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

«Реализация стека»

Выполнил: студент группы 381706-1
Митягина Дарья Сергеевна
_____ Подпись

Научный руководитель:
ассистент каф. МОСТ ИИТММ
_____ Лебедев И.Г

Нижний Новгород
2018.

Оглавление

1. Введение

Управление памятью одна из наиболее фундаментальных областей в программировании, эффективное ее использование является важной задачей программиста.

Динамические структуры данных – это структуры данных, память под которые выделяется и освобождается по мере необходимости.

Динамические структуры данных в процессе существования в памяти могут изменять не только число составляющих их элементов, но и характер связей между элементами.

Преимущества:

- размер занимаемой памяти всегда соответствует объему хранимой информации (память то выделяется, то освобождается)
- объем хранимой информации практически не ограничен

Недостатки:

- более сложный способ работы с динамическими структурами данных

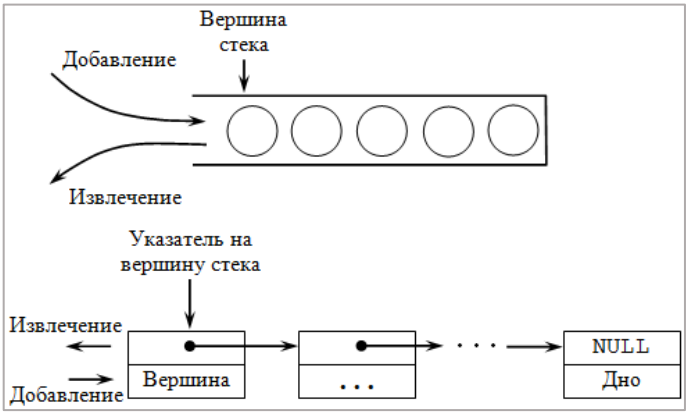

Рис. 1 – устройство стека

Стек - структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих

производится с одного конца, называемого вершиной стека. Притом первым из стека удаляется элемент, который был помещен туда последним, то есть в стеке реализуется стратегия «последним вошел — первым вышел» (last-in, first-out — LIFO).

Цель данной лабораторной работы: реализация такой структуры данных как стек.

2. Постановка задачи

В данной работе необходимо осуществить:

- Разработку спецификаций
- Выбор структуры хранения стека
- Программирование с защитой от ошибок
- Определение схемы наследования
- Реализацию тестов
- Разработку контрольного примера

3. Руководство пользователя

После запуска программы происходит следующее:

1. С пользователя запрашивается число элементов в стеке (размер стека)
2. Затем пользователь должен ввести элементы (в данном примере элты типа char)
3. После этого выводится полученный стек
4. Выполняется извлечение одного элемента
5. Демонстрируется работа конструктора копирования

В результате должно получиться нечто похожее:

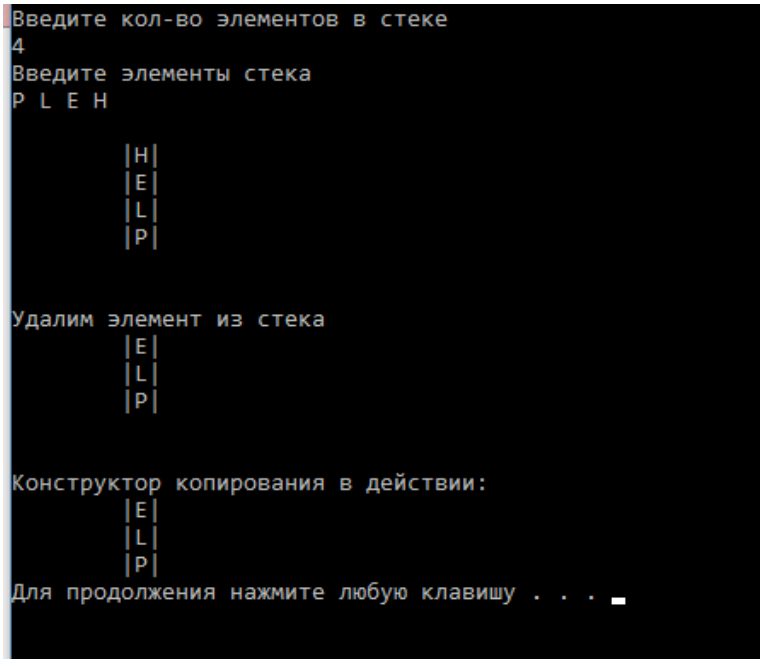

Рис. 2 – Пример использования стека

4. Руководство программиста

4.1. Описание структуры программы

Для реализации алгоритмов будет использован 1 класс TStack.

Лабораторная работа содержит следующие модули:

- StackLib

Этот модуль состоит из заголовочного файла Stack.h, отвечающего за определение интерфейса класса TStack и содержащего реализацию методов, и файла Stack.cpp. Реализованы следующие функции: конструкторы, деструктор, перегружены теоретико-множественные операторы (такие как сравнение, присваивание и т.д.), методы проверки ячеек стека на полноту/пустоту, добавления элемента, извлечения элемента из стека.

