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 № 3**

**по дисциплине «Программирование» на**

**тему:** **«Двусвязные списки в языке C/C++»**

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

Санкт-Петербург, 2020

Оглавление

[Введение 4](#_Toc41396023)

[1. Задание 4](#_Toc41396024)

[2. Уточнение задания 4](#_Toc41396025)

[3. Контрольные примеры 7](#_Toc41396026)

[4. Краткое описание алгоритма 8](#_Toc41396027)

[5. Структура вызова функций 9](#_Toc41396028)

[6. Функции 9](#_Toc41396029)

[6.1 Главная функция 10](#_Toc41396030)

[6.2 Функция Menu 11](#_Toc41396031)

[6.3 Функция OutputMenu 12](#_Toc41396032)

[6.4 Функция InputMenu 13](#_Toc41396033)

[6.5 Функция Help 14](#_Toc41396034)

[6.6 Функция ListActions 15](#_Toc41396035)

[6.7 Функция ListLen 16](#_Toc41396036)

[6.8 Функция DeleteItem 17](#_Toc41396037)

[6.9 Функция Sort 18](#_Toc41396038)

[6.10 Функция Swap 19](#_Toc41396039)

[6.11 Функция GetItem 20](#_Toc41396040)

[6.12 Функция PushBack 21](#_Toc41396041)

[6.13 Функция PushForward 22](#_Toc41396042)

[6.14 Функция PushAnyPlace 23](#_Toc41396043)

[6.15 Функция InputStations 24](#_Toc41396044)

[6.16 Функция Process 25](#_Toc41396045)

[6.17 Функция PrepareStruct 26](#_Toc41396046)

[6.18 Функция CopyStruct 27](#_Toc41396047)

[6.19 Функция OutputGasStationsTable 28](#_Toc41396048)

[6.20 Функция OutputGasStationsText 29](#_Toc41396049)

[6.21 Функция free\_station 30](#_Toc41396050)

[6.22 Функция free\_list 31](#_Toc41396051)

[6.23 Функция OutputGasStationsTableReverse 32](#_Toc41396052)

[6.24 Функция OutputGasStationsTextReverse 33](#_Toc41396053)

[6.25 Функция InputText 34](#_Toc41396054)

[6.26 Функция ReverseList 35](#_Toc41396055)

[8. Текст программы 36](#_Toc41396056)

[9. Пример работы программы 56](#_Toc41396057)

[9.1 Исходные данные 56](#_Toc41396058)

[9.2 Вывод программы 57](#_Toc41396059)

[9.3 Результат работы программы 58](#_Toc41396060)

[Заключение 59](#_Toc41396061)

Введение

Целью лабораторной работы является приобретение практических навыков в реализации двусвязного списка структур на языке программирования C/C++.

# 1. Задание

Написать программу, выполняющую ввод списка структур, длина которого заранее неизвестна. Контрольный вывод исходного списка, формирование новой списка из элементов исходного, поля которого удовлетворяют заданным условиям, совершение действий над исходным списком: узнать кол-во элементов списка, удалить элемент списка, поменять местами 2 элемента списка, сортировка списка, поиск элемента по заданным параметрам. Вывод полученного результата. В программе использовать меню и функции. Последовательность хранить в динамической памяти. Поля в структурах выбрать исходя из характеристик объектов предметной области. Условия для выбора элементов последовательности сформулировать самостоятельно.

# 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 начать обработку исходных данных и совершать действия над ними, пока он не введет входные данные
3. Из подменю должна быть возможность выйти
4. Вводимый текст не может быть пустым
5. Количество критериев для обработки исходных данных – 2: рейтинг выше 7 и цена на 98 бензин ниже 54

Выбранная предметная область – записи АЗС.

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

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип** | **Назначение** |
| name | char\* | Указатель на начало строки, содержащую название АЗС |
| address | char\* | Указатель на начало строки, содержащую адрес АЗС |
| fuelprices | float | Массив цен на 4 вида топлива |
| rating | int | Рейтинг |
| next | struct Gaslist\* | Указатель на след. структуру |
| prev | struct Gaslist\* | Указатель на пред. структуру |

Меню программы должно иметь подобную иерархию:

1 – Справка

2 – Ввод данных

2.1 Добавить в начало

2.2 Добавить в конец

2.3 Добавить на определенное место

2.4 Ввести полностью заново

2.5 Назад

3 – Контрольный вывод исходных данных

4 – Действия над списком

4.1 Узнать кол-во элементов в списке

4.2 Удалить элемент списка

4.3 Поменять местами 2 элемента

4.4 Отсортировать список по рейтингу

4.5 Поиск по определенному критерию

4.6 Назад

5 – Обработка данных по 2 условиям: рейтинг выше 7, цена на 98 бензин

ниже 54

6 – Вывод обработанной информации

0 - Выход

# 3. Контрольные примеры

Контрольные примеры приведены в таблице 1.

