Министерство науки и образования РФ

Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский государственный электротехнический университет «ЛЭТИ»

им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

**Отчет по лабораторной работе № 4**

**по дисциплине «Программирование» на**

**тему:** **«Стек и очередь в языке C/C++»**

|  |  |
| --- | --- |
| Выполнил студент гр. 9308 | Семенов А.И, |
|  |  |
| Проверил к.т.н., доцент | Сискович Т.И |

Санкт-Петербург, 2020

Оглавление

[Цель 3](#_Toc39254010)

[1. Задание 3](#_Toc39254011)

[2. Уточнение задания 3](#_Toc39254012)

[3. Контрольные примеры 5](#_Toc39254013)

[4. Краткое описание алгоритма 6](#_Toc39254014)

[5. Структура вызова функций 7](#_Toc39254015)

[6. Функции 8](#_Toc39254016)

[6.1 Главная функция 8](#_Toc39254017)

[6.2 Функция MainMenu 9](#_Toc39254018)

[6.3 Функция Menu 10](#_Toc39254019)

[6.4 Функция OutputMenu 11](#_Toc39254020)

[6.5 Функция Help 12](#_Toc39254021)

[6.6 Функция DeleteItem 13](#_Toc39254022)

[6.7 Функция GetItem 14](#_Toc39254023)

[6.8 Функция Push\_Queue 15](#_Toc39254024)

[6.9 Функция Push\_Stack 16](#_Toc39254025)

[6.10 Функция InputStation 17](#_Toc39254026)

[6.11 Функция PrepareStruct 18](#_Toc39254027)

[6.12 Функция CopyStruct 19](#_Toc39254028)

[6.13 Функция OutputGasStationsTable 20](#_Toc39254029)

[6.14 Функция OutputGasStationsText 21](#_Toc39254030)

[6.15 Функция free\_station 22](#_Toc39254031)

[6.16 Функция free\_list 23](#_Toc39254032)

[8. Текст программы 24](#_Toc39254033)

[9. Пример работы программы 35](#_Toc39254034)

[Исходные данные 35](#_Toc39254035)

[Вывод программы 36](#_Toc39254036)

[9.1 Результат работы программы 37](#_Toc39254037)

[10. Заключение 38](#_Toc39254038)

Цель

Целью лабораторной работы является приобретение практических навыков в реализации стека и очереди на языке программирования C/C++ с использованием односвязных списков

# 1. Задание

Разработать алгоритм и написать программу, выполняющую многократно по желанию пользователя следующие действия со стеком и очередью: добавление элемента, просмотр имеющихся элементов, удаление элемента.

# 2. Уточнение задания

При выполнении задания необходимо учитывать:

1. Максимальная длина вводимой строки – 80 символов.
2. Нужно построить меню таким образом, чтобы пользователь не смог совершать действия над данными, пока он не введет их
3. Из подменю должна быть возможность выйти
4. Вводимый текст не может быть пустым

Выбранная предметная область – записи АЗС.

Исходя из выбранной предметной области, были выбраны следующие поля структуры:

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип** | **Назначение** |
| name | char\* | Название АЗС |
| address | char\* | Адрес АЗС |
| fuelprices | float\* | Цены на 4 вида топлива |
| rating | int\* | Рейтинг |
| next | struct Gaslist\* | Указатель на след. структуру |

Меню программы должно иметь подобную иерархию:

1 – Справка

2 – Режим работы со стеком

2.1 Добавление элемента в стек

2.2 Вывод последнего элемента в стеке

2.3 Вывод всех элементов

2.4 Удалить последний элемент из стека

2.5 Назад к выбору режима работы

2.6 Завершение работы программы

3 – Режим работы с очередью

3.1 Добавление элемента в очередь

3.2 Вывод первого элемента в очереди

3.3 Вывод всех элементов

3.4 Удалить первый элемент из очереди

3.5 Назад к выбору режима работы

3.6 Завершение работы программы

4 – Очистить введенные данные

0 - Выход

# 3. Контрольные примеры

Контрольные примеры приведены в таблице 1.

