**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 8

**На тему:** *“*Cтруктури та об’єднання*”*

**З дисципліни:** *“Основи програмування”*

**Лектор:**

ст.викл. каф. ПЗ

Муха Т.О.

**Виконав:**

ст. гр. ПЗ-15

Марущак А.С.

**Прийняла:**

асист. каф. ПЗ

Заводовська Н.О.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2021 р.

∑= \_\_\_\_ .

Львів – 2021

**Тема роботи:** структури та об’єднання.

**Мета роботи:** навчитися створювати нові типи даних у вигляді структур та об’єднань, а також розробляти алгоритми їх обробки засобами мови С.

**Теоретичні відомості**

Структури – це особливий комбінований тип даних, який об’єднує у спільне ціле набір логічно пов’язаних між собою різнотипних компонентів. Складові частини структури називають полями або елементами. Кожне поле структури має свій тип і своє ім’я:

struct ім'я\_структури {

тип1 ім’я поля 1;

тип2 ім’я поля 2;

. . .

типN ім’я поля N;

};

Доступ до полів структури можна здійснити за допомогою оператора “.”.

Ім’я структурної змінної. Ім’я поля

В оголошеннях структурні змінні можна ініціалізувати, тобто присвоювати їх елементам початкові значення. Список значень, якими ініціалізується структура, вказується у фігурних дужках. Приклад ініціалізації структурної змінної:

struct student {

char surname[50]

char name[50];

int year;

} student1, student2 = {"Ivan", "Ivanenko",1};

struct student student1 = {"Petro", "Petrenko",42}

При цьому значення, якими ініціалізується структура записують в тій самій послідовності, в якій вказані поля у шаблоні структури та типи повинні бути сумісними з типами відповідних полів. Також можна виконувати часткову ініціалізацію.

Оскільки структурні змінні не є вказівниками, то до них можна застосовувати операцію визначення адреси &. Результатом операції буде адреса першого байта ділянки, яку займає в оперативній пам’яті структура-операнд. Значення адреси структури можна присвоїти вказівнику, базовий тип якої збігається з типом структурної змінної. Якщо оголошено структуру

struct point

{

int x;

int y;

int z;

} point1 = {-1,16,5};

то вказівник на неї виглядатиме так: struct point \*pnt; 5 Щоб присвоїти вказівнику pnt адресу екземпляра point1 структури point, слід записати: pnt = &point1; Доступ до полів структури через вказівник здійснюється за допомогою оператора ->.

Вказівник на структуру->Ім’я поля.

Структура може бути елементом структури іншого типу. Для вкладених структур діють ті ж правила оголошення, як і для зовнішніх: можна окремо оголошувати шаблон структури або задавати його всередині зовнішньої структури разом з оголошенням відповідних змінних.

Об’єднання, як і структура, дозволяє зберігати пов’язані дані всередині однієї змінної. Однак, розмір екземпляра структури рівний сумі розмірів усіх її полів, а розмір об’єднання рівний розміру найбільшого елемента, що входить у це об’єднання. Об’єднання застосовуються для більш економного використання пам’яті. 6 Якщо в екземплярі структури зберігаються усі її поля, то об’єднання зберігає лише один з елементів у кожен момент часу. Присвоєння значення іншому елементу об’єднання означає, що попереднє значення буде втрачене. Об’єднання створюється наступним чином:

union ім’я\_об’єднання

{

Тип1 ім’я\_елемента1;

Тип2 ім’я\_елемента2;

…

типN ім’я\_елементаN;

} імена\_екземплярів\_об’єднання;

Для змінних з типом об’єднання використовують ті ж операції, що й до структурних змінних: присвоєння, визначення адреси змінної, виділення окремих елементів (полів). Для звертання до елементів об’єднання використовують синтаксичні конструкції на основі операцій “.”, “->”, як і у випадку звертання до елементів структур:

Ім’я змінної об’єднання. Ім’я поля

Вказівник на об’єднання-> Ім’я поля

(\*Вказівник на об’єднання). Ім’я поля

Більшість функцій файлового обміну даними повертають значення, яке дає змогу перевірити, чи дана операція пройшла успішно, чи відбувся збій. Встановити конкретну причину збою можна через функції аналізу помилок файлового введення/виведення. Зокрема, функція

int feof(FILE\*fp)

перевіряє, чи досягнуто кінець файлу, пов’язаного з потоком fp. Повертає 0, якщо не встановлено ознаку кінця файлу, інакше – ненульове значення. Всі операції читання з файлу після досягнення його кінця вважаються помилковими.

