МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» им.В.И.УЛЬЯНОВА (ЛЕНИНА)

Кафедра вычислительной техники

Отчет по лабораторной работе № 9
по дисциплине «Программирование»
Тема: «Указатели на структуры и функции»

Студент гр. 8307 Никулин Л.А.

Преподаватель Перязева Ю.В.

Санкт-Петербург
2019

Содержание

Цель

Получить практические навыки в разработке алгоритма и написании программы на языке Си с использованием указателей на структуры и указателей на функции.

Задание

Выбор записей, в которых значение любого числового поля (выбор из меню) не превышает среднее значение в этом поле, сортировка результата по последнему символьному полю в обратном алфавитном порядке.

Для выбранной предметной области создать динамический массив структур, содержащих характеристики объектов предметной области. Написать программу, обеспечивающую начальное формирование массива структур при чтении из файла с последующим возможным дополнением элементов массива при вводе с клавиатуры. Следует использовать указатели на структуры и указатели на функции обработки массива в соответствии с вариантом задания. Во всех случаях, когда при поиске записей результат отсутствует, следует вывести сообщение.

Постановка задачи и описание решения

В структуре описываются свойства автомобиля. 5 полей – бренд и модель – динамические char-массивы, год производства – int, пробег – int, объём двигателя – float.

Объявляем указатель типа struct car с именем my_car. С помощью функции number_of_lines() (которая считает кол-во строк в файле) получаем кол-во элементов в массиве. Выделяем память для my_car в соответствии с полученным значением t. Считываем элементы массива структур из файла. Для динамических char-массивов выделяем соответствующие кол-во памяти. После считывания массива выводим его. После его спрашиваем пользователя, хотим ли добавить еще элемент. Если ответ положительный, выделяем дополнительную память для my_car с помощью функции realloc(), затем считываем с клавиатуры значение каждого поля и снова спрашиваем, хочет ли пользователь добавить новый элемент массива. После того как все дополнительные элементы массива были добавлены, сортируем получившейся массив по полю modelname в обратном алфавитном порядке методом пузырька и выводим полученный массив. Далее спрашиваем у пользователя, по какому полю искать среднее значение и вывести все элементы массива, значение соответствующего поля которых меньше среднего. В зависимости от

выбора, вызываем функцию print_array(), один из аргументов которой является указатель на соответствующую функцию (select_year, select_mileage или select_capacity). Прототипы всех трех функций идентичны. Функция принимает указатель на массив структур и кол-во элементов в нём. Считает среднее значение соответствующего поля (mean). Затем в созданный динамический массив int* array записываются все индексы элементов, значение поля которых меньше среднего. Также считается кол-во записанных индексов в массив (переменная k). Далее в структуру value типа selected (одно из полей – целочисленный указатель int* a, а второе - целочисленное поле int w) записывается указатель на массив индексов и их кол-во. Функция возвращает эту структуру. В функции print_array() funcName возвращает полученную структуру и записывает в переменную value. Если значение value.w равно 0, значит результат поиска отсутствует (выводиться сообщение). Далее, используя цикл с параметром, выводятся все элементы массива my_car, индексы которых были переданы в функцию при помощи поля a структуры value. После вывода очищается массив индексов. Далее очищаются все динамические поля структуры my_car и очищается массив структур. Файл закрывается.

Описание переменных

Описание структур:

```
struct car
{
    char* brandname;
    char* modelname;
    int year;
    int mileage;
    float capacity;
}
struct selected
{
    int* a;
    int w;
}.
```

Таблица 1. Описание переменных.

Имя переменной	**Тип**	**Назначение**
pF	*file	Указатель на исходный файл
my_car	struct car*	Указатель на массив структур
temp	struct car	Используется для сортировки
answer_add	int	Используется для запроса пользователя о добавлении новых элементов
answer_field	int	Используется для запроса пользователя о выборе поля для дальнейшей работы
t	int	Счётчик в цикле/Кол-во элементов в массиве
tt	int	Для удобного заполнении массива (на 1 меньше, чем t)
q	int	Счётчик в цикле
i	int	Счётчик в цикле
j	int	Счётчик в цикле
lexeme	char *	Лексема (то, что стоит между разделитеями)
text	char[1024]	Служит для хранении строки файла
sum	int/float	Сумма полей всех элементов массива
mean	float	Среднее значение поля
array	int*	Массив индексов
k	int	Кол-во индексов в массиве