Таблица 1. Контрольные примеры

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № при  мера | Исходные данные | | | |
| Название | Адрес | Цены на топливо | Рейтинг |
|
| 1 | Name | Address | 5 2 2,4 7 | 7 |
| 2 | Long name of GAS | smth | 57 58 58,8 60 | 9 |
| 3 | sadlksa | -557.1;51.0 | 52 53,1 53,4 52,7 | 8 |
| 4 | Dillimore | Dillimore village | 50 50,2 50,7 50,4 | 10 |
| 5 | Last | Any | 5 7 75 2 | 10 |

# 4. Краткое описание алгоритма

Начало программы.

Шаг №1. Вывод меню (функция MainMenu).

Шаг №2. Выбор пользователем пункта меню.

Шаг №3. Переход к пункту, выбранным пользователем:

Пункт 1-ый: Справка

Пункт 2-ый: Работа со списком как со стеком

Пункт 3-ый: Работа со списком как с очередью

Пункт 4-ый: Очистка введенных данных

Шаг №4: Если пользователь не захотел выйти, то переход к шагу 1.

Шаг №5. Иначе, конец программы.

# 5. Структура вызова функций

Рисунок 1. Структура вызова функций

# 6. Функции

## 6.1 Главная функция

**Назначение:**

Является точкой входа в программу.

**Прототип:**

int main(void)

**Пример вызова:**

Main();

**Описание переменных:**

Описание переменных приведено в таблице 2.

Таблица 2. Описание переменных главной функции

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип | Назначение |
| First | GSDesc\* | Указатель на начало списка структур |
| MenuItem | int | Выбранный пункт меню |

**Схема алгоритма:**

Схема алгоритма для функции main представлена на рисунке 2.

Рисунок 2. Схема алгоритма

## 6.2 Функция MainMenu

**Назначение:**

Главное меню, в котором выбирается режим работы со списком

**Прототип:**

int MainMenu();

**Пример вызова:**

MenuItem = MainMenu();

**Описание переменных:**

Описание переменных приведено в таблице 3.

Таблица 3. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Локальная | selected | int | Выбранный пункт меню |

## 6.3 Функция Menu

**Назначение:**

Меню выбора действий

**Прототип:**

int Menu();

**Пример вызова:**

MenuItem = Menu();

**Описание переменных:**

Описание переменных приведено в таблице 3.

Таблица 3. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Локальная | selected | int | Выбранный пункт меню |

## 6.4 Функция OutputMenu

**Назначение:**

Меню вывода

**Прототип:**

void OutputMenu(GSDesc\*);

**Пример вызова:**

OutputMenu(First);

**Описание переменных:**

Описание переменных приведено в таблице 4.

Таблица 4. Описание переменных главной функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | first | GSDesc\* | Указатель на начало списка структур |
| Локальная | selected | int | Выбранный пункт меню |

## 6.5 Функция Help

**Назначение:**

Справка

**Прототип:**

void Help();

**Пример вызова:**

Help();

## 6.6 Функция DeleteItem

**Назначение:**

Функция удаления элемента из стека/очереди

**Прототип:**

GSDesc\* DeleteItem(GSDesc\*);

**Пример вызова:**

First = DeleteItem(First);

**Описание переменных:**

Описание переменных приведено в таблице 6.

Таблица 6. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | First | GSDesc\* | Указатель на начало списка структур |
| Локальная | buff | GSDesc\* | Указатель на структуру, которую удаляем |

## 6.7 Функция GetItem

**Назначение:**

Функция вывода на экран элемента из стека/очереди

**Прототип:**

void GetItem(GSDesc\*);

**Пример вызова:**

GetItem(First);

**Описание переменных:**

Описание переменных приведено в таблице 7.

Таблица 7. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | first | GSDesc\* | Указатель на начало списка структур |
| Локальные | item | GSDesc\* | Указатель на структуру, которую выводим |

## 6.8 Функция Push\_Queue

**Назначение:**

Функция добавления элемента в очередь

**Прототип:**

GSDesc\* Push\_Queue(GSDesc\*);

**Пример вызова:**

First = Push\_Queue(First);

**Описание переменных:**

Описание переменных приведено в таблице 8.

Таблица 8. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | OtherStations | GSDesc\* | Указатель на исходный вектор структур |
| Локальная | connector | GSDesc\* | Указатель, связующий OtherStations и Station |
| Station | GSDesc\* | Указатель на структуру, которую добавляем |

## 6.9 Функция Push\_Stack

**Назначение:**

Функция добавления элемента в стек

**Прототип:**

GSDesc\* Push\_Stack(GSDesc\*);

**Пример вызова:**

First = Push\_Stack(First);

**Описание переменных:**

Описание переменных приведено в таблице 9.