Функція

int ferorr(FILE\*fp)

перевіряє, чи встановлено ознаку помилки в попередніх операціях звертання до потоків даних. Повертає нуль, якщо помилку незафіксовано, та ненульове значення, якщо виявлено помилку.

Причину збою в операціях потокового введення/виведення можна розшифрувати, використовуючи функцію

void perorr(char\*errtext)

Функція виводить у стандартний потік помилок stderr текст повідомлення, записаний в errtext , а за ним після двокрапки стандартне системне пояснення причини виникнення помилки. Після завершення роботи з файлом його потрібно закрити. Операційна система закриває файли після завершення роботи програми, однак хорошим тоном є виклик функції fclose у програмі:

int fclose(FILE\*fp)

Функція має один параметр – вказівник на потік, який необхідно закрити. У разі успішного виконання вона повертає значення 0 .

**ЛАБОРАТОРНЕ ЗАВДАННЯ**

1. Ознайомитися з теоретичним матеріалом викладеним вище в даній інструкції і виконати приклади програм.

2. Одержати індивідуальне завдання.

3. Скласти програму на мові С у відповідності з розробленим алгоритмом.

4. Виконати обчислення по програмі.

5. Підготувати та здати звіт про виконання лабораторної роботи.

**Індивідуальне завдання**

З текстового файлу зчитати послідовність записів, які містять дані про книгу: <Автор>, <Назва книги>, <Рік видання>, <Кількість сторінок>, <Вартість> .Роздрукувати введені дані у вигляді таблиці, а також подати інформацію згідно варіанту.

26. Відсортувавати за прізвищем в алфавітному порядку дані про книги, вартість яких менша середньої в бібліотеці.

Код програми:

(lab08\_lib.h):

#ifndef LAB08

#define LAB08

#define MAX\_NAME\_LEN 20

#define MAX\_SURNAME\_LEN 20

#define MAX\_FULLNAME\_LEN ((MAX\_NAME\_LEN) + (MAX\_SURNAME\_LEN))

#define MAX\_BOOK\_COUNT 100

#define MAX\_LINE\_SIZE 256

#define DEFAULT\_FILE\_PATH "books.txt"

#define MAX\_ERROR\_LENGTH 50

struct Book

{

    char title[50];

    struct Author

    {

        char name[MAX\_NAME\_LEN];

        char surname[MAX\_SURNAME\_LEN];

    }   author;

    double price;

    int pubYear;

    int pageCount;

};

#pragma region Macros

#define REPEAT\_CHAR(ch, i) \

    for(int z = 0; z < (i); z++){\

        printf("%c", ch);\

    }\

#define MAKE\_LINE(plainCh, nodeCh, autS, titS, yearS, pageS, priceS)\

    printf("%c", nodeCh);\

    REPEAT\_CHAR(plainCh, (autS));\

    printf("%c", nodeCh);\

    REPEAT\_CHAR(plainCh, (titS));\

    printf("%c", nodeCh);\

    REPEAT\_CHAR(plainCh, (yearS));\

    printf("%c", nodeCh);\

    REPEAT\_CHAR(plainCh, (pageS));\

    printf("%c", nodeCh);\

    REPEAT\_CHAR(plainCh, (priceS));\

    printf("%c\n", nodeCh);

