikipedia.org/wiki/%D0%AD%D0%92%D0%9C) и разрядностью указателей
* динамическое добавление и удаление элементов

Недостатки:

* сложность прямого доступа к элементу, а именно определения физического адреса по его индексу (порядковому номеру) в списке
* на поля-указатели (указатели на следующий и предыдущий элемент) расходуется дополнительная память (в массивах, например, указатели не нужны)
* некоторые операции со списками медленнее, чем с массивами, так как к произвольному элементу списка можно обратиться, только пройдя все предшествующие ему элементы
* соседние элементы списка могут быть распределены в памяти не локально, что снизит эффективность кэширования данных в процессоре
* над связными списками, по сравнению с массивами, гораздо труднее (хоть и возможно) производить параллельные векторные операции, такие, как вычисление суммы: накладные расходы на перебор элементов снижают эффективность распараллеливания

https://upload.wikimedia.org/wikipedia/commons/thumb/3/37/Singly_linked_list.png/220px-Singly_linked_list.png

Рисунок 2. Разновидность связного списка — односвязный список, содержащий 3 элемента.

**Постановка задачи**

Основная задача данной работы – создание и реализация такой структуры данных, как стек на списке. Включая обязательную реализацию следующих операций:

Добавление значения в стек

Извлечение значения из стека

Проверка стека на пустоту

Вывод стека на консоль в формате поэтапного извлечения элементов из стека

**Руководство пользователя**

При запуске данной программы будет произведено тестирование работы структуры хранения данных: стек на списке. Для проведения тестирования пользователю необходимо указать размер стека, который будет использоваться. После ввода размера стек автоматически станет заполняться случайно сгенерированными числами от 0 до 150.Для упрощения проводимых тестов, в стеке используется целочисленный тип данных в небольшом диапазоне. После заполнения данного стека значениями, будет произведено поэтапное извлечение всех значений из стека, которое будет выводиться на консоль для большего понимания работы такой структуры хранения данных.

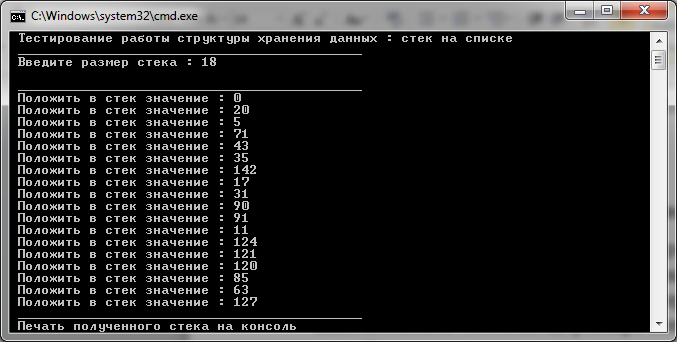


Рисунок 3. Тестирование заполнения стека

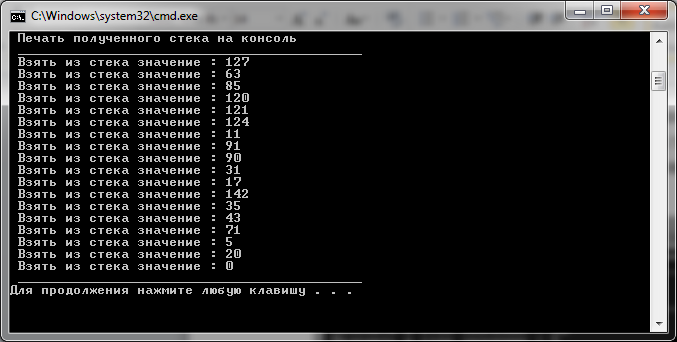


Рисунок 4. Тестирование освобождения стека

**Руководство программиста**

**Описание структур данных.**

**Класс TListStack**

Поля

TDatLink<ValType>\* firstLink – указатель на первое звено списка

int length – размер стека

Метод

TListStack() – конструктор класса

~TListStack() – деструктор класса

int IsEmpty() const – метод класса, отвечающий за переполнение

int GetLength() – метод класса, возвращающий размер стека

void Print() – метод класса, печатающий данный стек на консоль

void Put(ValType Val) – метод класса, добавляющий элемент в стек

ValType Get() – метод класса, извлекающий элемент из стека

**Описание алгоритмов.**

Добавление элемента в стек.

При добавлении элемента в стек, полученное значение хранится на вершине стека. Так как в стеке действует правило FIFO, то для данного класса добавление нового элемента в стек, элемент должен быть добавлен в начало списка, тем самым обеспечив ему первому выход из списка. То есть, таким образом, при добавлении нового элемента в стек выделяется память под него, как под новое звено списка, вкладывается значение нового элемента и указатель на первое звено в списке, и указатель прошлого первого звена сменяется на новый указатель пришедшего значения, тем самым добавляя новый элемент в начало списка.

**Описание структуры программы.**

liststack – модуль для тестирования работы структуры данных типа стек на списке. Главным, файлом которого является main.cpp

liststacklib – модуль для хранения реализации класса стек на списке. В нем содержится заголовочный файл – TListStack.h

liststacktest – модуль, содержащий тесты данного класса для прохождения их с помощью использования Google C++ Testing Framework. В файле liststack\_tests.cpp содержатся написанные тесты, а в test\_main.cpp код, запускающий тестирование

**Заключение**

**Заключение**

В ходе проведение данной лабораторной работы была создана и протестирована такая структура хранения данных как стек на списке. В классе список были реализованы такие операции как: добавление значения в стек, извлечение значения из стека, проверка стека на пустоту, вывод стека на консоль в формате поэтапного извлечения элементов из стека. Были усвоены все тонкости, с которыми пришлось столкнуться при выполнении данной лабораторной работы.

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. <https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BA>
3. <http://www.codenet.ru/progr/cpp/dlist.php> – Динамический список и его реализация