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

Кафедра вычислительной техники

Отчет по лабораторной работе № 11-12
по дисциплине «Программирование»
Тема: Линейные двусвязные списки. Кольцевые списки.

Студентка гр. 9305 Ковалева К.В.

Преподаватель Перязева Ю.В.

Санкт-Петербург

2020

Содержание

Введение

Цель работы – получить практические навыки в разработке алгоритма и написании программы на языке Си для знакомства с синтаксисом, в частности, с работой со списками: линейным двусвзяным списком, односвязным кольцевым (циклическим) списком, а также правилами написания кода на языке Си.

Задание

Перепроектировать структуру, созданную при выполнении лабораторной работы №9 (по выбранной предметной области), так, чтобы одно из информационных полей, содержащих характеристику группы объектов (издательство, модель, тип и т. п.), стало ссылкой на элемент двусвязного линейного списка и выполнить задание в соответствии с вариантом.

1) Разработать подалгоритм вывода значений заданного информационного поля элементов двусвязного списка в прямом или обратном направлении по желанию пользователя с одновременным удалением последнего выводимого на экран элемента.
2) Разработать подалгоритм создания односвязного кольцевого списка с обратным расположением элементов по отношению к полученному односвязному списку, но без элемента, номер которого получен. В случае отсутствия элемента с таким номером вывести сообщение.

Постановка задачи и описание решения

Дана структура с именем CARS (каталог автомобилей, выставленных на продажу), содержащая поля: Name – марка автомобиля, Type – тип кузова автомобиля, Characteristics – технические характеристики автомобиля, Year – год производства автомобиля, Cost – цена. Пользователю представляется выбор: загрузить данные из CSV-файла или ввести собственноручно, с клавиатуры.

В случае удачного открытия файла в этом файле подсчитывается количество строк, после чего выделяется нужный объем памяти для массива структур. При успешном выделении памяти строки из файла считываются по одной в переменную s1, каждая строка файла разделяется на элементы массива строк s2 по разделителям с помощью функции split, а затем структура заполняется данными с помощью функции file_fill. После заполнения полей элемента массива структур промежуточный массив строк очищается с помощью функции clear. Ввод данных с клавиатуры осуществляется за счет функции input_line, заполнение структуры - с помощью функции str_fill. Далее пользователь может воспользоваться основным меню: вывести исходный список в прямом или обратном порядке, удалить последний элемент списка и вывести его в прямом или обратном порядке, а также завершить работу программы. Для удаления программа вызывает функцию del_node, которая берет данные из головы списка, а затем удаляет из списка последний элемент и очищает память после него. Если работа производится с кольцевым списком, пользователь может удалить любой элемент, имеющийся в списке. В случае некорректного ввода пользователь получает соответствующее сообщение, и ему предлагается повторить попытку.

Вывод результата происходит следующим образом: вызывается функция str_out, отвечающая за вывод списка в прямом или обратном порядке в зависимости от выбора пользователя. Указатель присваивает значение головы и, в цикле спускаясь вниз (или вверх, если порядок обратный), выводит данные структуры.

Программа завершает свою работу, освобождая память, выделенную для массива структур, с использованием функции free и clear.

Описание структур

Таблица 1. Описание пользовательских типов данных CARS - структуры, описывающей информационное поле элемента списка.

Поле	Тип	Назначение
name	char*	Марка автомобиля
type	char*	Тип кузова
vin	int*	Характеристики
year	int	Год выпуска
cost	float	Цена

Таблица 2. Описание пользовательских типов данных NODE - структуры, описывающей элемент двусвязного списка.

Поле	Тип	Назначение
ID	Int	Индивидуальный идентификатор списка
content	cars*	Адрес информационного поля элемента списка
next	node*	Адрес следующего элемента списка
prev	node*	Адрес предыдущего элемента списка

Таблица 3. Описание пользовательских типов данных NODE - структуры, описывающей элемент кольцевого списка.

Поле	Тип	Назначение
ID	int	Индивидуальный идентификатор списка
content	cars*	Адрес информационного поля элемента списка
next	node*	Адрес следующего элемента списка

Таблица 4. Описание пользовательских типов данных HEAD - структуры, описывающей «голову» списка.

Поле	Тип	Назначение
Cnt	Int	Количество структур в списке
First	node*	Адрес первого элемента списка
Last	node*	Адрес последнего элемента списка

Описание функций

1. Функция main

Описание:
Точка входа в программу. Отвечает за открытие файла, содержащего данные для последующей работы.
Прототип:
int main()
Пример вызова:
main()
Описание переменных:

Имя	Тип	Назначение
start	head*	Адрес "головы" исходного списка
ph	FILE*	Адрес файла, в котором содержатся исходные данные
data	char*	Адрес первого элемента строки из файла ph
sep	char	Символ-разделитель
ans	int	Выбор пользователя
schet	int	Переменная для хранения последнего ID
count	int	Счетчик

Возвращает значение: 0, если работа программы завершена успешно.

2. Функция delay

Описание:
Функция, вызывающая задержку и очистку экрана при работе меню.
Прототип:
void delay();
Пример вызова:
delay();

3. Функция choose

Описание:
Функция, возвращающая выбор пользователя при работе меню. Проверка выбора на корректность.
Прототип:
int choose();

Пример вызова:

ans=choose();

Описание переменных:

Имя	Тип	Назначение
line	char*	Адрес строки, введенной пользователем
ans_coose	int	Числовой ввод пользователя

Возвращает значение: числового ввода пользователя.

4. **Функция input_line**

Описание:

Функция, осуществляющая считывание данных с клавиатуры.

Прототип:

char* input_line();

Пример вызова:

str->name=input_line();

Описание переменных:

Имя	Тип	Назначение
line_input	char*	Адрес первого элемента вводимой строки

Возвращает значение: line_input.

5. **Функция str_fill**

Описание:

Функция, заполняющая структуру при вводе данных с клавиатуры.

Прототип:

cars* str_fill();

Пример вызова:

sled->content=str_fill();

Описание переменных:

Имя	Тип	Назначение
str	cars*	Адрес структуры, в котором содержится информация об автомобиле
s	char*	Вспомогательная переменная
cop	int	Переменная для хранения ввода пользователя

Возвращает значение: str.

6. Функция str_out

Описание:

Функция вывода списка. Пока указатель на голову не NULL, выводит данные одной строки и переходит к следующей

Прототип:

void str_out(head *h);

Пример вызова:

str_out(start);

Описание переменных:

Имя	Тип	Назначение
p	node*	Адрес структуры для вывода элемента списка

7. Функция str_out_result

Описание:

Функция вывода рабочего списка (для кольцевого) в обратном порядке.

