

МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ЖЕЛЕЗНОДОРОЖНОГО ТРАНСПОРТА

Государственное бюджетное образовательное учреждение
высшего образования

«ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПУТЕЙ СООБЩЕНИЯ ИМПЕРАТОРА АЛЕКСАНДРА I»

Кафедра «ИНФОРМАЦИОННЫЕ И ВЫЧИСЛИТЕЛЬНЫЕ СИСТЕМЫ»

Дисциплина: «Программирование(С)»

О Т Ч Е Т
по лабораторной работе № 4

Вариант 19

Выполнил студент
Факультета *АИТ*
Группы *ИББ-211*

Шефнер А.

Санкт-Петербург
2023

Постановка задачи

Написать программу, которая читает данные произвольной размерности из одного файла, преобразует прочитанные данные и записывает получившийся результат в другой файл.

При выполнении задания продемонстрировать применение различных функций для работы с файлами (fprintf(), fscanf(), fgetc(), fputc(), fgets(), fputs(), fwrite(), fread()).

Данные представить тремя вариантами:

- a) Как двумерные динамические массивы
- b) Как строки
- c) Как структуры

Программа должна содержать несколько пользовательских функций, помещенных в соответствующие отдельные файлы.

Текст программ снабдить поясняющими комментариями.

19 Вариант

В файле содержится информация типа “автор” в следующем виде: фамилия автора, направление (физика, программирование и т.п.), год издания, число страниц. Считать информацию из файла, в другой файл записать только информацию об книгах последних 5 лет издания.

Пояснения

В главной функции вызываются функции `test_books_1` и `test_books_2`. Первая тестирует все функции из файлов `books_1.h` и `books_1.c`. В них используется структура из `book.h` и функции стандартной библиотеки `fprintf` и `fscanf`. Вторая тестирует все функции из файлов `books_2.h` и `books_2.c`. В них используются массив строк, двумерный массив строк и функции стандартной библиотеки `fgets`, `fputs`, `fputc`.

Код программы

c_lab_4.c (точка входа программы)

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>
#include <stdlib.h>
#include "books_1.h"
#include "books_2.h"

int filter_struct(book* book)
{
    return book->year >= 2000;
}

int filter_line(char** line)
{
    return atoi(line[2]) >= 2018;
}

void test_books_1(void);

void test_books_2(void);

int main(int argc, char* argv[])
{
    test_books_1();

    system("pause"); // NOLINT(concurrency-mt-unsafe)
    return 0;
}

void test_books_1(void)
{
    int count;
    book* books = get_books_from_file("books.txt", &count);
    for(int i = 0; i < count; i++)
    {
        printf("%s - %s %d year (%d pages)\n", books[i].surname,
books[i].theme, books[i].year, books[i].page_count);
    }

    printf("\n\nfiltered:\n\n");

    int filtered_count = 0;
    book* filtered = filter_books(books, filter_struct, count,
&filtered_count);
    write_books_to_file(filtered, filtered_count,
"filtered.txt");
    for(int i = 0; i < filtered_count; i++)
    {
        printf("%s - %s %d year (%d pages)\n",
filtered[i].surname, filtered[i].theme, filtered[i].year,
filtered[i].page_count);
    }
}
```

```

    }

    free(books);
    free(filtered);
}

void test_books_2(void)
{
    int count;
    char** lines = get_lines("books.txt", &count);

    printf("Lines read from file\n\n");
    for(int i = 0; i < count; i++)
    {
        printf("%s\n", lines[i]);
    }

    char*** chopped_lines = chop_lines(lines, count);

    printf("\n\nChopped lines:\n\n");
    for(int i = 0; i < count; i++)
    {
        printf("%s|%s|%s|%s\n",
            chopped_lines[i][0],
            chopped_lines[i][1],
            chopped_lines[i][2],
            chopped_lines[i][3]
        );
    }

    int filtered_count;
    char*** filtered_lines = filter_chopped_lines(
        chopped_lines,
        filter_line,
        count,
        &filtered_count
    );
    printf("\n\nFiltered lines:\n\n");
    for(int i = 0; i < filtered_count; i++)
    {
        printf("%s|%s|%s|%s\n",
            filtered_lines[i][0],
            filtered_lines[i][1],
            filtered_lines[i][2],
            filtered_lines[i][3]
        );
    }

    write_chopped_lines("output.txt", filtered_lines,
        filtered_count);

    printf("\n\nData successfully written to file.\n\n");
}

```

```
    free_lines(lines, count);  
    free_chopped_lines(chopped_lines, count);  
    free_chopped_lines(filtered_lines, filtered_count);  
}
```

book.h (структура book)

