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 № 2**

**по дисциплине «Программирование» на**

**тему:** **«Односвязные 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](#_Toc41159534)

[1. Задание 4](#_Toc41159535)

[2. Уточнение задания 4](#_Toc41159536)

[3. Контрольные примеры 6](#_Toc41159537)

[4. Краткое описание алгоритма 7](#_Toc41159538)

[5. Структура вызова функций 8](#_Toc41159539)

[6. Функции 8](#_Toc41159540)

[6.1 Главная функция 9](#_Toc41159541)

[6.2 Функция Menu 11](#_Toc41159542)

[6.3 Функция OutputMenu 12](#_Toc41159543)

[6.4 Функция InputMenu 13](#_Toc41159544)

[6.5 Функция Help 14](#_Toc41159545)

[6.6 Функция ListActions 15](#_Toc41159546)

[6.7 Функция ListLen 16](#_Toc41159547)

[6.8 Функция DeleteItem 17](#_Toc41159548)

[6.9 Функция SortByRating 18](#_Toc41159549)

[6.10 Функция Swap 19](#_Toc41159550)

[6.11 Функция GetItem 20](#_Toc41159551)

[6.12 Функция PushBack 21](#_Toc41159552)

[6.13 Функция PushForward 22](#_Toc41159553)

[6.14 Функция PushAnyPlace 23](#_Toc41159554)

[6.15 Функция InputStations 24](#_Toc41159555)

[6.16 Функция Process 25](#_Toc41159556)

[6.17 Функция PrepareStruct 26](#_Toc41159557)

[6.18 Функция CopyStruct 27](#_Toc41159558)

[6.19 Функция OutputGasStationsTable 28](#_Toc41159559)

[6.20 Функция OutputGasStationsText 29](#_Toc41159560)

[6.21 Функция free\_station 30](#_Toc41159561)

[6.22 Функция free\_list 31](#_Toc41159562)

[6.23 Функция InputText 32](#_Toc41159563)

[8. Текст программы 33](#_Toc41159564)

[9. Пример работы программы 51](#_Toc41159565)

[Исходные данные 51](#_Toc41159566)

[Вывод программы 52](#_Toc41159567)

[9.1 Результат работы программы 53](#_Toc41159568)

[Заключение 54](#_Toc41159569)

Введение

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

# 1. Задание

Написать программу, выполняющую ввод последовательности структур, длина которой заранее неизвестна. Контрольный вывод исходной последовательности, формирование новой последовательности из элементов исходной, поля которых удовлетворяют заданным условиям, совершение действий над исходной последовательностью: узнать кол-во элементов списка, удалить элемент списка, поменять местами 2 элемента в списке, 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
6. Сортировка производится по рейтингу, начиная с самого высокого

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\* | Цены на 4 вида топлива |
| rating | int\* | Рейтинг |
| next | struct Gaslist\* | Указатель на следующую 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 на 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

Пункт 2-ый: Ввод

Пункт 3-ый: Вывод введенных данных

Пункт 4-ый: Действия над списком

Пункт 5-ый: Обработка

Пункт 6-ый: Вывод результата обработки

Пункт 0-ой: Переход к шагу 4.

Шаг №4: Выход, 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 |

**Схема алгоритма:**

Схема алгоритма для функции main представлена на рисунке 2.

