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 № 1**

**по дисциплине «Программирование» на**

**тему:** **«Структуры в языке C/C++»**

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

Санкт-Петербург, 2020

Оглавление

[Цель 3](#_Toc39156028)

[1. Задание 3](#_Toc39156029)

[2. Уточнение задания 3](#_Toc39156030)

[3. Контрольные примеры 5](#_Toc39156031)

[4. Краткое описание алгоритма 6](#_Toc39156032)

[5. Структура вызова функций 7](#_Toc39156033)

[6. Функции 8](#_Toc39156034)

[6.1 Главная функция 8](#_Toc39156035)

[6.2 Функция Menu 10](#_Toc39156036)

[6.3 Функция OutputMenu 11](#_Toc39156037)

[6.4 Функция Info 12](#_Toc39156038)

[6.5 Функция OutputGasStationsTable 13](#_Toc39156039)

[6.6 Функция OutputGasStationsText 14](#_Toc39156040)

[6.7 Функция InputInfo 15](#_Toc39156041)

[6.8 Функция InputText 16](#_Toc39156042)

[6.9 Функция Process 17](#_Toc39156043)

[6.10 Функция PrepareStruct 18](#_Toc39156044)

[6.11 Функция free\_struct 19](#_Toc39156045)

[8. Текст программы 20](#_Toc39156046)

[9. Пример работы программы 28](#_Toc39156047)

[Исходные данные 28](#_Toc39156048)

[Вывод программы 29](#_Toc39156049)

[9.1 Результат работы программы 30](#_Toc39156050)

[10. Заключение 31](#_Toc39156051)

Цель

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

# 1. Задание

Написать программу, выполняющую ввод последовательности структур, длина которой заранее неизвестена. Контрольный вывод исходной последовательности, формирование новой последовательности из элементов исходной, поля которых удовлетворяют заданным условиям. 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, пока он не введет входные данные
3. Из подменю должна быть возможность выйти
4. Вводимый текст не может быть пустым
5. Количество критериев для обработки исходных данных – 2: рейтинг выше 7 и цена на 98 бензин ниже 54

Выбранная предметная область – записи АЗС.

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

|  |  |  |
| --- | --- | --- |
| **Имя поля** | **Тип** | **Назначение** |
| name | char\* | Название АЗС |
| address | char\* | Адрес АЗС |
| fuelprices | float\* | Цены на 4 вида топлива |
| rating | int\* | Рейтинг |

Меню программы должно иметь подобную иерархию:

1 – Справка

2 – Ввод данных

2.1 Ввод названия

2.2 Ввод адреса

2.3 Ввод цен на топливо

2.4 Ввод рейтинга

2.5 Предложение продолжить ввод

3 – Контрольный вывод исходных данных

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

ниже 54

5 – 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 подходит | |
| 2 | Long name of GAS | smth | 57 58 58,8 60 | 9 | Не подходит | |
| 3 | sadlksa | -557.1;51.0 | 52 53,1 53,4 52,7 | 8 | Подходит | |
| 4 | Dillimore | Dillimore village | 50 50,2 50,7 50,4 | 10 | Подходит | |
| 5 | Last | Any | 5 7 75 2 | 10 | Не подходит | |

# 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: Обработка

Пункт 5-ый: Вывод результата обработки

Пункт 0-ой: Выход, завершение программы

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

Шаг №5. Иначе, конец программы.

# 5. Структура вызова функций

Рисунок 1. Структура вызова функций

# 6. Функции

## 6.1 Главная функция

**Назначение:**

Является точкой входа в программу.

**Прототип:**

int main(void)

**Пример вызова:**

Main();

**Описание переменных:**

Описание переменных приведено в таблице 2.

Таблица 2. Описание переменных главной функции

|  |  |  |
| --- | --- | --- |
| Имя переменной | Тип | Назначение |
| GasStations | GSDesc\* | Указатель на вектор структур |
| GSResult | GSDesc\* | Указатель на вектор результирующих структур |
| i | int | Кол-во исходных структур |
| ri | int | Кол-во результирующих структур |
| MenuItem | int | Выбранный пункт меню |

**Схема алгоритма:**

Схема алгоритма для функции main представлена на рисунке 2.