Контрольные примеры

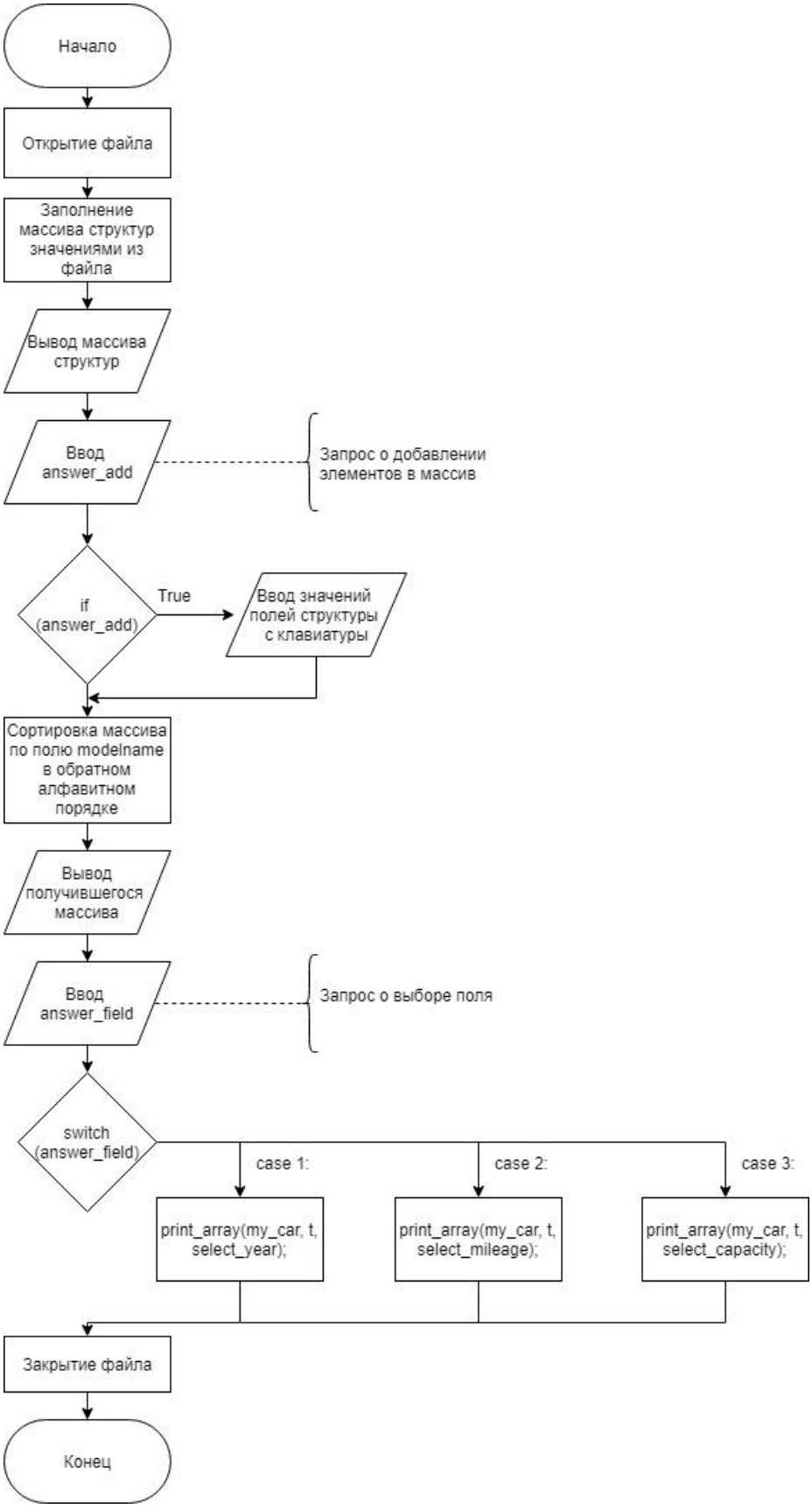
```
Porsche;Cayenne;2012;300000;3.6;
Nissan;Teana;2010;30000;2.5;
Opel;Meriva;2008;50000;1.6;
Chery;Kimo;2011;120000;1.3;
Lada;Vesta;2017;10000;1.6;
Ford;Focus;2004;182000;1.6;
Lada;Granta;2013;111000;1.6;
Chevrolet;Lacetti;2009;22000;1.6;
Renault;Megane;2001;260000;1.4;
Kia;Rio;2013;102500;1.4;
```