#define MAX(a, b) a > b ? a : b

#pragma endregion

#pragma region Functions

void arrcpy(struct Book dest[], struct Book source[], int size);

int readBooksFromTextFile(char fileName[], struct Book output[], int\* output\_size);

int readBooksFromBinFile(char fileName[], struct Book output[], int\* output\_size);

void writeBooksToTextFile(char fileName[], struct Book input[], int input\_size);

void writeBooksToBinFile(char fileName[], struct Book input[], int input\_size);

struct Book strToBook(char str[]);

void formTable(struct Book books[], int size);

void performActions(struct Book input[], int input\_size, struct Book output[], int \* output\_size, int printMiddleSteps, ...);

#pragma endregion

#endif

(actions.h):

#ifndef ACTIONS

#define ACTIONS

void getBooksBelowAverage(struct Book input[], int input\_size, struct Book output[], int \* output\_size);

void sortBooksBySurname(struct Book input[], int input\_size, struct Book output[], int \* output\_size);

#endif

(lab08\_lib.c):

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include <stdarg.h>

#include "lab08\_lib.h"

#pragma region Utils

void arrcpy(struct Book dest[], struct Book source[], int size){

    for(int i = 0; i < size; i++){

        dest[i] = source[i];

    }

}

int numLen(int num){

    int result = 0;

    while(num > 0){

        num /= 10; result++;

    }

    return result;

}

#pragma endregion

#pragma region TablePrint

int getAuthorFullNameLen(struct Book book){

    return strlen(book.author.name) + strlen(book.author.surname) + 1;

}

int longestAuthorFieldSize(struct Book books[], int size){

    int result = getAuthorFullNameLen(books[0]);

    for(int i = 1; i < size; i++){

        int curr = getAuthorFullNameLen(books[i]);

        if(curr > result) result = curr;

    }

    return result;

}

int longestBookTitleSize(struct Book books[], int size){

    int result = strlen(books[0].title);

    for(int i = 1; i < size; i++){

        int curr = strlen(books[i].title);

        if(curr > result) result = curr;

    }

    return result;

}

void longestNumberFieldsSize(struct Book books[], int size, int \* yearSize, int \* pageSize, int \* priceSize){

    int

        maxYearLen = 0,

        maxPageLen = 0,

        maxPriceLen = 0,

        yearLen = 0,

        pageLen = 0,

        priceLen = 0;

    for(int i = 0; i < size; i++){

        yearLen = numLen(books[i].pubYear);

        pageLen = numLen(books[i].pageCount);

        priceLen = numLen((int)books[i].price) + 3;

        if(yearLen > maxYearLen) maxYearLen = yearLen;

        if(pageLen > maxPageLen) maxPageLen = pageLen;

        if(priceLen > maxPriceLen) maxPriceLen = priceLen;

    }

    \*yearSize = maxYearLen;

    \*pageSize = maxPageLen;

    \*priceSize = maxPriceLen;

}

void formTable(struct Book books[], int size){

    int authorColumnWidth = longestAuthorFieldSize(books, size);

    int titleColumnWidth = longestBookTitleSize(books, size);

    int yearColumnWidth = 0, pageColumnWidth = 0, priceColumnWidth = 0;

    longestNumberFieldsSize(books, size, &yearColumnWidth, &pageColumnWidth, &priceColumnWidth);

    authorColumnWidth = MAX(authorColumnWidth, strlen("Author"));

    titleColumnWidth = MAX(titleColumnWidth, strlen("Title"));

    yearColumnWidth = MAX(yearColumnWidth, strlen("Pub. year"));

    pageColumnWidth = MAX(pageColumnWidth, strlen("Pages"));

    priceColumnWidth = MAX(priceColumnWidth, strlen("Price"));

    printf("\n");

    MAKE\_LINE('=','+', authorColumnWidth + 2, titleColumnWidth + 2, yearColumnWidth + 2, pageColumnWidth + 2, priceColumnWidth + 2);

    printf("| %-\*s | %-\*s | %-\*s | %-\*s | %-\*s |\n",

        authorColumnWidth, "Author",

        titleColumnWidth,"Title",

        yearColumnWidth, "Pub. year",

        pageColumnWidth, "Pages",

        priceColumnWidth, "Price");