```
#pragma once

#define SURNAME_CHAR_NUMBER 20
#define THEME_CHAR_NUMBER 50

#define SURNAME_FORMAT "%20s"
#define THEME_FORMAT "%50s"
#define YEAR_FORMAT "%5hu"
#define PAGE_FORMAT "%5hu"

typedef struct book
{
    char surname[SURNAME_CHAR_NUMBER];
    char theme[THEME_CHAR_NUMBER];
    unsigned short year;
    unsigned short page_count;
} book;
```

books_1.h (через структуру, fprintf и fscanf)

```
#pragma once
```

```
#include "book.h"
```

```
book* get_books_from_file(const char* path, int* count);
```

```
book* filter_books(book* books, int (*filter)(book*), int count,  
int* out_count);
```

```
void write_books_to_file(book* books, int count, char* path);
```


books_1.c

```
#define _CRT_SECURE_NO_WARNINGS

#include "books_1.h"

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

void replace_char(char str[], char from, char to, int size)
{
    for(int i = 0; i < size; i++)
    {
        if(str[i] == from) str[i] = to;
    }
}

book* get_books_from_file(const char* path, int* count)
{
    FILE* file;
    file = fopen(path, "r");
    fscanf(file, "%i\n", count);
    book* books = malloc(sizeof(book) * *count);
    for(int i = 0; i < *count; i++)
    {
        char* surname[SURNAME_CHAR_NUMBER];
        char* theme[THEME_CHAR_NUMBER];
        unsigned short year;
        unsigned short page_count;
        fscanf(file, SURNAME_FORMAT " " THEME_FORMAT " "
YEAR_FORMAT " " PAGE_FORMAT "\n",
            surname,
            theme,
            &year,
            &page_count
        );
        strcpy(books[i].surname, surname);
        strcpy(books[i].theme, theme);
        replace_char(books[i].theme, '_', ' ',
THEME_CHAR_NUMBER);
        books[i].year = year;
        books[i].page_count = page_count;
    }
    fclose(file);
    return books;
}

book* filter_books(book* books, int(* filter)(book*), int count,
int* out_count)
{
    *out_count = 0;
    short* suitable_books = malloc(sizeof(short) * count);
    for(int i = 0; i < count; i++)
```

```

    {
        suitable_books[i] = filter(&books[i]);
        if(suitable_books[i]) (*out_count)++;
    }

    book* filtered_books = malloc(sizeof(book) * (*out_count));
    for(int i = 0, j = 0; i < count; i++)
    {
        if(suitable_books[i])
        {
            filtered_books[j] = books[i];
            j++;
        }
    }
    free(suitable_books);
    return filtered_books;
}

void write_books_to_file(book* books, int count, char* path)
{
    FILE* file;
    file = fopen(path, "w");
    fprintf(file, "%d\n", count);
    for(int i = 0; i < count; i++)
    {
        char theme[THEME_CHAR_NUMBER];
        strcpy(theme, books[i].theme);
        replace_char(theme, ' ', '_', THEME_CHAR_NUMBER);
        fprintf(file, SURNAME_FORMAT " " THEME_FORMAT " "
YEAR_FORMAT " " PAGE_FORMAT "\n", books[i].surname, theme,
books[i].year, books[i].page_count);
    }
    fclose(file);
}

```

books_2.h (через массив строк, двумерный массив, fgets, fputc и fputc)

```
#pragma once
```

```
#define SURNAME_CHARS 20
```

```
#define THEME_CHARS 50
```

```
#define YEARS_CHARS 5
```

```
#define PAGES_CHARS 5
```

```
// lines are allocated with malloc, so don't forget to free.
```

```
char** get_lines(char* path, int* count);
```

```
char*** chop_lines(char** lines, int count);
```

```
char*** filter_chopped_lines(  
    char*** chopped_lines,  
    int (*filter)(char**),  
    int count,  
    int* filtered_count  
);
```