input.txt

№	answer_add	Добавленные элементы	answer_field	Результат
1	0	-	1	Opel; Meriva; 2008; 50000; 1.6 Renault; Megane; 2001; 260000; 1.4 Chevrolet; Lacetti; 2009; 22000; 1.6 Ford; Focus; 2004; 182000; 1.6

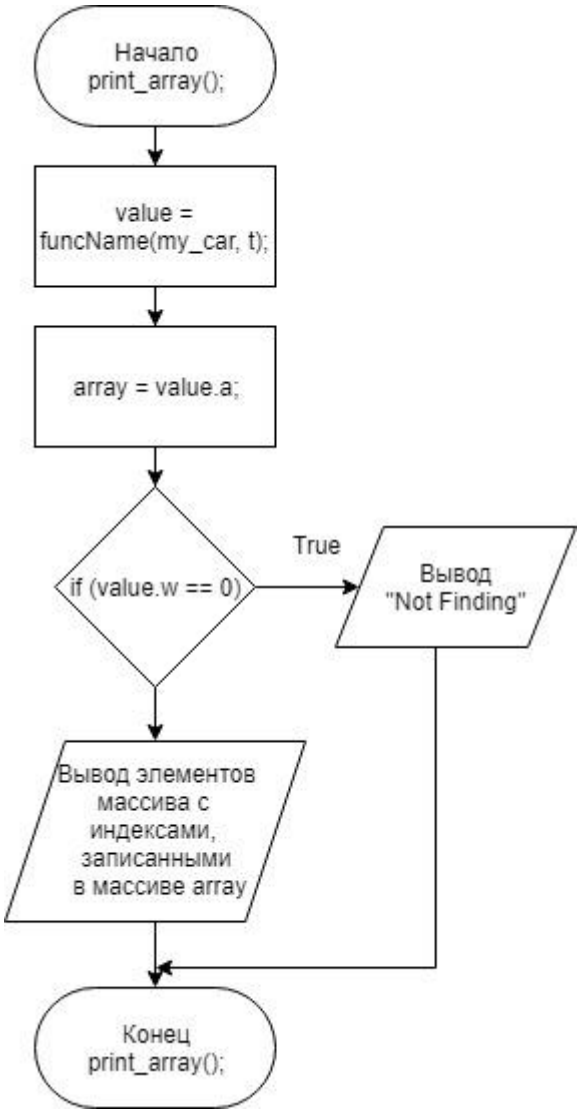
2	0	-	3	Lada; Vesta; 2017; 10000; 1.6 Nissan; Teana; 2010; 30000; 2.5 Kia; Rio; 2013; 102500; 1.4 Opel; Meriva; 2008; 50000; 1.6 Chevrolet; Lacetti; 2009; 22000; 1.6 Lada; Granta; 2013; 111000; 1.6
3	1	Audi A5 2008 0 2.0	2	Mean value: 107954.0 Lada; Vesta; 2017; 10000; 1.6 Nissan; Teana; 2010; 30000; 2.5 Kia; Rio; 2013; 102500; 1.4 Opel; Meriva; 2008; 50000; 1.6 Chevrolet; Lacetti; 2009; 22000; 1.6 Audi; A5; 2008; 0; 2.0

Схема алгоритма



main()

number_of_lines()


Начало
print_array();
value =
funcName(my_car, t);
array = value.a;
if (value.w == 0)
True
Вывод
"Not Finding"
Вывод элементов
массива с
индексами,
записанными
в массиве array
Конец
print_array();

print_array()


Начало
select_year()
sum = 0;
k = 0;
Суммирование всех значений полей year и сохранение результата в sum
mean = sum/t;
Вывод значения mean
Добавление всех индексов элементов, удолетворяющих условию
Сохранение кол-ва добавленных индексов в массив в переменную k
value.a = array;
value.w = k;
return value;
Конец
select_year()

select_year()

select_mireage()

select_capacity()