Прототип:

void str_out(head *h);

Пример вызова:

str_out(start);

Описание переменных:

Имя	Тип	Назначение
p	node*	Адрес структуры для вывода элемента списка
i	int	Параметры цикла
j	int	

8. Функция clear_str

Описание:

Функция, очищающая структуры.

Прототип:

void clear_str(head *h);

Пример вызова:

clear_str(start);

Описание переменных:

Имя	Тип	Назначение
p	node*	Адрес структуры, для которой необходимо освободить память

9. Функция clear

Описание:

Функция, освобождающая память из-под двумерного массива строк.

Прототип:

void clear(char** data1, int schet);

Пример вызова:

clear(str, count);

Описание переменных:

Имя	Тип	Назначение
i	int	Параметр цикла

10.Функция split

Описание:

Функция, делящая строку по знаку разделителю. Каждая строка файла разделяется на элементы промежуточного массива строк s2 по разделителям с помощью функции, и в зависимости от типа поля элемента массива структур выполняется преобразование элемента массива строк в поле отдельной структуры.

Прототип:

char **split(char *data0, char sep);

Пример вызова:

str=split(data, sep);

Описание переменных:

Имя	Тип	Назначение
str	char*	Строка, содержащая структурные данные
i	int	Параметр цикла
j	int	Параметр цикла
k	int	Параметр для столбца
m	int	Количество символов разделителей в строке
key	int	Флаг для выделения памяти
count	int	Количество строк

Возвращаемое значение: массив строк.

11. Функция file_fill

Описание:

Функция, заполняющая структуру из файла.

Прототип:

void file_fill(head *h, char *data, char sep, int schet);

Пример вызова:

file_fill(start, data, sep, schet);

Описание переменных:

Имя	Тип	Назначение
ks	cars*	Переменная для заполнения информационного поля
str	char **	Адрес первого элемента первой строки разбиения исходной строки по знаку разделителю

Возвращает значение: массив строк.

12.Функция create_head

Описание:

Функция, создающая «голову» списка

Прототип:

head *create_head();

Пример вызова:

start=create_head();

Описание переменных:

Имя	Тип	Назначение
h	head*	Указатель на «голову» списка

Возвращает значение: указатель на «голову».

13.Функция create_elem

Описание:

Функция, создающая элемент списка

Прототип:

void create_elem(head* h, int k, int g, cars *ks);

Пример вызова:

create_elem(h, schet, 0, ks);

Описание переменных:

Имя	Тип	Назначение
sled	node*	Адрес структуры для создания исходного списка

Возвращает значение: указатели на элементы.

14.Функция create_elem

Описание:

Функция, создающая элемент списка

Прототип:

void del_node(head *st, int id);

Пример вызова:

del_node(start, start->cnt);

Описание переменных:

Имя	Тип	Назначение
p0	node*	Адрес структуры для удаления эл-та списка
p1	node*	Адрес структуры для удаления эл-та списка

Возвращает значение: список без удаленного элемента.

15.Функция copy_str

Описание:

Функция, создающая копию исходного списка. Она позволяет сохранять исходные данные, а работу производить с копией списка.

Прототип:

void copy_str(head *h1, head *h2);

Пример вызова:

copy_str(start, worker);

Описание переменных:

Имя	Тип	Назначение
Tmp	cars**	Адрес первого элемента массива указателей на структуры
P	node*	Адрес элемента списка для копирования
Count	int	Счетчик структур, для которых память выделена успешно
I	int	Параметр цикла

16. Функция copy

Описание:

Функция, копирующая содержимое структур.

Прототип:

void copy(cars *first, cars *second);

Пример вызова:

copy(p->content, tmp[i]);

Контрольные примеры

Двусвязный список

Входные данные: файл

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42
Hyundai;estate;3;1;1;2003;0,21

Выходные данные: прямой порядок

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42

Входные данные: клавиатура

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42
Hyundai;estate;3;1;1;2003;0,21

Выходные данные: обратный порядок

Hyundai;estate;3;1;1;2003;0,21
Hyundai;hatchback;6;5;0;2013;0,42
Porsche;crossover;7;7;7;2015;3,49
Cadillac;sedan;1;2;6;2020;2,70
Volkswagen;estate;3;1;4;2005;0,52
Hyundai;crossover;2;2;8;2011;1,25

Кольцевой список

Входные данные: файл, 4 элемент

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42
Hyundai;estate;3;1;1;2003;0,21

Выходные данные:

Hyundai;estate;3;1;1;2003;0,21
Hyundai;hatchback;6;5;0;2013;0,42
Porsche;crossover;7;7;7;2015;3,49
Volkswagen;estate;3;1;4;2005;0,52
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;sedan;8;7;6;2017;0,87

Входные данные: файл, 9 элемент

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42
Hyundai;estate;3;1;1;2003;0,21

Выходные данные:

Некорректный ввод. Попробуйте еще раз

Входные данные: клавиатура, 1 элемент

Volkswagen;sedan;8;7;6;2017;0,87
Hyundai;crossover;2;2;8;2011;1,25
Volkswagen;estate;3;1;4;2005;0,52
Cadillac;sedan;1;2;6;2020;2,70
Porsche;crossover;7;7;7;2015;3,49
Hyundai;hatchback;6;5;0;2013;0,42
Hyundai;estate;3;1;1;2003;0,21

Выходные данные:

Hyundai;estate;3;1;1;2003;0,21
Hyundai;hatchback;6;5;0;2013;0,42
Porsche;crossover;7;7;7;2015;3,49
Cadillac;sedan;1;2;6;2020;2,70
Volkswagen;estate;3;1;4;2005;0,52
Hyundai;crossover;2;2;8;2011;1,25

Алгоритм на естественном языке

Начало.

Шаг 1. Вывод меню. Если пользователь выбирает пункт 1), переход на шаг 2, если пользователь выбирает пункт 2), переход на шаг 3, если пользователь выбирает пункт 3), переход на шаг 4, если пользователь выбирает пункт 4), переход на шаг 5, если пользователь выбирает пункт 5), переход на шаг 6, если пользователь выбирает пункт 6), переход на шаг 7. Если ввод некорректен, выводится сообщение об ошибке, выполняется переход на шаг 1.

Шаг 2. Вывод меню. Если пользователь выбирает пункт 1), то происходит считывание данных из CSV-файла при помощи разделения информации символом-разделителем. Если пункт 2), то происходит ввод исходной последовательности структур с клавиатур, проверка на корректность введенных данных, если данные некорректны, выводится сообщение об ошибке, ввод повторяется до тех пор, пока данные не станут корректными. После ввода очередной последовательности, узнается желание пользователя продолжить ввод. Если пункт 3), то возвращение к главному меню.

