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»
Тема: «Линейные и кольцевые списки»

Студент гр. 8307 _____ Овечко Д.А.

Преподаватель _____ Перязева Ю.В.

Санкт-Петербург
2019

Содержание

Цель

Получить практические навыки в разработке алгоритма и написании программы на языке Си с использованием структур и массивов структур, а также указателями на структуры и функции, потому что эти знания необходимы при написании практических заданий.

Задание

1) Разработать подалгоритм удаления элементов с заданным содержимым указанного информационного поля из односвязного списка. При отсутствии таких элементов в списке вывести соответствующее сообщение.

2)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.

3)Разработать подалгоритм вывода значений информационного поля элементов односвязного кольцевого списка в прямом или обратном направлении по желанию пользователя с одновременным удалением последнего выводимого на экран элемента.

Исходные данные: Структура типа GameDev, считываемая с файла.

Результаты:

1)Удаление элементов с заданным содержанием.

2)Вывод значений в направлении, выбранном пользователем с одновременным удалением последнего выводимого на экран элемента.

3) Вывод значений в направлении, выбранном пользователем с одновременным удалением последнего выводимого на экран элемента.

Постановка задачи и описание решения

Для всех 3-х лабораторных, чтобы программа работала корректно, необходимо считать информацию с файла. Для этого, после проверки на открытие заданного файла, проверяем прошло ли считывание первой строки с него удачно и если да, то переходим на новую строку, иначе завершаем считывание. (функция read)

В лабораторных мы работаем с 3 разными видами списка: линейный односвязный, линейный двусвязный, кольцевой односвязный.

Так как все лабораторные объединены в одну работу, то необходимо меню, для выбора действия (функция menu). В меню выбирается тип взаимодействия с данным списком.

Далее будет описан процесс работы лабораторной работы 10.

Если в главном меню был выбран 1-й процесс, который определяет лабораторная работа 10, запустится главное меню для работы с односвязным списком (функция main_lab10). Пользователю предлагается выбор – «по какому критерию удалять элемент» (функция menu_delete). После выбора пункта, пользователю необходимо ввести значение, которое соответствует элементу, который необходимо удалить. Далее происходит удаление (для этого используется несколько функций вида delete_*, где * - это наименование столбца, по которому и происходит поиск и удаление). После завершения удаления будет показан изменённый список (функция display).

Процесс работы лабораторной работы 11.

Если в главном меню был выбран 2-й процесс, который определяет лабораторная работа 11, запустится главное меню для работы с линейным двусвязным списком. Процесс работы схож с предыдущим. Происходит вывод списка. Но, в отличие от лаб.10 конечным выводом будет не полный список а значения поля, которое выбирает пользователь (выбор осуществляется функцией menu_choose). После выбора поля вывода, предлагается выбор порядка, в котором будет произведён вывод. В зависимости от выбора порядка функция delete произведёт удаление элемента (первого или последнего). Далее, в зависимости от выбора поля, будет запущена одна из функций (output_*, где * - название поля, с которым происходит взаимодействие). Функции учитывают порядок вывода.

Процесс работы лабораторной работы 12.

Если в главном меню был выбран 3-й процесс, который определяет лабораторная работа 12, запустится главное меню для работы с кольцевым односвязным списком (main_lab12). Процесс почти полностью идентичен работе предыдущей лаб.11, с тем лишь отличием, что программа будет прогонять значения выбранного поля по кольцу (так, как список кольцевой). За вывод полей отвечают функции derivation_*, по аналогии с предыдущими заданиями. За выбор поля для вывода отвечает функция menu_select, за удаление последнего выводимого элемента функция uninstall.

Описание переменных

Таблица 1. Описание переменных.

Имя переменной	Тип	Назначение
c	int	Для выбора действия в меню
f	FILE*	Указатель на файл
h	head*	Заголовочный элемент (заголовок списка)
current	GameDev*	Элемент списка
r	int	«Флаг»
i	int	Счетчик “for”
j	Int	Счетчик ”for”

Таблица 2. Описание функций.

Имя переменной	Тип	Назначение
main()	Int	Главное меню
MakeHead()	head	Создание головы
read(head *h)	int	Считывание с файла
display(head *h)	void	Вывод списка
add(GameDev *Game, head *h)	void	Запись строки в список
delete(head *h)	int/void	Удаление элемента
main_lab10()	int	Главное меню для линейного односвязного списка
main_lab11()	int	Главное меню для линейного двусвязного списка
main_lab12()	int	Главное меню для кольцевого односвязного списка
menu_delete(head*)	int	Меню для выбора удаляемого элемента(10)
menu_choose(head*)	int	Меню для выбора выводимого поля(11)
menu_selsect(head*)	int	Меню для выбора выводимого поля(12)
add10(GameDev *Game, head *h) по аналогии add11 и add12 для заданий 1, 2, 3 соответственно	void	Добавление элемента в список (1, 2, 3) Различия вызваны разными типами списков.
delete_*([тип переменной название поля],head *h)	int	Функции для удаления элемента в задании 1
output_*(head*)	void	Вывод поля * по заданию 2
derivation_*(head*)	void	Вывод поля * по заданию 3
delete(head*)	void	Удаление последнего выводимого элемента в задании 2, в зависимости от условий
display_ring(head *h)	void	Вывод кольцевого списка

uninstall(head*)	void	Удаление последнего элемента в задании 3 в зависимости от условий

Таблица 3. Описание структуры HEAD

Имя переменной	Тип	Назначение
size	int	Количество элементов списка
first	GameDev *	Первый элемент списка
last	GameDev *	Последний элемент списка

Таблица 4. Описание структуры RUS

Имя переменной	Тип	Назначение
id	Int	Номер элемента
name	char*	Название игры (серии)
chapter	char*	Часть серии
developer	char*	Разработчик
date[3]	int	Дата выпуска (целочисленный масив)
Publishrate	float	Средняя оценка, полученная из оценок двух популярных игровых изданий (x/100)
Gamersrate	int	Оценка игроков (x/100)
bothrate	float	Средняя оценка (игроки+издания/2) (x/100)
Publisher1	int	Оценка первого издания
Publisher2	int	Оценка второго издания
next	GameDev*	Указатель на следующий элемент
prev	GameDev*	Указатель на предыдущий элемент

CSV-файл:

Diablo;2;BlizzardEntertainment;29,06,2000;88.5;91;89.75;89;88
MaxPayne;1;3DRealmsEntertainment;23,07,2001;90.5;94;92.25;89;92
The_Elder_Scrolls;3;BethesdaSoftworks;04,10,2002;89.5;93;91.25;89;90
Warcraft;3_TheFrozenThrone;BlizzardEntertainment;22,06,2003;89.5;89;89.25;91;88
GrandTheftAuto;SanAndreas;RockstarGames;29,10,2004;95.5;90;92.75;96;95
NeedforSpeed;MostWanted;ElectronicArts;15,11,2005;83.5;86;84.75;85;82
The_Elder_Scrolls;IV_Oblivion;BethesdaSoftworks;20,03,2006;93.5;90;91.75;93;94
Call_of_Duty;4_ModernWarfare;InfinityWard;09,11,2007;93.5;88;90.75;94;93
Fallout;3;BethesdaSoftworks;31,10,2008;92;86;89;91;93
Call_of_Duty;ModernWarfare_2;Infinity Ward;10,11,2009;92;87;89.5;94;90
Mafia;2;2K_Czech;27,08,2010;81.5;85;83.25;78;85
The_Elder_Scrolls;5_Skyrim;Bethesda Softworks;11,11,2011;95;89;92;95;95
FarCry;3;UbisoftEntertainment;30,11,2012;88;88;88;88;88
GrandTheftAuto;5;RockstarGames;17,09,2013;100;87;93.5;100;100
DarkSouls;2;FromSoftware;14,03,2014;90;78;84;89;91
Witcher;3_Wild Hunt;CDProjektRED;19,05,2015;94.5;86;90.25;92;97
DarkSouls;3;FromSoftware;12,04,2016;91;82;86.5;89;93
Resident_Evil;7_Biohazard;Capcom;24,01,2017;85;77;81;87;83
FarCry;5;UbisoftEntertainment;27,03,2018;81;70;75.5;84;78
Metro;Exodus;4A_Games;15,02,2019;83;78;80.5;84;82

Контрольные примеры

1. **Пример 1.1**
   Исходные данные:
   - Содержимое CSV-файла
   - В главном меню выбираем пункт 1
   - Выбираем удаление по имени (пункт 2)
   - Вписываем FarCry

   Результат:
   - Выведен список без 13 и 19 элементов

2. **Пример 1.2**
   Исходные данные:
   - Содержимое CSV-файла
   - В главном меню выбираем пункт 1
   - Выбираем удаление по оценке геймеров (пункт 7)
   - Вписываем значение 79
   - **Результат:**
     - Список не изменится. Будет выведено сообщение об ошибке поиска.

3. **Пример 2.1**
   Исходные данные:
   - Содержимое CSV-файла
   - В главном меню выбираем пункт 2
   - Выбираем вывод по дате (пункт 5)
   - Выбираем прямой вывод – 1

   Результат:
   - Выведено поле release_date в прямом порядке без последнего элемента списка.
4. **Пример 2.2**

Исходные данные:

- Содержимое CSV-файла:
- В главном меню выбираем пункт 2
- Выбираем вывод по оценке первого издания (пункт 9)
- Выбираем обратный вывод -2

Результат:

- Выведено поле first publisher rate в обратном порядке без первого (в списке), последнего в выводе элемента.

5. **Пример 3.1**

Исходные данные:

- Содержимое CSV-файла
- В главном меню выбираем пункт 3
- Выбираем вывод по разработчику (пункт 4)
- Выбираем обратный порядок - 2

Результат:

- Выведено поле developer в обратном порядке без первого (в списке) последнего в выводе элемента.

6. **Пример 3.2**

Исходные данные:

- Содержимое CSV-файла
- В главном меню выбираем пункт 3
- Выбираем вывод по названию (пункт 2)
- Выбираем прямой порядок – 1

Результат:

- Выведено поле name в прямом порядке без последнего элемента списка.

Текст программы

Общие функции

main.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include "lab_10.h"
#include "lab_11.h"
#include "lab_12.h"
#include "struct.h"

int main() {
    printf("Select the list you
want to continue working with:\n1
- Linear simply linked list
(lab10)\n"
           "2 - The linear double
linked list (lab11)\n3 - Ring
single-linked list (lab12)\nPrint
parametr: " );
    int c;
    scanf("%d", &c);
    while ((c!=1) && (c!=2) &&
(c!=3)) {
        printf("Incorrect input!
Try again: ");
        scanf("%d", &c);
    }
    if(c == 1)
        c = main_lab10();
    else if(c == 2)
        c = main_lab11();
    else if(c == 3)
        c = main_lab12();

    puts("Enter '1' to exit
program");
    int h;
    scanf("%d", &h);
    return c;
}
```

struct.h

```
#ifndef LAB10_12_OVECHKO_STRUCT_H
#define LAB10_12_OVECHKO_STRUCT_H

typedef struct GameDev {
    int id;                         //
Id
    char *name;                     //
Name of Game
    char *chapter;                  //
Type of Chapter
    char *developer;                //
Developer of Game
    int date[3];                    //
Release date
    float Publisherrate;           //
Publisher rate of Game
    int Gamersrate;                 //
Gamers rate of Game
    float bothrate;                //
Both rate of Game(gamers and
Publishers)
    int Publisher1;                 //
First Publisher
    int Publisher2;                 //
Second Publisher
    struct GameDev *next;           //
Link to next
    struct GameDev *prev;           //
Link to prev
} GameDev;


typedef struct head {          //the
header element
    int size;                 //the
number of the list
    struct GameDev *first;
     //first element
    struct GameDev *last;     //last
element
}head;