```
void write_chopped_lines(char* path, char*** chopped_lines, int  
count);
```

```
void free_lines(char** lines, int count);
```

```
void free_chopped_lines(char*** lines, int count);
```

books_2.c

```
// ReSharper disable CppDeprecatedEntity
// ReSharper disable
CppClangTidyClangDiagnosticDeprecatedDeclarations
#define _CRT_SECURE_NO_WARNINGS

#include "books_2.h"

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

const int len = SURNAME_CHARS + THEME_CHARS + YEARS_CHARS +
PAGES_CHARS + 3;

char** get_lines(char* path, int* count)
{
    FILE* file = fopen(path, "r");

    char len_str[10];
    fgets(len_str, 10, file);
    *count = atoi(len_str);
    char** lines = malloc(sizeof(char*) * *count);
    for(int i = 0; i < *count; i++)
    {
        char* tmp = malloc(sizeof(char) * (len + 2));
        fgets(tmp, len + 2, file);
        tmp[len] = '\0';
        lines[i] = (char*)malloc(sizeof(char) * len + 1);
        strcpy(lines[i], tmp);
        free(tmp);
    }
    return lines;
}

char*** chop_lines(char** lines, int count)
{
    char*** chopped_lines = malloc(sizeof(char**) * count);
    for(int i = 0; i < count; i++)
    {
        chopped_lines[i] = malloc(sizeof(char*) * 4);

        chopped_lines[i][0] = malloc(sizeof(char) *
SURNAME_CHARS + 1);
        chopped_lines[i][1] = malloc(sizeof(char) * THEME_CHARS
+ 1);
        chopped_lines[i][2] = malloc(sizeof(char) * YEARS_CHARS
+ 1);
        chopped_lines[i][3] = malloc(sizeof(char) * PAGES_CHARS
+ 1);

        int str_index = 0;
        for(int j = 0; j < SURNAME_CHARS; j++)
```

```

        {
            chopped_lines[i][0][j] = lines[i][str_index++];
        }
        str_index++;
        for(int j = 0; j < THEME_CHARS; j++)
        {
            chopped_lines[i][1][j] = lines[i][str_index++];
        }
        str_index++;
        for(int j = 0; j < YEARS_CHARS; j++)
        {
            chopped_lines[i][2][j] = lines[i][str_index++];
        }
        str_index++;
        for(int j = 0; j < PAGES_CHARS; j++)
        {
            chopped_lines[i][3][j] = lines[i][str_index++];
        }

        chopped_lines[i][0][SURNAME_CHARS] = '\\0';
        chopped_lines[i][1][THEME_CHARS] = '\\0';
        chopped_lines[i][2][YEARS_CHARS] = '\\0';
        chopped_lines[i][3][PAGES_CHARS] = '\\0';
    }
    return chopped_lines;
}

char*** filter_chopped_lines(char*** chopped_lines,
int(*filter)(char**), int count, int* filtered_count)
{
    *filtered_count = 0;
    short* suitable_lines = (short*)malloc(sizeof(short) *
count);
    for(int i = 0; i < count; i++)
    {
        suitable_lines[i] = filter(chopped_lines[i]);
        if(suitable_lines[i]) (*filtered_count)++;
    }

    char*** filtered_lines = malloc(sizeof(char**) *
(*filtered_count));
    for(int i = 0, j = 0; i < count; i++)
    {
        if(suitable_lines[i])
        {
            filtered_lines[j] = malloc(sizeof(char*) * 4);

            filtered_lines[j][0] = malloc(sizeof(char) *
SURNAME_CHARS + 1);
            filtered_lines[j][1] = malloc(sizeof(char) *
THEME_CHARS + 1);
            filtered_lines[j][2] = malloc(sizeof(char) *
YEARS_CHARS + 1);

```