    MAKE\_LINE('=','+', authorColumnWidth + 2, titleColumnWidth + 2, yearColumnWidth + 2, pageColumnWidth + 2, priceColumnWidth + 2);

    char fullName[MAX\_FULLNAME\_LEN];

    for(int i = 0; i < size; i++){

        sprintf(fullName, "%s %s", books[i].author.name, books[i].author.surname);

        printf("| %-\*s | %-\*s | %-\*d | %-\*d | %-\*.2lf |\n",

            authorColumnWidth, fullName,

            titleColumnWidth, books[i].title,

            yearColumnWidth, books[i].pubYear,

            pageColumnWidth, books[i].pageCount,

            priceColumnWidth, books[i].price

            );

    }

    MAKE\_LINE('=','+', authorColumnWidth + 2, titleColumnWidth + 2, yearColumnWidth + 2, pageColumnWidth + 2, priceColumnWidth + 2);

    printf("\n");

}

#pragma endregion

#pragma region FileSystemWork

struct Book strToBook(char str[]){

    struct Book result;

    char \*pPart = strtok(str, ",");

    sscanf(pPart, "%s %s", &result.author.name, &result.author.surname);

    pPart = strtok(NULL, ",");

    strcpy(result.title, pPart);

    pPart = strtok(NULL, ",");

    result.pubYear = atoi(pPart);

    pPart = strtok(NULL, ",");

    result.pageCount = atoi(pPart);

    pPart = strtok(NULL, ",");

    result.price = strtod(pPart, NULL);

    return result;

}

int readBooksFromTextFile(char fileName[], struct Book output[], int\* output\_size){

    char buffer[MAX\_LINE\_SIZE];

    FILE \*fin = fopen(fileName, "r");

    if(fin == NULL) return -1;

    while(fgets(buffer, MAX\_LINE\_SIZE, fin) != NULL){

        if(strlen(buffer) <= 2) continue;

        output[(\*output\_size)++] = strToBook(buffer);

    }

    fclose(fin);

    return 0;

}

int readBooksFromBinFile(char fileName[], struct Book output[], int\* output\_size){

    FILE \*fin = fopen(fileName, "rb");

    if(fin == NULL) return -1;

    struct Book temp;

    while(!feof(fin)){

        fread(&temp, sizeof(struct Book), 1, fin);

        if(!feof(fin)){

            output[\*output\_size] = temp;

            (\*output\_size)++;

        }

    }

    fclose(fin);

    return 0;

}

void writeBooksToBinFile(char fileName[], struct Book input[], int input\_size){

    FILE \*fout = fopen(fileName, "wb");

    fwrite(input, sizeof(struct Book), input\_size, fout);

    fclose(fout);

}

void writeBooksToTextFile(char fileName[], struct Book input[], int input\_size){

    FILE \*fout = fopen(fileName, "w");

    for(int i = 0; i < input\_size; i++){

        fprintf(fout, "%s %s,%s,%d,%d,%.2lf\n", input[i].author.name, input[i].author.surname, input[i].title, input[i].pubYear, input[i].pageCount, input[i].price);

    }

    fclose(fout);

}

#pragma endregion

void performActions(struct Book input[], int input\_size, struct Book output[], int \* output\_size, int printMiddleSteps, ...){

    va\_list params;

    va\_start(params, printMiddleSteps);

    arrcpy(output, input, input\_size);

    \*output\_size = input\_size;

    struct Book tmp[MAX\_BOOK\_COUNT];

    int tmp\_size;

    void (\*action)(struct Book[], int , struct Book[], int \*);

    int first\_action = 1;

    while(action = va\_arg(params, void (\*)(struct Book[], int , struct Book[], int \*))){

        if(printMiddleSteps && !first\_action){

            formTable(output, \*output\_size);

        }

        action(output, \*output\_size, tmp, &tmp\_size);

        arrcpy(output, tmp, tmp\_size);

        \*output\_size = tmp\_size;

        tmp\_size = 0;

        first\_action = 0;

    }

    va\_end(params);

}

(actions.c):

#include <stdio.h>

#include <string.h>

#include "lab08\_lib.h"