#endif //LAB10_12_OVECHKO_STRUCT_H
```

read.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "struct.h"
#include "read.h"
#include "lab_10.h"
#include "lab_11.h"
#include "lab_12.h"

head MakeHead()  // Создание головы
{
    head h;
    h.first = NULL;
    h.size = 1;
    return h;
}

int read(head *h, int j) {    // Считывание
    FILE *f;
    int r=1;
    if ((f = fopen("rates.csv", "r")) != NULL) {
        fseek(f,0,SEEK_END);
        long pos = ftell(f);
        if(pos>0) // Файл непустой
        {
            rewind(f);
            while (!(feof(f))) {
                h = entry(h, f, j);    // Добавление по строке
            }
        } else h->size = 0;
        if (fclose(f) == EOF) {
            puts("Error closing file");
            r=0;
        }
    }
    else {
        puts("Error opening file");
        r=0;
    }
    return r+1;
}

////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////

head *entry(head *h, FILE *f, int j) {      // Запись с файла
    GameDev *current;
    current = (GameDev*)malloc(sizeof(GameDev));
    current->name = (char*)malloc(256 * sizeof(char));
    current->chapter = (char*)malloc(256 * sizeof(char));
    current->developer = (char*)malloc(256 * sizeof(char));
    if (current && current->name) {
        fscanf(f, "%[^;];%[^;];%[^;];%d,%d,%d;%f;%d;%f;%d;%d",
current->name, current->chapter,
                current->developer, &(current->date[0]), &(current-
>date[1]), &(current->date[2]),
                &(current->Publisherrate), &(current->Gamersrate),
&(current->bothrate),
```

```
                &(current->Publisher1), &(current->Publisher2));
        fgetc(f);
        if (j == 10)
            add10(current, h);
        else if (j == 11)
            add11(current, h);
        else if (j == 12)
            add12(current, h);
    }
    else{
        puts("Memory allocation error!");
        if (current)
            free(current);
        if (current->name)
            free(current->name);
    }
    return h;
}

////////////////////////////////////////////////////////////////////////
////////////////////////////////////

void display(head *h) {
    GameDev *current = h->first;
    if (current == NULL) {
        printf("List is empty\n");
        return;
    }

printf("_____________________________________________________________________
_____________"
"_____________________________________________________________________
_________"
 "_____________________________________________\n");
    printf("Id |Game\t\t|  Chapter\t\t  |  Developer\t\t    |Release
date "
           "|Publishers average rating|Gamers average rating|Both
average rating|"
           "Rate of first Publisher|Rate of second Publisher|\n");

printf("_____________________________________________________________________
_____________"

"_____________________________________________________________________
_________"
           "_____________________________________________\n");
    while (current != NULL) {
        printf("%-3d|%-20s|%-25s|%-25s|%d.%d.%d\t  |%-25.2f|%-21d|%-
19.2f|%-23d|%-23d|\n", current->id,
               current->name, current->chapter, current->developer,
current->date[0], current->date[1],
               current->date[2], current->Publisherrate, current-
>Gamersrate, current->bothrate,
               current->Publisher1, current->Publisher2);
        current = current->next;
        printf("---+--------------------+-----------------------+---
---------------------+----"
               "---------+------------------------+-----------------
```

```
---+-------------------+-----"
                "------------------+----------------------+\n");
    }
}
```

read.h

```
#ifndef LAB10_12_OVECHKO_READ_H
#define LAB10_12_OVECHKO_READ_H

#include "struct.h"


head MakeHead(); // Создание головы
int read(head *h, int); // Считывание с файла
head *entry(head *h, FILE *f, int ); // Добавление по строке
void add(GameDev *Game, head *h);
void display(head *h); // Вывод списка


#endif //LAB10_12_OVECHKO_READ_H
```

Код к лабораторной №10

lab10.h

```
#ifndef LAB10_12_OVECHKO_LAB_10_H
#define LAB10_12_OVECHKO_LAB_10_H

#include "struct.h"

int main_lab10(); // Мэйн для линейного односвязного списка
int menu_delete(head*); // Меню для выбора удаляемого элемента

void add10(GameDev *Game, head *h); // Добавление элемента в список

int delete_id(int id, head *h); // Удаление элемента
int delete_name(char* c, head *h); // Удаление элемента
int delete_chapter(char *c, head *h); // Удаление элемента
int delete_dev(char *c, head *h); // Удаление элемента
int delete_release(int year, head *h); // Удаление элемента
int delete_Publisherrate(float rate, head *h); // Удаление элемента
int delete_Gamersrate(int rate, head *h); // Удаление элемента
int delete_bothrate(float rate, head *h); // Удаление элемента
int delete_Publisher1(int pub, head *h); // Удаление элемента
int delete_Publisher2(int pub, head *h); // Удаление элемента

#endif //LAB10_12_OVECHKO_LAB_10_H
```

lab10.c

```
#include "lab_10.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "struct.h"
#include "read.h"
#define eps 0.000001

int main_lab10() { // Мэйн для линейного односвязного списка
    head h = MakeHead();
    int r, j=10;
    r = read(&h, j);
    if (r == 2) {
        if (h.size == 0) {
            puts("Error. List is empty");
        } else {
            puts("Your list");
            display(&h);
            r = menu_delete(&h);
            if (r == 1) {
                puts("Removal was successful, the final list");
                display(&h);
            }
        }
        return 0;
    }
}


void add10(GameDev *Game, head *h) { // Добавление елемента в список
```

```
    GameDev *current = NULL;
    Game->next = NULL;
    Game->id = 1;

    if (h->first == NULL) {
        h->first = Game;
    } else {
        Game->id++;
        (h->size)++;
        current = h->first;
        while (current->next != NULL) {
            Game->id++;
            current = current->next;
        }
        current->next = Game;
    }
}

int menu_delete(head *h) { // Меню для выбора удаляемого элемента
    int k, u;
    char c[100];
    float f;
    printf("1 - delete by id\n2 - delete by name\n3 - delete by
chapter\n4 - delete by developer\n5 - delete by year"
           "\n6 - delete by publisher rate\n7 - delete by gamers
rate\n8 - delete by both rate\n"
           "9 - delete by first publisher rate\n10 - delete by second
publisher rate\nPrint parametr: ");
    scanf("%d", &k);
    while (k < 1 || k > 10) {
        printf("Incorrect input! Try again: ");
        scanf("%d", &k);
    }
    if (k==1) {
        printf("Print id");
        scanf("%d", &k);
        u = delete_id(k, h);
    }
    else if (k==2)
    {
        printf("Print name");
        scanf("%s", c);
        u = delete_name(c, h);
    }
    else if (k==3)
    {
        printf("Print chapter");
        scanf("%s", c);
        u = delete_chapter(c, h);
    }
    else if (k==4)
    {
        printf("Print developer");
        scanf("%s", c);
        u = delete_dev(c, h);
    }
    else if (k==5) {
        printf("Print year");
        scanf("%d", &k);
        u = delete_release(k, h);
```

```
    }
    else if (k==6) {
        printf("Print publisher rate");
        scanf("%f", &f);
        u = delete_Publisherrate(f, h);
    }
    else if (k==7) {
        printf("Print gamers rate");
        scanf("%d", &k);
        u = delete_Gamersrate(k, h);
    }
    else if (k==8) {
        printf("Print both rate");
        scanf("%f", &f);
        u = delete_bothrate(f, h);
    }
    else if (k==9) {
        printf("Print rate first publisher");
        scanf("%d", &k);
        u = delete_Publisher1(k, h);
    }
    else if (k==10) {
        printf("Print rate second publisher");
        scanf("%d", &k);
        u = delete_Publisher2(k, h);
    }
    if (u == 0) {
        printf("Not found");
    }
    return u;
}

int delete_id(int id, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (current->id == id)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u =1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}

int delete_name(char* c, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (strcmp(current->name, c) == 0)
        {
```

```
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}


int delete_chapter(char *c, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (strcmp(current->chapter, c) == 0)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}


int delete_dev(char *c, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (strcmp(current->developer, c) == 0)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}

int delete_release(int year, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
```

```
        if (current->date[2] == year)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}
int delete_Publisherrate(float rate, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (fabs(current->Publisherrate - rate)<eps)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}
int delete_Gamersrate(int rate, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (current->Gamersrate == rate)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u =1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}


int delete_bothrate(float rate, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (fabs(current->bothrate - rate)<eps)
```

```
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}


int delete_Publisher1(int pub, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (current->Publisher1 == pub)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}

int delete_Publisher2(int pub, head *h) {  // Удаление элемента
    GameDev *current = h->first;
    GameDev *previous = current;
    int u = 0;
    while (current != NULL)
    {
        if (current->Publisher2 == pub)
        {
            previous->next = current->next;
            if (current == h->first)
                h->first = current->next;
            free(current);
            u = 1;
        }
        previous = current;
        current = current->next;
    }
    return u;
}
```

