

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

Кафедра вычислительной техники

Отчет по лабораторным работам №10-12
по дисциплине «Программирование»
Темы: «Линейный односвязный список как АД»
«Линейный двусвязный список»
«Кольцевые (циклические) списки»

Студент гр. 8307

Никулин Л.А.

Преподаватель

Перязева Ю.В.

Содержание

Цель	3
Задание.....	3
Постановка задачи и описание решения.....	3
Описание переменных	6
Контрольные примеры	9
Текст программы	12
Пример работы программы.....	35
Выводы:	38

Цель

Получить практические навыки в разработке алгоритмов и написании программ на языке Си с использованием линейных односвязных, двусвязных и кольцевых списков.

Задание

1 (лаб.10). Разработать подалгоритм создания односвязного списка из имеющегося односвязного списка путем копирования заданных элементов. Номера копируемых элементов содержатся в полученном массиве. Порядок копируемых элементов должен соответствовать их порядку в исходном списке. В случае отсутствия элемента с заданным номером вывести сообщение.

2 (лаб.11). Разработать подалгоритм, обеспечивающий копирование элемента линейного двусвязного списка с заданным id в заданную позицию (0 считается позицией «перед первым элементом») с одновременным автоинкрементом поля id. Если номер позиции превышает количество элементов списка, копирование делается в позицию «после последнего».

3 (лаб.12). Разработать подалгоритм удаления в односвязном кольцевом списке «предпоследнего» элемента. При недостаточном количестве элементов в списке вывести соответствующее сообщение.

Постановка задачи и описание решения

При запуске программы пользователю надо выбрать модуль, отвечающий за определённую лаб. работу (1 -10 лаба, 2 – 11 лаба, 3 – 12 лаба).

1. Создаём односвязный список элементов, тип данных которых `car`. Считываем изначальный список элементов из файла `input.txt`. После того, как считали весь список, спрашивается, хотим ли мы добавить дополнительные элементы с клавиатуры. При выборе 1 (или любого числа, отличного от 0) начинается заполнение следующего элемента списка. Как только пользователь в диалоге о дальнейшем вводе элемента выберет 0, выведется сообщение об окончании заполнения списка и выведется этот список. Далее, спрашивается, сколько элементов хочет иметь пользователь в новом списке. Далее, вводятся номера этих элементов. По условию задачи *«порядок копируемых элементов должен соответствовать их порядку в исходном списке»*, поэтому полученный массив номеров сортируем, используя функцию

qsort() (один из параметров – указатель на функцию compareVozr(), которую мы описали ранее). После сортировки, создаем новый односвязный список. Далее, мы проходим по изначальному списку, проверяя номер каждого элемента, если он имеется в массиве, введенном ранее, мы добавляем его в новый список. У нас есть переменная flag – она нужна нам для определения порядка элемента в новый список, т.е. если ее значение 1 (изначальное) и мы нашли элемент, который хотим добавить в новый список, значит это первый найденный элемент и мы сбрасываем её значение в 0. И при дальнейшем нахождении нового элемента, который мы будем добавлять в новый список, её значение будет равно 0 – значит это не первый найденный элемент. Мы используем функцию AddFirst(), когда добавляем первый элемент в список, и функцию InsertAfter(), если очередной найденный элемент не первый в новом списке. Также имеется переменная flag2, отвечающая за то, нашли ли мы i-ый элемент в старом списке – изначально её значение 0, но если мы нашли i-ый элемент, её значение становится 1. Если flag2 после прохождения по всему старому списку осталось равным 0, выводим сообщение о том, что i-ый элемент не был найден. Перед началом поиска i+1 элемента сбрасываем значение flag2 в ноль и указатель p на начало списка. Далее выводим новый полученный список после копирования элементов. Очищаем память, выделенную для старого и нового списков, а также для массива array. Закрываем файл.