#include "actions.h"

double averagePrice(struct Book books[], int size){

    double result = 0;

    for(int i = 0 ; i < size; i++){

        result += books[i].price;

    }

    result /= size;

    return result;

}

void getBooksBelowAverage(struct Book input[], int input\_size, struct Book output[], int \* output\_size){

    int res\_size = 0;

    double average = averagePrice(input, input\_size);

    for(int i = 0; i < input\_size; i++){

        if(input[i].price < average){

            output[res\_size] = input[i];

            res\_size++;

        }

    }

    \*output\_size = res\_size;

    printf("Average price: %.2lf; There %s %d %s, which price less than average left.\n", average, \*output\_size > 1 ? "are" : "is", \*output\_size, \*output\_size > 1 ? "elements" : "element");

}

void swap(struct Book \*a, struct Book \*b){

    struct Book tmp = \*a;

    \*a = \*b;

    \*b = tmp;

}

void sortBooksBySurname(struct Book input[], int input\_size, struct Book output[], int \* output\_size){

    arrcpy(output, input, input\_size);

    for(int i = 0; i < input\_size; i++){

        for(int j = 1; j < input\_size - i; j++){

            if(strcmp(output[j].author.surname, output[j-1].author.surname) < 0) swap(&output[j], &output[j-1]);

        }

    }

    \*output\_size = input\_size;

    printf("Sorting books by authors' surnames.\n");

}

(lab08.c):

#include <stdio.h>

#include <string.h>

#include "lab08\_lib.h"

#include "actions.h"

int main(int argc, char \*argv[]){

    char \*filePath = DEFAULT\_FILE\_PATH, \*outputFilePath;

    int printMiddleSteps = 0, textInput = 1, output = 0, textOutput = 1;

    for(int i = 1; i < argc; i++){

        if(strcmp(argv[i], "-m") == 0) printMiddleSteps = 1;

        else if((strcmp(argv[i], "-f") == 0  || strcmp(argv[i], "-ft") == 0) && i + 1 < argc){

            filePath = argv[i + 1];

            textInput = 1;

        }

        else if(strcmp(argv[i], "-fb") == 0 && i + 1 < argc){

            filePath = argv[i + 1];

            textInput = 0;

        }

        else if((strcmp(argv[i], "-o") == 0  || strcmp(argv[i], "-ot") == 0) && i + 1 < argc){

            outputFilePath = argv[i + 1];

            output = 1;

            textOutput = 1;

        }

        else if(strcmp(argv[i], "-ob") == 0 && i + 1 < argc){

            outputFilePath = argv[i + 1];

            output = 1;

            textOutput = 0;

        }

    }

    struct Book books[MAX\_BOOK\_COUNT];

    int book\_count;

    char errorMessage[MAX\_ERROR\_LENGTH];

    if(textInput && readBooksFromTextFile(filePath, books, &book\_count) == -1){

        sprintf(errorMessage, "\nError accessing the text file \"%s\"", filePath);

        perror(errorMessage);

        return -1;

    }

    else if(!textInput && readBooksFromBinFile(filePath, books, &book\_count) == -1){

        sprintf(errorMessage, "\nError accessing the text file \"%s\"", filePath);

        perror(errorMessage);

        return -1;

    }

    printf("\nInput:");

    formTable(books, book\_count);

    struct Book result[MAX\_BOOK\_COUNT];

    int result\_size = 0;

    performActions(books, book\_count, result, &result\_size, printMiddleSteps, getBooksBelowAverage, sortBooksBySurname, NULL);

    printf("\nOutput:");

    formTable(result, result\_size);

    if(output){

        if(textOutput) writeBooksToTextFile(outputFilePath, result, result\_size);

