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

Пояснительная записка

к курсовой работе

на тему «Электронная картотека»

по дисциплине «Программирование»

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

Санкт-Петербург,

2020

Оглавление

[Введение 3](#_Toc41760248)

[1. Задание 3](#_Toc41760249)

[2. Уточнение задания 3](#_Toc41760250)

[3. Контрольные примеры 6](#_Toc41760251)

[4. Краткое описание алгоритма 7](#_Toc41760252)

[5. Структура вызова функций 9](#_Toc41760253)

[6. Функции 10](#_Toc41760254)

[6.1 Главная функция 10](#_Toc41760255)

[6.2 Функция Menu 11](#_Toc41760256)

[6.3 Функция OutputMenu 12](#_Toc41760257)

[6.4 Функция InputMenu 13](#_Toc41760258)

[6.5 Функция Help 14](#_Toc41760259)

[6.6 Функция ListActions 15](#_Toc41760260)

[6.7 Функция ListLen 16](#_Toc41760261)

[6.8 Функция DeleteItem 17](#_Toc41760262)

[6.9 Функция Sort 18](#_Toc41760263)

[6.10 Функция Swap 19](#_Toc41760264)

[6.11 Функция GetItem 20](#_Toc41760265)

[6.12 Функция PushBack 21](#_Toc41760266)

[6.13 Функция PushForward 22](#_Toc41760267)

[6.14 Функция PushAnyPlace 23](#_Toc41760268)

[6.15 Функция InputStations 24](#_Toc41760269)

[6.16 Функция Process 25](#_Toc41760270)

[6.17 Функция PrepareStruct 26](#_Toc41760271)

[6.18 Функция CopyStruct 27](#_Toc41760272)

[6.19 Функция OutputGasStationsTable 28](#_Toc41760273)

[6.20 Функция OutputGasStationsText 29](#_Toc41760274)

[6.21 Функция free\_station 30](#_Toc41760275)

[6.22 Функция free\_list 31](#_Toc41760276)

[6.23 Функция OutputGasStationsTableReverse 32](#_Toc41760277)

[6.24 Функция OutputGasStationsTextReverse 33](#_Toc41760278)

[6.25 Функция InputText 34](#_Toc41760279)

[6.26 Функция ReverseList 35](#_Toc41760280)

[6.27 Функция EditItem 36](#_Toc41760281)

[6.28 Функция InputTextFile 37](#_Toc41760282)

[6.29 Функция OutputFile 38](#_Toc41760283)

[6.30 Функция fillStruct 39](#_Toc41760284)

[7. Текст программы 40](#_Toc41760285)

[8. Результаты тестирования программы 66](#_Toc41760286)

[9. Инструкция пользователю 69](#_Toc41760287)

[Заключение 70](#_Toc41760288)

Введение

Целью курсовой работы является законченное поэтапное решение содержательной задачи (постановка задачи, спецификация, выбор структур данных и разработка алгоритма, программная реализация, тестирование).

# 1. Задание

Создать электронную картотеку, хранящуюся на диске. С картотекой возможны следующие действия: занесение данных, внесение изменений(исключение, корректировка, добавление), поиск данных, сортировка, отбор по определенным условиям. Вывод результаты работы и сделать возможность сохранения на диске.

# 2. Уточнение задания

При выполнении задания следует учесть:

1. Максимальный размер строки - 80 символов, т. к. стандартный размер строки в командной оболочке – 80.
2. Нужно построить меню таким образом, чтобы пользователь не смог начать изменять картотеку, пока он не введет ее.
3. Количество критериев для обработки исходных данных – 2: рейтинг выше 7 и цена на 98 бензин ниже 54
4. Из подменю должна быть возможность выйти.