Код к лабораторной №11

lab11.h

```
#ifndef LAB10_12_OVECHKO_LAB_11_H
#define LAB10_12_OVECHKO_LAB_11_H

#include "struct.h"


int main_lab11(); // Мэйн для линейного двусвязного списка
void add11(GameDev*, head*); // Добавление елемента в список

void menu_choose(head*); // Меню для выбора выводимого поля

void delete(head*, int); // Удаление последнего элемента, в
зависимости от выбранного вывода

void output_id(head*, int); // Вывод поля с id
void output_name(head*, int); // Вывод поля с названием игры
void output_chapter(head *h, int p); // Вывод поля с названием части
игры
void output_dev(head *h, int p);  // Вывод поля с названием
разработчика игры
void output_date(head *h, int p); // Вывод поля с датой выпуска игры
void output_Publisherrate(head *h, int p); // Вывод поля с средними
оценками изданий
void output_Gamersrate(head *h, int p);  // Вывод поля с оценками
игроков
void output_bothrate(head *h, int p); // Вывод поля основной оценкой
игры
void output_Publisher1(head *h, int p); // Вывод поля оценкой первого
издания
void output_Publisher2(head *h, int p);  // Вывод поля оценкой второго
издания

#endif //LAB10_12_OVECHKO_LAB_11_H
```