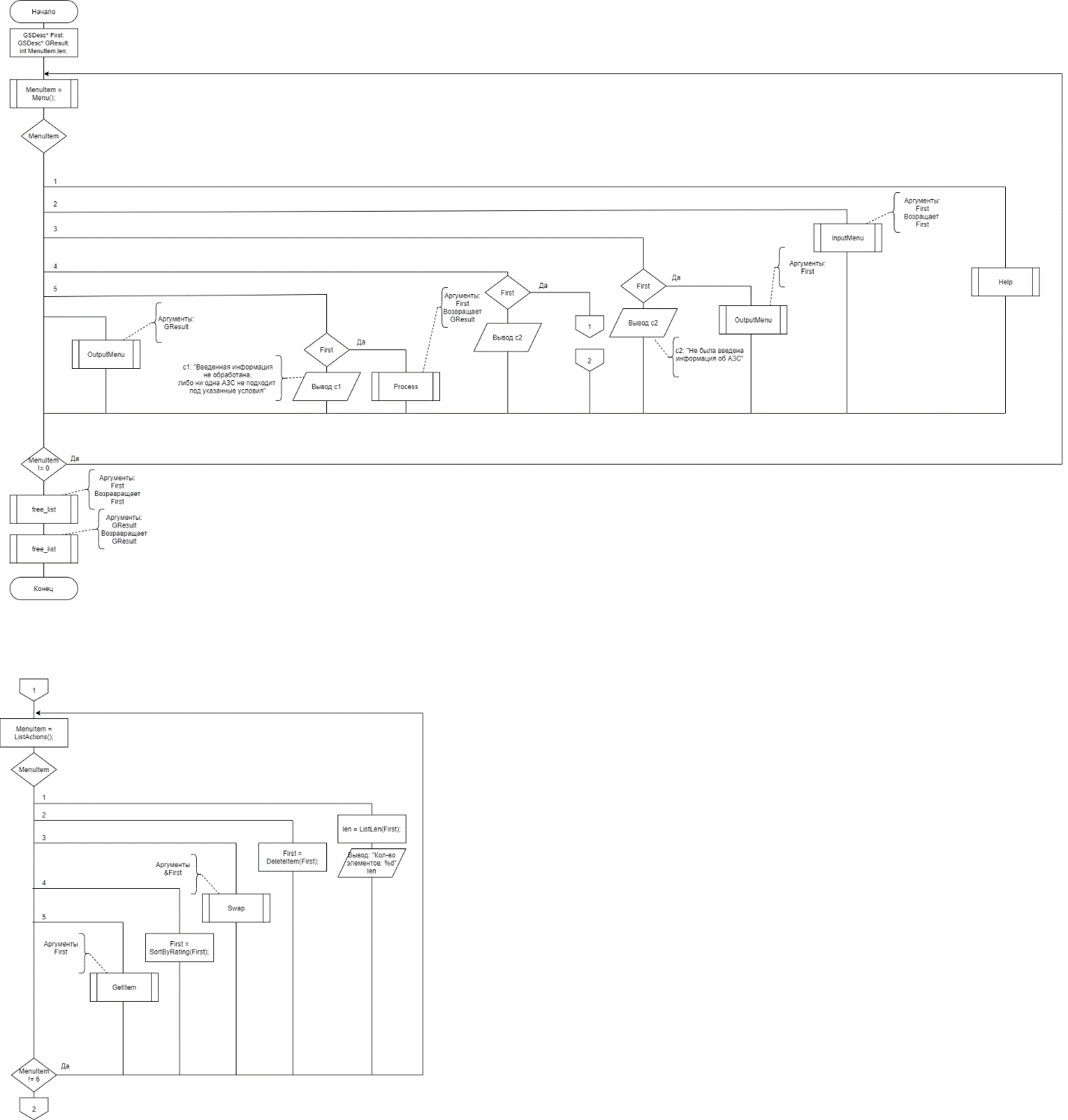


Рисунок 2. Схема алгоритма

## 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\*);

**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 |

## 6.4 Функция InputMenu

**Назначение:**

Меню ввода данных

**Прототип:**

GSDesc\* InputMenu(GSDesc\*);

**Пример вызова:**

First = InputMenu(First);

**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();

**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\*);

**Пример вызова:**

len = ListLen(First);

**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\*);

**Пример вызова:**

First = DeleteItem(First);

**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 SortByRating

**Назначение:**

Функция сортировки списка по убыванию рейтинга

**Прототип:**

GSDesc\* SortByRating(GSDesc\*);

**Пример вызова:**

First = SortByRating(First);

**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 |
| 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 |

## 6.10 Функция Swap

**Назначение:**

Функция перестановки 2 элементов местами

**Прототип:**

void Swap(GSDesc\*\*);

**Пример вызова:**

Swap(&First);

**Описание переменных:**

Описание переменных приведено в таблице 10.