Шаг 3. Вывод исходной последовательности структур. Возвращение к шагу 1.

Шаг 4 (кольцевой). Копирование исходного списка в рабочий список. Возвращение к шагу 1.

Шаг 5. Удаление последнего элемента списка (для двусвязного), либо удаление любого элемента (для кольцевого). Для второго случая выводится количество доступных для удаления ID, и пользователь вводит требуемый. Если данные некорректны, выводится сообщение об ошибке. Возвращение к шагу 1.

Шаг 6. Ввод условия вывода результата (для двусвязного), проверка введенного значения. Если данные некорректны, выводится сообщение об ошибке. Выводится результирующий массив структур. Возвращение к шагу 1.

Шаг 7. Завершение работы программы.

Конец.

Текст программы

Двусвязный список. main.c

```
#include "my_lib.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <locale.h>

#include <errno.h>


int main()

{

    head *start=NULL;       ///Адрес "головы" исходного списка

    FILE *ph;              ///Адрес файла, в котором содержатся исходные
данные

    char *data,             ///Адрес первого элемента строки, из файла
ph

        sep;                ///Символ-разделитель

    int ans,                ///Выбор пользователя

        schet,              ///Переменная для хранения последнего ID

        count;              ///Счетчик


    setlocale(LC_ALL, "RUS");
```

```
    count=0;
    schet=0;

    do
    {
        puts("Меню:");
        puts("1) - Ввод последовательности структур");
        puts("2) - Вывод введенной последовательности");
        puts("3) - Удаление элемента");
        puts("4) - Конец");
        ans=choose();
        system("cls");
        switch(ans)
        {
            ///-------------------------------------------------------
-
            case(1):
                do
                {
                    puts("Подменю:");
                    puts("1) - Ввод структур из файла");
                    puts("2) - Ввод структур с клавиатуры");
```

```
                puts("3) - Возвращение в главное меню");
                ans=choose();
                system("cls");
                switch(ans)
                {
                    ///-----------------------------------------
-
                    case(1):
                        if (start)
                            clear_str(start);

                        start=create_head();
                        ph=fopen("car_catalog.csv", "r");
                        if (ph)
                        {
                            sep=';';
                            data=(char*)malloc(80*sizeof(char));
                            if (data)
                            {
                                while(fgets(data, 80, ph)!=NULL)
                                {
                                    schet++;
```

```
                                    file_fill(start,   data,   sep,
schet);
                                }
                                puts("Ввод      данных      успешно
завершен!");
                                delay();
                            }
                            else
                            {
                                puts("Ошибка выделения памяти");
                                fclose(ph);
                                delay();
                            }
                        }
                        else
                        {
                            perror("Произошла ошибка: ");
                            puts("Попробуйте   ввести   данные   с
клавиатуры.");
                            delay();
                        }
                        break;
```

```
                        ///-----------------------------------------
-
                        case(2):
                            if (start)
                            {
                                ans=choose();
                                if (ans!=0)
                                    count=0;
                            }
                            else
                                count=0;

                            do
                            {
                                count++;
                                schet++;
                                if ((count!=1)&&(start))
                                {
                                    create_elem(start, schet, 1, NULL);
                                    puts("Хотите ввести еще структуры?
(1)");
                                    ans=choose();
                                }
```

```
                                else if(start==NULL)
                                {
                                    start=create_head();
                                    create_elem(start, schet, 1, NULL);
                                    puts("Хотите ввести еще структуры?
(1)");
                                    ans=choose();
                                }
                                else
                                {
                                    clear_str(start);
                                    start=create_head();
                                    create_elem(start, schet, 1, NULL);
                                    puts("Хотите ввести еще структуры?
(1)");
                                    ans=choose();
                                }
                            }while(ans==1);
                            puts("\nВвод структур завершен");
                            delay();
                            break;
                        ///-----------------------------------------
-
```

```
                        case(3):
                            puts("Возвращение в главное меню");
                            delay();
                            break;
                        ///-----------------------------------------
-
                        default:
                            puts("Введено некорректное значение.
Попробуйте ещё раз");
                            delay();
                    }
                }while(ans!=3);
                ans=1;

                break;
            ///-------------------------------------------------------
-
            case(2):
                if (start!=NULL)
                {
                    puts("Введенная последовательность структур:");
                    str_out(start);
                }
```

```
            else
            {
                puts("Вы ещё не ввели ни одной структуры");
                delay();
            }
            break;
        ///-------------------------------------------------------
-
        case(3):
            if (start!=NULL)
            {
                puts("Удаление элемента...");
                del_node(start, start->cnt);
                puts("Успешно!");
                str_out(start);
            }
            else
                puts("Вы ещё не ввели ни одной структуры");
            delay();
            break;
        ///-------------------------------------------------------
-
        case(4):
```

```
                puts("Завершение программы");
                if (start)
                    clear_str(start);
                delay();
                break;
            ///-------------------------------------------------------
-
            default:
                puts("Некорректный ввод. Попробуйте ещё раз");
                delay();
        }
    }while(ans!=4);

    return 0;
}
```

Двусвязный список. my_lib.h

```
#ifndef FUNC_H_INCLUDED
#define FUNC_H_INCLUDED

///Структура, описывающая информационное поле элемента списка
typedef struct C
{
```

```
    char* name;             ///Название автомобиля
    char* type;             ///Тип кузова
    int* vin;               ///Характеристики автомобиля
    int year;               ///Год выпуска
    float cost;             ///Цена автомобиля
}cars;

///Структура, описывающая элемент списка
typedef struct e
{
    int ID;                 ///Индивидуальный идентификатор списка
    cars *content;         ///Адрес информационного поля элемента списка
    struct e *next;         ///Адрес следующего элемента списка
    struct e *prev;         ///Адрес предыдущего элемента списка
}node;

///Структура, описывающая "голову" списка
typedef struct h
{
    int cnt;                ///Количество структур в списке
    node *first;            ///Адрес первого элемента списка
    node *last;             ///Адрес последнего элемента списка
```

```
}head;

///Функция, вызывающая задержку и очистку экрана
void delay();

///Функция, возвращающая выбор пользователя
int choose();

///Функция, осуществлющая считывание
char* input_line();

///Функция, заполняющая структуру
cars* str_fill();

///Функция, выводящая список
void str_out(head *h);

///Функция очищающая структуры
void clear_str(head *h);

///Функция, освобождающая память из-под двумерного массива строк
void clear(char** data1, int schet);
```

```
///Функция, делящая строку по знаку разделителю
char **split(char *data0, char sep);

///Функция, заполняющая структуру из файла
void file_fill(head *h, char *data, char sep, int schet);

///Функция, создающая "голову" списка
head *create_head();

///Функция, создающая лемент списка
void create_elem(head* h, int k, int g, cars *ks);

///Функция, удаляющая элемент списка
void del_node(head *st, int id);

#endif // FUNC_H_INCLUDED
```