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MS 80
#define LINE 1024

struct car
{
   char* brandname;
   char* modelname;
   int year;
   int mileage;
   float capacity;
};

struct selected
{
   int* a;
   int w;
};

int number_of_lines(FILE* file);
void print_array(struct car* my_car, int t, struct selected (funcName)(struct car*, int));
struct selected select_year(struct car* my_car, int t);
struct selected select_mileage(struct car* my_car, int t);
struct selected select_capacity(struct car* my_car, int t);
```

```
int main()
{
    FILE* pF = fopen("input.txt", "r");
    struct car* my_car = NULL;
    struct car temp;
    int answer_add, answer_field;
    if(pF != NULL)
    {
        char text[LINE];
        int t = 0;
        int tt;
        t = number_of_lines(pF);
        my_car = (struct car*)malloc(t*sizeof(struct car));
        if (my_car != NULL)
        {
            for (int i = 0; i < t; i++)
            {
                fgets(text, LINE, pF);
                int q = 0;
                char *lexeme = strtok(text, ";");
                while(lexeme != NULL)
                {
                    switch (q)
                    {
                    case 0:
                        my_car[i].brandname =
(char*)malloc(strlen(lexeme)*sizeof(char));
                        strcpy(my_car[i].brandname, lexeme);
                        break;
                    case 1:
```

```
                    my_car[i].modelname =
(char*)malloc(strlen(lexeme)*sizeof(char));
                    strcpy(my_car[i].modelname, lexeme);
                    break;
                case 2:
                    my_car[i].year = atoi(lexeme);
                    break;
                case 3:
                    my_car[i].mileage = atoi(lexeme);
                    break;
                case 4:
                    my_car[i].capacity = atof(lexeme);
                    break;
                }
                lexeme = strtok(NULL, ";");
                q++;
            }
        }

        printf("List:\n");
        for (int i = 0; i < t; i++)
            printf("%s; %s; %d; %d; %.1f\n", my_car[i].brandname,
my_car[i].modelname, my_car[i].year, my_car[i].mileage,
my_car[i].capacity);
        puts("");

        printf("Do you want to add element? (0 - no, 1 - yes)\n");
        scanf("%d", &answer_add);
        getchar();
        while (answer_add)
```

```
        {
            t++;
            my_car = (struct car*)realloc(my_car, t*sizeof(struct car));
            if (my_car != NULL)
            {
                tt = t -1;
                puts("Enter brandname");
                my_car[tt].brandname = (char*)malloc(MS*sizeof(char));
                if (my_car[tt].brandname != NULL)
                {
                    fgets(my_car[tt].brandname, MS, stdin);
                    my_car[tt].brandname[strlen(my_car[tt].brandname) - 1]
= '\0';
                    puts("Enter modelname");
                    my_car[tt].modelname = (char*)malloc(MS*sizeof(char));
                    if (my_car[tt].modelname != NULL)
                    {
                        fgets(my_car[tt].modelname, MS, stdin);
                        my_car[tt].modelname[strlen(my_car[tt].modelname) -
1] = '\0';
                        puts("Enter year");
                        scanf("%d", &my_car[tt].year);
                        puts("Enter mileage");
                        scanf("%d", &my_car[tt].mileage);
                        puts("Enter capacity");
                        scanf("%f", &my_car[tt].capacity);
                        printf("Do you want to add element? (0 - no, 1 -
yes)\n");
                        scanf("%d", &answer_add);
                        getchar();
```

```
                }
                else puts("Error Memory!!!");
            }
            else puts("Error Memory!!!");
        }
        else puts("Error Memory!");
    }

    for (int i = 0; i < t - 1; i++)
        for (int j = 0; j < t - i - 1; j++)
            if (strcmp(my_car[j].modelname, my_car[j+1].modelname) <
0)
            {
                temp = my_car[j];
                my_car[j] = my_car[j+1];
                my_car[j+1] = temp;
            }
    printf("\nNew sorted list:\n");
    for (int i = 0; i < t; i++)
        printf("%s; %s; %d; %d; %.1f\n", my_car[i].brandname,
my_car[i].modelname, my_car[i].year, my_car[i].mileage,
my_car[i].capacity);
    puts("");

    puts("Choose the field (1 - year, 2 - mileage, 3 - capacity):");
    scanf("%d", &answer_field);
    switch(answer_field)
    {
    case 1:
        print_array(my_car, t, select_year);
```