Таблица 1. Контрольные примеры

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № при  ме  ра | Исходные данные | | | | Результаты | |
| Название | Адрес | Цены на топливо | Рей  тинг | Условия | |
| Рейтинг выше 7 | Цена на 98 бензин ниже 54 |
| 1 | Name | Address | 5 2 2,4 7 | 7 | Не входит в результирующий список АЗС | |
| 2 | Long name of GAS | smth | 57 58 58,8 60 | 9 | Не входит в результирующий список АЗС | |
| 3 | sadlksa | -557.1;51.0 | 52 53,1 53,4 52,7 | 8 | Входит в результирующий список АЗС | |
| 4 | Dillimore | Dillimore village | 50 50,2 50,7 50,4 | 10 | Входит в результирующий список АЗС | |
| 5 | Last | Any | 5 7 75 2 | 10 | Не входит в результирующий список АЗС | |

# 4. Краткое описание алгоритма

Начало программы.

Шаг №1. Вывод меню (функция *Menu*).

Шаг №2. Выбор пользователем пункта меню.

Шаг №3. Переход к пункту, выбранным пользователем:

Пункт 1-ый: Справка. Переход к шагу 4.

Пункт 2-ый: Ввод. Переход к шагу 4.

Пункт 3-ый: Вывод введенных данных. Переход к шагу 4.

Пункт 4-ый: Действия над списком. Переход к шагу 4.

Пункт 5-ый: Обработка. Переход к шагу 4.

Пункт 6-ый: Вывод результата обработки. Переход к шагу 4.

Пункт 0-ой: Переход к шагу 4.

Шаг №4: Проверка на выбранный пункт меню. Если пользователь выбрал пункт 0, то переход к шагу 5, иначе – переход к шагу 1

Шаг №5: Завершение 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 |
| GSResult | GSDesc\* | Указатель на начало списка результирующих структур |
| MenuItem | int | Выбранный пункт меню |
| len | int | Кол-во элементов в списке |
| index1 | int | Индекс первого элемента для смены |
| index2 | int | Индекс второго элемента для смены |
| field | int | Поле сортировки |
| mode | int | Режим сортировки |

## 6.2 Функция Menu

**Назначение:**

Главное меню

**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 |

## 6.3 Функция OutputMenu

**Назначение:**

Меню вывода

**Прототип:**

void OutputMenu(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

OutputMenu(First); First описана в таблице 2

**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 |

## 6.4 Функция InputMenu

**Назначение:**

Меню ввода данных

**Прототип:**

GSDesc\* InputMenu(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

First = InputMenu(First); First описана в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 5.

Таблица 5. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| Локальные | NewStations | GSDesc\* | Указатель на начало списка вновь введенных структур |
| item | int | Выбранный пункт меню |

## 6.5 Функция Help

**Назначение:**

Справка

**Прототип:**

void Help();

**Пример вызова:**

Help();

## 6.6 Функция ListActions

**Назначение:**

Меню действий над списком

**Прототип:**

int ListActions();

**Пример вызова:**

MenuItem = ListActions(); MenuItem описана в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 6.

Таблица 6. Описание переменных функции

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

## 6.7 Функция ListLen

**Назначение:**

Функция вычисления кол-ва структур в списке

**Прототип:**

int ListLen(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

len = ListLen(First); First, len описаны в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 7.

Таблица 7. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| Локальная | len | int | Кол-во структур в списке |

## 6.8 Функция DeleteItem

**Назначение:**

Функция удаления элемента из списка

**Прототип:**

GSDesc\* DeleteItem(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

First = DeleteItem(First); First описана в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 8.

Таблица 8. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| Локальные | temp | GSDesc\* | Служебный указатель для связки после удаления элемента |
| toDel | GSDesc\* | Указатель на структуру для удаления |
| num | int | Номер элемента для удаления |

## 6.9 Функция Sort

**Назначение:**

Функция сортировки списка по полю

**Прототип:**

void Sort(GSDesc\*\*, int, int); Передаются указатель на первую структуру списка и поле, по которому сортируется список

**Пример вызова:**

Sort(&First, field, mode); First, field, mode описаны в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 9.