lab11.c

```
#include "lab_11.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "struct.h"
#include "read.h"


int main_lab11() {  // Мэйн для линейного двусвязного списка
    head h = MakeHead();
    int r, j=11;
    r = read(&h, j);
    puts("Your list");
    display(&h);
    if (r == 2) {
        if (h.size == 0) {
            puts("Error. List is empty");
```

```
        } else {
            menu_choose(&h);
        }
    }
    return 0;
}


void add11(GameDev *Game, head *h) {   // Добавление елемента в список
    GameDev *current = NULL;
    Game->next = NULL;
    Game->id = 1;
    if (h->first == NULL) {
        h->first = Game;
    } else {
        Game->id++;
        (h->size)++;
        current = h->first;
        while (current->next != NULL) {
            Game->id++;
            current = current->next;
        }
        Game->prev = current;
        h->last = Game;
        current->next = Game;
    }
}

void menu_choose(head *h){ // Меню для выбора выводимого поля
    int k, p;
    printf("1 - output id\n2 - output name\n3 - output chapter\n4 -
output developer\n5 - output date\n"
           "6 - output publisher rate\n7 - output gamers rate\n8 -
output both rate\n"
           "9 - output first publisher rate\n10 - output second
publisher rate\nPrint parametr: ");
    scanf("%d", &k);
    while (k < 1 || k > 10) {
        printf("Incorrect input! Try again: ");
        scanf("%d", &k);
    }
    printf("\n1 - output in direct order\n2 - output in reverse
order\nPrint parametr: ");
    scanf("%d", &p);
    while (p < 1 || p > 2) {
        printf("Incorrect input! Try again: ");
        scanf("%d", &p);
    }
    delete(h, p);
    if (k == 1)
        output_id(h,p);
    else if (k == 2)
        output_name(h, p);
    else if (k == 3)
        output_chapter(h, p);
    else if (k == 4)
        output_dev(h, p);
    else if (k == 5)
        output_date(h, p);
    else if (k == 6)
```

```
        output_Publisherrate(h, p);
    else if (k == 7)
        output_Gamersrate(h, p);
    else if (k == 8)
        output_bothrate(h, p);
    else if (k == 9)
        output_Publisher1(h, p);
    else if (k == 10)
        output_Publisher2(h, p);
}

void delete(head* h, int p) { // Удаление последнего элемента, в
зависимости от выбранного вывода
    GameDev *current = h->first;
    GameDev *previous = current;
    if (p == 1) {
        while (current != NULL) {
            if (current->id == h->size) {
                previous->next = current->next;
                if (current == h->first)
                    h->first = current->next;
                free(current);
            }
            previous = current;
            current = current->next;
        }
    } else if (p == 2){
        while (current != NULL) {
            if (current->id == 1) {
                previous->next = current->next;
                if (current == h->first)
                    h->first = current->next;
                free(current);
            }
            previous = current;
            current = current->next;
        }
    }
}

void output_id(head *h, int p){ // Вывод поля с id
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output id in direct order with the last item removed:
\n");
        while (current != NULL) {
            printf("|%3d |\n", current->id);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output id in reverse order with the last item removed:
\n");
        while (previous->prev != NULL) {
            printf("|%3d |\n", previous->id);
            previous = previous->prev;
        }
    }
}
```

```
void output_name(head *h, int p){ // Вывод поля с названием игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output name in direct order with the last item
removed: \n");
        while (current != NULL) {
            printf("|%20s |\n", current->name);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output name in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%20s |\n", previous->name);
            previous = previous->prev;
        }
    }
}


void output_chapter(head *h, int p){ // Вывод поля с названием части
игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output chapter in direct order with the last item
removed: \n");
        while (current != NULL) {
            printf("|%12s |\n", current->chapter);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output chapter in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%12s |\n", previous->chapter);
            previous = previous->prev;
        }
    }
}


void output_dev(head *h, int p){ // Вывод поля с названием
разработчика игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output developer in direct order with the last item
removed: \n");
        while (current != NULL) {
            printf("|%32s |\n", current->developer);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
```

```
        printf("Output developer in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%32s |\n", previous->developer);
            previous = previous->prev;
        }
    }
}

void output_date(head *h, int p){ // Вывод поля с датой выпуска игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output date in direct order with the last item
removed: \n");
        while (current != NULL) {
            printf("|%3d.%2d.%4d |\n", current->date[0], current-
>date[1], current->date[2]);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output date in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%3d.%2d.%4d |\n", previous->date[0], previous-
>date[1], previous->date[2]);
            previous = previous->prev;
        }
    }
}

void output_Publisherrate(head *h, int p){  // Вывод поля с средними
оценками изданий
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output publisher rate in direct order with the last
item removed: \n");
        while (current != NULL) {
            printf("|%3.2f |\n", current->Publisherrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output publisher rate in reverse order with the last
item removed: \n");
        while (previous->prev != NULL) {
            printf("|%3.2f |\n", previous->Publisherrate);
            previous = previous->prev;
        }
    }
}

void output_Gamersrate(head *h, int p){  // Вывод поля с оценками
игроков
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output gamers rate in direct order with the last item
removed: \n");
```

```
        while (current != NULL) {
            printf("|%3d |\n", current->Gamersrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output gamers rate in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%3d |\n", previous->Gamersrate);
            previous = previous->prev;
        }
    }
}

void output_bothrate(head *h, int p){ // Вывод поля основной оценкой
игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output both rate in direct order with the last item
removed: \n");
        while (current != NULL) {
            printf("|%3.2f |\n", current->bothrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output both rate in reverse order with the last item
removed: \n");
        while (previous->prev != NULL) {
            printf("|%3.2f |\n", previous->bothrate);
            previous = previous->prev;
        }
    }
}

void output_Publisher1(head *h, int p){ // Вывод поля оценкой первого
издания
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output rate first publisher in direct order with the
last item removed: \n");
        while (current != NULL) {
            printf("|%3d |\n", current->Publisher1);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output rate first publisher in reverse order with the
last item removed: \n");
        while (previous->prev != NULL) {
            printf("|%3d |\n", previous->Publisher1);
            previous = previous->prev;
        }
    }
}
```

```
void output_Publisher2(head *h, int p){ // Вывод поля оценкой второго
издания
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output rate second publisher in direct order with the
last item removed: \n");
        while (current != NULL) {
            printf("|%3d |\n", current->Publisher2);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *previous = h->last;
        printf("Output rate second publisher in reverse order with the
last item removed: \n");
        while (previous->prev != NULL) {
            printf("|%3d |\n", previous->Publisher2);
            previous = previous->prev;
        }
    }
}
```

Код к лабораторной №12

lab12.h

```
#ifndef LAB10_12_OVECHKO_LAB_12_H
#define LAB10_12_OVECHKO_LAB_12_H

#include "struct.h"

int main_lab12(); // Мэйн для кольцевой односвязного списка
void add12(GameDev*, head*); // Добавление елемента в список
void display_ring(head *h);

void menu_select(head*);
void uninstall(head*, int); // Удаление последнего элемента, в
зависимости от выбранного вывода

void derivation_id(head*, int); // Вывод поля с id
void derivation_name(head*, int); // Вывод поля с названием игры
void derivation_chapter(head *h, int p); // Вывод поля с названием
части игры
void derivation_dev(head *h, int p);  // Вывод поля с названием
разработчика игры
void derivation_date(head *h, int p); // Вывод поля с датой выпуска
игры
void derivation_Publisherrate(head *h, int p); // Вывод поля с
средними оценками изданий
void derivation_Gamersrate(head *h, int p);  // Вывод поля с оценками
игроков
void derivation_bothrate(head *h, int p); // Вывод поля основной
оценкой игры
void derivation_Publisher1(head *h, int p); // Вывод поля оценкой
первого издания
void derivation_Publisher2(head *h, int p);  // Вывод поля оценкой
второго издания

#endif //LAB10_12_OVECHKO_LAB_12_H
```

lab11.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "struct.h"
#include "read.h"
#include "lab_12.h"


int main_lab12() { // Мэйн для кольцевой односвязного списка
    head h = MakeHead();
    int r, j=12;
    r = read(&h, j);
    puts("Your list");
    display_ring(&h);
    if (r == 2) {
        if (h.size == 0) {
            puts("Error. List is empty");
        } else {
            menu_select(&h);
```

```
        }
    }

    return 0;
}


void add12(GameDev* Game, head* h ) { // Добавление елемента в список
    GameDev *current = NULL;
    Game->next = NULL;
    Game->id = 1;

    if (h->first == NULL) {
        h->first = Game;
        Game->next = h->first;
    } else {
        Game->next = h->first;
        Game->id++;
        (h->size)++;
        current = h->first;
        while (current->next != h->first) {
            Game->id++;
            current = current->next;
        }
        h->last = current;
        current->next = Game;
    }
}

void display_ring(head *h) {
    GameDev *current = h->first;
    if (current == NULL) {
        printf("List is empty\n");
        return;
    }

printf("_____________________________________________________________________
_________________"

"_____________________________________________________________________
__________"
           "_____________________________________________\n");
    printf("Id |Game\t\t|  Chapter\t\t  |  Developer\t\t    |Release
date "
           "|Publishers average rating|Gamers average rating|Both
average rating|"
           "Rate of first Publisher|Rate of second Publisher|\n");

printf("_____________________________________________________________________
_____________"

"_____________________________________________________________________
__________"
           "_____________________________________________\n");
    printf("%-3d|%-20s|%-25s|%-25s|%d.%d.%d\t  |%-25.2f|%-21d|%-
19.2f|%-23d|%-23d|\n", current->id,
           current->name, current->chapter, current->developer,
current->date[0], current->date[1],
           current->date[2], current->Publisherrate, current-
>Gamersrate, current->bothrate,
```