```
            break;
        case 2:
            print_array(my_car, t, select_mileage);
            break;
        case 3:
            print_array(my_car, t, select_capacity);
            break;
        }

        if(fclose(pF)==EOF)
            puts("Input closing error");
        for (int i = 0; i < t; i++)
        {
            free(my_car[i].modelname);
            free(my_car[i].brandname);
        }
        free(my_car);
    }
    else puts("Error Memory!");
}

else
    printf("Opening Error");

return 0;
}

int number_of_lines(FILE* file)
{
```

```
    int t = 0;
    char text[LINE];
    while(fgets(text, LINE, file))
        t++;
    rewind(file);
    return t;
}

void print_array(struct car* my_car, int t, struct selected
(*funcName)(struct car*, int))
{
    struct selected value;
    value = funcName(my_car, t);
    int i;
    int* array = value.a;
    if (value.w == 0) puts("Not Finding");
    else
    for (int j = 0; j < value.w; j++)
    {
        i = array[j];
        printf("%s; %s; %d; %d; %.1f\n", my_car[i].brandname,
my_car[i].modelname, my_car[i].year, my_car[i].mileage,
my_car[i].capacity);
    }
    free(array);
}

struct selected select_year(struct car* my_car, int t)
{
    int sum = 0;
```

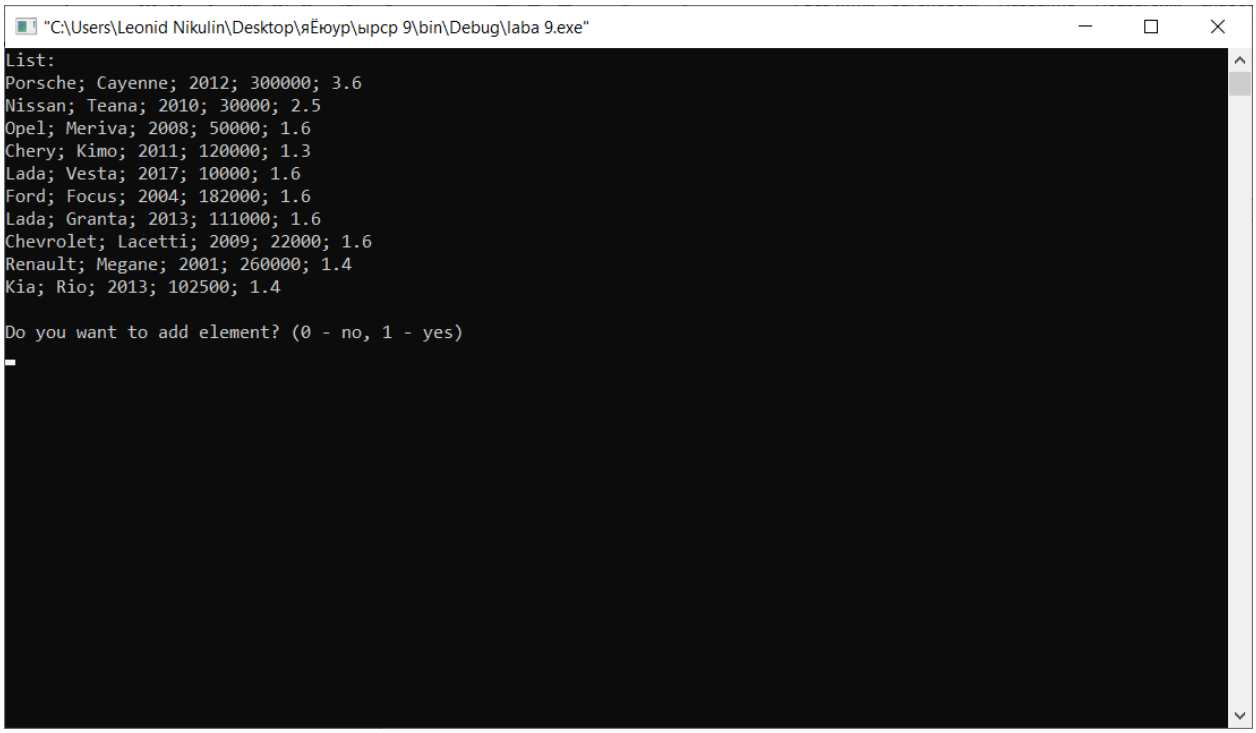
```
    float mean;
    struct selected value;
    int *array = (int*)calloc(t, sizeof(int));
    if (array != NULL)
    {
        int k = 0;
        for (int i = 0; i < t; i++)
            sum+=my_car[i].year;
        mean = sum / t;
        printf("Mean value: %.1f\n", mean);
        for (int i = 0; i < t; i++)
            if (my_car[i].year <= mean)
            {
                array[k] = i;
                k++;
            }
        value.a = array;
        value.w = k;
    }
    else puts("Error Memory!");
    return value;
}