Меню программы имеет следующую иерархию:

1 – Справка

2 – Ввод данных

2.1 – Добавить в начало

2.2 – Добавить в конец

2.3 – Добавить на определенное место

2.4 – Ввод полностью заново

2.5 – Получить данные из файла

2.6 - Назад

3 – Контрольный вывод данных

3.1 – Вывод в виде таблицы

3.2 – Вывод в виде блоков построчно

3.3 – Вывод в виде таблицы, начиная с конца

3.4 – Вывод в виде блоков построчно. начиная с конца

3.5 – Вывод в файл

3.6 - Назад

4 – Действия над картотекой

4.1 – Узнать количество элементов картотеки

4.2 – Удалить элемент картотеки

4.3 – Поменять местами 2 элемента картотеки

4.4 – Сортировка картотеки

4.5 – Поиск элементов по заданным параметрам

4.6 – Изменить данные элемента картотеки

4.7 – Назад

5 – Обработка данных

6 – Вывод результата

6.1 – Вывод в виде таблицы

6.2 – Вывод в виде блоков построчно

6.3 – Вывод в виде таблицы, начиная с конца

6.4 – Вывод в виде блоков построчно. начиная с конца

6.5 – Вывод в файл

6.6 - Назад

0 – выход.

Пункты меню 3,4,5,6 недоступны, пока пользователь не введет все исходные данные, также выводится соответствующее сообщение. Пункт 6 недоступен, пока пользователь не выполнит обработку(пункт 5). Если пользователь выберет несуществующий пункт меню, то нужно ему сообщить об этом и предложить выбрать пункт меню заново.

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 |

# 3. Контрольные примеры

1. Исходный список:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| test1 | ad1 | 1 57 24 23 | 8 |
| this | smth | 60 60 60 60 | 10 |
| outthis | sdjghdfkhsfs | 2 7 5,7 5,9 | 9 |

Изменим во 2-м элементе цены на топлива на 10 10 10 10

Ожидаемый результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| test1 | ad1 | 1 57 24 23 | 8 |
| this | smth | 10 10 10 10 | 10 |
| outthis | sdjghdfkhsfs | 2 7 5,7 5,9 | 9 |

2) Исходный список:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| name | a1 | 1 1 1 1 | 2 |
| name2 | a2 | 2 2 87 5 | 6 |
| name3 | a3 | 6,7 57 51 5,9 | 7 |

Поменяем местами 2 и 3 структуры, предварительно изменив рейтинг 2-й с 6 на 3. Далее удалим первый элемент списка

Ожидаемый результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| name3 | a3 | 6,7 57 51 5,9 | 7 |
| name2 | a2 | 2 2 87 5 | 3 |

3) Исходный список:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |

Проведем поиск по картотеке по следующим критериям: искать во всех полях значение «10»

Ожидаемый результат:

Программа должна вывести все три структуры.

4) Исходный список:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |

Проведем сортировку по название по алфавиту

Ожидаемый результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |

# 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: Вывод справки для пользователя(функция *Help*)

Пункт 2-ой: Ввод исходных данных(функция *InputMenu*), открытие подменю.

Пункт 2.1-ый: Добавление данных в начало(функция *PushForward*) и переход к шагу 2.

Пункт 2.2-ой: Добавление данных в конец(функция *PushBack*) и переход к шагу 2.

Пункт 2.3-ий: Добавление данных на определенное место(функция *PushAnyPlace*) и переход к шагу 2.

Пункт 2.4-ый: Ввод данных полностью заново(функция *InputStations*) и переход к шагу 2.

Пункт 2.5-ый: Получение данных из файла(функция *InputTextFile*) и переход к шагу 2.

Пункт 2.0-ый: Переход к шагу 1.

Пункт 3-ий: Контрольный вывод данных(функция *OutputMenu*) открытие подменю.

Пункт 3.1-ый: Вывод данных в виде таблицы(функция *OutputGasStationsTable*) и переход к шагу 3.

Пункт 3.2-ой: Вывод данных в виде блоков построчно(функция *OutputGasStationText*) и переход к шагу 3.

Пункт 3.3-ий: Вывод данных в виде таблицы, начиная с конца(функция *OutputGasStationsTableReverse*) и переход к шагу 3.

Пункт 3.4-ый: Вывод данных в виде блоков построчно, начиная с конца(функция *OutputGasStationTextReverse*) и переход к шагу 3.

Пункт 3.0-ый: Переход к шагу 1.

Пункт 4-ый: Вывод подменю действий над картотекой(функция *ListActions*)

Пункт 4.1-ый: Вывод количества данных в картотеке(функция *ListLen*) и переход к шагу 4.

Пункт 4.2-ой: Удаление элемента картотеки(функция *DeleteItem*) и переход к шагу 4.

Пункт 4.3-ий: Смена мест 2-х элементов(функция *Swap*) и переход к шагу 4.