Таблица 9. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | OtherStations | GSDesc\* | Указатель на исходный вектор структур |
| Локальная | Station | GSDesc\* | Указатель на структуру, которую добавляем |

## 6.10 Функция InputStation

**Назначение:**

Функция ввода информации в структуру

**Прототип:**

GSDesc\* InputStation();

**Пример вызова:**

Station = InputStation();

**Описание переменных:**

Описание переменных приведено в таблице 10.

Таблица 10. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Локальные | Station | GSDesc\* | Указатель на структуру |
| i | int | Переменная для цикла for |

## 6.11 Функция PrepareStruct

**Назначение:**

Функция выделения памяти под поля структуры. Возвращает 1, если выделение памяти прошло успешно, иначе 0.

**Прототип:**

int PrepareStruct(GSDesc\*);

**Пример вызова:**

if(PrepareStruct(Station))

**Описание переменных:**

Описание переменных приведено в таблице 11.

Таблица 11. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Station | GSDesc\* | Указатель на структуру |
| Локальные | res | int | Возвращаемое число, показывающее успех выделения памяти |
| i | int | Переменная для цикла for |

## 6.12 Функция CopyStruct

**Назначение:**

Функция копирования данных из одной структуры в другую

**Прототип:**

void CopyStruct(GSDesc\*, GSDesc\*);

**Пример вызова:**

CopyStruct(item, first);

**Описание переменных:**

Описание переменных приведено в таблице 12.

Таблица 12. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | ThisStation | GSDesc\* | Указатель на структуру, в которую копируем |
| OtherStation | GSDesc\* | Указатель на структуру, которую копируем |
| Локальная | i | int | Переменная для цикла for |

## 6.13 Функция OutputGasStationsTable

**Назначение:**

Функция вывода структур в виде таблице

**Прототип:**

void OutputGasStationsTable(GSDesc\*);

**Пример вызова:**

OutputGasStationsTable(first);

**Описание переменных:**

Описание переменных приведено в таблице 13.

Таблица 13. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | first | GSDesc\* | Указатель на начало списка структур |
| Локальные | buff | GSDesc\* | Указатель на структуру списка |
| namelen | int | Максимальная длина названия |
| addresslen | int | Максимальная длина адреса |

## 6.14 Функция OutputGasStationsText

**Назначение:**

Функция вывода структур в виде блоков построчно

**Прототип:**

void OutputGasStationsText(GSDesc\*);

**Пример вызова:**

OutputGasStationsText(first);

**Описание переменных:**

Описание переменных приведено в таблице 14.

Таблица 14. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | first | GSDesc\* | Указатель на начало списка структур |
| Локальные | i | int | Переменная для цикла for |
| buff | GSDesc\* | Указатель на структуру списка |

## 6.15 Функция free\_station

**Назначение:**

Функция освобождения памяти одной структуры. Возвращает всегда NULL

**Прототип:**

GSDesc\* free\_station(GSDesc\*);

**Пример вызова:**

buff = free\_station(buff);

**Описание переменных:**

Описание переменных приведено в таблице 15.

Таблица 15. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Station | GSDesc\* | Указатель на структуру |

## 6.16 Функция free\_list

**Назначение:**

Функция освобождения памяти списка структур. Возвращает всегда NULL

**Прототип:**

GSDesc\* free\_list(GSDesc\*);

**Пример вызова:**

First = free\_list(First);

**Описание переменных:**

Описание переменных приведено в таблице 16.

Таблица 16. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | item | GSDesc\* | Указатель на структуру |
| Локальная | buff | GSDesc\* | Указатель на следующую после item структуру |

# 8. Текст программы

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <locale.h>

#include <string.h>

#define MAXLEN 80

typedef struct Gaslist

{

char\* name; //Название

char\* address; //Адрес

float fuelPrices[4]; //Ниже цены на топливо(92,95,98,дизель)

int rating; //Рейтинг АЗС(1-10)

struct Gaslist\* next; //Ссылка на следующую структуру

} GSDesc;