struct selected select_mileage(struct car* my_car, int t)
{
    int sum = 0;
    float mean;
    struct selected value;
    int *array = (int*)calloc(t, sizeof(int));
```

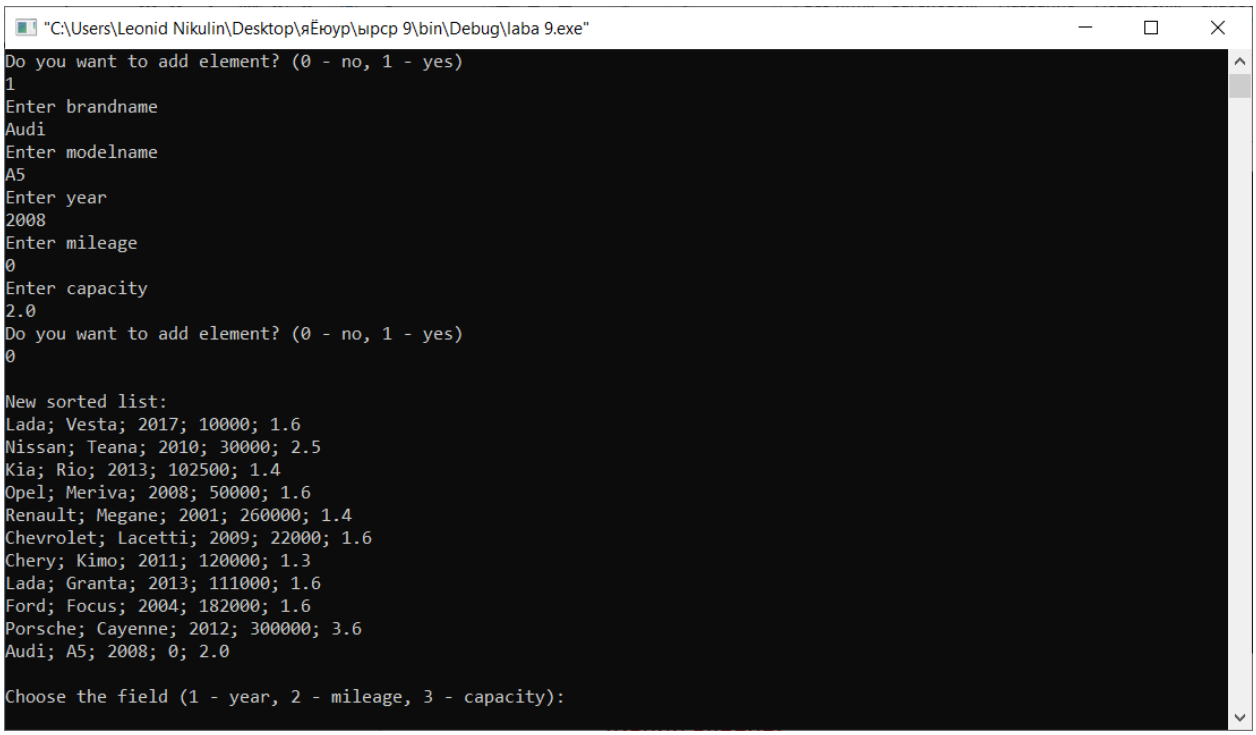
```
    if (array != NULL)
    {
        int k = 0;
        for (int i = 0; i < t; i++)
            sum+=my_car[i].mileage;
        mean = sum / t;
        printf("Mean value: %.1f\n", mean);
        for (int i = 0; i < t; i++)
            if (my_car[i].mileage <= mean)
            {
                array[k] = i;
                k++;
            }
        value.a = array;
        value.w = k;
    }
    else puts("Error Memory!");
    return value;
}

struct selected select_capacity(struct car* my_car, int t)
{
    float sum = 0;
    float mean;
    struct selected value;
    int *array = (int*)calloc(t, sizeof(int));
    if (array != NULL)
    {
        int k = 0;
```