- Stack

Этот модуль содержит пример использования стека, описанный в пункте 3.

- StackTest

Содержит 12 тестов, проверяющих работу методов класса TStack.

- Exception

Содержит класс исключений.

4.2. Описание структур данных

Класс TStack:

Этот класс является шаблонным. В него включены следующие поля (*protected - часть*):

int Size; - размер стека

int Top; - элемент, расположенный на вершине стека

T* Mas; - элементы стека

Public – часть:

1) int GetSize() { return Size; }

Возвращает размер стека.

2) TStack(int n = 0);

Конструктор инициализации.

3) TStack(TStack<T> &S);

Конструктор копирования.

4) T Get();

Возвращает элемент, расположенный на вершине стека.

5) void PrintStack();

Выводит стек на экран.

6) void Put(T A);

Добавляет элемент на вершину стека, если стек не полон.

7) bool IsFull();

Проверка на полноту.

8) bool IsEmpty();

Проверка на пустоту. Стек пуст, если в нем нет ни одного элемента, т.е. когда количество элементов равно нулю.

9) int operator!=(const TStack<T>& stack) const;

Принимает ссылку на объект класса TStack, выполняет проверку на неравенство.

10) int operator==(const TStack<T>& stack) const;

Принимает ссылку на объект класса TStack, выполняет проверку на равенство.

11) TStack& operator=(const TStack<T>& stack);

Принимает ссылку на объект класса TStack, приравнивает исходный объект к полученному.

12) virtual ~TStack() - деструктор

4.3. Описание алгоритмов

В данном разделе не будут рассматриваться тривиальные методы.

1. Put(T A)

Принцип работы:

При добавлении элемента в стек необходимо переместить указатель вершины стека, записать элемент в соответствующую позицию динамического массива и увеличить количество элементов.

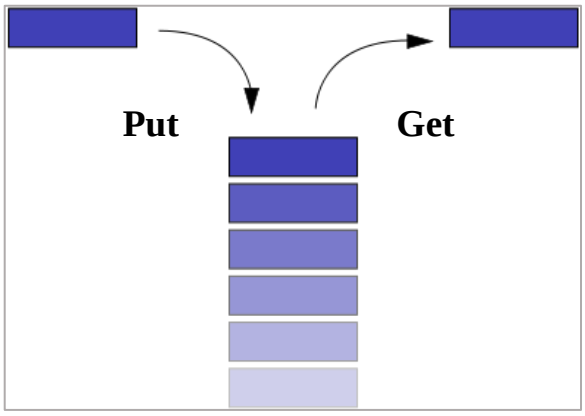

Рис. 3 – схема реализации двух методов (Put и Get)

2. T Get()

Подход аналогичный, из Рис.3 все должно быть ясно.

Если стек не пуст: при удалении элемента из стека необходимо возвратить значение из динамического массива по индексу вершины стека, переместить указатель вершины стека и уменьшить количество элементов. Рассмотрим наглядный пример:

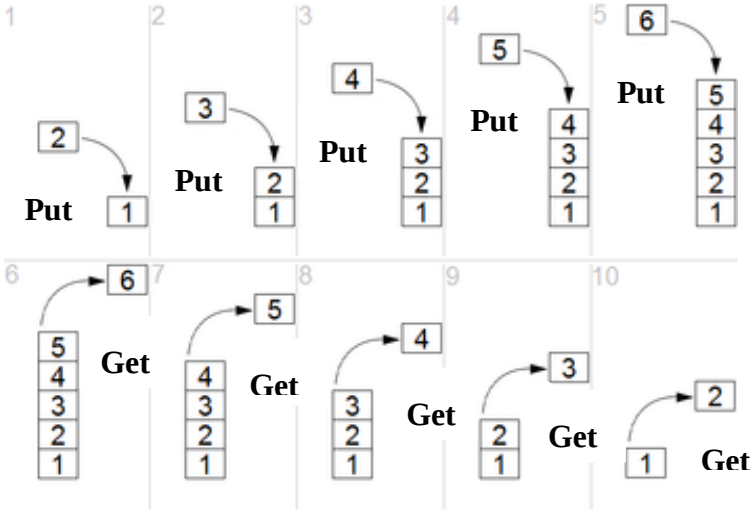

Рис. 4 - Пример

5. Заключение

В данной лабораторной работе была выполнена задача разработки программы, реализующей структуру хранения данных, а именно стек. Получены новые знания и навыки.

Результат: завершена реализация класса TStack, освоена техника составления *тестов* путем самостоятельного составления нескольких из них на базе GT.

6. Литература

1. Гергель В.П. Методические материалы по курсу «Методы программирования 2», 2015.

2. Ахо А., Хопкрофт Д., Ульман Д. Структуры данных и алгоритмы.: Пер. с англ. М.: Издат. дом «Вильямс», 2000. С. 58–76

Интернет-ресурсы:

3. Статья о стеке в Викиконспектах [https://neerc.ifmo.ru/wiki/index.php?title=%D0%A1%D1%82%D0%B5%D0%BA]