Таблица 9. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| field | int | Поле, по которому сортируем |
| Локальные | result | GSDesc\* | Указатель на начало отсортированного списка структур |
| temp | GSDesc\* | Указатель на структуру исходного списка |
| buff | GSDesc\* | Указатель на структуру отсортированного списка |
| num | int | Переменная для цикла for |
| i | int | Индекс сравниваемого элемента |
| j | int | Индекс элемента, с которым сравниваем |
| n | int | Длина списка |

## 6.10 Функция Swap

**Назначение:**

Функция перестановки 2 элементов местами

**Прототип:**

void Swap(GSDesc\*, int, int); Передаются указатель на первую структуру списка, номера структур в списке для смены

**Пример вызова:**

Swap(First, index1, index2); First, index1, index2 описаны в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 10.

Таблица 10. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| first | int | Номер 1-го элемента для смены |
| second | int | Номер 2-го элемента для смены |
| Локальная | gFirst | GSDesc\* | Указатель на первый элемент для смены |
| gSecond | GSDesc\* | Указатель на второй элемент для смены |
| buff | GSDesc\* | Указатель на структуру, стоящую перед первым элементом для смены |
| temp | int | Переменная для смены местами first и second, если first > second |
| i | int | Переменная для цикла for |

## 6.11 Функция GetItem

**Назначение:**

Функция вывода структур по заданным параметрам

**Прототип:**

void GetItem(GSDesc\*, int); Передается указатель на первую структуру списка и номер поля, по которому будет осуществляться поиск

**Пример вызова:**

GetItem(First, field); First, field описаны в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 11.

Таблица 11. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на вектор структур |
| field | int | Поле для поиска |
| Локальные | result | GSDesc\* | Указатель на начало списка структур, удовлетворяющим условиям |
| buff | GSDesc\* | Указатель на структуру из результирующего списка |
| buffThis | GSDesc\* | Указатель на элемент исходного списка |
| string | char\* | Указатель на строку, которую ищем |

## 6.12 Функция PushBack

**Назначение:**

Функция вставки в конец списка

**Прототип:**

GSDesc\* PushBack(GSDesc\*, GSDesc\*); Передаются указатель на список структур, который добавляется к исходному, и указатель на список, в конец которого добавляются новые элементы

**Пример вызова:**

Stations = PushBack(NewStations, Stations); NewStations, Stations описаны в таблице 5

**Описание переменных:**

Описание переменных приведено в таблице 12.

Таблица 12. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | NewStations | GSDesc\* | Указатель на вектор структур, который вставляем |
| Stations | GSDesc\* | Указатель на начало списка имеющихся структур |
| Локальная | connector | GSDesc\* | Указатель, связующий NewStations и Stations |

## 6.13 Функция PushForward

**Назначение:**

Функция вставки в начало списка

**Прототип:**

GSDesc\* PushForward(GSDesc\*, GSDesc\*); Передаются указатель на список структур, который добавляется к исходному, и указатель на список, в начало которого добавляются новые элементы

**Пример вызова:**

Stations = PushForward(NewStations, Stations); NewStations, Stations описаны в таблице 5

**Описание переменных:**

Описание переменных приведено в таблице 13.

Таблица 13. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | NewStations | GSDesc\* | Указатель на вектор структур, который вставляем |
| Stations | GSDesc\* | Указатель на начало списка имеющихся структур |
| Локальная | connector | GSDesc\* | Указатель, связующий NewStations и Stations |

## 6.14 Функция PushAnyPlace

**Назначение:**

Функция вставки в любое место списка

**Прототип:**

GSDesc\* PushAnyPlace(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

Stations = PushAnyPlace(Stations); Stations описана в таблице 5

**Описание переменных:**

Описание переменных приведено в таблице 14.

Таблица 14. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало списка структур |
| Локальные | NewStations | GSDesc\* | Указатель на начало списка структур, которые вставляем в исходный список |
| connector | GSDesc\* | Указатель, связующий NewStations и Stations |
| buff | GSDesc\* | Указатель на структуру, после которой будет вставка |
| buff2 | GSDesc\* | Указатель на структуру, перед которой будет вставка |
| ans | int | Место для вставки |
| i | int | Переменная для цикла for |
| len | int | Кол-во элементов в списке Stations |

## 6.15 Функция InputStations

**Назначение:**

Функция ввода структур

**Прототип:**

GSDesc\* InputStations();

**Пример вызова:**

NewStations = InputStations(); NewStations описана в таблице 5

**Описание переменных:**

Описание переменных приведено в таблице 15.