Таблица 10. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Stations | GSDesc\*\* | Указатель на указатель начала списка структур |
| 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 |
| buff2 | GSDesc\* | Указатель на структуру, стоящую после второго элемента для смены |
| buff3 | GSDesc\* | Указатель на структуру, стоящую после первого элемента для смены |
| first | int | Номер 1-го элемента для смены |
| second | int | Номер 2-го элемента 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\*);

**Пример вызова:**

GetItem(First);

**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 |
| Локальные | result | GSDesc\* | Указатель на начало списка структур, удовлетворяющим условиям |
| buff | GSDesc\* | Указатель на структуру из результирующего списка |
| ans | int | Категория для поиска |
| num | int | Число для поиска |
| price | float | Цена топлива для поиска |

## 6.12 Функция PushBack

**Назначение:**

Функция вставки в конец списка

**Прототип:**

GSDesc\* PushBack(GSDesc\*, GSDesc\*);

**Пример вызова:**

Stations = PushBack(NewStations, Stations);

**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\*);

**Пример вызова:**

Stations = PushForward(NewStations, Stations);

**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\*);

**Пример вызова:**

Stations = PushAnyPlace(Stations);

**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();

**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\*);

**Пример вызова:**

GResult = Process(First);

**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\*);

**Пример вызова:**

if(PrepareStruct(buff))

**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\*);

**Пример вызова:**

CopyStruct(buff, temp);

**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\*);

**Пример вызова:**

OutputGasStationsTable(first);

**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\*);

**Пример вызова:**

OutputGasStationsText(first);

**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\*);

**Пример вызова:**

toDel = free\_station(toDel);

**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\*);

**Пример вызова:**

result = free\_list(result);

**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 InputText

**Назначение:**

Функция ввода строки

**Прототип:**

void InputText(char\*);

**Пример вызова:**

InputText(buff->name);

**Описание переменных:**

Описание переменных приведено в таблице 23.

Таблица 23. 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 |

# 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 Menu(); //Меню

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

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

void Help(); //Справка

int ListActions(); //Действия со списком

int ListLen(GSDesc\*); //Длина списка

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

GSDesc\* SortByRating(GSDesc\*); //Сортировка по убыванию рейтинга

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

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

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

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, len;

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:

Swap(&First);

break;

case 4:

First = SortByRating(First);

break;

case 5:

GetItem(First);

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 подходит под условия!");

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 обработанных 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("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);

}

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("имеющим 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;

temp = free\_station(temp);

}

else

{

toDel = temp->next;

while(num > 2)

{

temp = temp->next;

toDel = temp->next;

num--;

}

temp->next = toDel->next;

toDel = free\_station(toDel);

}

return Stations;

}

GSDesc\* SortByRating(GSDesc\* Stations) //Сортировка по рейтингу по убыванию

{

GSDesc\* result = NULL;

GSDesc\* temp = NULL;

GSDesc\* buff = NULL;

int num;

for(num = 10; ListLen(result) != ListLen(Stations) ; num--)

{

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

{

if(temp->rating == num)

{

if(result == NULL)

{

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

buff = result;

CopyStruct(result, temp);

}

else

{

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

buff = buff->next;

CopyStruct(buff, temp);

}

}

}

}

if(result == NULL) puts("Упс... Что-то пошло не так! Попробуйте еще раз!");

else puts("Сортировка прошла успешно!");

Stations = free\_list(Stations);

system("pause");

return result;

}

void GetItem(GSDesc\* Stations)

{

system("cls");

GSDesc\* result = NULL;

GSDesc\* buff = NULL;

int ans,

num;

float price;

puts("Выберете категорию, по которой мы ищем");

puts("1 - Цена на топливо");

puts("2 - Рейтинг");

puts("3 - Отмена");

do

{

scanf("%d", &ans);

if(ans < 1 || ans > 3) puts("Введите от 1 до 3");

} while(ans < 1 || ans > 3);

if(ans == 1)

{

puts("\nВыберете, по цене какого топлива будем искать\n1 - 92 бензин\n2 - 95 бензин\n3 - 98 бензин\n4 - Дизель");

do

{

scanf("%d", &num);

if(num < 1 || num > 4) puts("Введите от 1 до 4");