Пункт 4.4-ый: Сортировка картотеки(функция *Sort*) и переход к шагу 4.

Пункт 4.5-ый: Поиск элементов по заданным параметрам(функция *GetItem*) и переход к шагу 4.

Пункт 4.6-ый: Изменение данных элемента(функция *EditItem*) и переход к шагу 4.

Пункт 4.0-ый: Переход к шагу 1.

Пункт 5-ый: Обработка результата(функция *Process*)

Пункт 6-ый: Вывод результата(функция *OutputMenu*), открытие подменю

Пункт 6.1-ый: Вывод данных в виде таблицы(функция *OutputGasStationsTable*) и переход к шагу 6.

Пункт 6.2-ой: Вывод данных в виде блоков построчно(функция *OutputGasStationText*) и переход к шагу 6.

Пункт 6.3-ий: Вывод данных в виде таблицы, начиная с конца(функция *OutputGasStationsTableReverse*) и переход к шагу 6.

Пункт 6.4-ый: Вывод данных в виде блоков построчно, начиная с конца(функция *OutputGasStationTextReverse*) и переход к шагу 3.

Пункт 6.0-ый: Переход к шагу 6.

Пункт 0: Переход к шагу 1.

Пункт 0-ой: Переход к шагу 4.

Шаг №4. 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 |
| 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 |

**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(); MenuItem описана в таблице 2.

**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 убыванию рейтинга

**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 |

## 6.27 Функция EditItem

**Назначение:**

Функция изменения данных элемента картотеки

**Прототип:**

void EditItem(GSDesc\*, int); Передается указатель на первую структуру списка и номер элемента, который необходимо изменить

**Пример вызова:**

EditItem(First, index1); First и index1 описаны в таблице 2.

**Описание переменных:**

Описание переменных приведено в таблице 27.

Таблица 27. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | First | GSDesc\* | Указатель на начало списка |
| index | int | Номер элемента, который изменяем |
| Локальная | buff | GSDesc\* | Указатель на структуру, информацию в котором меняем |
| i | int | Переменная для цикла for |

## 6.28 Функция InputTextFile

**Назначение:**

Функция считывания данных с файла

**Прототип:**

GSDesc\* InputTextFile();

**Пример вызова:**

Stations = InputTextFile(); Stations описана в таблице 5.

**Описание переменных:**

Описание переменных приведено в таблице 28.

Таблица 28. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Локальная | newFirst | GSDesc\* | Указатель на первую структуру, считанную с файла |
| buff | GSDesc\* | Указатель на текущую структуру, считанную с файла |
| text | char\* | Указатель на строку, содержащую всю информацию из файла |
| string | char\* | Указатель на строку, содержащую всю информацию о текущей структуре, считанной с файла |
| file | FILE\* | Указатель на файл |
| filename | char | Название файла |
| ans | int | Флаг для ввода собственного имени файла |
| i | int | Переменная для цикла for |
| f | int | Флаг конца файла |
| size | int | Количество символов в файле |
| frows | int | Количество строк в файле |
| strsize | int | Размер текущей строки файла |
| linebeg | int | Индекс начала текущей строки |

## 6.29 Функция OutputFile

**Назначение:**

Функция вывода информации в файл

**Прототип:**

void OutputFile(GSDesc\*); Передается адрес первого элемента картотеки

**Пример вызова:**

OutputFile(first); first описана в таблице 4.

**Описание переменных:**

Описание переменных приведено в таблице 29.

Таблица 29. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | first | GSDesc\* | Указатель на начало списка |
| Локальная | file | FILE\* | Указатель на файл |
| buff | GSDesc\* | Указатель на текущую структуру из списка |
| filename | char | Массив, хранящий имя файла |
| ans | int | Флаг ввода своего имени файла |
| namelen | int | Максимальная длина названия |
| addresslen | int | Максимальная длина адреса |

## 6.30 Функция fillStruct

**Назначение:**

Функция вывода информации в файл

**Прототип:**

GSDesc\* fillStruct(char\* string); Передается строка, содержащая информацию для заполнения полей структуры

**Пример вызова:**

newFirst = fillStruct(string); newFirst и string описаны в таблице 28

**Описание переменных:**

Описание переменных приведено в таблице 30.