```
        for (int i = 0; i < t; i++)
            sum+=my_car[i].capacity;
        mean = sum / t;
        printf("Mean value: %.1f\n", mean);
        for (int i = 0; i < t; i++)
            if (my_car[i].capacity <= mean)
            {
                array[k] = i;
                k++;
            }
        value.a = array;
        value.w = k;
    }
    else puts("Error Memory!");
    return value;
}
```

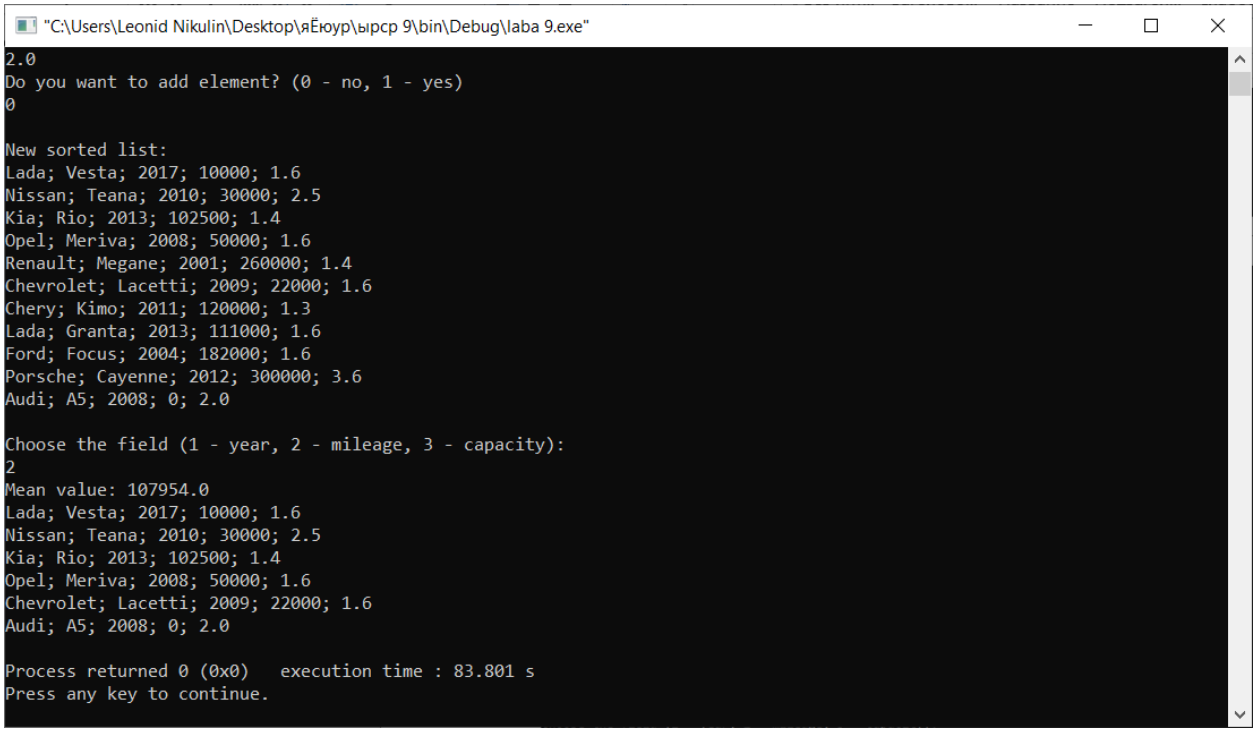
Пример работы программы

Исходные данные





Вывод программы



Выводы:

При выполнении лабораторной работы были получены практические навыки в использовании указателей на структуры и указателей на функции.