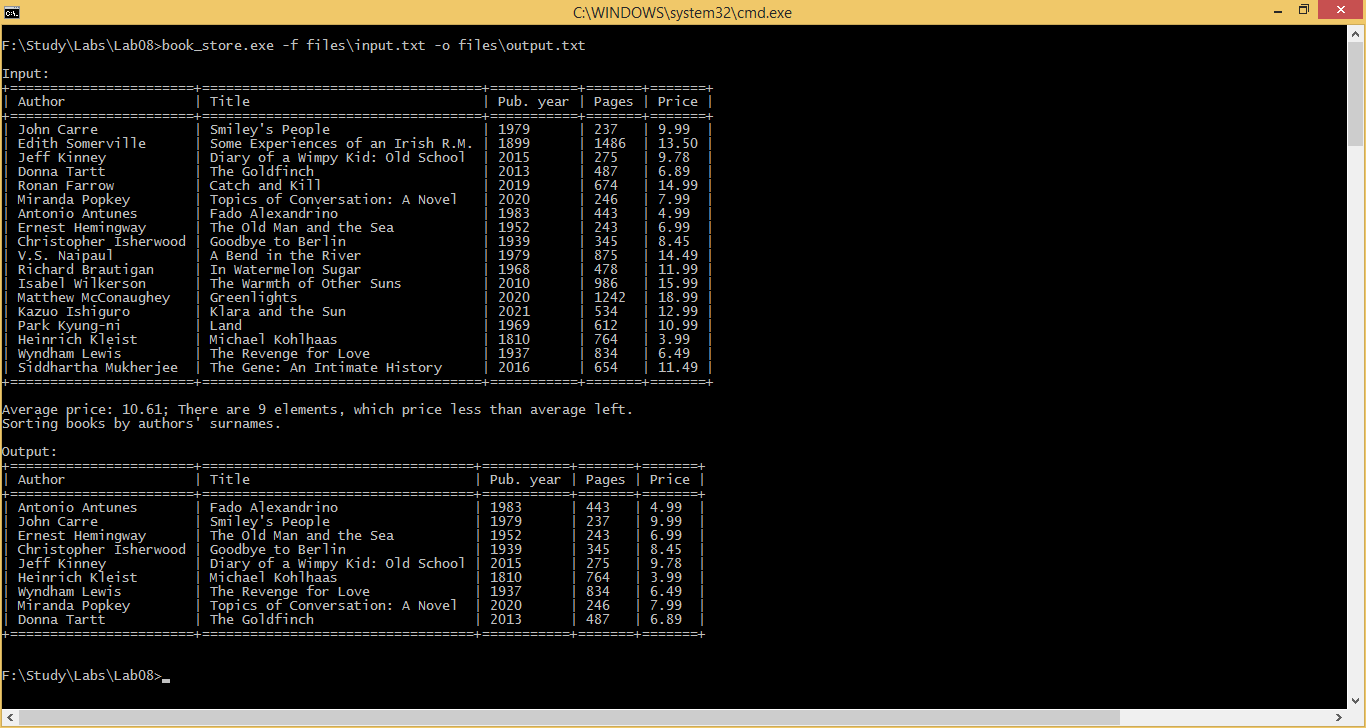
        else writeBooksToBinFile(outputFilePath, result, result\_size);

    }

}

**Протокол роботи:**

Програма приймає опції виконання з командного рядка(в цьому випадку вхідний файл та вихідний), якщо їх не вказати, то за замовчуванням дані беруться з файлу books.txt, а виводу в файл не відбувається 0 лише на екран. Скріншот виконання:



Вхідний та вихідний файл відповідно:

(input.txt):

John Carre,Smiley's People,1979,237,9.99

Edith Somerville,Some Experiences of an Irish R.M.,1899,1486,13.5

Jeff Kinney,Diary of a Wimpy Kid: Old School,2015,275,9.78

Donna Tartt,The Goldfinch,2013,487,6.89

Ronan Farrow,Catch and Kill,2019,674,14.99

Miranda Popkey,Topics of Conversation: A Novel,2020,246,7.99

Antonio Antunes,Fado Alexandrino,1983,443,4.99

Ernest Hemingway,The Old Man and the Sea,1952,243,6.99

Christopher Isherwood,Goodbye to Berlin,1939,345,8.45

V.S. Naipaul,A Bend in the River,1979,875,14.49

Richard Brautigan,In Watermelon Sugar,1968,478,11.99

Isabel Wilkerson,The Warmth of Other Suns,2010,986,15.99

Matthew McConaughey,Greenlights,2020,1242,18.99

Kazuo Ishiguro,Klara and the Sun,2021,534,12.99

Park Kyung-ni,Land,1969,612,10.99

Heinrich Kleist,Michael Kohlhaas,1810,764,3.99

Wyndham Lewis,The Revenge for Love,1937,834,6.49

Siddhartha Mukherjee,The Gene: An Intimate History,2016,654,11.49

(output.txt):