Двусвязные списки. func.c

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

```
#include "my_lib.h"


void delay()
{
    system("pause");
    system("cls");
}


int choose()
{
    char* line;         ///Адрес строки, введенной пользователем
    int ans_coose;      ///Числовой ввод пользователя


    line=(char*)malloc(15*sizeof(char));
    if (line!=NULL)
    {
        fflush(stdin);
        fgets(line, 15, stdin);
        ans_coose=atoi(line);
    }
    else
```

```
    {
        puts("Ошибка выделения памяти");
        ans_coose=5;
    }

    free(line);

    return ans_coose;
}

char* input_line()
{
    char* line_input;          ///Адрес первого элемента вводимой строки

    line_input=(char*)malloc(80*sizeof(char));
    if (line_input!=NULL)
    {
        fflush(stdin);
        fgets(line_input, 79, stdin);
        line_input[strlen(line_input)-1]='\0';
        line_input=(char*)realloc(line_input,
(strlen(line_input)+1)*sizeof(char));
```

```
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }
    return line_input;
}

cars* str_fill()
{
    cars *str; ///Адрес структуры, в которой содержится информация о мотоцикле
    char *s;
    int cop;            ///Переменная для хранения ввода пользователя

    str=(cars*)malloc(sizeof(cars));
    if (str!=NULL)
    {
        str->name=(char*)malloc(80*sizeof(char));
        if (str->name!=NULL)
        {
```

```
            puts("Введите структуру:");
            printf("Введите название автомобиля - ");
            str->name=input_line();
            str->name=(char*)realloc(str->name,           (strlen(str-
>name)+1)*sizeof(char));

            printf("Введите тип кузова - ");
            str->type=input_line();
            str->type=(char*)realloc(str->type,           (strlen(str-
>type)+1)*sizeof(char));
            str->vin=(int*)malloc(3*sizeof(int));

            do
            {
                printf("Как бы Вы оценили состояние автомобиля от 0 до
10? - ");
                cop=choose();
                if ((cop<0)||(cop>10))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>10));
            str->vin[0]=cop;

            do
```

```
            {
                printf("Коробка передач: 0 - автомат, 1 - механика, 2 - вариантор, 3 - робот. ");
                cop=choose();
                if ((cop<0)||(cop>3))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>3));
            str->vin[1]=cop;

            do
            {
                printf("Тип двигателя: 0 - бензиновый, 1 - дизельный, 2 - газ, 3 - гибрид, 4 - электро. ");
                cop=choose();
                if ((cop<0)||(cop>4))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>4));
            str->vin[2]=cop;

            do
            {
                printf("Введите год выпуска автомобиля - ");
                cop=choose();
```

```
        if ((cop<1885)||(cop>2020))
            puts("Некоррекнтый ввод, попробуйте снова");
    }while((cop<1885)||(cop>2020));
    str->year=cop;

    do
    {
        printf("Введите стоимость автомобиля - ");
        s=input_line();
        cop=atof(s);
        if ((cop<20)||(cop>1000000))
            puts("Некоррекнтый ввод, попробуйте снова");
    }while((cop<20)||(cop>1000000));
    str->cost=cop;
    free(s);
}
else
{
    puts("Ошибка выделения памяти для поля name");
    delay();
    free(str);
}
```

```
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }
    return str;
}

void print_header()
{
    printf("|%20s                |%10s                |%18s|%6s|%6s|\n",
"NAME","TYPE","CHARACTERISTICS","YEAR","COST");
    printf("+--------------------+----------+------------------+----
--+------+\n");
}

void str_out(head *h)
{
    node *p;            ///Адрес структуры для вывода эл-та списка
    int ans_out;        ///Выбор пользователя1

    p=(node*)malloc(sizeof(node));
```

```
    if (p)
    {
        puts("В каком порядке вы хотите вывести исходный список? (0 -
прямой, 1 - обратный)");
        ans_out=choose();
        do
        {
            if ((ans_out!=0)&&(ans_out!=1))
            {
                puts("Некорректное значение. Попробуйте снова");
                ans_out=choose();
            }
        }while((ans_out!=0)&&(ans_out!=1));

        system("cls");
        if (ans_out)
        {
            p=h->last;
            puts("Исходный список в обратном порядке");
            print_header();
            while(p)
            {
```

```
                printf("|%20s |%10s |%5d %5d %5d |%5d |%5.2f |\n",
                    p->content->name,p->content->type,p->content->vin[0],p->content->vin[1],p->content->vin[2],p->content->year,p->content->cost);
                p=p->prev;
            }
        }
        else
        {
            p=h->first;
            puts("Исходный список в прямом порядке");
            print_header();
            while(p)
            {
                printf("|%20s |%10s |%5d %5d %5d |%5d |%5.2f |\n",
                    p->content->name,p->content->type,p->content->vin[0],p->content->vin[1],p->content->vin[2],p->content->year,p->content->cost);
                p=p->prev;
            }
        }
        delay();
    }
```

```
}


void clear_str(head *h)
{
    node *p=NULL;          ///Адрес структуры, для которой необходимо освобождить память


    p=(node*)malloc(sizeof(node));
    if(p)
    {
        p=h->first;
        while(p)
        {
            h->first=p->next;
            free(p->content->name);
            free(p->content);
            free(p);
            p=h->first;
            h->cnt=h->cnt-1;
        }
        free(h);
        h=NULL;
```

```
    }
}

void clear(char** data1, int schet)
{
    int i;

    for(i=0;i<schet;i++)
        free(data1[i]);

    free(data1);
}

char **split(char *data0, char sep)
{
    char **str=NULL;            ///Адрес первого элемента первой строки
    int i,                      ///Параметр цикла
        j,                      ///Параметр цикла
        k,                      ///
        m;                       ///Количество символов разделителей в
строке
    int key,                     ///Переменная, показывающая, полностью
ли выделилась память
```

```
    count;                  ///Счетчик введенных структур


for(j=0,m=0;j<strlen(data0);j++)
    if(data0[j]==sep)
        m++;


key=1;
str=(char**)malloc((m+1)*sizeof(char*));
if(str!=NULL)
{
    for(i=0,count=0;i<=m;i++,count++)
    {
        str[i]=(char*)malloc(strlen(data0)*sizeof(char));
        if(str[i]==NULL)
        {
            key=0;
            i=m;
        }
    }


    if(key)
    {
```

```
            k=0;
            m=0;
            for(j=0;j<strlen(data0);j++)
            {
                if(data0[j]!=sep)
                    str[m][j-k]=data0[j];
                else
                {
                    str[m][j-k]='\0';
                    k=j+1;
                    m++;
                }
            }
            str[m][j-k]='\0';
        }
        else
            clear(str, count);
    }
    return str;
}

head *create_head()
```

```
{
    head *h=NULL;                ///"Голова" списка


    h=(head*)malloc(sizeof(head));
    if (h)
    {
        h->cnt=0;
        h->first=NULL;
        h->last=NULL;
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }


    return h;
}


void create_elem(head* h, int k, int g, cars *ks)
{
```

```
    node *sled=NULL;           ///Адрес структуры для создания исходного
списка


    sled=(node*)malloc(sizeof(node));
    if (sled)
    {
        if (h->first==NULL)
        {
            sled->prev=NULL;
            h->first=sled;
        }
        else
        {
            h->last->next=sled;
            sled->prev=h->last;
        }


        h->cnt++;
        sled->ID=k;
        sled->next=NULL;
        h->last=sled;
        if (g)
```

```
            sled->content=str_fill();
        else
            sled->content=ks;
    }
}

void file_fill(head *h, char *data, char sep, int schet)
{
    cars *ks;              ///Переменная для заполнения информационного поля
    char **str;        ///Адрес первого элемента первой строки разбиения исходной строки по знаку разделителю

    data[strlen(data)-1]='\0';
    str=split(data, sep);
    ks=(cars*)malloc(sizeof(cars));
    ks->name=(char*)malloc((strlen(str[0])+1)*sizeof(char));
    strcpy(ks->name, str[0]);
    strcpy(ks->type, str[1]);
    ks->vin[0]=atoi(str[2]);
    ks->vin[1]=atoi(str[3]);
    ks->vin[2]=atoi(str[4]);
    ks->year=atoi(str[5]);
```

```
    ks->cost=atof(str[6]);
    create_elem(h, schet, 0, ks);
    clear(str, 7);
}

void del_node(head *h, int id)
{
    node *p0=NULL,          ///Адрес структуры для удаления эл-та списка
        *p1=NULL;           ///Адрес структуры для удаления эл-та списка

    p0=(node*)malloc(sizeof(node));
    p1=(node*)malloc(sizeof(node));
    if (p0&&p1)
    {
        p1=h->first;
        p0=p1;
        while(p1->ID!=id)
        {
            p0=p1;
            p1=p1->next;
        }
```

```
if (h->cnt==1)
{
    h->first=NULL;
    h->last=NULL;
    free(p1->content->name);
    free(p1->content);
    free(p1);
}
else if (p1==h->last)
{
    free(p1->content->name);
    free(p1->content);
    free(p1);
    p0->next=NULL;
    h->last=p0;
}
else if (p1==h->first)
{
    h->first=p1->next;
    free(p1->content->name);
    free(p1->content);
    free(p1);
```

```
        }
        else
        {
            p0->next=p1->next;
            free(p1->content->name);
            free(p1->content);
            free(p1);
        }
        h->cnt=h->cnt-1;
    }
}
```