Таблица 15. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Локальные | first | GSDesc\* | Указатель на начало списка структур, которые ввели |
| buff | GSDesc\* | Указатель на последнюю структуру в вводимом списке |
| i | int | Переменная для цикла for |
| f | int | Флаг повтора ввода |

## 6.16 Функция Process

**Назначение:**

Функция обработки списка структур

**Прототип:**

GSDesc\* Process(GSDesc\*); Передается указатель на первую структуру исходного списка

**Пример вызова:**

GResult = Process(First); First, GResult описаны в таблице 2

**Описание переменных:**

Описание переменных приведено в таблице 16.

Таблица 16. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\* | Указатель на начало исходного списка структур |
| Локальные | result | GSDesc\* | Указатель на начало списка отобранных структур |
| buff | GSDesc\* | Указатель на последнюю структуру в результирующем списке |

## 6.17 Функция PrepareStruct

**Назначение:**

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

**Прототип:**

int PrepareStruct(GSDesc\*); Передается указатель на структуру

**Пример вызова:**

if(PrepareStruct(buff))

buff описана в таблице 10

**Описание переменных:**

Описание переменных приведено в таблице 17.

Таблица 17. 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 |

## 6.18 Функция CopyStruct

**Назначение:**

Функция копирования данных из одной структуры в другую

**Прототип:**

void CopyStruct(GSDesc\*, GSDesc\*); Передаются указатель на структуру, в которую копируем, и указатель на структуру, которую копируем

**Пример вызова:**

CopyStruct(buff, gFirst); buff, gFirst описаны в таблице 10

**Описание переменных:**

Описание переменных приведено в таблице 18.

Таблица 18. 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 |

## 6.19 Функция OutputGasStationsTable

**Назначение:**

Функция вывода структур в виде таблице

**Прототип:**

void OutputGasStationsTable(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

OutputGasStationsTable(first); first описана в таблице 4

**Описание переменных:**

Описание переменных приведено в таблице 19.

Таблица 19. 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 |

## 6.20 Функция OutputGasStationsText

**Назначение:**

Функция вывода структур в виде блоков построчно

**Прототип:**

void OutputGasStationsText(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

OutputGasStationsText(first); first описана в таблице 4

**Описание переменных:**

Описание переменных приведено в таблице 20.

Таблица 20. 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 |

## 6.21 Функция free\_station

**Назначение:**

Функция освобождения памяти одной структуры. Возвращает всегда NULL

**Прототип:**

GSDesc\* free\_station(GSDesc\*); Передается указатель на структуру

**Пример вызова:**

toDel = free\_station(toDel); toDel описана в таблице 8

**Описание переменных:**

Описание переменных приведено в таблице 21.

Таблица 21. Описание переменных функции

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

## 6.22 Функция free\_list

**Назначение:**

Функция освобождения памяти списка структур. Возвращает всегда NULL

**Прототип:**

GSDesc\* free\_list(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

result = free\_list(result); result описана в таблице 11

**Описание переменных:**

Описание переменных приведено в таблице 22.

Таблица 22. Описание переменных функции

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

## 6.23 Функция OutputGasStationsTableReverse

**Назначение:**

Функция вывода структур в виде таблице, начиная с конца

**Прототип:**

void OutputGasStationsTableReverse(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

OutputGasStationsTableReverse(first); first описана в таблице 4

**Описание переменных:**

Описание переменных приведено в таблице 23.

Таблица 23. 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 |

## 6.24 Функция OutputGasStationsTextReverse

**Назначение:**

Функция вывода структур в виде блоков построчно, начиная с конца

**Прототип:**

void OutputGasStationsText(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

OutputGasStationsText(first); first описана в таблице 4

**Описание переменных:**

Описание переменных приведено в таблице 24.

Таблица 24. 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 |

## 6.25 Функция InputText

**Назначение:**

Функция ввода строки

**Прототип:**

void InputText(char\*); Передается указатель на строку

**Пример вызова:**

InputText(buff->name); name указана в описании структуры

**Описание переменных:**

Описание переменных приведено в таблице 25.

Таблица 25. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | string | char\* | Указатель на начало строки |
| Локальная | c | char | Введенный символ |
| i | int | Переменная для цикла for |

## 6.26 Функция ReverseList

**Назначение:**

Функция реверса списка

**Прототип:**

GSDesc\* ReverseList(GSDesc\*); Передается указатель на первую структуру списка

**Пример вызова:**

\*Stations = ReverseList(\*Stations); Stations описана в таблице 9

**Описание переменных:**

Описание переменных приведено в таблице 26.