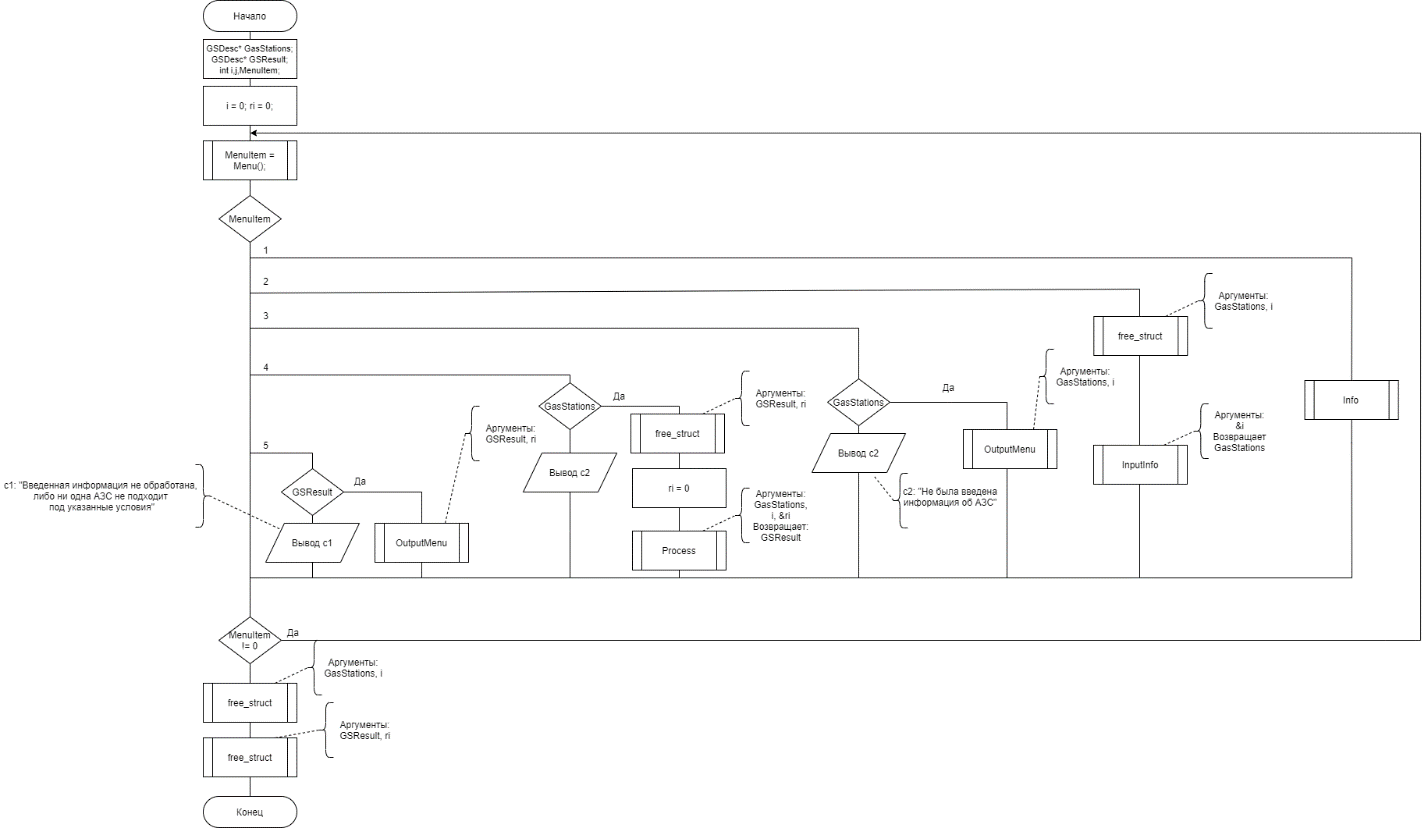


Рисунок 2. Схема алгоритма

## 6.2 Функция Menu

**Назначение:**

Главное меню

**Прототип:**

int Menu();

**Пример вызова:**

MenuItem = Menu();

**Описание переменных:**

Описание переменных приведено в таблице 2.