2. Создаём двусвязный список элементов, тип данных которых car. Заполнение происходит тем же самым образом, что и в лабораторной работе 10. В переменную n записываем кол-во элементов в полученном списке. Спрашиваем пользователя об id элемента, который хотим скопировать. Если id меньше 1 или больше кол-ва элементов – выводим сообщение об ошибке, иначе запрашиваем номер позиции, куда надо вставить скопированный элемент. Если позиция меньше 0, выводим сообщение об ошибке. Если позиция равна 0 – вставляем перед первым (используем функцию InsertBefore2()), а также автоматически инкремируем все id элементов, стоящих после первого (используем функцию autoinkAll()). Если позиция больше количества элементов списка, вставляем после последнего (используем функцию InsertAfter2()). В ином случае, вставляем скопированный элемент в позицию, заданную ранее (используем функцию InsertBefore2()), изменяем ему id в соответствии с его позицией, а также автоматически инкремируем все id элементов, стоящих после вставленного (используем функцию autoinkN()). Выводим получившейся список после вставки

элемента и очищаем память, выделенную для этого списка. Закрываем файл.

3. Создаём односвязный кольцевой список элементов, тип данных которых car. Заполнение происходит тем же самым образом, что и в лабораторной работе 10. После заполнения односвязного списка, делаем его кольцевым при помощи функции `makeCircular()`. В переменную `n` записываем количество элементов в списке. Если в списке всего 1 элемент, выводим соответствующее сообщение и не удаляем предпоследний элемент (т.к. он отсутствует). В ином случае, ставим указатель в начало списка и двигаемся по нему, пока `id` элемента не будет равно $(n-1)$ (т.е. пока указатель не будет указывать на предпоследний элемент). Удаляем этот элемент при помощи функции `deleteSelected()`. Уведомляем об успешном выполнении и выводим полученный список после удаления и очищаем память, выделенную для этого списка. Закрываем файл.

Описание переменных

10 Л/Р:

Используемые структуры:

```
car                                Head
{                                  {
int id;                           int cnt;
char* brandname;                  struct s_car *first;
int year;                        struct s_car *last;
int mileage;                     };
struct s_car* next;
};
```

Таблица 1. Описание переменных.

Имя переменной	Тип	Назначение
pF	FILE*	Указатель на файл input.txt
p0	Head	Голова изначального списка
np0	Head	Голова нового списка
p, np, p1, MyNode	car*	Указатели на элементы списка
brand	char[MS]	Промежуточное хранение текстового поля элемента
slen	int	Длина текстового поля
y, m	int	Промежуточное хранение целочисленных полей элемента
answer	int	Служит для обработки ответа пользователя о вводе очередного элемента
n	int	Кол-во элементов в новом списке
flag	int	Флаг для определения функции, используемой для добавления элемента в новый список
flag2	int	Флаг, отвечающий за то, нашли ли мы i-ый элемент в списке
array	int*	Массив элементов нового списка

11 Л/Р:

Используемые структуры:

```
car
{
int id;
char* brandname;
int year;
int mileage;
struct s_car* next;
struct s_car* prev;
};
```

```
Head
{
int cnt;
struct s_car *first;
struct s_car *last;
};
```

Таблица 2. Описание переменных.

Имя переменной	Тип	Назначение
pF	FILE*	Указатель на файл input.txt
p0	Head	Голова изначального списка
p, p1, MyNode	car*	Указатели на элементы списка
brand	char[MS]	Промежуточное хранение текстового поля элемента
slen	int	Длина текстового поля
y, m	int	Промежуточное хранение целочисленных полей элемента
answer	int	Служит для обработки ответа пользователя о вводе очередного элемента
n	int	Кол-во элементов в списке
search_id	int	Номер элемента, который нужно скопировать
position	int	Номер позиции, куда надо вставить элемент
array	int*	Массив элементов нового списка

12 Л/Р:

Используемые структуры:

```
car
{
int id;
char* brandname;
int year;
int mileage;
struct s_car* next;
};
```

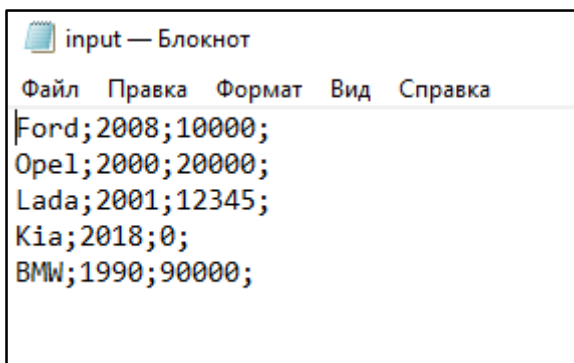
```
Head
{
int cnt;
struct s_car *first;
struct s_car *last;
};
```

Таблица 3. Описание переменных.