Кольцевой список. main.c

```
#include "my_lib.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <locale.h>
#include <errno.h>

int main()
{
```

```
    head *start=NULL,       ///Адрес "головы" исходного списка
        *worker=NULL;       ///Адрес "головы" рабочего списка
    FILE *ph;              ///Адрес файла, в котором содержатся исходные
данные
    node *elem=NULL;        ///Элемент списка
    char *data,             ///Адрес первого элемента строки, из файла
ph
        sep;                ///Символ-разделитель
    int *id=NULL,           ///Указатель на первый элемент массива ID
элементов списка
        ans,                ///Выбор пользователя
        k,                  ///Выбор пользователя на удаление элемента
списка
        schet,              ///Переменная для хранения последнего ID
        flag1,              ///Флаг, отвечающий за проверку вывода
        count,              ///Счетчик
        i;                  ///Параметр цикла


    setlocale(LC_ALL, "RUS");
    count = 0;
    schet = 0;
    flag1 = 1;
```

```
    do
    {
        puts("Меню:");
        puts("1) - Ввод последовательности структур");
        puts("2) - Вывод введенной последовательности");
        puts("3) - Копирование исходного массива в рабочий");
        puts("4) - Удаление элемента");
        puts("5) - Вывод результата");
        puts("6) - Конец");
        ans=choose();
        system("cls");
        switch(ans)
        {
            ///-------------------------------------------------------
-
            case(1):
                do
                {
                    puts("Подменю:");
                    puts("1) - Ввод структур из файла");
                    puts("2) - Ввод структур с клавиатуры");
                    puts("3) - Возвращение в главное меню");
```

```
                    ans=choose();
                    system("cls");
                    switch(ans)
                    {
                        ///-----------------------------------------
-
                        case(1):
                            if (start)
                                clear_str(start);

                            start=create_head();
                            ph=fopen("car_catalog.csv", "r");
                            if (ph)
                            {
                                sep=';';
                                data=(char*)malloc(80*sizeof(char));
                                if (data)
                                {
                                    while(fgets(data, 80, ph)!=NULL)
                                    {
                                        schet++;
                                        file_fill(start,   data,   sep,
schet);
```

```
                    }
                    puts("Ввод данных успешно
завершен!");
                    delay();
                    fclose(ph);
                }
                else
                {
                    puts("Ошибка выделения памяти");
                    fclose(ph);
                    delay();
                }
            }
            else
            {
                perror("Произошла ошибка : ");
                delay();
            }
            break;
        ///-----------------------------------------
-
        case(2):
            if (start)
```

```
                    {
                        puts("Хотите дополнить текущий список?
(0)");
                        ans=choose();
                        if (ans!=0)
                            count=0;
                    }
                    else
                        count=0;

                    do
                    {
                        count++;
                        schet++;
                        if ((count!=1)&&(start))
                        {
                            create_elem(start, schet, 1, NULL);
                            puts("Хотите ввести еще структуры?
(1)");
                            ans=choose();
                        }
                        else if(start==NULL)
                        {
```

```
                start=create_head();
                create_elem(start, schet, 1, NULL);
                puts("Хотите ввести еще структуры?
(1)");
                ans=choose();
            }
            else
            {
                clear_str(start);
                start=create_head();
                create_elem(start, schet, 1, NULL);
                puts("Хотите ввести еще структуры?
(1)");
                ans=choose();
            }
        }while(ans==1);
        puts("\nВвод структур завершен");
        delay();
        break;
    ///-----------------------------------------
-
    case(3):
        puts("Возвращение в главное меню");
```

```
                            delay();
                            break;
                        ///-----------------------------------------
-
                        default:
                            puts("Введено некорректное значение.
Попробуйте ещё раз");
                            delay();
                    }
                }while(ans!=3);

                if (start)
                {
                    if (worker)
                        clear_str(worker);

                    worker=create_head();
                    copy_str(start, worker);
                    flag1 = 1;
                }
                ans=1;
                break;
```

```
            ///-------------------------------------------------------
-
            case(2):
                if (start!=NULL)
                {
                    puts("Введенная последовательность структур:");
                    str_out(start);
                }
                else
                {
                    puts("Вы ещё не ввели ни одной структуры");
                    delay();
                }
                break;
            ///-------------------------------------------------------
-
            case(3):
                if (start!=NULL)
                {
                    if (worker)
                        clear_str(worker);
                    worker=create_head();
                    copy_str(start, worker);
```

```
                    flag1 = 1;
                    puts("Список успешно скопирован");
                    delay();
                }
                else
                {
                    puts("Вы ещё не сформировали исходный список");
                    delay();
                }
                break;
            ///-------------------------------------------------------
-
            case(4):
                if (start!=NULL)
                {
                    if(worker->cnt!=0)
                    {
                        id=(int*)realloc(id, worker->cnt*sizeof(int));
                        if (id)
                        {
                            puts("В списке находятся элементы со
следующими ID:");
                            elem=worker->first;
```

```
                    for(i=0;i<worker->cnt;i++)
                    {
                        printf("ID элемента %d-го по списку =
%d\n", (i+1), elem->ID);
                        id[i]=elem->ID;
                        elem=elem->next;
                    }

                    puts("Выберете ID элемента, который нужно
удалить:");
                    do
                    {
                        k=choose();
                        if(k<1)
                        {
                            puts("Некорректный ввод. Попробуйте
еще раз");
                            k=0;
                        }
                        else
                        {
                            for(i=0;i<worker->cnt;i++)
                            if(id[i]==k)
```

```
                    i=worker->cnt+1;


                    if(i!=worker->cnt+2)
                    {
                        puts("Некорректный ввод.
Попробуйте еще раз");
                        k=0;
                    }
                    else
                        del_node(worker, k);
                }
            }while(k==0);
            free(id);
            id = NULL;
            puts("Улемент успешно удален");
            delay();
        }
        else
        {
            puts("Ошибка при выделении памяти");
            system("pause");
        }
```

```
                    }
                else
                {
                    puts("Рабочий список пуст, невозможно выполнить
действие");
                    delay();
                }
            }
            else
            {
                puts("Вы ещё не ввели ни одной структуры");
                delay();
            }
            ans=4;
            break;
            ///-------------------------------------------------------
-
            case(5):
                if (flag1==1)
                {
                    if (worker->cnt!=0)
                    {
                        puts("Сформированный по запросу результат:");
```

```
                    str_out_result(worker);
                }
                else
                {
                    puts("Рабочий список пуст");
                    delay();
                }
            }
            else
            {
                puts("Вы еще не ввели условия вывода");
                delay();
            }
            break;
        ///-------------------------------------------------------
-
        case(6):
            puts("Завершение программы");
            if (start)
                clear_str(start);

            if (worker)
```

```
                    clear_str(worker);


                if (elem)
                {
                    free(elem);
                    free(id);
                }
                delay();
                break;
            ///-------------------------------------------------------
-
            default:
                puts("Некорректный ввод. Попробуйте ещё раз");
                delay();
        }
    }while(ans!=6);


    return 0;
}
```