Antonio Antunes,Fado Alexandrino,1983,443,4.99

John Carre,Smiley's People,1979,237,9.99

Ernest Hemingway,The Old Man and the Sea,1952,243,6.99

Christopher Isherwood,Goodbye to Berlin,1939,345,8.45

Jeff Kinney,Diary of a Wimpy Kid: Old School,2015,275,9.78

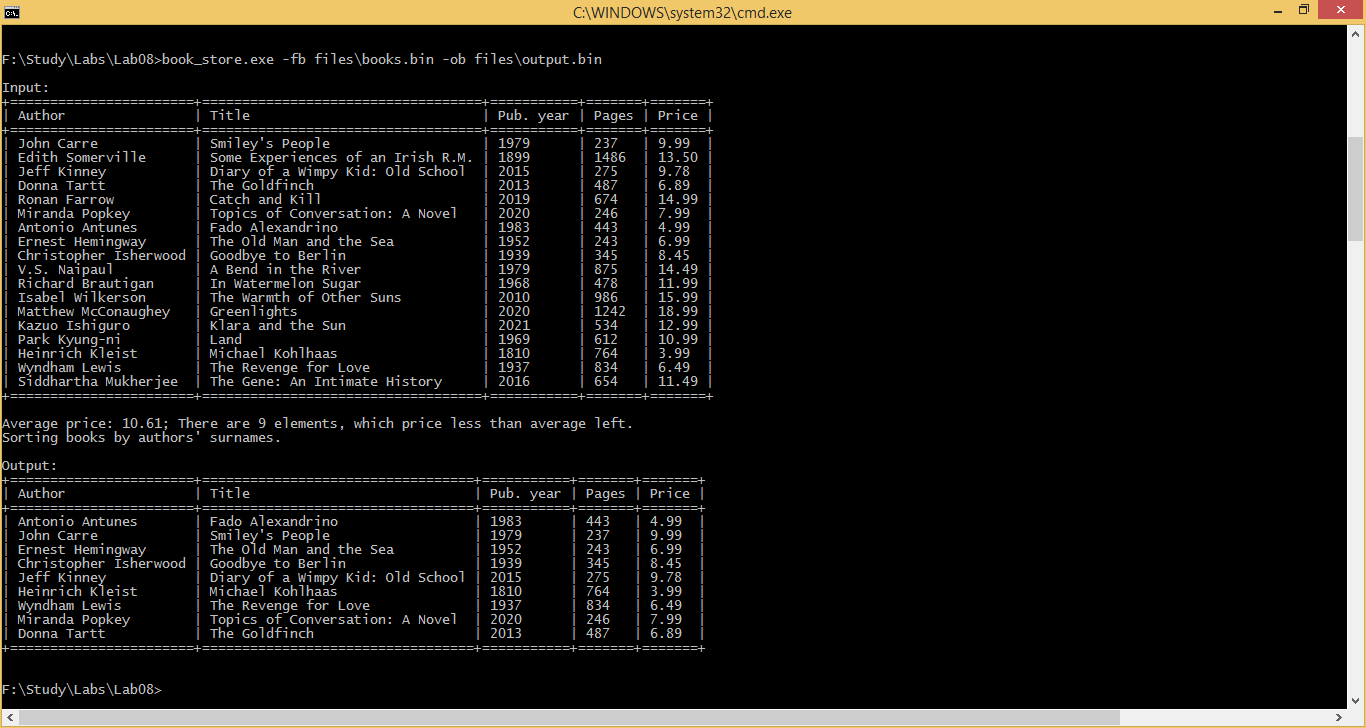
Heinrich Kleist,Michael Kohlhaas,1810,764,3.99