Имя переменной	Тип	Назначение
pF	FILE*	Указатель на файл input.txt
p0	Head	Голова изначального списка
p, p1, MyNode	car*	Указатели на элементы списка
brand	char[MS]	Промежуточное хранение текстового поля элемента
slen	int	Длина текстового поля
y, m	int	Промежуточное хранение целочисленных полей элемента
answer	int	Служит для обработки ответа пользователя о вводе очередного элемента
n	int	Кол-во элементов в новом списке

Контрольные примеры

Файл input.txt:



```
input — Блокнот
Файл  Правка  Формат  Вид  Справка
Ford;2008;10000;
Opel;2000;20000;
Lada;2001;12345;
Kia;2018;0;
BMW;1990;90000;
```

10 Л/Р:

Пример 1:

Ввод дополнительных элементов списка:

Toyota	2019	100
--------	------	-----

Сколько элементов будет в новом списке: 2

Номера элементов: 1 6

Полученный список:

id: 1; brandname: Ford; year: 2008; mileage: 10000;

id: 2; brandname: Toyota; year: 2019; mileage: 100;

Пример 2:

Ввод дополнительных элементов списка:

Toyota	2019	100
Honda	2000	30000

Кол-во элементов в новом списке: 0

Номера элементов: -

Полученный список: -

Пример 3:

Ввод дополнительных элементов списка: -

Кол-во элементов в новом списке: 3

Номера элементов: 1 10 100

Вывод:

The 10 id was not found.

The 100 id was not found.

id: 1; brandname: Ford; year: 2008; mileage: 10000;

11 Л/Р:

Пример 1:

Ввод дополнительных элементов списка:

Toyota	2019	100
--------	------	-----

Выбор элемента для копирования: 1

Позиция для вставки: 3

Полученный список:

id: 1; brandname: Ford; year: 2008; mileage: 10000;

id: 2; brandname: Opel; year: 2000; mileage: 20000;

id: 3; brandname: Ford; year: 2008; mileage: 10000;

id: 4; brandname: Lada; year: 2001; mileage: 12345;

id: 5; brandname: Kia; year: 2018; mileage: 0;

id: 6; brandname: BMW; year: 1990; mileage: 90000;

id: 7; brandname: Toyota; year: 2019; mileage: 100;

Пример 2:

Ввод дополнительных элементов списка:

Toyota	2019	100
Honda	2000	30000

Выбор элемента для копирования: 100

Вывод:

This id does not exist.

Пример 3:

Ввод дополнительных элементов списка: -

Выбор элемента для копирования: 1

Позиция для вставки: 100

Полученный список:

id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 5; brandname: BMW; year: 1990; mileage: 90000;
id: 6; brandname: Ford; year: 2008; mileage: 10000;

12 Л/Р:

Пример 1:

Ввод дополнительных элементов списка:

Toyota	2019	100
--------	------	-----

Полученный список:

id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 6; brandname: Toyota; year: 2019; mileage: 100;

Пример 2:

Ввод дополнительных элементов списка:

Toyota	2019	100
Honda	2000	30000

Вывод:

id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;

id: 4; brandname: Kia; year: 2018; mileage: 0;

id: 5; brandname: BMW; year: 1990; mileage: 90000;

id: 7; brandname: Honda; year: 2000; mileage: 30000;

Пример 3:

Ввод дополнительных элементов списка: -

Вывод:

id: 1; brandname: Ford; year: 2008; mileage: 10000;

id: 2; brandname: Opel; year: 2000; mileage: 20000;

id: 3; brandname: Lada; year: 2001; mileage: 12345;

id: 5; brandname: BMW; year: 1990; mileage: 90000;