Таблица 30. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | string | char\* | Указатель на строку, содержащую информацию о структуре |
| Локальная | result | GSDesc\* | Указатель на структуру |
| i | int | Переменная для цикла for |
| len | int | Длина строки |
| j | int | Переменная для цикла for |
| index | int | Индекс начала в строке текущего значения для поля |
| buffi | int | Количество полученных значений полей |
| empty | char | Строка “Unknown” на случай, когда адрес/название не указаны |
| buff | char\*\* | Указатель на вектор строк, содержащих информацию для стуктуры |

# 7. 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\*, int); //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

void EditItem(GSDesc\*, int); //Изменение элемента

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

void InputText(char\*); //Ввод строки

GSDesc\* InputTextFile(); //Ввод текста с файла

GSDesc\* fillStruct(char\* string); //Заполняем структуры из данных файла

void OutputFile(GSDesc\*); //Вывод в файл

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

{

len = ListLen(First);

MenuItem = ListActions();

switch(MenuItem)

{

case 1:

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

system("pause");

break;

case 2: //удалить

system("cls");

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

do

{

scanf("%d", &index1);

if(index1 < 1 || index1 > len)

puts("Нет элемента под таким номером");

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

First = DeleteItem(First, index1);

break;

case 3: //поменять местами

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: //сортировка

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: //поиск

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;

case 6: //изменение

system("cls");

printf("Введите номер элемента, который хотите изменить(от 1 до %d): ", len);

do

{

scanf("%d", &index1);

if(index1 < 1 || index1 > len)

printf("Введенный номер должен быть от 1 до %d!\n", len);

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

EditItem(First, index1);

break;

}

} while(MenuItem != 0);

if(MenuItem == 0) MenuItem = 1284;

}

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;

}

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("5 - В файл");

puts("0 - 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 == 1) OutputGasStationsTable(first);

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

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

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

else if(item == 5) OutputFile(first);

}while(item);

}

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 - Изменить данные АЗС");

puts("0 - 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;

}

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 - Получить данные из файла");

puts("0 - 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 > 0 && 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();

}

else if(item == 5)

{

Stations = free\_list(Stations);

Stations = InputTextFile();

}

}while(item != 0);

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 список");

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

puts("поиск АЗС по определенному параметру, изменение данных одной АЗС\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, int index)

{

system("cls");

GSDesc\* temp = Stations;

GSDesc\* toDel = NULL;

if(index == 1)

{

Stations = Stations->next;

if(Stations) Stations->prev = NULL;

temp = free\_station(temp);

}

else

{

toDel = temp->next;

while(index > 2)

{

temp = temp->next;

toDel = temp->next;

index--;

}

temp->next = toDel->next;

if(temp->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));

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);

}

void EditItem(GSDesc\* Stations, int index)

{

GSDesc\* buff = Stations;

int i;

for(i = 1; i < index; i++, buff = buff->next)

system("cls");

printf("Введите название АЗС. Текущее название: %s\n", buff->name);

InputText(buff->name);

printf("\nВведите адрес. Текущий адрес: %s\n", buff->address);

InputText(buff->address);

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

printf("Текущие значения: %.2f %.2f %.2f %.2f\n", buff->fuelPrices[0], buff->fuelPrices[1], buff->fuelPrices[2], buff->fuelPrices[3]);

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);

}

printf("Введите рейтинг АЗС от 1 до 10. Текущий рейтинг: %d\n", buff->rating);

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);

}

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;

PrepareStruct(buff);

CopyStruct(buff, Stations);

buff->prev = NULL;

}

else

{

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

buff->next->prev = buff;

buff = buff->next;

PrepareStruct(buff);

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;

}

GSDesc\* InputTextFile()

{

system("cls");

GSDesc\* newFirst = NULL;

GSDesc\* buff = NULL;

char \*text = NULL,

\*string = NULL;

FILE \*file = NULL;

char filename[128];

int ans,

i,

f,

size,

frows,

strsize,

linebeg; //индекс начала очередной строки

puts("Хотите указать свое имя файла? Введите любое число, отличное от нуля\nИначе будет использован файл in.txt");

scanf("%d", &ans);

fflush(stdin);

if(ans)