Таблица 26. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | First | GSDesc\* | Указатель на начало списка |
| Локальная | result | GSDesc\* | Указатель на начало реверсивного списка |
| buff | GSDesc\* | Указатель на последний элемент реверсивного списка |
| buffThis | GSDesc\* | Указатель на последний элемент исходного списка |

# 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

struct Gaslist\* prev; //Ссылка на предыдущую структуру

} GSDesc;

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

int Menu(); //Меню

int ListActions(); //Действия со списком

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

GSDesc\* InputMenu(GSDesc\*); //Меню выбора ввода

void Help(); //Справка

int ListLen(GSDesc\*); //Длина списка

GSDesc\* DeleteItem(GSDesc\*); //Удаление элемента из списка

void Sort(GSDesc\*\*, int, int); //Сортировка по полю

void Swap(GSDesc\*, int, int); //Поменять местами 2 элемента

void GetItem(GSDesc\*, int); //Вывод АЗС по заданным параметрам

GSDesc\* ReverseList(GSDesc\*); //Реверс списка

GSDesc\* PushBack(GSDesc\*, GSDesc\*); //Добавление в конец

GSDesc\* PushForward(GSDesc\*, GSDesc\*); //Добавление в начало

GSDesc\* PushAnyPlace(GSDesc\*); //Добавление в любое место

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

GSDesc\* Process(GSDesc\*); //Обработка данных

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

void OutputGasStationsTableReverse(GSDesc\*); //Вывод информации в виде таблицы с конца

void OutputGasStationsTextReverse(GSDesc\*); //Вывод информации в виде текста с конца

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

GSDesc\* GResult = NULL; //Начало списка результата

int MenuItem, //Пункт меню

len, //Длина списка

field, //Поле поиска

index1, //Индекс первого элемента для смены

index2, //Индекс второго элемента для смены

mode; //Режим сортировки

do

{

MenuItem = Menu();

switch(MenuItem)

{

case 1:

Help();

break;

case 2:

First = InputMenu(First);

break;

case 3:

if(First != NULL) OutputMenu(First);

else

{

puts("Нет данных для вывода, список пуст!");

system("pause");

}

break;

case 4:

if(First != NULL)

{

do

{

MenuItem = ListActions();

switch(MenuItem)

{

case 1:

len = ListLen(First);

printf("Количество элементов в списке: %d\n",len);

system("pause");

break;

case 2:

First = DeleteItem(First);

break;

case 3:

system("cls");

printf("Введите два номера элементов для перестановки(от 1 до %d): ", len);

do

{

scanf("%d %d", &index1, &index2);

if(index1 < 1 || index1 > len || index2 < 1 || index2 > len)

printf("Введенные номера должны быть от 1 до %d!\n", len);

} while(index1 < 1 || index1 > len || index2 < 1 || index2 > len);

Swap(First, index1, index2);

break;

case 4:

system("cls");

puts("Выберете поле для сортировки:\n1 - название, 2 - адрес, 3 - цена на 92 бензин,4 - цена на 95 бензин\n5 - цена на 98 бензин, 6 - цена на дизель, 7 - рейтинг");

do

{

scanf("%d", &field);

if(field < 1 || field > 7) puts("Поля с таким номером нет!");

} while(field < 1 || field > 7);

puts("Режим сортировки: 1 - по возрастанию, 2 - по убыванию");

do

{

scanf("%d", &mode);

if(mode < 1 || mode > 2) puts("Режима с таким номером нет!");

} while(mode < 1 || mode > 2);

mode--;

Sort(&First, field, mode);

break;

case 5:

system("cls");

puts("Выберете поле для поиска:\n1 - название, 2 - адрес, 3 - цена на 92 бензин,4 - цена на 95 бензин\n5 - цена на 98 бензин, 6 - цена на дизель, 7 - рейтинг\n0 - искать везде");

do

{

scanf("%d", &field);

if(field < 0 || field > 7) puts("Поля с таким номером нет!");

} while(field < 0 || field > 7);

GetItem(First, field);

break;

}

} while(MenuItem != 6);

}

else

{

puts("Сначала введите список АЗС!");

system("pause");

}

break;

case 5:

if(First != NULL)

{

GResult = free\_list(GResult);

GResult = Process(First);

}

else

{

puts("Сначала введите список АЗС!");

system("pause");

}

break;

case 6:

if(GResult != NULL) OutputMenu(GResult);

else

{

puts("Не была обработана информация, либо ни одна АЗС не удовлетворяет условиям!");

system("pause");

}

break;

}

} while(MenuItem);

First = free\_list(First);

GResult = free\_list(GResult);

return 0;

}

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

int Menu()

{

system("cls");

int selected;

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

puts("1 - Справка");

puts("2 - Ввод информации об АЗС");

puts("3 - Вывод информации об АЗС");

puts("4 - Действия над списком АЗС");

puts("5 - Обработка данных");

puts("6 - Вывод данных");

puts("0 - Выход");

do

{

scanf("%d", &selected);