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\*, int);

**Пример вызова:**

OutputMenu(GasStations, i);

**Описание переменных:**

Описание переменных приведено в таблице 2.

Таблица 4. Описание переменных главной функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | GasStations | GSDesc\* | Указатель на вектор структур |
| lines | int | Кол-во структур |
| Локальная | selected | int | Выбранный пункт меню |

## 6.4 Функция Info

**Назначение:**

Вывод справки

**Прототип:**

void Info();

**Пример вызова:**

Info();

## 6.5 Функция OutputGasStationsTable

**Назначение:**

Функция вывода структур в виде таблице

**Прототип:**

void OutputGasStationsTable(GSDesc\*, int);

**Пример вызова:**

OutputGasStationsTable(GasStations, lines);

**Описание переменных:**

Описание переменных приведено в таблице 4.

Таблица 4. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | GasStations | GSDecs\* | Указатель на вектор структур |
| lines | int | Кол-во структур |
| Локальные | i | int | Переменная для цикла for |
| namelen | int | Максимальная длина названия |
| addresslen | int | Максимальная длина адреса |

## 6.6 Функция OutputGasStationsText

**Назначение:**

Функция вывода структур в виде блоков построчно

**Прототип:**

void OutputGasStationsText(GSDesc\*, int);

**Пример вызова:**

OutputGasStationsText(GasStations, lines);

**Описание переменных:**

Описание переменных приведено в таблице 5.

Таблица 5. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | GasStations | GSDesc\* | Указатель на вектор структур |
| lines | int | Кол-во структур |
| Локальные | i | int | Переменная для цикла for |
| j | int | Переменная для цикла for |

## 6.7 Функция InputInfo

**Назначение:**

Функция ввода структур.

**Прототип:**

GSDesc\* InputInfo(int\*);

**Пример вызова:**

GasStations = InputInfo(&i);

**Описание переменных:**

Описание переменных приведено в таблице 6.

Таблица 6. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | amount | int\* | Указатель на участок памяти, в котором хранится кол-во структур |
| Локальные | Stations | GSDesc\* | Указатель на вектор структур |
| i | int | Переменная для цикла for |
| j | int | Переменная для цикла for |
| f | int | Флаг повтора ввода |

## 6.8 Функция InputText

**Назначение:**

Функция ввода строки

**Прототип:**

void InputText(char\*);

**Пример вызова:**

InputText(Stations[j].name);

**Описание переменных:**

Описание переменных приведено в таблице 7.

Таблица 7. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | string | char\* | Указатель на строку |
| Локальные | c | char | Введенный символ |
| i | int | Переменная для цикла For |

## 6.9 Функция Process

**Назначение:**

Функция для обработки данных.

**Прототип:**

GSDesc\* Process(GSDesc\*, int, int\*);

**Пример вызова:**

GSResult = Process(GasStations, i, &ri);

**Описание переменных:**

Описание переменных приведено в таблице 8.

Таблица 8. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | GasStations | GSDesc\* | Указатель на вектор структур |
| lines | int | Кол-во исходных структур |
| rlines | int\* | Указатель на участок памяти, где хранится кол-во результирующих структур |
| Локальные | result | GSDesc\* | Указатель на вектор результирующих структур |
| i | int | Переменная для цикла for |
| j | int | Кол-во результирующих структур |
| k | int | Переменная для цикла for |

## 6.10 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(&Stations[j]))

**Описание переменных:**

Описание переменных приведено в таблице 9.

Таблица 9. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальная | Station | GSDesc\* | Указатель на структуру |
| Локальная | res | int | Возвращаемое число, показывающее успех выделения памяти |

## 6.11 Функция free\_struct

**Назначение:**

Функция освобождения памяти, выделенной под структуры

**Прототип:**

GSDesc\* free\_struct(GSDesc\*, int);

**Пример вызова:**

GasStations = free\_struct(GasStations, i);

**Описание переменных:**

Описание переменных приведено в таблице 9.