Кольцевой список. my_lib.h

```
#ifndef FUNC_H_INCLUDED
```

```
#define FUNC_H_INCLUDED


///Структура, описывающая информационное поле элемента списка
typedef struct C
{
    char* name;             ///Название автомобиля
    char* type;             ///Тип кузова
    int* vin;               ///Характеристики автомобиля
    int year;               ///Год выпуска
    float cost;             ///Цена автомобиля
}cars;


///Структура, описывающая элемент списка
typedef struct e
{
    int ID;                 ///Индивидуальный идентификатор списка
    cars *content;         ///Адрес информационного поля элемента списка
    struct e *next;         ///Адрес следующего элемента списка
}node;


///Структура, описывающая "голову" списка
typedef struct h
```

```
{
    int cnt;                ///Количество структур в списке
    node *first;            ///Адрес первого элемента списка
    node *last;             ///Адрес последнего элемента списка
}head;

///Функция, вызывающая задержку и очистку экрана
void delay();

///Функция, возвращающая выбор пользователя
int choose();

///Функция, осуществлющая считывание
char* input_line();

///Функция, заполняющая структуру
cars* str_fill();

///Функции, выводящие список
void str_out(head *h);
void str_out_result(head *h);
```

```
///Функция очищающая структуры
void clear_str(head *h);

///Функция, освобождающая память из-под двумерного массива строк
void clear(char** data1, int schet);

///Функция, делящая строку по знаку разделителю
char **split(char *data0, char sep);

///Функция, заполняющая структуру из файла
void file_fill(head *h, char *data, char sep, int schet);

///Функция, создающая "голову" списка
head *create_head();

///Функция, создающая элемент списка
void create_elem(head* h, int k, int g, cars *ks);

///Функция, удаляющая элемент списка
void del_node(head *st, int id);

///Функция, копирующая содержимое структур
```

```
void copy(cars *first, cars *second);


///функция, копирующая список
void copy_str(head *h1, head *h2);


#endif // FUNC_H_INCLUDED
```

Кольцевой список. func.c

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


#include "my_lib.h"


void delay()
{
    system("pause");
    system("cls");
}


int choose()
{
```

```
    char* line;         ///Адрес строки, введенной пользователем
    int ans_coose;      ///Числовой ввод пользователя


    line=(char*)malloc(15*sizeof(char));
    if (line!=NULL)
    {
        fflush(stdin);
        fgets(line, 15, stdin);
        ans_coose=atoi(line);
    }
    else
    {
        puts("Ошибка выделения памяти");
        ans_coose=5;
    }


    free(line);


    return ans_coose;
}


char* input_line()
```

```
{
    char* line_input;          ///Адрес первого элемента вводимой строки


    line_input=(char*)malloc(80*sizeof(char));
    if (line_input!=NULL)
    {
        fflush(stdin);
        fgets(line_input, 79, stdin);
        line_input[strlen(line_input)-1]='\0';
        line_input=(char*)realloc(line_input,
(strlen(line_input)+1)*sizeof(char));
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }
    return line_input;
}


cars* str_fill()
{
```

```
    cars *str;
    char *s;          ///Адрес структуры, в которой содержится информация
о мотоцикле
    int cop;             ///Переменная для хранения ввода пользователя


    str=(cars*)malloc(sizeof(cars));
    if (str!=NULL)
    {
        str->name=(char*)malloc(80*sizeof(char));
        if (str->name!=NULL)
        {
             puts("Введите структуру:");
            printf("Введите название автомобиля - ");
            str->name=input_line();
            str->name=(char*)realloc(str->name,            (strlen(str-
>name)+1)*sizeof(char));


            printf("Введите тип кузова - ");
            str->type=input_line();
            str->type=(char*)realloc(str->type,            (strlen(str-
>type)+1)*sizeof(char));
            str->vin=(int*)malloc(3*sizeof(int));
```