{

puts("\nВведите имя файла для чтения. Имя файла не должно превышать 128 символов!");

gets(filename);

fflush(stdin);

file = fopen(filename, "r");

}

else file = fopen("in.txt", "r");

if(file != NULL)

{

fflush(file);

fseek(file, 0, SEEK\_SET);

fseek(file,0,SEEK\_END);

size = ftell(file);

rewind(file);

if(size == 0)

puts("Файл пуст!");

text = (char\*)malloc(size\*sizeof(char));

if(text != NULL && size)

{

fread(text, size, 1, file);

if(fclose(file) == EOF) puts("Error closing!");

frows = 0; f = 0;

for(i = 0; i < size+1 && !f; i++)

{

if(text[i] <= 0) f = 1, text[i] = '\0';

if(text[i] == '\n' || text[i] == '\0')

frows++;

}

if(frows > 0)

{

linebeg = 0;

do

{

strsize = 0;

while(text[linebeg+strsize] != '\n' && text[linebeg+strsize] != '\0') strsize++;

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

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

string[i] = text[linebeg+i];

string[i] = '\0';

if(newFirst)

{

buff->next = fillStruct(string);

buff->next->prev = buff;

buff = buff->next;

}

else

{

newFirst = fillStruct(string);

buff = newFirst;

buff->prev = NULL;

}

frows--;

while(text[linebeg] != '\n' && text[linebeg] != '\0') linebeg++;

linebeg++;

free(string);

} while(frows != 0);

puts("Список заполнен!");

}

else

puts("Все строки пустые!");

}

free(text);

text = NULL;

}

else puts("Не удалось открыть файл с таким именем!");

system("pause");

return newFirst;

}

GSDesc\* fillStruct(char\* string)

{

GSDesc\* result = NULL;

int i,

len,

j,

index,

buffi;

char empty[] = "Unknown\0";

char\*\* buff = NULL;

len = strlen(string);

index = 0; buffi = 0;

buff = (char\*\*)malloc(7\*sizeof(char\*));

do

{

i = 0;

while(string[i+index] != ';' && string[i+index] != '\0') i++;

buff[buffi] = (char\*)malloc((i+1)\*sizeof(char));

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

buff[buffi][j] = string[index+j];

buff[buffi][j] = '\0';

buffi++;

index += i+1;

} while(index < len);

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

if(PrepareStruct(result))

{

if(buffi != 0)

strcpy(result->name, buff[0]);

else

strcpy(result->name, empty);

if(buffi > 1 && strlen(buff[1]) > 0)

strcpy(result->address, buff[1]);

else

strcpy(result->address, empty);

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

{

if(buffi > 2+i)

if(atof(buff[2+i]) > 0)

result->fuelPrices[i] = atof(buff[2+i]);

else

result->fuelPrices[i] = 0.0;

else

result->fuelPrices[i] = 0.0;

}

if(buffi > 6)

if(atoi(buff[6]) > 0 && atoi(buff[6]) < 11)

result->rating = atoi(buff[6]);

else if(atoi(buff[6]) > 0 && atoi(buff[6]) > 10)

result->rating = 10;

else

result->rating = 0;

else

result->rating = 0;

}

for(i = 0; i < 7; i++)

free(buff[i]);

free(buff);

return result;

}

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

}

void OutputFile(GSDesc\* first)

{

system("cls");

FILE \*file = NULL;

GSDesc\* buff = first;

char filename[128];

int ans,

namelen,

addresslen;

ans = 0;

puts("Хотите указать имя файла самостоятельно? Введите любое число, отличное от нуля.\nВ противном случае все сохранится в out.txt");

scanf("%d", &ans);

fflush(stdin);

if(ans)

{

puts("Введите имя файла для сохранения");

gets(filename);

file = fopen(filename, "w");

}

else file = fopen("out.txt", "w");

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);

}

fprintf(file, "|%\*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)

fprintf(file, "|%\*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);

fclose(file);

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;

}

# 8. Результаты тестирования программы

Тестирование программного обеспечения — это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом.