/\*----------------------------ФУНКЦИИ------------------------------\*/

int MainMenu(); //Меню выбора работы(стек или очередь)

int Menu(); //Меню действий

void OutputMenu(GSDesc\*); //Меню вывода

void Help(); //Справка

GSDesc\* DeleteItem(GSDesc\*); //Удаление элемента из стека/очереди

void GetItem(GSDesc\*); //Вывод крайнего элемента стека/очереди

GSDesc\* Push\_Stack(GSDesc\*); //Добавление в стек

GSDesc\* Push\_Queue(GSDesc\*); //Добавление в очередь

GSDesc\* InputStation(); //Ввод данных об АЗС

int PrepareStruct(GSDesc\*); //Выделение памяти для полей структуры

void CopyStruct(GSDesc\*, GSDesc\*); //Копирование структуры

void OutputGasStationsTable(GSDesc\*); //Вывод информации в виде таблицы

void OutputGasStationsText(GSDesc\*); //Вывод информации в виде текста

GSDesc\* free\_station(GSDesc\*); //Освобождение памяти одной АЗС

GSDesc\* free\_list(GSDesc\*); //Освобождение памяти списка АЗС

/\*-----------------------------------------------------------------\*/

int main()

{

setlocale(LC\_ALL, "russian");

GSDesc\* First = NULL; //Начало списка

int MenuItem;

do

{

MenuItem = MainMenu();

switch(MenuItem)

{

case 1:

Help();

break;

case 2:

do

{

MenuItem = Menu();

switch(MenuItem)

{

case 1:

First = Push\_Stack(First);

break;

case 2:

if(First) GetItem(First);

else

{

puts("Вы ничего не ввели, список АЗС пуст! Выполните пункт 1");

system("pause");

}

break;

case 3:

if(First) OutputMenu(First);

else

{

puts("Вы ничего не ввели, список АЗС пуст! Выполните пункт 1");

system("pause");

}

break;

case 4:

if(First) First = DeleteItem(First);

else

{

puts("Вы ничего не ввели, список АЗС пуст! Выполните пункт 1");

system("pause");

}

break;

}

fflush(stdin);

} while(MenuItem < 5);

if(MenuItem == 6) MenuItem = 0;

break;

case 3:

do

{

MenuItem = Menu();

switch(MenuItem)

{

case 1:

First = Push\_Queue(First);

break;

case 2:

if(First) GetItem(First);

else

{

puts("Вы ничего не ввели, список АЗС пуст! Выполните пункт 1");

system("pause");

}

break;

case 3:

if(First) OutputMenu(First);

else

{

puts("Нет данных для вывода! Выполните пункт 1");

system("pause");

}

break;

case 4:

if(First) First = DeleteItem(First);

else

{

puts("Вы ничего не ввели, список АЗС пуст! Выполните пункт 1");

system("pause");

}

break;

}

fflush(stdin);

} while(MenuItem < 5);

if(MenuItem == 6) MenuItem = 0;

break;

case 4:

First = free\_list(First);

puts("Данные очищены!");

system("pause");

break;

}

} while(MenuItem);

First = free\_list(First);

return 0;

}

//------------------------------------------------------МЕНЮ------------------------------------------------------

int MainMenu()

{

system("cls");

int selected;

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*ГЛАВНОЕ МЕНЮ\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

puts("1 - Справка");

puts("2 - Работать с данными, как со стеком");

puts("3 - Работать с данными, как с очередью");

puts("4 - Очистить введенные данные");

puts("0 - Выход");

do

{

scanf("%d", &selected);

if(selected < 0 || selected > 4) puts("Данного пункта меню не существует");

} while(selected < 0 || selected > 4);

fflush(stdin);

return selected;

}

int Menu()

{

system("cls");

int selected;

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*МЕНЮ\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

puts("1 - Добавление АЗС к имеющемуся списку");

puts("2 - Вывести элемент");

puts("3 - Вывод полного списка АЗС");

puts("4 - Удалить элемент списка");

puts("5 - Назад к выбору режима работы");

puts("6 - Завершить работу программы");

do

{

scanf("%d", &selected);

if(selected < 0 || selected > 6) puts("Данного пункта меню не существует");

} while(selected < 0 || selected > 6);

fflush(stdin);

return selected;

}

void OutputMenu(GSDesc\* first)

{

int item;

do

{

system("cls");

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Вывод\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

puts("1 - В виде таблицы");

puts("2 - В виде текста");

puts("0 - Назад");

do

{

scanf("%d", &item);

if(item < 0 || item > 2) puts("Данного пункта меню не существует");

} while(item < 0 || item > 2);

fflush(stdin);

if(item == 1) OutputGasStationsTable(first);

else if(item == 2) OutputGasStationsText(first);

