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(C)»

ОТЧЕТ

по лабораторной работе № 4

Вариант *19*

Выполнил студент Шефнер А.

Факультета *АИТ*

Группы *ИВБ-211*

Санкт-Петербург

2023

**Постановка задачи**

Написать программу, которая читает данные произвольной размерности из одного файла, преобразует прочитанные данныеи 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 readden 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 readden 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

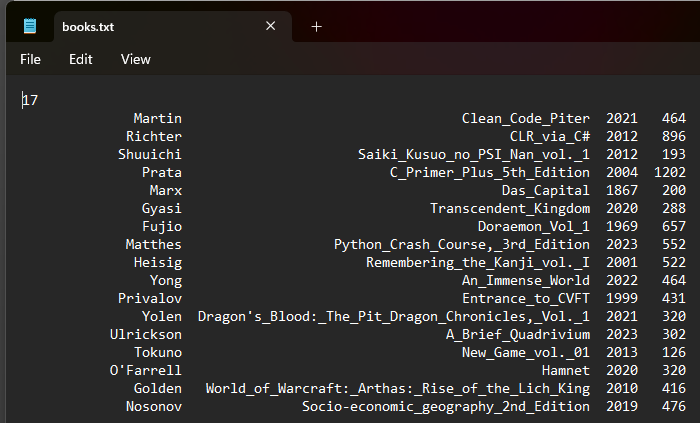
Nosonov| Socio-economic\_geography\_2nd\_Edition| 2019| 476

Data successfully written to file.

Press any key to continue . . .

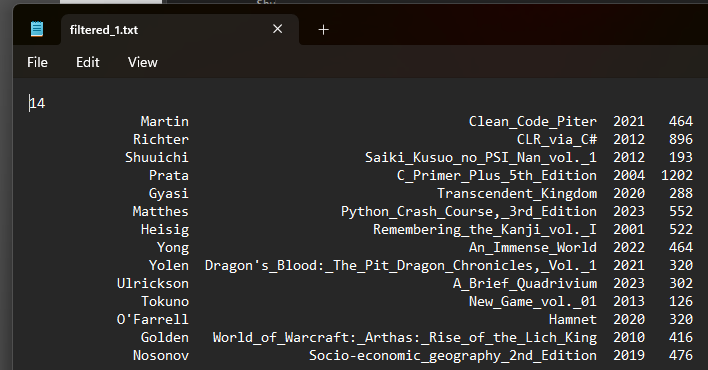
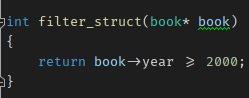
**Содержимое файлов.**

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



filtered\_1.txt – результат работы функций из books\_1.

Условие: год выпуска книги больше или равен 2000.



filtered\_2.txt – результат работы функций из books\_2.

Условие: год книги больше или равен 2018.