```
            do
            {
                printf("Как бы Вы оценили состояние автомобиля от 0 до
10? - ");
                cop=choose();
                if ((cop<0)||(cop>10))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>10));
            str->vin[0]=cop;

            do
            {
                printf("Коробка передач: 0 - автомат, 1 - механика, 2 -
вариантор, 3 - робот. ");
                cop=choose();
                if ((cop<0)||(cop>3))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>3));
            str->vin[1]=cop;

            do
            {
```

```
                printf("Тип двигателя: 0 - бензиновый, 1 - дизельный, 2
- газ, 3 - гибрид, 4 - электро. ");
                cop=choose();
                if ((cop<0)||(cop>4))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<0)||(cop>4));
            str->vin[2]=cop;

            do
            {
                printf("Введите год выпуска автомобиля - ");
                cop=choose();
                if ((cop<1885)||(cop>2020))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<1885)||(cop>2020));
            str->year=cop;

            do
            {
                printf("Введите стоимость автомобиля - ");
                s=input_line();
                cop=atof(s);
```

```
                if ((cop<20)||(cop>1000000))
                    puts("Некоррекнтый ввод, попробуйте снова");
            }while((cop<20)||(cop>1000000));
            str->cost=cop;
            free(s);
        }
        else
        {
            puts("Ошибка выделения памяти для поля name");
            delay();
            free(str);
        }
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }
    return str;
}

void print_header()
```

```
{
    printf("|%20s                |%10s                |%18s|%6s|%6s|\n",
"NAME","TYPE","CHARACTERISTICS","YEAR","COST");
    printf("+--------------------+----------+------------------+------
--+------+\n");
}

void str_out(head *h)
{
       node *p;            ///Адрес структуры для вывода эл-та списка
    int i;

    p=(node*)malloc(sizeof(node));
    if (p)
    {
        p=h->first;
        print_header();
        for(i=h->cnt;i>0;i--)
        {
            printf("|%20s |%10s |%5d %5d %5d |%5d |%6.2f |\n",
                p->content->name,p->content->type,p->content-
>vin[0],p->content->vin[1],p->content->vin[2],p->content->year,p-
>content->cost);
```

```
            p=p->next;
        }
        delay();
    }
}

void str_out_result(head *h)
{
    node *p;            ///Адрес структуры для вывода эл-та списка
    int i,j;

    p=(node*)malloc(sizeof(node));
    if (p)
    {
        print_header();
        for(i=h->cnt-1;i>=0;i--)
        {
            p=h->first;
            for(j=0;j<i;j++)
                p=p->next;
            printf("|%20s |%10s |%5d %5d %5d |%5d |%6.2f |\n",
```

```
                p->content->name,p->content->type,p->content-
>vin[0],p->content->vin[1],p->content->vin[2],p->content->year,p-
>content->cost);
        }
        delay();
    }
}

void clear_str(head *h)
{
    node *p=NULL;          ///Адрес структуры, для которой необходимо
освобождить память
    int i;

    p=(node*)malloc(sizeof(node));
    if(p)
    {
        p=h->first;
        for(i=0;i<h->cnt;i++)
        {
            h->first=p->next;
            free(p->content->name);
            free(p->content);
```

```
            free(p);
            p=h->first;
        }
        free(h);
        h=NULL;
    }
}

void clear(char** data1, int schet)
{
    int i;

    for(i=0;i<schet;i++)
        free(data1[i]);

    free(data1);
}

char **split(char *data0, char sep)
{
    char **str=NULL;            ///Адрес первого элемента первой строки
    int i,                      ///Параметр цикла
```

```
        j,                      ///Параметр цикла
        k,                      ///
        m;                       ///Количество символов разделителей в
строке
    int key,                     ///Переменная, показывающая, полностью
ли выделилась память
        count;                  ///Счетчик введенных структур


    for(j=0,m=0;j<strlen(data0);j++)
        if(data0[j]==sep)
            m++;


    key=1;
    str=(char**)malloc((m+1)*sizeof(char*));
    if(str!=NULL)
    {
        for(i=0,count=0;i<=m;i++,count++)
        {
            str[i]=(char*)malloc(strlen(data0)*sizeof(char));
            if(str[i]==NULL)
            {
                key=0;
                i=m;
```

```
    }
}


if(key)
{
    k=0;
    m=0;
    for(j=0;j<strlen(data0);j++)
    {
        if(data0[j]!=sep)
            str[m][j-k]=data0[j];
        else
        {
            str[m][j-k]='\0';
            k=j+1;
            m++;
        }
    }
    str[m][j-k]='\0';
}
else
    clear(str, count);
```

```
     }
     return str;
}

head *create_head()
{
    head *h=NULL;                ///"Голова" списка

    h=(head*)malloc(sizeof(head));
    if (h)
    {
        h->cnt=0;
        h->first=NULL;
        h->last=NULL;
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }

    return h;
```

```
}

void create_elem(head* h, int k, int g, cars *ks)
{
    node *sled=NULL;           ///Адрес структуры для создания исходного
списка

    sled=(node*)malloc(sizeof(node));
    if (sled)
    {
        if (h->first==NULL)
            h->first=sled;
        else
            h->last->next=sled;;

        h->cnt++;
        sled->ID=k;
        sled->next=h->first;
        h->last=sled;

        if (g)
            sled->content=str_fill();
```

```
        else
            sled->content=ks;
    }
}

void file_fill(head *h, char *data, char sep, int schet)
{
    cars *ks;              ///Переменная для заполнения информационного поля
    char **str;        ///Адрес первого элемента первой строки разбиения исходной строки по знаку разделителю

    data[strlen(data)-1]='\0';
    str=split(data, sep);
    ks=(cars*)malloc(sizeof(cars));
    ks->name=(char*)malloc((strlen(str[0])+1)*sizeof(char));
    strcpy(ks->name, str[0]);
    ks->type=(char*)malloc((strlen(str[1])+1)*sizeof(char));
    strcpy(ks->type, str[1]);
    ks->vin=(int*)malloc(3*sizeof(int));
    ks->vin[0]=atoi(str[2]);
    ks->vin[1]=atoi(str[3]);
    ks->vin[2]=atoi(str[4]);
```