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

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

fflush(stdin);

return selected;

}

int ListActions()

{

system("cls");

int selected;

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

puts("1 - Узнать количество элементов списка АЗС");

puts("2 - Удалить элемент списка АЗС");

puts("3 - Поменять местами элементы списка АЗС");

puts("4 - Сортировка элементов списка АЗС по рейтингу(по убыв.)");

puts("5 - Поиск АЗС по определенному критерию");

puts("6 - Назад");

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("3 - В виде таблицы, начиная с конца");

puts("4 - В виде текста, начиная с конца");

puts("0 - Назад");

do

{

scanf("%d", &item);

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

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

fflush(stdin);

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

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

else if(item == 3) OutputGasStationsTableReverse(first);

else if(item == 4) OutputGasStationsTextReverse(first);

}while(item);

}

GSDesc\* InputMenu(GSDesc\* Stations) //Меню выбора ввода

{

system("cls");

GSDesc\* NewStations = NULL;

int item;

do

{

system("cls");

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Ввод\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

puts("1 - Добавить в начало");

puts("2 - Добавить в конец");

puts("3 - Добавить на определенное место");

puts("4 - Ввести полностью заново");

puts("5 - Назад");

do

{

scanf("%d", &item);

if(item < 0 || item > 5) puts("Данного пункта меню не существует");

} while(item < 0 || item > 5);

fflush(stdin);

if(item < 3) NewStations = InputStations();

if(item == 1) Stations = PushForward(NewStations, Stations);

else if(item == 2) Stations = PushBack(NewStations, Stations);

else if(item == 3) Stations = PushAnyPlace(Stations);

else if(item == 4)

{

Stations = free\_list(Stations);

Stations = InputStations();

}

}while(item != 5);

return Stations;

}

void Help()

{

system("cls");

puts(" Данная программа расчитана для работы над списком АЗС,");

puts("имеющий следующую структуру: название, адрес, цены на топливо и рейтинг");

puts(" Пункт 'обработка' подразумевает под собой отбор тех АЗС,");

puts("которые удовлетворяют следующим условиям:");

puts("рейтинг выше 7, цена на 98 бензин ниже 54");

puts(" Также Вам доступны следующие действия над списком:");

puts("узнать кол-во АЗС в списке, удалить одну АЗС из списка,");

puts("поменять местами 2 АЗС в списке, отсортировать по убыванию рейтинга");

puts("(при сортировке изменяется ИЗНАЧАЛЬНЫЙ список, будьте внимательны!)");

puts("поиск АЗС по определенному параметру\n");

puts(" Внимание! Кол-во букв в названии и адресе не могут превышать 80!\n");

system("pause");

}

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

int ListLen(GSDesc\* Stations)

{

int len = 0;

for(;Stations != NULL; Stations = Stations->next) len++;

return len;

}

GSDesc\* DeleteItem(GSDesc\* Stations)

{

system("cls");

GSDesc\* temp = Stations;

GSDesc\* toDel = NULL;

int num;

printf("Введите номер элемента, который хотите удалить(всего %d элементов): ", ListLen(Stations));

do

{

scanf("%d", &num);

if(num < 1 || num > ListLen(Stations)) printf("Элемента под таким номером нет. Введите от 1 до %d\n", ListLen(Stations));

} while(num < 1 || num > ListLen(Stations));

if(num == 1)

{

Stations = Stations->next;

Stations->prev = NULL;

temp = free\_station(temp);

}

else

{

toDel = temp->next;

while(num > 2)

{

temp = temp->next;

toDel = temp->next;

num--;

}

temp->next = toDel->next;

temp->next->prev = temp;

toDel = free\_station(toDel);

}

return Stations;

}

void Sort(GSDesc\*\* Stations, int field, int mode) //Сортировка: адрес, название, цена на 92,95,98,дизель, рейтинг

{

system("cls");

GSDesc\* result = NULL;

GSDesc\* temp = NULL;

GSDesc\* buff = NULL;

GSDesc\* buffThis = NULL;

int num,

i,

j,

n;

n = ListLen(\*Stations);

if(field == 1)

{

for(i = 1, buffThis = \*Stations; i <= n; buffThis = buffThis->next, i++)

{

for(j = 1, temp = \*Stations; temp->next != NULL; temp = temp->next, j++)

if(strcmp(buffThis->name, temp->name) < 0)

Swap(\*Stations, i,j);

}

}

else if(field == 2)

{

for(i = 1, buffThis = \*Stations; i <= n; buffThis = buffThis->next, i++)