Для поиска ошибок были выбраны тесты:

1) Тест первый:

Список:

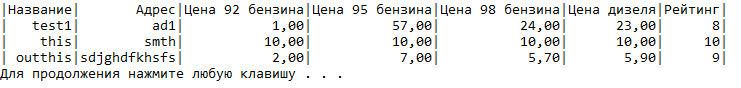
|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| test1 | ad1 | 1 57 24 23 | 8 |
| this | smth | 60 60 60 60 | 10 |
| outthis | sdjghdfkhsfs | 2 7 5,7 5,9 | 9 |

Изменим во 2-м элементе цены на топлива на 10 10 10 10

Ожидаемый результат:

После изменения 2-го элемента картотеки программа должна вывести все три структуры после обработки.

Программа отработала правильно при таком тесте. Результат выполнения приведен на рисунке.



*Рисунок . Пример вывода программы*

2) Тест второй

Список:

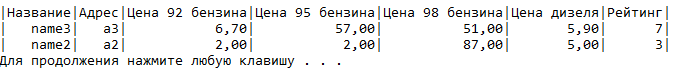
|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| name | a1 | 1 1 1 1 | 2 |
| name2 | a2 | 2 2 87 5 | 6 |
| name3 | a3 | 6,7 57 51 5,9 | 7 |

Поменяем местами 2 и 3 структуры, предварительно изменив рейтинг 2-й с 6 на 3. Далее удалим первый элемент списка

Ожидаемый результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| name3 | a3 | 6,7 57 51 5,9 | 7 |
| name2 | a2 | 2 2 87 5 | 3 |

Программа отработала правильно при таком тесте. Результат выполнения приведен на рисунке.

****

*Рисунок . Пример вывода программы*

3) Тест третий

Список:

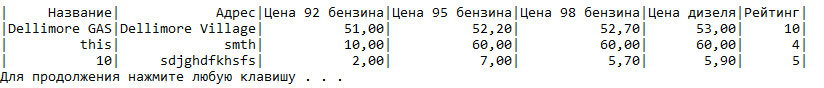
|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |

Проведем поиск по картотеке по следующим критериям: искать во всех полях значение «10»

Ожидаемый результат:

Программа должна вывести все три структуры.

Программа отработала правильно при таком тесте. Результат выполнения приведен на рисунке.



*Рисунок . Пример вывода программы*

4) Тест четвертый

Список:

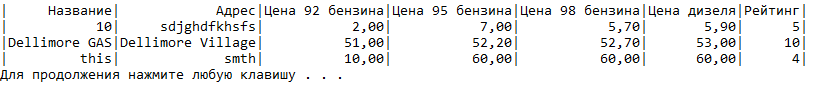
|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |

Проведем сортировку по название по алфавиту

Ожидаемый результат:

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Адрес** | **Цены на топливо** | **Рейтинг** |
| 10 | sdjghdfkhsfs | 2 7 5,7 5,9 | 5 |
| Dellimore GAS | Dellimore Village | 51 52,2 52,7 53 | 10 |
| this | smth | 10 60 60 60 | 4 |

Программа отработала правильно при таком тесте. Результат выполнения приведен на рисунке.



*Рисунок . Пример вывода программы*

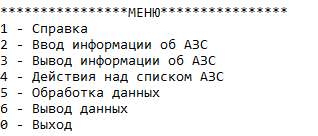
# 9. Инструкция пользователю

При запуске программы Вы увидите меню, такое же, как показано на рисунке . Вас просят выбрать пункт меню. При вводе 1, Вы получите справку о том, для чего предназначена программа. При вводе 2, Вам откроется подменю, в котором Вы сможете выбрать, как ввести элементы картотеки.

В 3-м пункте меню Вы можете посмотреть, что было введено Вами и проверить правильность ввода.

В 4-м пункте меню Вы можете изменять введенную Вами картотеку: удалить элемент, поменять местами 2 любых элемента, отсортировать картотеку, найти элементы по определенным критериям, изменить данные элемента.

При выборе пункта 5 программа обработает заданную Вами картотеку по правилам, указанным в справке(пункт 1 меню). Если Вы не заполните картотеку, то обработка будет недоступна, и Вам будет об этом сообщено. После обработки Вы сможете выбрать пункт 6, в котором представится выбор, куда вывести результат обработки: в консоль или в файл. При выборе пункта 0 Вы, соответственно, завершите работу программы.



*Рисунок . Меню программы*

# Заключение

При выполнении курсовой работы были получены практические навыки поэтапного решение содержательной задачи, связанной с обработкой текстовой информации.