```
    ks->year=atoi(str[5]);
    ks->cost=atof(str[6]);
    create_elem(h, schet, 0, ks);
    clear(str, 7);
}

void del_node(head *h, int id)
{
    node *p0=NULL,          ///Адрес структуры для удаления эл-та списка
        *p1=NULL;           ///Адрес структуры для удаления эл-та списка

    p1=h->first;
    p0=p1;
    while(p1->ID!=id)
    {
        p0=p1;
        p1=p1->next;
    }

    if (h->cnt==1)
    {
        h->first=NULL;
```

```
    h->last=NULL;
    free(p1->content->name);
    free(p1->content);
    free(p1);
}
else if (p1==h->last)
{
    free(p1->content->name);
    free(p1->content);
    free(p1);
    p0->next=NULL;
    h->last=p0;
}
else if (p1==h->first)
{
    h->first=p1->next;
    free(p1->content->name);
    free(p1->content);
    free(p1);
}
else
{
```

```
        p0->next=p1->next;
        free(p1->content->name);
        free(p1->content);
        free(p1);
    }
    h->cnt=h->cnt-1;
}

void copy(cars *first, cars *second)
{
    strcpy(second->name, first->name);
    strcpy(second->type, first->type);
    second->vin[0]=first->vin[0];
    second->vin[1]=first->vin[1];
    second->vin[2]=first->vin[2];
    second->year=first->year;
    second->cost=first->cost;
}

void copy_str(head *h1, head *h2)
{
```

```
    cars **tmp;         ///Адрес первого элемента массива указателей на структуры
    node *p;          ///Адрес элемента списка, служащего для копирования
    int count,          ///Счетчик структур, для которых память выделена успешно
        i;              ///Параметр цикла

    tmp=(cars**)malloc(h1->cnt*(sizeof(cars*)));
    if (tmp)
    {
        p=h1->first;
        for(i=0, count=0;i<h1->cnt;i++, count++)
        {
            tmp[i]=(cars*)malloc(sizeof(cars));
            tmp[i]->name=(char*)malloc((strlen(p->content->name)+1)*sizeof(char));
            tmp[i]->type=(char*)malloc((strlen(p->content->type)+1)*sizeof(char));
            tmp[i]->vin=(int*)malloc(3*sizeof(int));
            if (tmp[i])
            {
                copy(p->content, tmp[i]);
                create_elem(h2, p->ID, 0, tmp[i]);
```

```
            }
            else
            {
                puts("Ошибка выделения памяти");
                delay();
                i=h1->cnt;
            }
            p=p->next;
        }
        if (count!=h1->cnt)
        {
            for(i=0;i<count;i++)
                free(tmp[i]);
            free(tmp);
        }
    }
    else
    {
        puts("Ошибка выделения памяти");
        delay();
    }
}
```

Пример работы программы

Контрольный пример №1 (Двусвязный список)

Исходные данные:

	A	B	C	D	E	F	G
1	Volkswagen	sedan	8	7	6	2017	0,87
2	Hyundai	crossover	2	2	8	2011	1,25
3	Volkswagen	estate	3	1	4	2005	0,52
4	Cadillac	sedan	1	2	6	2020	2,7
5	Porsche	crossover	7	7	7	2015	3,49
6	Hyundai	hatchback	6	5	0	2013	0,42
7	Hyundai	estate	3	1	1	2003	0,21

Вывод программы:

```
How do you want to print your list?
 1 - in direct order
 2 - in reverse order
1
|                NAME |      TYPE |            NUMBER|  YEAR|  COST|
+--------------------+-----------+------------------+------+------+
|          Volkswagen |     sedan |    8     7     6 | 2017 | 0.87 |
|             Hyundai | crossover |    2     2     8 | 2011 | 1.25 |
|          Volkswagen |    estate |    3     1     4 | 2005 | 0.52 |
|            Cadillac |     sedan |    1     2     6 | 2020 | 2.70 |
|             Porsche | crossover |    7     7     7 | 2015 | 3.49 |
|             Hyundai | hatchback |    6     5     0 | 2013 | 0.42 |

Process returned 0 (0x0)   execution time : 1.474 s
Press any key to continue.
```

Контрольный пример №2 (Двусвязный список)

Вывод программы:

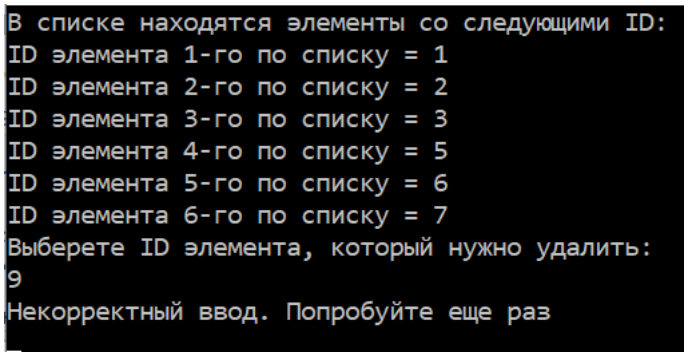
```
How do you want to print your list?
 1 - in direct order
 2 - in reverse order
2
|                NAME |      TYPE |            NUMBER|  YEAR|  COST|
+--------------------+-----------+------------------+------+------+
|             Hyundai |    estate |    3     1     1 | 2003 | 0.21 |
|             Hyundai | hatchback |    6     5     0 | 2013 | 0.42 |
|             Porsche | crossover |    7     7     7 | 2015 | 3.49 |
|            Cadillac |     sedan |    1     2     6 | 2020 | 2.70 |
|          Volkswagen |    estate |    3     1     4 | 2005 | 0.52 |
|             Hyundai | crossover |    2     2     8 | 2011 | 1.25 |

Process returned 0 (0x0)   execution time : 1.250 s
Press any key to continue.
```

Контрольный пример №3 (Кольцевой список)

Вывод программы:

Контрольный пример №4 (Кольцевой список)

Вывод программы:



Заключение

При выполнении лабораторной работы были получены практические навыки в разработке алгоритма и написании программы на языке Си, в частности, в работе со структурами и линейным двусвязным и кольцевым списком, чтением данных из CSV-файла или с клавиатуры, а также в работе с функциями в структурах.