Текст программы

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MS 64
#include "single_linked_linear.h"
#include "doubly_linked_linear.h"
#include "single_linked_circular.h"

void single_linear();
void doubly_linear();
void single_circular();

int main(){
    int mode;
    int successful = 0;

    puts("Select lab:");
    puts("Enter 1 for 10 lab - single linked linear list");
    puts("Enter 2 for 11 lab - doubly linked linear list");
    puts("Enter 3 for 12 lab - single linked circular list");
    scanf("%d", &mode);
    while(!successful) {
        switch (mode) {
            case 1:
                single_linear();
                successful = 1;
                break;
            case 2:
                doubly_linear();
                successful = 1;
                break;
            case 3:
                single_circular();
                successful = 1;
                break;
            default:
                puts("You enter wrong number, please, enter
again");
        }
    }

    return 0;
}

void single_linear()
{
    Head *p0, *np0;
    car *p, *np, *p1, *MyNode=NULL;
    char brand[MS];
    int slen, y, m;
    int answer, n, flag = 1, flag2 = 0;
    int* array;
    FILE* pF = fopen("input.txt", "r");
    p0 = MakeHead();
}
```

```

p = readFromFile(pF, p0);

puts("Enter 1 to add elements or 0 to not add");
scanf("%d", &answer);
getchar();
while (answer)
{
    puts("Enter car brand");
    fgets(brand, MS, stdin);
    slen = strlen(brand);
    brand[slen-1] = '\0';
    puts("Enter car year");
    scanf("%d", &y);
    puts("Enter car mileage");
    scanf("%d", &m);
    MyNode = CreateNode(brand, slen, y, m);
    InsertAfter(p0, MyNode, p);
    p=MyNode;
    puts("Enter 1 if you want to continue, 0 to end");
    scanf("%d", &answer);
    getchar();
}

printf("\n--Input teminated. Your data are:--\n");

p=p0->first;
while(p!=NULL)
{
    printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", p->id, p->brandname, p->year, p->mileage);
    p=p->next;
}

puts("How many elements do you want to have in new List?");
scanf("%d", &n);
array = (int*)malloc(n*sizeof(int));
for (int i = 0; i < n; i++)
{
    printf("Enter the id, which you want to copy to new
List\n");
    scanf("%d", &array[i]);
}
qsort(array, n, sizeof(int), compareVozr);

np0 = MakeHead();
p = p0->first;
for (int i = 0; i < n; i++) {
    while (p != NULL) {
        if (p->id == array[i]) {
            flag2 = 1;
            if (flag) {
                np = CreateNode(p->brandname, strlen(p-
>brandname), p->year, p->mileage);
                AddFirst(np0, np);
                flag = 0;
                np = np0->first;
            } else {

```

```

        MyNode = CreateNode(p->brandname, strlen(p-
>brandname), p->year, p->mileage);
        InsertAfter(np0, MyNode, np);
        np = MyNode;
    }
    }
    p = p->next;
}
if (!flag2) printf("The %d id was not found\n", ar-
ray[i]);
flag2 = 0;
p = p0->first;
}
np = np0->first;
if (np0->first != NULL) puts("New List:");
while(np!=NULL)
{
    printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", np->id, np->brandname, np->year, np->mileage);
    np=np->next;
}

p = p0->first;
while(p!=NULL)
{
    p1=p->next;
    free(p);
    p=p1;
}
free(p0);

np = np0->first;
while(np!=NULL)
{
    p1=np->next;
    free(np);
    np=p1;
}
free(np0);

free(array);
if(fclose(pF)==EOF)
    puts("Input closing error");
}

void doubly_linear()
{
    int slen, y, m, answer, search_id, position, n;
    Head* p0;
    car *p, *p1, *MyNode;
    char brand[MS];
    p0 = MakeHead2();
    FILE* pF = fopen("input.txt", "r");
    p = readFromFile2(pF, p0);

    puts("Enter 1 if you want to continue, 0 to end");
    scanf("%d", &answer);
    getchar();

```