Таблица 9. Описание переменных функции

|  |  |  |  |
| --- | --- | --- | --- |
| Вид переменной | Имя переменной | Тип | Назначение |
| Формальные | Stations | GSDesc\* | Указатель на вектор структур |
| lines | int | Кол-во структур |
| Локальная | 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

{

char\* name; //Название

char\* address; //Адрес

float\* fuelPrices; //Ниже цены на топливо(92,95,98,дизель)

int\* rating; //Рейтинг АЗС(1-10)

} GSDesc;

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

int Menu(); //Меню

void OutputMenu(GSDesc\*, int); //Подменю вывода, int - кол-во элементов

void Info(); //Справка

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

кол-во элементов

void OutputGasStationsText(GSDesc\*, int); //Вывод информации в виде текста, int - кол-

во элементов

GSDesc\* InputInfo(int\*); //Ввод элементов

void InputText(char\*); //Ввод строки

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

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

void free\_struct(GSDesc\*, int); //Освобождение памяти

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

int main()

{

setlocale(LC\_ALL, "russian");

GSDesc \*GasStations = NULL, //АЗС

\*GSResult = NULL; //Результат отбора

int i = 0, //Кол-во АЗС

ri = 0, //Кол-во АЗС после обработки

MenuItem; //Выбранный пункт меню

do

{

MenuItem = Menu();

switch(MenuItem)

{

case 1:

Info();

break;

case 2:

free\_struct(GasStations, i);

GasStations = InputInfo(&i);

break;

case 3:

if(GasStations != NULL) OutputMenu(GasStations, i);

else

{

puts("Не была введена информация об АЗС");

system("pause");

}

break;

case 4:

if(GasStations != NULL)

{

free\_struct(GSResult, ri);

ri = 0;

GSResult = Process(GasStations, i, &ri);

}

else puts("Не была введена информация об АЗС"),system("pause");

break;

case 5:

if(GSResult != NULL) OutputMenu(GSResult, ri);

else

{

puts("Введенная информация не обработана, либо ни одна АЗС не подходит под

указанные условия");

system("pause");

}

break;

}

} while(MenuItem);

free\_struct(GasStations, i);

free\_struct(GSResult, ri);

return 0;

}

GSDesc\* InputInfo(int\* amount)

{

GSDesc\* Stations = NULL;

int i,

j = 0,

f;

do

{

system("cls");

Stations = (GSDesc\*)realloc(Stations, (j+1)\*sizeof(GSDesc));

if(PrepareStruct(&Stations[j]))

{

puts("Введите название АЗС: ");

InputText(Stations[j].name);

puts("\nВведите адрес: ");

InputText(Stations[j].address);

printf("\nВведите цены на топливо(92,95,98,дизель): ");

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

{

do

{

scanf("%f", Stations[j].fuelPrices+i);

if(Stations[j].fuelPrices[i] < 1 || Stations[j].fuelPrices[i] > 10000) puts("1-10000");

} while(Stations[j].fuelPrices[i] < 1 || Stations[j].fuelPrices[i] > 10000);

}

printf("Введите рейтинг АЗС: ");

do

{

scanf("%d", Stations[j].rating);

if(\*(Stations[j].rating) < 1 || \*(Stations[j].rating) > 10) puts("1-10");

} while(\*(Stations[j].rating) < 1 || \*(Stations[j].rating) > 10);

j++;

puts("Введите любое число, отличное от нуля, если хотите продолжить ввод данных");

scanf("%d", &f);

}

else f = 0;

} while(f);

\*amount = j;

system("pause");

return Stations;

}

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);

}

void OutputGasStationsTable(GSDesc\* GasStations, int lines)

{

system("cls");

int i,

namelen,

addresslen;

namelen = 0, addresslen = 0;

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

{

if(strlen(GasStations[i].name) > namelen) namelen = strlen(GasStations[i].name);

if(strlen(GasStations[i].address) > addresslen) addresslen = strlen(GasStations[i].address);

}

if(namelen < 8) namelen = 8;

if(addresslen < 5) addresslen = 5;