```

        filtered_lines[j][3] = malloc(sizeof(char) *
PAGES_CHARS + 1);
        strcpy(filtered_lines[j][0], chopped_lines[i][0]);
        strcpy(filtered_lines[j][1], chopped_lines[i][1]);
        strcpy(filtered_lines[j][2], chopped_lines[i][2]);
        strcpy(filtered_lines[j][3], chopped_lines[i][3]);

        j++;
    }
}
free(suitable_lines);
return filtered_lines;
}

void write_chopped_lines(char* path, char*** chopped_lines, int
count)
{
    FILE* file = fopen(path, "w");
    char count_str[10];
    _itoa(count, count_str, 20);
    fputs(count_str, file);
    fputc('\n', file);

    for(int i = 0; i < count; i++)
    {
        fputs(chopped_lines[i][0], file);
        fputc(' ', file);
        fputs(chopped_lines[i][1], file);
        fputc(' ', file);
        fputs(chopped_lines[i][2], file);
        fputc(' ', file);
        fputs(chopped_lines[i][3], file);
        fputc(' ', file);
        fputc('\n', file);
    }
}

void free_lines(char** lines, int count)
{
    for(int i = 0; i < count; i++)
    {
        free(lines[i]);
    }
    free(lines);
}

void free_chopped_lines(char*** lines, int count)
{
    for(int i = 0; i < count; i++)
    {
        for(int j = 0; j < 4; j++)
        {
            free(lines[i][j]);

```

```
        }
        free(lines[i]);
    }
    free(lines);
}
```


Отладка приложения.

Далее представлен вывод консоли, поскольку он не помещается в скриншоты. Вы можете проверить вывод консоли самостоятельно с помощью exe файла.

books_1

```
Martin - Clean Code Piter 2021 year (464 pages)
Richter - CLR via C# 2012 year (896 pages)
Shuuichi - Saiki Kusuo no PSI Nan vol. 1 2012 year (193 pages)
Prata - C Primer Plus 5th Edition 2004 year (1202 pages)
Marx - Das Capital 1867 year (200 pages)
Gyasi - Transcendent Kingdom 2020 year (288 pages)
Fujio - Doraemon Vol 1 1969 year (657 pages)
Matthes - Python Crash Course, 3rd Edition 2023 year (552 pages)
Heisig - Remembering the Kanji vol. I 2001 year (522 pages)
Yong - An Immense World 2022 year (464 pages)
Privalov - Entrance to CVFT 1999 year (431 pages)
Yolen - Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 year (320 pages)
Ulrickson - A Brief Quadrivium 2023 year (302 pages)
Tokuno - New Game vol. 01 2013 year (126 pages)
O'Farrell - Hamnet 2020 year (320 pages)
Golden - World of Warcraft: Arthas: Rise of the Lich King 2010 year (416 pages)
Nosonov - Socio-economic geography 2nd Edition 2019 year (476 pages)
```

filtered:

```
Martin - Clean Code Piter 2021 year (464 pages)
Richter - CLR via C# 2012 year (896 pages)
Shuuichi - Saiki Kusuo no PSI Nan vol. 1 2012 year (193 pages)
Prata - C Primer Plus 5th Edition 2004 year (1202 pages)
Gyasi - Transcendent Kingdom 2020 year (288 pages)
Matthes - Python Crash Course, 3rd Edition 2023 year (552 pages)
Heisig - Remembering the Kanji vol. I 2001 year (522 pages)
Yong - An Immense World 2022 year (464 pages)
Yolen - Dragon's Blood: The Pit Dragon Chronicles, Vol. 1 2021 year (320 pages)
Ulrickson - A Brief Quadrivium 2023 year (302 pages)
Tokuno - New Game vol. 01 2013 year (126 pages)
O'Farrell - Hamnet 2020 year (320 pages)
Golden - World of Warcraft: Arthas: Rise of the Lich King 2010 year (416 pages)
Nosonov - Socio-economic geography 2nd Edition 2019 year (476 pages)
```

books_2

Lines readen from file

Martin	Clean_Code_Piter	2021	464
Richter	CLR_via_C#	2012	896
Shuuichi	Saiki_Kusuo_no_PSI_Nan_vol._1	2012	193
Prata	C_Primer_Plus_5th_Edition	2004	1202
Marx	Das_Capital	1867	200
Gyasi	Transcendent_Kingdom	2020	288
Fujio	Doraemon_Vol_1	1969	657
Matthes	Python_Crash_Course,_3rd_Edition	2023	552
Heisig	Remembering_the_Kanji_vol._I	2001	522
Yong	An_Immense_World	2022	464
Privalov	Entrance_to_CVFT	1999	431
Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
Ulrickson	A_Brief_Quadrivium	2023	302
Tokuno	New_Game_vol._01	2013	126
O'Farrell	Hamnet	2020	320
Golden	World_of_Warcraft:_Arthas:_Rise_of_the_Lich_King	2010	416
Nosonov	Socio-economic_geography_2nd_Edition	2019	476

Chopped lines:

Martin	Clean_Code_Piter	2021	464
Richter	CLR_via_C#	2012	896