{

for(j = 1, temp = \*Stations; temp->next != NULL; temp = temp->next, j++)

if(strcmp(buffThis->address, temp->address) < 0)

Swap(\*Stations, i,j);

}

}

else if(field > 2 && field < 7)

{

field-=3;

for(i = 1, buffThis = \*Stations; i <= n; buffThis = buffThis->next, i++)

{

for(j = 1, temp = \*Stations ; temp->next != NULL; temp = temp->next, j++)

{

if(buffThis->fuelPrices[field] < temp->fuelPrices[field])

Swap(\*Stations, i,j);

}

}

}

else if(field == 7)

{

for(i = 1, buffThis = \*Stations; i <= n; i++, buffThis = buffThis->next)

for(j = 1, temp = \*Stations; j <= n; j++, temp = temp->next)

{

if(buffThis->rating < temp->rating)

Swap(\*Stations, i, j);

}

}

if(mode) //убывание

\*Stations = ReverseList(\*Stations);

puts("Сортировка прошла успешно!");

system("pause");

}

GSDesc\* ReverseList(GSDesc\* First)

{

GSDesc\* result = NULL;

GSDesc\* buff = NULL;

GSDesc\* buffThis = NULL;

for(buffThis = First; buffThis->next != NULL; buffThis = buffThis->next);

for(; buffThis != NULL; buffThis = buffThis->prev)

{

if(result == NULL)

{

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

buff = result;

PrepareStruct(buff);

CopyStruct(buff, buffThis);

buff->prev = NULL;

}

else

{

buff->next = (GSDesc\*)malloc(sizeof(GSDesc));

if(buff->next) buff->next->prev = buff;

buff = buff->next;

PrepareStruct(buff);

CopyStruct(buff, buffThis);

}

}

return result;

}

void GetItem(GSDesc\* Stations, int field) //Поиск: 1: адрес, 2: название, 3-6: цена на 92,95,98,дизель, 7: рейтинг, -1: все

{

system("cls");

GSDesc\* result = NULL;

GSDesc\* buff = NULL;

GSDesc\* buffThis = NULL;

char\* string = NULL;

string = (char\*)malloc((MAXLEN+1)\*sizeof(char));

puts("Введите, какое значение будем искать");

InputText(string);

puts("");

for(buffThis = Stations; buffThis != NULL; buffThis = buffThis->next)

{

if( ((field == 0 || field == 1) && !strcmp(string,buffThis->name)) || ((field == 0 || field == 2) && !strcmp(string,buffThis->address)) || ((field == 0 || field == 3) && atof(string) == buffThis->fuelPrices[0]) || ((field == 0 || field == 4) && atof(string) == buffThis->fuelPrices[1]) || ((field == 0 || field == 5) && atof(string) == buffThis->fuelPrices[2]) || ((field == 0 || field == 6) && atof(string) == buffThis->fuelPrices[3]) || ((field == 0 || field == 7) && atoi(string) == buffThis->rating) )

{

if(result == NULL)

{

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

buff = result;

PrepareStruct(buff);

CopyStruct(buff, buffThis);

buff->prev = NULL;

}

else

{

buff->next = (GSDesc\*)malloc(sizeof(GSDesc));

buff->next->prev = buff;

buff = buff->next;

PrepareStruct(buff);

CopyStruct(buff, buffThis);

}

}

}

if(result != NULL) OutputMenu(result);

else

puts("Ничего не найдено!"), system("pause");

result = free\_list(result);

}

GSDesc\* Process(GSDesc\* Stations) //Обработка данных

{

GSDesc\* result = NULL;

GSDesc\* buff = NULL;

do

{

if(Stations->rating > 7 && Stations->fuelPrices[2] < 54.0)

{

if(result == NULL)

{

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

buff = result;

CopyStruct(buff, Stations);

buff->prev = NULL;

}

else

{

buff->next = (GSDesc\*)malloc(sizeof(GSDesc));

buff->next->prev = buff;

buff = buff->next;

CopyStruct(buff, Stations);

}

}

Stations = Stations->next;

} while(Stations != NULL);

if(result == NULL) puts("Ни одна АЗС не подходит под критерии");

else puts("Обработка прошла успешно!");

system("pause");

return result;

}

GSDesc\* PushForward(GSDesc\* NewStations, GSDesc\* Stations) //Добавление в начало

{

GSDesc\* connector = NULL;

NewStations->prev = NULL;

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

connector->next = Stations;

if(Stations != NULL) Stations->prev = connector;

return NewStations;

}

GSDesc\* PushBack(GSDesc\* NewStations, GSDesc\* Stations) //Добавление в конец

{

GSDesc\* connector = NULL;