}while(item);

}

void OutputItem(GSDesc\* item)

{

int selected;

do

{

system("cls");

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Вывод\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

puts("1 - В виде таблицы");

puts("2 - В виде текста");

puts("0 - Назад");

do

{

scanf("%d", &selected);

if(selected < 0 || selected > 2) puts("Данного пункта меню не существует");

} while(selected < 0 || selected > 2);

fflush(stdin);

if(selected == 1) OutputGasStationsTable(item);

else if(selected == 2) OutputGasStationsText(item);

}while(selected);

}

void Help()

{

system("cls");

puts(" Данная программа предназначена для работы над списком АЗС, имеющим следующую структуру:");

puts("название, адрес, цены на топливо и рейтинг - в двух режимах:");

puts("стек(добавление, удаление и получение последнего элемента)");

puts("и очередь(те же действия, но с первым элементом). В каждом режиме можно вывести полный список АЗС");

puts(" В главном меню Вы можете выбрать режим работы");

puts("Стоит заметить, что, поработав в одном режиме, можно поменять его в том же главном меню");

puts("а введенные ранее данные не нужно будет вводить заново");

puts(" При желании можно очистить все данные, выбрав в главном меню подходящий пункт\n");

system("pause");

}

//------------------------------------------------------ДЕЙСТВИЯ СО СПИСКОМ------------------------------------------------------

GSDesc\* DeleteItem(GSDesc\* First)

{

system("cls");

GSDesc\* buff = NULL;

buff = First;

First = First->next;

buff = free\_station(buff);

return First;

}

void GetItem(GSDesc\* first)

{

system("cls");

GSDesc\* item = NULL;

item = (GSDesc\*)malloc(sizeof(GSDesc));

CopyStruct(item, first);

item->next = NULL;

OutputItem(item);

item = free\_station(item);

}

GSDesc\* Push\_Stack(GSDesc\* OtherStations) //Добавление в начало

{

GSDesc\* Station = NULL;

Station = InputStation();

Station->next = OtherStations;

return Station;

}

GSDesc\* Push\_Queue(GSDesc\* OtherStations) //Добавление в конец

{

GSDesc\* Station = NULL;

GSDesc\* connector = NULL;

Station = InputStation();

if(OtherStations)

{

for(connector = OtherStations; connector->next != NULL ; connector = connector->next);

connector->next = Station;

}

else

OtherStations = Station;

return OtherStations;

}

//------------------------------------------------------ВВОД------------------------------------------------------

void InputText(char\* string)

{

char c; //Введенный символ

int i;

i = 0;

do

{

c = getch();

if(c != 13 && c != 8)

{

string[i] = c;

printf("%c", c);

i++;

}

else if(c == 8)

{

if(i > 0)

{

string[i-1] = '\0';

printf("\r%s %c", string, c);

i--;

}

}

if(c == 13 && i == 0)

puts("Строка не может быть пустой!");

} while(i < MAXLEN && (c != 13 || i == 0));

string[i] = '\0';

fflush(stdin);

}

GSDesc\* InputStation()

{

system("cls");

int i;

GSDesc\* Station = NULL;

Station = (GSDesc\*)malloc(sizeof(GSDesc));

if(Station != NULL)

{

if(PrepareStruct(Station))

{

system("cls");

puts("Введите название АЗС");

InputText(Station->name);

puts("\nВведите адрес");

InputText(Station->address);

puts("\nВведите цены на топливо(92,95,98,Дизель). Цена не выше 10000");

for(i = 0; i < 4; i++)

{

do

{

scanf("%f", &(Station->fuelPrices[i]));

if(Station->fuelPrices[i] < 1.0 || Station->fuelPrices[i] > 10000.0) puts("Цена от 1 до 10000");

} while(Station->fuelPrices[i] < 1.0 || Station->fuelPrices[i] > 10000.0);

}

puts("Введите рейтинг АЗС от 1 до 10");

do

{

scanf("%d", &(Station->rating));

if(Station->rating < 1 || Station->rating > 10) puts("Рейтинг от 1 до 10!");

} while(Station->rating < 1 || Station->rating > 10);

}

}

system("pause");

return Station;

}

//------------------------------------------------------ВЫВОД------------------------------------------------------

void OutputGasStationsTable(GSDesc\* first)