} while(num < 1 || num > 4);

printf("\nВведите цену, по которой будем искать: ");

do

{

scanf("%f", &price);

if(price < 0) puts("Цена должна быть выше 0!");

} while(price < 0);

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

{

if(Stations->fuelPrices[num-1] == price)

{

if(result == NULL)

{

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

buff = result;

CopyStruct(buff, Stations);

}

else

{

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

buff = buff->next;

CopyStruct(buff, Stations);

}

}

}

}

else if(ans == 2)

{

printf("Введите рейтинг, по которому будем искать(от 1 до 10): ");

do

{

scanf("%d", &num);

if(num < 1 || num > 10) puts("Рейтинг должен быть от 1 до 10!");

} while(num < 1 || num > 10);

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

{

if(Stations->rating == num)

{

if(result == NULL)

{

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

buff = result;

CopyStruct(buff, Stations);

}

else

{

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

buff = buff->next;

CopyStruct(buff, Stations);

}

}

}

}

if(ans != 3)

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);

}

else

{

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

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;

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

connector->next = Stations;

return NewStations;

}

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

{

GSDesc\* connector = NULL;

connector = NewStations;

if(Stations != NULL)

{

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

connector->next = NewStations;

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;

}

return Stations;

}

void Swap(GSDesc\*\* Stations) //Поменять местами 2 элемента

{

GSDesc\* gFirst = NULL;

GSDesc\* gSecond = NULL;

GSDesc\* buff = NULL;

GSDesc\* buff2 = NULL;

GSDesc\* buff3 = NULL;

int first,

second,

temp,

i;

printf("Введите 2 номера элементов, которые хотите поменять местами(всего %d элементов): ", ListLen(\*Stations));

do

{

scanf("%d %d", &first, &second);

if(first < 1 || second < 1 || first > ListLen(\*Stations) || second > ListLen(\*Stations)) printf("Данных элементов нет в списке, выберите от 1 до %d\n", ListLen(\*Stations));

} while(first < 1 || second < 1 || first > ListLen(\*Stations) || second > ListLen(\*Stations));

if(first != second)

{

if(first > second)

{

temp = second;

second = first;

first = second;

}

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

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

if(first == 1)

{

for(buff = \*Stations; buff->next != gSecond; buff = buff->next);

buff->next = gFirst;

buff = gFirst->next;

buff2 = gSecond->next;

gFirst->next = buff2;

gSecond->next = buff;

\*Stations = gSecond;

}

else if(second - first == 1)

{

for(buff = \*Stations; buff->next != gFirst; buff = buff->next);

buff->next = gSecond;

buff2 = gSecond->next;

gSecond->next = gFirst;

gFirst->next = buff2;

}

else

{

for(buff = \*Stations; buff->next != gFirst; buff = buff->next);

for(buff2 = \*Stations; buff2->next != gSecond; buff2 = buff2->next);

buff3 = gFirst->next;

buff->next = gSecond;

buff2->next = gFirst;

gFirst->next = gSecond->next;

gSecond->next = buff3;

}

}

system("pause");

}

//------------------------------------------------------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)

{

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 = 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");

}

//------------------------------------------------------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 = 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  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 | |

## Вывод программы

Рисунок 3. Пример ввода в программе

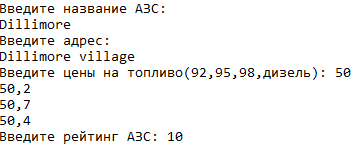


Рисунок 4. Вывод исходной информации в виде таблицы

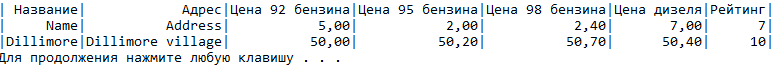
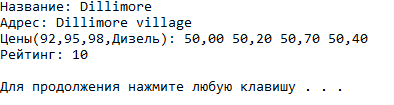


Рисунок 5. Вывод обработанной информации в виде блоков из строк



## 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.

# Заключение

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