```

while (answer)
{
    puts("Enter car brand");
    fgets(brand, MS, stdin);
    slen = strlen(brand);
    brand[slen-1] = '\0';
    puts("Enter car year");
    scanf("%d", &y);
    puts("Enter car mileage");
    scanf("%d", &m);
    MyNode = CreateNode2(brand, slen, y, m);
    InsertAfter2(p0, MyNode, p);
    p=MyNode;
    puts("Enter 1 if you want to continue, 0 to end");
    scanf("%d", &answer);
    getchar();
}
printf("\n--Input teminated. Your data are:--\n");

p=p0->first;
while(p!=NULL)
{
    printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", p->id, p->brandname, p->year, p->mileage);
    p=p->next;
}

n = number_cars(p0);
puts("Choose the id");
scanf("%d", &search_id);
getchar();
if (search_id < 1 || search_id > n) puts("This id does not
exist.");
else{
    MyNode = (car*)malloc(sizeof(car));
    MyNode = CreateNode2(SelectByOrder(p0, search_id)-
>brandname, strlen(SelectByOrder(p0, search_id)->brandname), Se-
lectByOrder(p0, search_id)->year, SelectByOrder(p0, search_id)-
>mileage);
    puts("Choose the position");
    scanf("%d", &position);
    getchar();
    if (position < 0) puts("Wrong position");
    else
    {
        if (position == 0) {
            InsertBefore2(p0, MyNode, p0->first); p0->first-
>id = 1; autoinkAll(p0);
        }
        else if (position > n) {
            MyNode->next = NULL; InsertAfter2(p0, MyNode,
p0->last);
        }
        else{
            p = SelectByOrder(p0, position);
            InsertBefore2(p0, MyNode, p);
            autoinkN(p0, position);
            MyNode->id = position;
        }
    }
}

```



```

        }
    }

}

p=p0->first;
while(p!=NULL)
{
    printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", p->id, p->brandname, p->year, p->mileage);
    p=p->next;
}

p=p0->first;
while(p!=NULL)
{
    p1=p->next;
    free(p);
    p=p1;
}
free(p0);
if(fclose(pF)==EOF)
    puts("Input closing error");
}

void single_circular()
{
    Head *p0;
    car *p, *p1, *MyNode=NULL;
    char brand[MS];
    int slen, y, m, answer;
    FILE* pF = fopen("input.txt", "r");
    p0 = MakeHead_c();
    p = readFromFile_c(pF, p0);

    puts("Enter 1 if you want to continue, 0 to end");
    scanf("%d", &answer);
    getchar();
    while (answer)
    {
        puts("Enter car brand");
        fgets(brand, MS, stdin);
        slen = strlen(brand);
        brand[slen-1] = '\0';
        puts("Enter car year");
        scanf("%d", &y);
        puts("Enter car mileage");
        scanf("%d", &m);
        MyNode = CreateNode_c(brand, slen, y, m);
        InsertAfter_c(p0, MyNode, p);
        p=MyNode;
        puts("Enter 1 if you want to continue, 0 to end");
        scanf("%d", &answer);
        getchar();
    }
    printf("\n--Input teminated. Your data are:--\n");

    p=p0->first;

```

```

while(p!=NULL)
{
    printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", p->id, p->brandname, p->year, p->mileage);
    p=p->next;
}

makeCircular_c(p0);
int n = p0->cnt;
if (n == 1) puts("You have only 1 element in list.");
else{
    p = p0->first;
    while (p->id != n-1)
        p=p->next;
    deleteSelected_c(p0, p);
    puts("Deletion complete");
    puts("New list: ");
    p=p0->first;
    do
    {
        printf("id: %d; brandname: %s; year: %d; mileage:
%d;\n", p->id, p->brandname, p->year, p->mileage);
        p=p->next;
    }while(p!=p0->last->next);
}

p0->last->next=NULL;
p = p0->first;
while(p!=NULL)
{
    p1=p->next;
    free(p);
    p=p1;
}
free(p0);
if(fclose(pF)==EOF)
    puts("Input closing error");
}

```

Пример работы программы

10 Л/Р:

Исходные данные