```
            current->Publisher1, current->Publisher2);
    current = current->next;
    printf("---+--------------------+------------------------+-------
-----------------+----"
           "--------+------------------------+--------------------
+-------------------+-----"
           "-----------------+----------------------+\n");
    while (current != h->first) {
        printf("%-3d|%-20s|%-25s|%-25s|%d.%d.%d\t  |%-25.2f|%-21d|%-
19.2f|%-23d|%-23d|\n", current->id,
               current->name, current->chapter, current->developer,
current->date[0], current->date[1],
               current->date[2], current->Publisherrate, current-
>Gamersrate, current->bothrate,
               current->Publisher1, current->Publisher2);
        current = current->next;
        printf("---+--------------------+------------------------+---
---------------------+----"
               "--------+------------------------+-----------------
---+-------------------+-----"
               "-----------------+----------------------+\n");
    }
}

void menu_select(head *h){ // Меню для выбора выводимого поля
    int k, p;
    printf("1 - derivation id\n2 - derivation name\n3 - derivation
chapter\n4 - derivation developer\n5 - derivation date\n"
           "6 - derivation publisher rate\n7 - derivation gamers
rate\n8 - derivation both rate\n"
           "9 - derivation first publisher rate\n10 - derivation
second publisher rate\nPrint parametr: ");
    scanf("%d", &k);
    while (k < 1 || k > 10) {
        printf("Incorrect input! Try again: ");
        scanf("%d", &k);
    }
    printf("\n1 - derivation in direct order\n2 - derivation in
reverse order\nPrint parametr: ");
    scanf("%d", &p);
    while (p < 1 || p > 2) {
        printf("Incorrect input! Try again: ");
        scanf("%d", &p);
    }
    uninstall(h, p);
    display_ring(h);
    if (k == 1)
        derivation_id(h,p);
    else if (k == 2)
        derivation_name(h, p);
    else if (k == 3)
        derivation_chapter(h, p);
    else if (k == 4)
        derivation_dev(h, p);
    else if (k == 5)
        derivation_date(h, p);
    else if (k == 6)
        derivation_Publisherrate(h, p);
    else if (k == 7)
        derivation_Gamersrate(h, p);
```

```
    else if (k == 8)
        derivation_bothrate(h, p);
    else if (k == 9)
        derivation_Publisher1(h, p);
    else if (k == 10)
        derivation_Publisher2(h, p);
}

void uninstall(head* h, int p) { // Удаление последнего элемента, в
зависимости от выбранного вывода
    GameDev *current = h->first;
    GameDev *previous = current;
    GameDev *remember = NULL;
    int c = 0;
    if (p == 1) {
        while (current != h->first || c == 0) {
            c++;
            if (current->id == h->size) {
                previous->next = current->next;
                if (current == h->first)
                    h->first = current->next;
                free(current);
            }
            previous = current;
            current = current->next;
        }
    } else if (p == 2) {
        while (current != h->first || c == 0) {
            c++;
            if (current->id == c) {
                free(remember);
            }
        }
    }
}

void derivation_id(head *h, int p){ // Вывод поля с id
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output id in direct order with the last item removed:
\n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3d |\n", current->id);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output id in reverse order with the last item removed:
\n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3d |\n", current->id);
```

```
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}


void derivation_name(head *h, int p){ // Вывод поля с названием игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output name in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%20s |\n", current->name);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output name in reverse order with the last item
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%20s |\n", current->name);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}


void derivation_chapter(head *h, int p){ // Вывод поля с названием
части игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output chapter in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%12s |\n", current->chapter);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output chapter in reverse order with the last item
```

```
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%12s |\n", current->chapter);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}


void derivation_dev(head *h, int p){ // Вывод поля с названием
разработчика игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output developer in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%32s |\n", current->developer);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output developer in reverse order with the last item
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%32s |\n", current->developer);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}

void derivation_date(head *h, int p){ // Вывод поля с датой выпуска
игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output date in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3d.%2d.%4d |\n", current->date[0], current-
>date[1], current->date[2]);
```

```
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output date in reverse order with the last item
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3d.%2d.%4d |\n", current->date[0], current-
>date[1], current->date[2]);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}

void derivation_Publisherrate(head *h, int p){  // Вывод поля с
средними оценками изданий
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output publisher rate in direct order with the last
item removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3.2f |\n", current->Publisherrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output publisher rate in reverse order with the last
item removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3.2f |\n", current->Publisherrate);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}

void derivation_Gamersrate(head *h, int p){  // Вывод поля с оценками
игроков
    if (p == 1) {
```

```
        GameDev *current = h->first;
        printf("Output gamers rate in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3d |\n", current->Gamersrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output gamers rate in reverse order with the last item
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3d |\n", current->Gamersrate);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}

void derivation_bothrate(head *h, int p){ // Вывод поля основной
оценкой игры
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output both rate in direct order with the last item
removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3.2f |\n", current->bothrate);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output both rate in reverse order with the last item
removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3.2f |\n", current->bothrate);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
```

```
    }
}

void derivation_Publisher1(head *h, int p){ // Вывод поля оценкой
первого издания
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output rate first publisher in direct order with the
last item removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3d |\n", current->Publisher1);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output rate first publisher in reverse order with the
last item removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
                printf("|%3d |\n", current->Publisher1);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}

void derivation_Publisher2(head *h, int p){ // Вывод поля оценкой
второго издания
    if (p == 1) {
        GameDev *current = h->first;
        printf("Output rate second publisher in direct order with the
last item removed: \n");
        int c = 0;
        while (current != h->first || c == 0) {
            c++;
            printf("|%3d |\n", current->Publisher2);
            current = current->next;
        }
    }
    else if (p == 2) {
        GameDev *current = h->first;
        GameDev *previous = h->last;
        int c = 0, k = h->size;
        printf("Output rate second publisher in reverse order with the
last item removed: \n");
        while (current != previous || c == 0) {
            c++;
            if (k == 1)
                return;
            if (current->id == k) {
```