NewStations->prev = NULL;

connector = NewStations;

if(Stations != NULL)

{

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

connector->next = NewStations;

NewStations->prev = connector;

connector = Stations;

}

return connector;

}

GSDesc\* PushAnyPlace(GSDesc\* Stations) //Добавление в любое место

{

system("cls");

GSDesc\* NewStations = NULL;

GSDesc\* connector = NULL;

GSDesc\* buff = NULL;

GSDesc\* buff2 = NULL;

int ans,

i,

len;

len = ListLen(Stations);

printf("Введите номер места, на которое хотите добавить новые АЗС(от 1 до %d)\n", len+1);

do

{

scanf("%d", &ans);

if(ans < 1 || ans > len+1) printf("Введите от 1 до %d\n", len+1);

} while(ans < 1 || ans > len+1);

NewStations = InputStations();

if(ans == 1)

Stations = PushForward(NewStations, Stations);

else if(ans == len+1)

Stations = PushBack(NewStations, Stations);

else

{

buff = Stations;

for(i = 1; i < ans-1; i++)

buff = buff->next;

buff2 = buff->next;

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

buff->next = NewStations;

connector->next = buff2;

NewStations->prev = buff;

buff2->prev = connector;

}

return Stations;

}

void Swap(GSDesc\* Stations, int first, int second) //Поменять местами 2 элемента

{

GSDesc\* gFirst = NULL;

GSDesc\* gSecond = NULL;

GSDesc\* buff = NULL;

int i,

temp;

if(first > second)

{

temp = first;

first = second;

second = temp;

}

if(first != second)

{

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

PrepareStruct(buff);

for(i = 1, gFirst = Stations; i < first; i++, gFirst = gFirst->next);

for(i = 1, gSecond = Stations; i < second; i++, gSecond = gSecond->next);

CopyStruct(buff, gFirst);

CopyStruct(gFirst, gSecond);

CopyStruct(gSecond, buff);

free(buff);

}

}

//------------------------------------------------------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\* InputStations()

{

system("cls");

int i,

f;

GSDesc\* first = NULL;

GSDesc\* buff = NULL;

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

if(first != NULL)

{

first->prev = NULL;

for(f = 1, buff = first ; buff != NULL && f ;)

{

if(PrepareStruct(buff))

{

system("cls");

puts("Введите название АЗС");

InputText(buff->name);

puts("\nВведите адрес");

InputText(buff->address);

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

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

{

do

{

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

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

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

}

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

do

{

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

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

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

puts("Если хотите продолжить ввод данных, введите любое число, отличное от нуля");

scanf("%d", &f);

}

if(f)

{

buff->next = (GSDesc\*)malloc(sizeof(GSDesc));

buff->next->prev = buff;

buff = buff->next;

}

}

}

system("pause");

return first;

}

//------------------------------------------------------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");

}

void OutputGasStationsTableReverse(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->next != NULL; buff = buff->next);

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

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 OutputGasStationsTextReverse(GSDesc\* first)

{

fflush(stdout);

system("cls");

GSDesc\* buff = first;

int i;

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

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

{

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;

Station->prev = 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;

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

## 9.1 Исходные данные

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № при  ме  ра | Исходные данные | | | | Результаты | |
| Название | Адрес | Цены на топливо | Рей  тинг | Условия | |
| Рейтинг выше 7 | Цена на 98 бензин ниже 54 |
| 1 | Name | Address | 5 2 2,4 7 | 7 | Не подходит | |
| 2 | Dillimore | Dillimore village | 50 50,2 50,7 50,4 | 10 | Подходит | |

## 9.2 Вывод программы

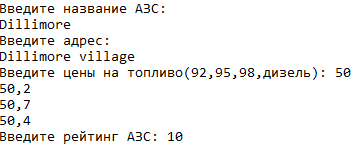


Рисунок 2. Пример ввода в программе

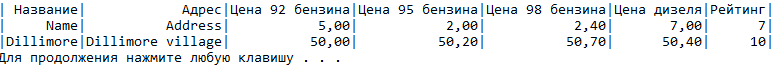
**

Рисунок 3. Вывод исходной информации в виде таблицы

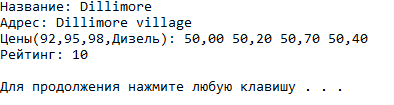
**

Рисунок 4. Вывод обработанной информации в виде блоков из строк

## 9.3 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 2, 3, 4.

# Заключение

При выполнении лабораторной работы получены практические навыки в реализации двусвязного списка структур на языке программирования C/C++.