```
"C:\Users\Leonid Nikulin\Documents\CodeBlocks\lab10 to screen\bin\Debug\lab10 to screen.exe"
Enter 1 to add elements or 0 to not add
1
Enter car brand
Toyota
Enter car year
2019
Enter car mileage
100
Enter 1 if you want to continue, 0 to end
0
--Input terminated. Your data are:--
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 5; brandname: BMW; year: 1990; mileage: 90000;
id: 6; brandname: Toyota; year: 2019; mileage: 100;
How many elements do you want to have in new List?
2
Enter the id, which you want to copy to new List
1
Enter the id, which you want to copy to new List
6
```

Ввод 1.

Вывод программы

```
New List:
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Toyota; year: 2019; mileage: 100;

Process returned 0 (0x0)   execution time : 39.530 s
Press any key to continue.
```

Вывод 1.

11 Л/Р:

Исходные данные

```
"C:\Users\Leonid Nikulin\Documents\CodeBlocks\lab 11 to screen\bin\Debug\lab 11 to screen.exe"
Enter 1 if you want to continue, 0 to end
1
Enter car brand
Toyota
Enter car year
2019
Enter car mileage
100
Enter 1 if you want to continue, 0 to end
0
--Input terminated. Your data are:--
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 5; brandname: BMW; year: 1990; mileage: 90000;
id: 6; brandname: Toyota; year: 2019; mileage: 100;
Choose the id
1
Choose the position
3_
```

Ввод 2.

Вывод программы

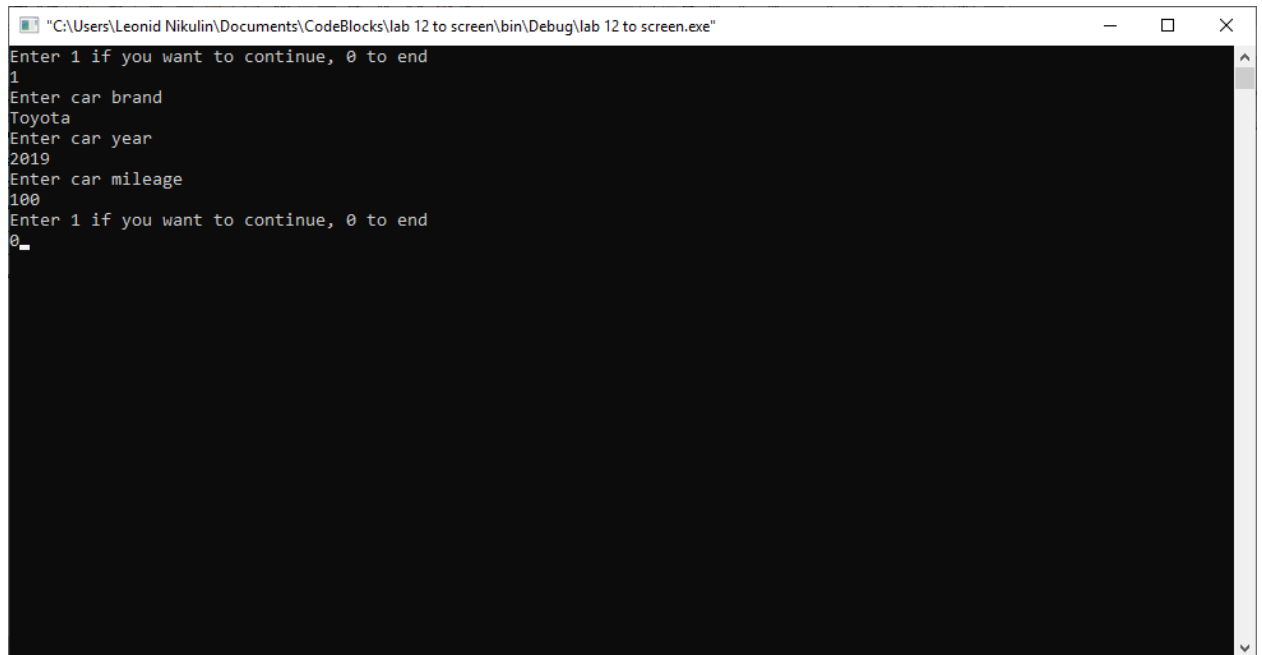
```
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Ford; year: 2008; mileage: 10000;
id: 4; brandname: Lada; year: 2001; mileage: 12345;
id: 5; brandname: Kia; year: 2018; mileage: 0;
id: 6; brandname: BMW; year: 1990; mileage: 90000;
id: 7; brandname: Toyota; year: 2019; mileage: 100;