Shuuichi	Saiki_Kusuo_no_PSI_Nan_vol._1	2012	193
Prata	C_Primer_Plus_5th_Edition	2004	1202
Marx	Das_Capital	1867	200
Gyasi	Transcendent_Kingdom	2020	288
Fujio	Doraemon_Vol_1	1969	657
Matthes	Python_Crash_Course,_3rd_Edition	2023	552
Heisig	Remembering_the_Kanji_vol._I	2001	522
Yong	An_Immense_World	2022	464
Privalov	Entrance_to_CVFT	1999	431
Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
Ulrickson	A_Brief_Quadrivium	2023	302
Tokuno	New_Game_vol._01	2013	126
O'Farrell	Hamnet	2020	320
Golden	World_of_Warcraft:_Arthas:_Rise_of_the_Lich_King	2010	416
Nosonov	Socio-economic_geography_2nd_Edition	2019	476

Filtered lines:

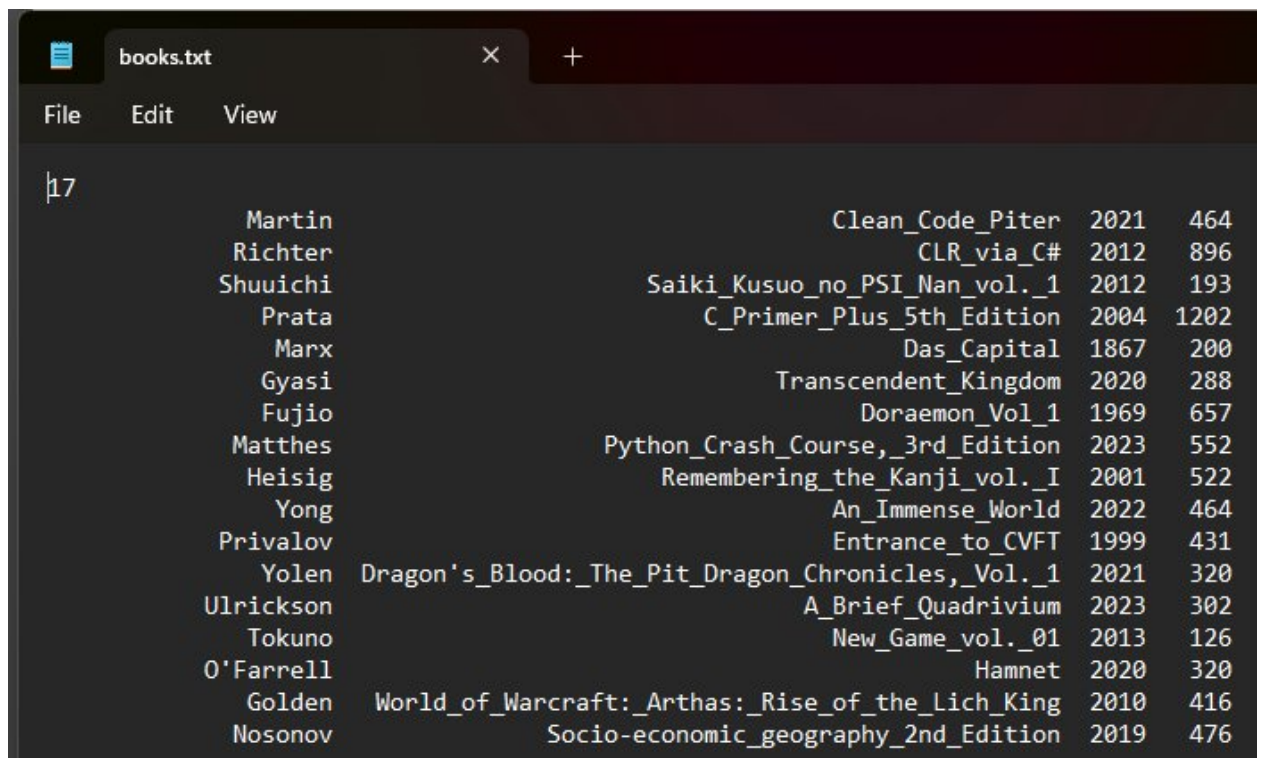
Martin	Clean_Code_Piter	2021	464
Gyasi	Transcendent_Kingdom	2020	288
Matthes	Python_Crash_Course,_3rd_Edition	2023	552
Yong	An_Immense_World	2022	464
Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
Ulrickson	A_Brief_Quadrivium	2023	302
O'Farrell	Hamnet	2020	320
Nosonov	Socio-economic_geography_2nd_Edition	2019	476

Data successfully written to file.

Press any key to continue . . .

Содержимое файлов.

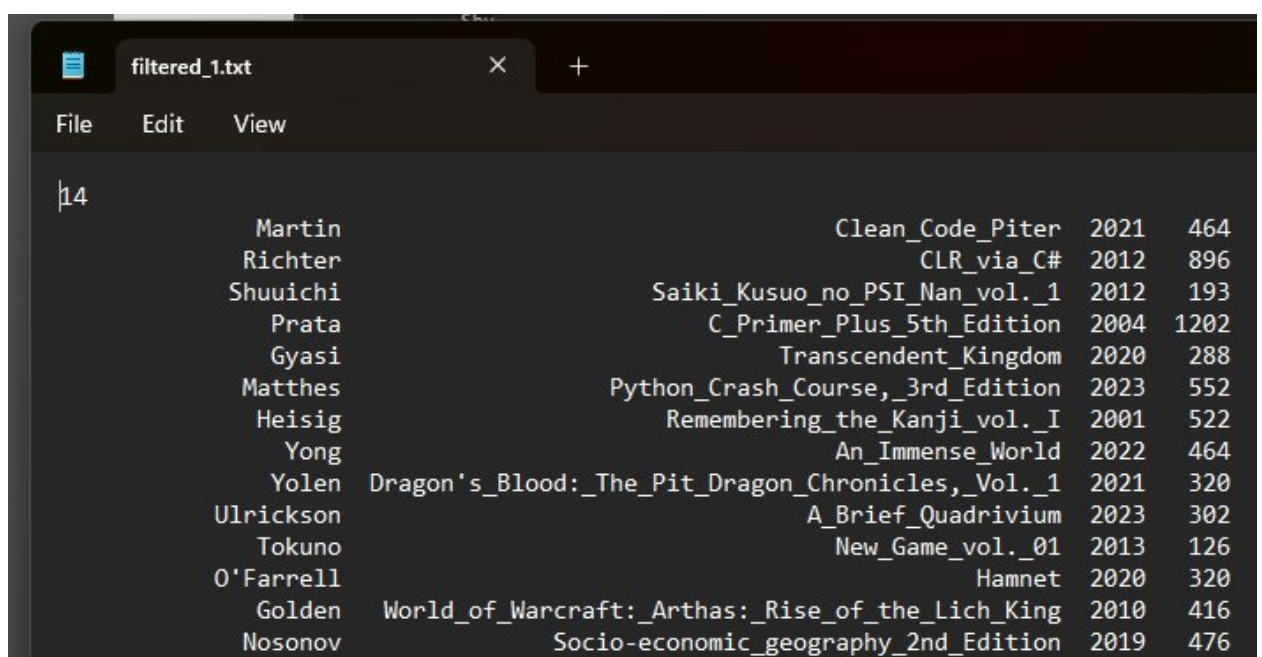
books.txt – исходный файл, из которого считываются книги.



17	Martin	Clean_Code_Piter	2021	464
	Richter	CLR_via_C#	2012	896
	Shuuichi	Saiki_Kusuo_no_PSI_Nan_vol._1	2012	193
	Prata	C_Primer_Plus_5th_Edition	2004	1202
	Marx	Das_Capital	1867	200
	Gyasi	Transcendent_Kingdom	2020	288
	Fujio	Doraemon_Vol_1	1969	657
	Matthes	Python_Crash_Course,_3rd_Edition	2023	552
	Heisig	Remembering_the_Kanji_vol._I	2001	522
	Yong	An_Immense_World	2022	464
	Privalov	Entrance_to_CVFT	1999	431
	Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
	Ulrickson	A_Brief_Quadrivium	2023	302
	Tokuno	New_Game_vol._01	2013	126
	O'Farrell	Hamnet	2020	320
	Golden	World_of_Warcraft:_Arthas:_Rise_of_the_Lich_King	2010	416
	Nosonov	Socio-economic_geography_2nd_Edition	2019	476

filtered_1.txt – результат работы функций из books_1.
Условие: год выпуска книги больше или равен 2000.

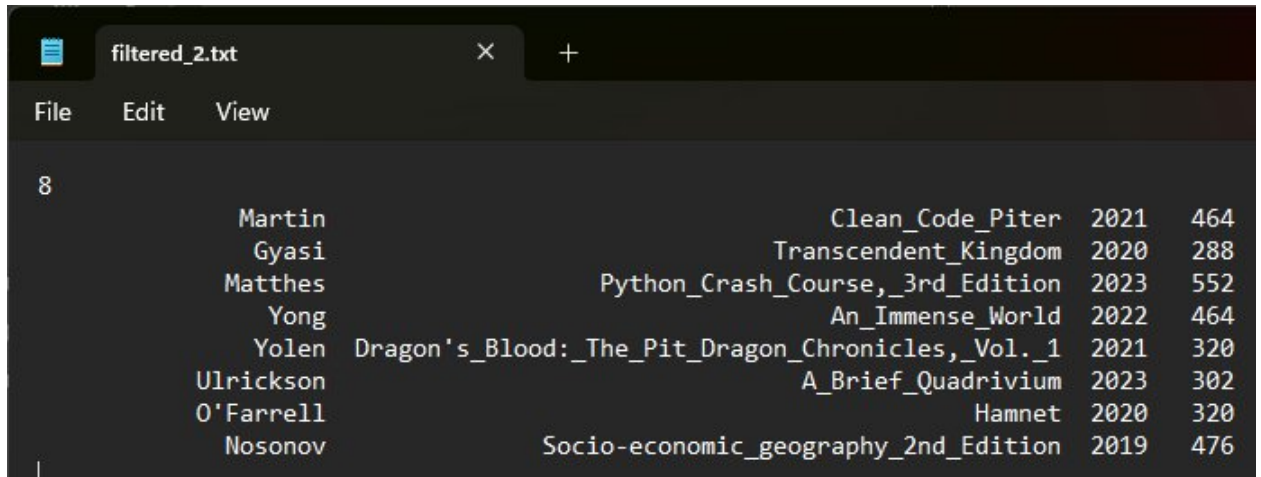
```
int filter_struct(book* book)
{
    return book->year ≥ 2000;
}
```



14	Martin	Clean_Code_Piter	2021	464
	Richter	CLR_via_C#	2012	896
	Shuuichi	Saiki_Kusuo_no_PSI_Nan_vol._1	2012	193
	Prata	C_Primer_Plus_5th_Edition	2004	1202
	Gyasi	Transcendent_Kingdom	2020	288
	Matthes	Python_Crash_Course,_3rd_Edition	2023	552
	Heisig	Remembering_the_Kanji_vol._I	2001	522
	Yong	An_Immense_World	2022	464
	Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
	Ulrickson	A_Brief_Quadrivium	2023	302
	Tokuno	New_Game_vol._01	2013	126
	O'Farrell	Hamnet	2020	320
	Golden	World_of_Warcraft:_Arthas:_Rise_of_the_Lich_King	2010	416
	Nosonov	Socio-economic_geography_2nd_Edition	2019	476

filtered_2.txt – результат работы функций из books_2.
Условие: год книги больше или равен 2018.

```
int filter_line(char** line)
{
    return atoi(line[2]) ≥ 2018;
}
```



8	Martin	Clean_Code_Piter	2021	464
	Gyasi	Transcendent_Kingdom	2020	288
	Matthes	Python_Crash_Course,_3rd_Edition	2023	552
	Yong	An_Immense_World	2022	464
	Yolen	Dragon's_Blood:_The_Pit_Dragon_Chronicles,_Vol._1	2021	320
	Ulrickson	A_Brief_Quadrivium	2023	302
	O'Farrell	Hamnet	2020	320
	Nosonov	Socio-economic_geography_2nd_Edition	2019	476