{

fflush(stdout);

system("cls");

int namelen,

addresslen;

GSDesc\* buff = first;

namelen = 8, addresslen = 5;

for(; buff != NULL; buff = buff->next)

{

if(strlen(buff->name) > namelen) namelen = strlen(buff->name);

if(strlen(buff->address) > addresslen) addresslen = strlen(buff->address);

}

printf("|%\*s|%\*s|Цена 92 бензина|Цена 95 бензина|Цена 98 бензина|Цена дизеля|Рейтинг|\n", namelen, "Название", addresslen, "Адрес");

for(buff = first; buff != NULL; buff = buff->next)

printf("|%\*s|%\*s|%15.2f|%15.2f|%15.2f|%11.2f|%7d|\n", namelen, buff->name, addresslen, buff->address,

buff->fuelPrices[0], buff->fuelPrices[1], buff->fuelPrices[2], buff->fuelPrices[3], buff->rating);

system("pause");

}

void OutputGasStationsText(GSDesc\* first)

{

fflush(stdout);

system("cls");

GSDesc\* buff = first;

int i;

for(; buff != NULL; buff = buff->next)

{

printf("Название: %s", buff->name);

printf("\nАдрес: %s", buff->address);

printf("\nЦены(92,95,98,Дизель): ");

for(i = 0; i < 4; i++) printf("%.2f ", buff->fuelPrices[i]);

printf("\nРейтинг: %d", buff->rating);

printf("\n\n");

}

system("pause");

}

//------------------------------------------------------ПАМЯТЬ------------------------------------------------------

int PrepareStruct(GSDesc\* Station)

{

int res = 0,

i;

if(Station != NULL)

{

Station->name = (char\*)malloc((MAXLEN+1)\*sizeof(char));

Station->address = (char\*)malloc((MAXLEN+1)\*sizeof(char));

for(i = 0; i < 4; i++) Station->fuelPrices[i] = 0;

Station->rating = 0;

Station->next = NULL;

res++;

}

return res;

}

GSDesc\* free\_station(GSDesc\* Station)

{

if(Station != NULL)

{

free(Station->name);

Station->name = NULL;

free(Station->address);

Station->address = NULL;

Station->next = NULL;

free(Station);

Station = NULL;

}

return NULL;

}

GSDesc\* free\_list(GSDesc\* item)

{

if(item != NULL)

{

GSDesc\* buff = NULL;

for(; item != NULL; item = buff)

{

buff = item->next;

item = free\_station(item);

}

}

return NULL;

}

void CopyStruct(GSDesc\* ThisStation, GSDesc\* OtherStation)

{

int i;

if(PrepareStruct(ThisStation))

{

strcpy(ThisStation->name, OtherStation->name);

strcpy(ThisStation->address, OtherStation->address);

for(i = 0; i < 4; i++)

ThisStation->fuelPrices[i] = OtherStation->fuelPrices[i];

ThisStation->rating = OtherStation->rating;

}

}

# 9. Пример работы программы

## Исходные данные

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № при  мера | Исходные данные | | | |
| Название | Адрес | Цены на топливо | Рейтинг |
|
| 1 | Name | Address | 5 2 2,4 7 | 7 |
| 2 | Dillimore | Dillimore village | 50 50,2 50,7 50,4 | 10 |

## Вывод программы

Рисунок 3. Главное меню

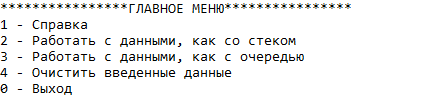


Рисунок 4. Вывод первого элемента стека

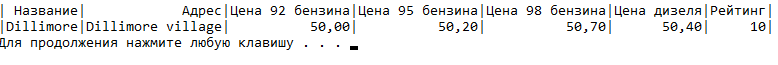
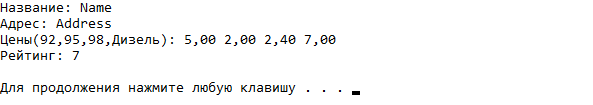


Рисунок 5. Вывод первого элемента в очереди



## 9.1 Результат работы программы

При выполнении программы получены результаты, совпадающие со значениями, приведенными в Таблице 1. Ошибки не обнаружены. Пример протокола выполнения программы приведены на рисунках 3, 4, 5.

# 10. Заключение

При выполнении лабораторной работы получены практические навыки в реализации стека и очереди на языке программирования C/C++.