Process returned 0 (0x0)   execution time : 97.512 s
Press any key to continue.
```

Вывод 2.

12 Л/Р:

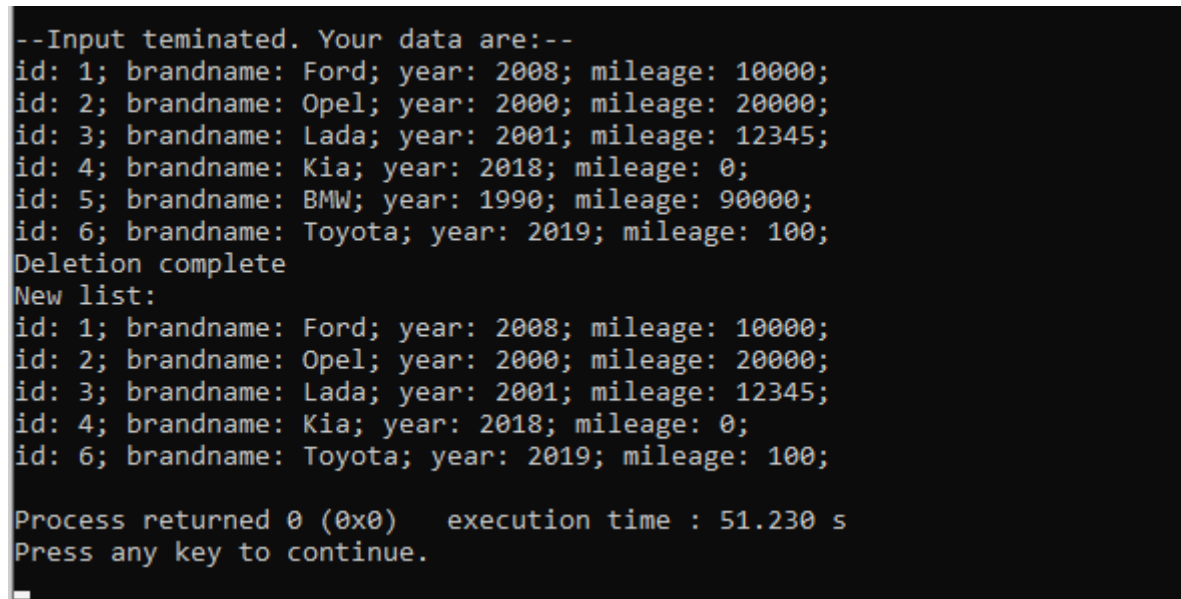
Исходные данные



```
"C:\Users\Leonid Nikulin\Documents\CodeBlocks\lab 12 to screen\bin\Debug\lab 12 to screen.exe"
Enter 1 if you want to continue, 0 to end
1
Enter car brand
Toyota
Enter car year
2019
Enter car mileage
100
Enter 1 if you want to continue, 0 to end
0
```

Ввод 3.

Вывод программы



```
--Input teminated. Your data are:--
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 5; brandname: BMW; year: 1990; mileage: 90000;
id: 6; brandname: Toyota; year: 2019; mileage: 100;
Deletion complete
New list:
id: 1; brandname: Ford; year: 2008; mileage: 10000;
id: 2; brandname: Opel; year: 2000; mileage: 20000;
id: 3; brandname: Lada; year: 2001; mileage: 12345;
id: 4; brandname: Kia; year: 2018; mileage: 0;
id: 6; brandname: Toyota; year: 2019; mileage: 100;

Process returned 0 (0x0)   execution time : 51.230 s
Press any key to continue.
```

Вывод 3.

Выводы:

При выполнении лабораторной работы были получены практические навыки в использовании линейных односвязных, двусвязных и кольцевых списков.