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

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

printf("|%\*s|%\*s|%15.2f|%15.2f|%15.2f|%11.2f|%7d|\n", namelen, GasStations[i].name, addresslen, GasStations[i].address,

GasStations[i].fuelPrices[0], GasStations[i].fuelPrices[1], GasStations[i].fuelPrices[2], GasStations[i].fuelPrices[3], \*(GasStations[i].rating));

system("pause");

}

void OutputGasStationsText(GSDesc\* GasStations, int lines)

{

system("cls");

int i,

j;

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

{

printf("Название: %s", GasStations[i].name);

printf("\nАдрес: %s", GasStations[i].address);

printf("\nЦены(92,95,98,Дизель): ");

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

printf("\nРейтинг: %d", \*(GasStations[i].rating));

printf("\n\n");

}

system("pause");

}

GSDesc\* Process(GSDesc\* GasStations, int lines, int\* rlines)

{

GSDesc\* result = NULL;

int i,

j = 0,

k;

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

if(\*(GasStations[i].rating) > 7 && GasStations[i].fuelPrices[2] < 54.0)

{

result = (GSDesc\*)realloc(result, (j+1)\*sizeof(GSDesc));

if(PrepareStruct(result+j))

{

strcpy(result[j].name, GasStations[i].name);

strcpy(result[j].address, GasStations[i].address);

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

result[j].fuelPrices[k] = GasStations[i].fuelPrices[k];

\*(result[j].rating) = \*(GasStations[i].rating);

j++;

}

}

\*rlines = j;

if(result == NULL) puts("Нет АЗС, удовлетворяющим поставленным условиям");

else puts("Обработка прошла успешно");

system("pause");

return result;

}

int PrepareStruct(GSDesc\* Station)

{

int res = 0;

if(Station != NULL)

{

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

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

Station->fuelPrices = (float\*)malloc(4\*sizeof(float));

Station->rating = (int\*)malloc(sizeof(int));

res++;

}

return res;

}

/\*------------------------------------------------Меню и справка---------------------------------------------\*/

int Menu()

{

system("cls");

int selected;

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

puts("1 - Справка");

puts("2 - Ввод информации об АЗС");

puts("3 - Контрольный вывод введенной информации");

puts("4 - Обработка информации");

puts("5 - Вывод обработанной информации");

puts("0 - Выход");

do

{

scanf("%d", &selected);

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

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

fflush(stdin);

return selected;

}

void OutputMenu(GSDesc\* GasStations, int lines)

{

int selected;

do

{

system("cls");

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

puts("1 - В виде таблицы");

puts("2 - В виде текста");

puts("0 - Назад");

do

{

scanf("%d", &selected);

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

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

fflush(stdin);

if(selected == 1) OutputGasStationsTable(GasStations, lines);

else if(selected == 2) OutputGasStationsText(GasStations, lines);

}while(selected);

}

void Info()

{

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\n");

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

system("pause");

}

void free\_struct(GSDesc\* Stations, int lines)

{

if(Stations != NULL)

{

int i;

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

{

free(Stations[i].name);

Stations[i].name = NULL;

free(Stations[i].address);

Stations[i].address = NULL;

free(Stations[i].fuelPrices);

Stations[i].fuelPrices = NULL;

free(Stations[i].rating);

Stations[i].rating = NULL;

free(Stations+i);

}

Stations = NULL;

}

}

# 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

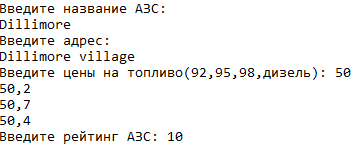


Рисунок 4. Вывод исходной информации в виде таблицы

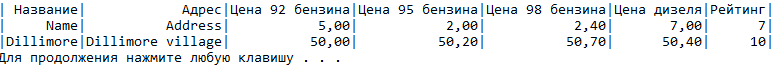
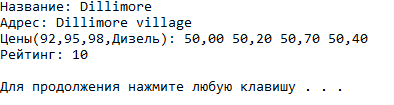


Рисунок 5. Вывод обработанной информации в виде блоков из строк



## 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.

# 10. Заключение

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