Wyndham Lewis,The Revenge for Love,1937,834,6.49

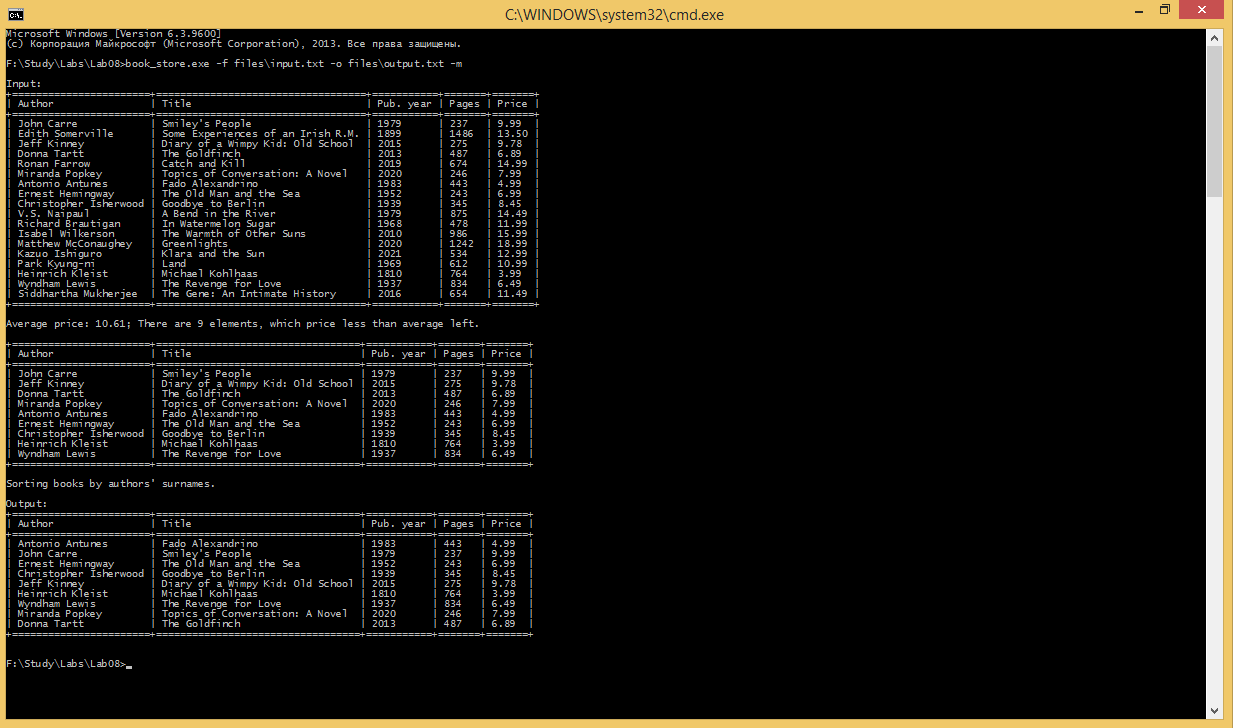
Miranda Popkey,Topics of Conversation: A Novel,2020,246,7.99

Donna Tartt,The Goldfinch,2013,487,6.89

Також є функціонал по роботі з бінарними файлами:



А ще можливо вказати, щоб програма виводила результати проміжних кроків:



Як бачимо, програма успішно відпрацьовує у всіх цих випадках. Як видно з коду, функціонал реалізовано через передачу необхідних операцій над масивом книг у парматери іншої функції. Це необхідно для значного збільшення гнучкості – іншому програмісту, який буде використовувати цю бібліотеку в своєму коді необхідно буде лише реалізувати функцію за заданим шаблоном і передати вказівник на неї в конвеєр обробки – все інше зробить бібліотека.

**Висновок:** при виконанні цього завдання ми навчились використовувати структури – формувати екземпляри по считаній з файлу стрічці, сортувати, виводити та багато іншого.