```
                printf("|%3d |\n", current->Publisher2);
                k--;
            }
            current = current->next;
            previous = previous->next;
        }
    }
}
```

Пример работы программы

1. Исходные данные (Пример 1.1)

- Содержимое CSV-файла
- В главном меню выбираем пункт 1
- Выбираем удаление по имени (пункт 2)
- Вписываем FarCry

Вывод программы:

```
Select the list you want to continue working with:
1 - Linear simply linked list (lab10)
2 - The linear double linked list (lab11)
3 - Ring single-linked list (lab12)
Print parametr: 1
Your list
```

Id	Game	Chapter	Developer	Release date	Publishers average rating	Gamers average rating	Both average rating	Rate of first Publisher	Rate of second Publisher
1	Diablo	2	BlizzardEntertainment	29.6.2000	88.50	91	89.75	89	88
2	MaxPayne	1	3DRealmsEntertainment	23.7.2001	90.50	94	92.25	89	92
3	The_Elder_Scrolls	3	BethesdaSoftworks	4.10.2002	89.50	93	91.25	89	90
4	Warcraft	3_TheFrozenThrone	BlizzardEntertainment	22.6.2003	89.50	89	89.25	91	88
5	GrandTheftAuto	SanAndreas	RockstarGames	29.10.2004	95.50	90	92.75	96	95
6	NeedforSpeed	MostWanted	ElectronicArts	15.11.2005	83.50	86	84.75	85	82
7	The_Elder_Scrolls	IV_Oblivion	BethesdaSoftworks	20.3.2006	93.50	90	91.75	93	94
8	Call_of_Duty	4_ModernWarfare	InfinityWard	9.11.2007	93.50	88	90.75	94	93
9	Fallout	3	BethesdaSoftworks	31.10.2008	92.00	86	89.00	91	93
10	Call_of_Duty	ModernWarfare_2	Infinity Ward	10.11.2009	92.00	87	89.50	94	90
11	Mafia	2	2K_Czech	27.8.2010	81.50	85	83.25	78	85
12	The_Elder_Scrolls	5_Skyrim	Bethesda Softworks	11.11.2011	95.00	89	92.00	95	95
13	FarCry	3	UbisoftEntertainment	30.11.2012	88.00	88	88.00	88	88
14	GrandTheftAuto	5	RockstarGames	17.9.2013	100.00	87	93.50	100	100
15	DarkSouls	2	FromSoftware	14.3.2014	90.00	78	84.00	89	91
16	Witcher	3_Wild Hunt	CDProjektRED	19.5.2015	94.50	86	90.25	92	97
17	DarkSouls	3	FromSoftware	12.4.2016	91.00	82	86.50	89	93
18	Resident_Evil	7_Biohazard	Capcom	24.1.2017	85.00	77	81.00	87	83
19	FarCry	5	UbisoftEntertainment	27.3.2018	81.00	70	75.50	84	78
20	Metro	Exodus	4A_Games	15.2.2019	83.00	78	80.50	84	82

```
1 - delete by id
2 - delete by name
3 - delete by chapter
4 - delete by developer
5 - delete by year
6 - delete by publisher rate
7 - delete by gamers rate
8 - delete by both rate
9 - delete by first publisher rate
10 - delete by second publisher rate
Print parametr: 2
Print nameFarCry
Removal was successful, the final list
```

Id	Game	Chapter	Developer	Release date	Publishers average rating	Gamers average rating	Both average rating	Rate of first Publisher	Rate of second Publisher
1	Diablo	2	BlizzardEntertainment	29.6.2000	88.50	91	89.75	89	88
2	MaxPayne	1	3DRealmsEntertainment	23.7.2001	90.50	94	92.25	89	92
3	The_Elder_Scrolls	3	BethesdaSoftworks	4.10.2002	89.50	93	91.25	89	90
4	Warcraft	3_TheFrozenThrone	BlizzardEntertainment	22.6.2003	89.50	89	89.25	91	88
5	GrandTheftAuto	SanAndreas	RockstarGames	29.10.2004	95.50	90	92.75	96	95
6	NeedforSpeed	MostWanted	ElectronicArts	15.11.2005	83.50	86	84.75	85	82
7	The_Elder_Scrolls	IV_Oblivion	BethesdaSoftworks	20.3.2006	93.50	90	91.75	93	94
8	Call_of_Duty	4_ModernWarfare	InfinityWard	9.11.2007	93.50	88	90.75	94	93
9	Fallout	3	BethesdaSoftworks	31.10.2008	92.00	86	89.00	91	93
10	Call_of_Duty	ModernWarfare_2	Infinity Ward	10.11.2009	92.00	87	89.50	94	90
11	Mafia	2	2K_Czech	27.8.2010	81.50	85	83.25	78	85
12	The_Elder_Scrolls	5_Skyrim	Bethesda Softworks	11.11.2011	95.00	89	92.00	95	95
14	GrandTheftAuto	5	RockstarGames	17.9.2013	100.00	87	93.50	100	100
15	DarkSouls	2	FromSoftware	14.3.2014	90.00	78	84.00	89	91
16	Witcher	3_Wild Hunt	CDProjektRED	19.5.2015	94.50	86	90.25	92	97
17	DarkSouls	3	FromSoftware	12.4.2016	91.00	82	86.50	89	93
18	Resident_Evil	7_Biohazard	Capcom	24.1.2017	85.00	77	81.00	87	83
20	Metro	Exodus	4A_Games	15.2.2019	83.00	78	80.50	84	82

```
Enter '1' to exit program
```

2. **Исходные данные (Пример 2.1)**

- Содержимое CSV-файла
- В главном меню выбираем пункт 2
- Выбираем вывод по дате (пункт 5)
- Выбираем прямой вывод – 1

Вывод программы:

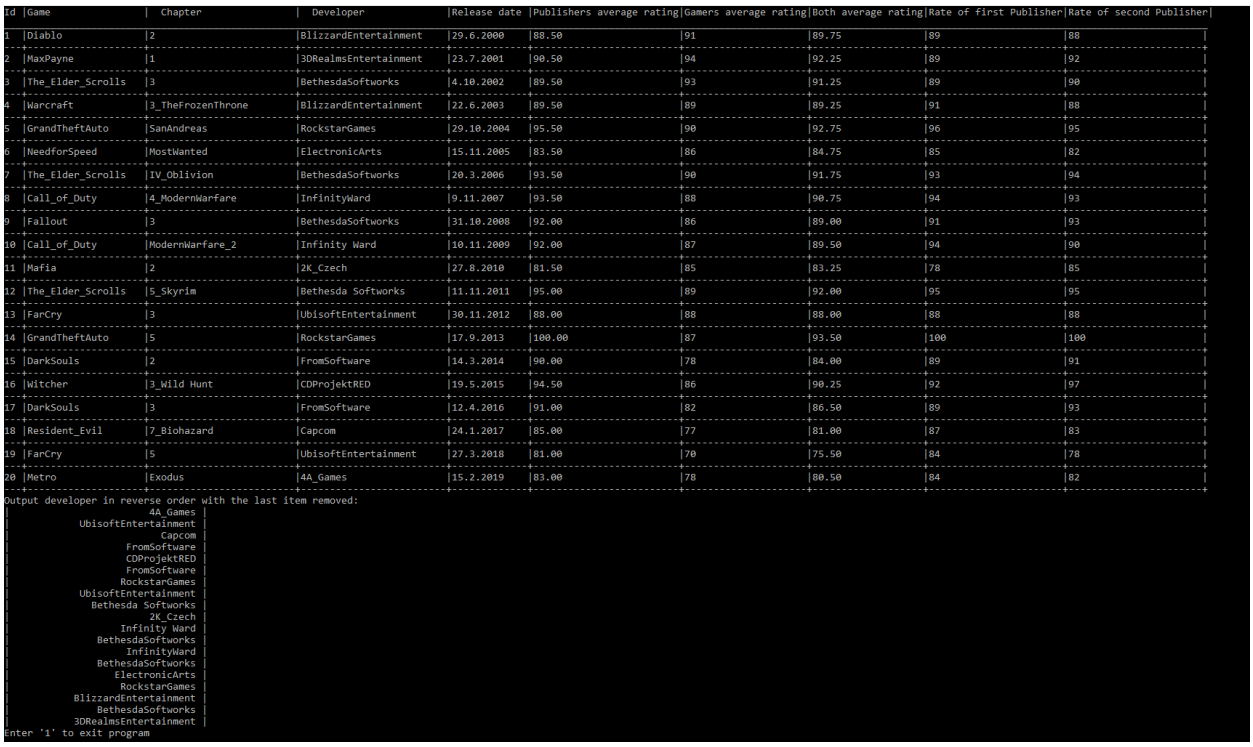
```
8 - output both rate
9 - output first publisher rate
10 - output second publisher rate
Print parametr: 5

1 - output in direct order
2 - output in reverse order
Print parametr: 1
Output date in direct order with the last item removed:
| 29. 6.2000 |
| 23. 7.2001 |
|  4.10.2002 |
| 22. 6.2003 |
| 29.10.2004 |
| 15.11.2005 |
| 20. 3.2006 |
|  9.11.2007 |
| 31.10.2008 |
| 10.11.2009 |
| 27. 8.2010 |
| 11.11.2011 |
| 30.11.2012 |
| 17. 9.2013 |
| 14. 3.2014 |
| 19. 5.2015 |
| 12. 4.2016 |
| 24. 1.2017 |
| 27. 3.2018 |
Enter '1' to exit program
```

3. Исходные данные (Пример 3.1)

- Содержимое CSV-файла
- В главном меню выбираем пункт 3
- Выбираем вывод по разработчику (пункт 4)
- Выбираем обратный порядок - 2

Вывод программы:



Id	Game	Chapter	Developer	Release date	Publishers average rating	Gamers average rating	Both average rating	Rate of first Publisher	Rate of second Publisher
1	Diablo	2	BlizzardEntertainment	29.6.2000	88.50	91	89.75	89	88
2	MaxPayne	1	3DRealmsEntertainment	23.7.2001	90.50	94	92.25	89	92
3	The_Elder_Scrolls	3	BethesdaSoftworks	4.10.2002	89.50	93	91.25	89	90
4	Warcraft	3_TheFrozenThrone	BlizzardEntertainment	22.6.2003	89.50	89	89.25	91	88
5	GrandTheftAuto	SanAndreas	RockstarGames	29.10.2004	95.50	90	92.75	96	95
6	NeedforSpeed	MostWanted	ElectronicArts	15.11.2005	83.50	86	84.75	85	82
7	The_Elder_Scrolls	IV_Oblivion	BethesdaSoftworks	20.3.2006	93.50	90	91.75	93	94
8	Call_of_Duty	4_ModernWarfare	InfinityWard	9.11.2007	93.50	88	90.75	94	93
9	Fallout	3	BethesdaSoftworks	31.10.2008	92.00	86	89.00	91	93
10	Call_of_Duty	ModernWarfare_2	Infinity Ward	10.11.2009	92.00	87	89.50	94	90
11	Mafia	2	2K_Czech	27.8.2010	81.50	85	83.25	78	85
12	The_Elder_Scrolls	5_Skyrim	Bethesda Softworks	11.11.2011	95.00	89	92.00	95	95
13	FarCry	3	UbisoftEntertainment	30.11.2012	88.00	88	88.00	88	88
14	GrandTheftAuto	5	RockstarGames	17.9.2013	100.00	87	93.50	100	100
15	DarkSouls	2	FromSoftware	14.3.2014	90.00	78	84.00	89	91
16	Witcher	3_Wild Hunt	CDProjektRED	19.5.2015	94.50	86	90.25	92	97
17	DarkSouls	3	FromSoftware	12.4.2016	91.00	82	86.50	89	93
18	Resident_Evil	7_Biohazard	Capcom	24.1.2017	85.00	77	81.00	87	83
19	FarCry	5	UbisoftEntertainment	27.3.2018	81.00	70	75.50	84	78
20	Metro	Exodus	4A_Games	15.2.2019	83.00	78	80.50	84	82

```
Output developer in reverse order with the last item removed:
|                    4A_Games |
|           UbisoftEntertainment |
|                      Capcom |
|              FromSoftware |
|              CDProjektRED |
|              FromSoftware |
|             RockstarGames |
|           UbisoftEntertainment |
|             Bethesda Softworks |
|                   2K_Czech |
|               Infinity Ward |
|           BethesdaSoftworks |
|                InfinityWard |
|           BethesdaSoftworks |
|              ElectronicArts |
|              RockstarGames |
|         BlizzardEntertainment |
|           BethesdaSoftworks |
|         3DRealmsEntertainment |
Enter '1' to exit program
```

Вывод

При выполнении лабораторной работы были получены практические навыки в работе со списками разных типов.