МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Московский авиационный институт

(национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»

Кафедра 806: «Вычислительная математика и программирование»

Курсовой проект

по курсу «Основы информатики»

I семестр

Задание 6. «Обработка последовательной файловой структуры на языке СИ.»

|  |  |
| --- | --- |
| Группа: | М8о-107б-18 |
| Студент: | Тояков Артем |
| Преподаватель: | Ридли Александра Николаевна |
| Оценка: |  |
| Дата: |  |

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc7255726)

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc7255727)

[СТРУКТУРА ПРОГРАММЫ 3](#_Toc7255728)

[ОПИСАНИЕ ПРОГРАММЫ 4](#_Toc7255729)

[ПРИМЕР РАБОТЫ 11](#_Toc7255730)

[ВЫВОД 12](#_Toc7255731)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 12](#_Toc7255732)

# ПОСТАНОВКА ЗАДАЧИ

Разработать последовательную структуру данных для представления простейшей базы данных на файлах в СП СИ в соответствии с заданным вариантом. Составить программу генерации внешнего нетекстового файла заданной структуры. Распечатать содержимое сгенерированного файла в виде таблицы.

Вариант №26:

Найти абитуриентов-немедалистов, суммарный балл которых выше среднего.

# СТРУКТУРА ПРОГРАММЫ

В данной работе мой кп будет состоять из 8 элементов:

1. Текстовый файл с таблицей, подающейся на вход.
2. Заголовочный файл io.h, который служит для подключения нужных мне функций.
3. io.c, функции считывания текстового файла, записи и считывания бинарного файла, а также печати.
4. print.c, функция печати бинарного файла.
5. Makefile.
6. person.h, наш собственный «созданный» класс.

7. generate.c, генератор текстового файла в бинарный.

8. execute.c, выполнение нашего условия и вывод результата.

# ОПИСАНИЕ ПРОГРАММЫ

Сначала я опишу файл **person.h**. Он выглядит вот так:

#ifndef person\_h\_

#define person\_h\_

#include <stdlib.h>

#include <stdio.h>

#include <string.h>

#define surname\_LENGTH 32

#define initials\_LENGTH 5

typedef struct {

char surname[surname\_LENGTH];

char initials[initials\_LENGTH];

char gender;

char medal;

int school\_number;

int literature;

int russian\_language;

int informatics;

int physics;

char essay;

} person;

#endif

В нём я создал свою переменную (класс) с помощью typedef struct, которая содержит информацию про абитуриента. В таблице содержится фамилия, инициалы, пол, наличие медали, номер школы, оценки по четырём предметам и информация о сдаче сочинения.

Дальше моя программа содержит **io.h**:

#ifndef IO\_H\_

#define IO\_H\_

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "person.h"

int readtxt(person \*person, FILE \*in);

int readbin(person \*person, FILE \*in);

void writebin(person \*person, FILE \*out);

void print(person \*person);

#endif

В этом файле содержатся названия функций, которые мы подключаем при компиляции, а именно (**io.c**):

#include "io.h"

int readtxt(person \*person, FILE \*in)

{

fscanf(in, "%s", person->surname);

fscanf(in, "%s", person->initials);

fscanf(in, " %c", &(person->gender));

fscanf(in, "%d", &(person->school\_number));

fscanf(in, " %c", &(person->medal));

fscanf(in, "%d", &(person->literature));

fscanf(in, "%d", &(person->russian\_language));

fscanf(in, "%d", &(person->informatics));

fscanf(in, "%d", &(person->physics));

char fail\_or\_pass[4];

fscanf(in, "%s", fail\_or\_pass);

person->essay = strcmp(fail\_or\_pass, "pass");

return !feof(in);

}

int readbin(person \*person, FILE \*in)

{

fread(person->surname, sizeof(char), surname\_LENGTH, in);

fread(person->initials, sizeof(char), initials\_LENGTH, in);

fread(&(person->gender), sizeof(char), 1, in);

fread(&(person->medal), sizeof(char), 1, in);

fread(&(person->school\_number), sizeof(int), 1, in);

fread(&(person->literature), sizeof(int), 1, in);

fread(&(person->russian\_language), sizeof(int), 1, in);

fread(&(person->informatics), sizeof(int), 1, in);

fread(&(person->physics), sizeof(int), 1, in);

fread(&(person->essay), sizeof(char), 1, in);

return !feof(in);

}

void writebin(person \*person, FILE \*out)

{

fwrite(person->surname, sizeof(char), surname\_LENGTH, out);

fwrite(person->initials, sizeof(char), initials\_LENGTH, out);

fwrite(&(person->gender), sizeof(char), 1, out);

fwrite(&(person->medal), sizeof(char), 1, out);

fwrite(&(person->school\_number), sizeof(int), 1, out);

fwrite(&(person->literature), sizeof(int), 1, out);

fwrite(&(person->russian\_language), sizeof(int), 1, out);

fwrite(&(person->informatics), sizeof(int), 1, out);

fwrite(&(person->physics), sizeof(int), 1, out);

fwrite(&(person->essay), sizeof(char), 1, out);

}

void print(person \*person) {

printf(

"Фамилия: %s Инициалы: %s Пол: %c. Есть медаль: %c ",

person->surname,

person->initials,

person->gender,

person->medal

);

printf(

"Все оценки: %d %d %d %d ",

person->literature,

person->russian\_language,

person->informatics,

person->physics

);

if (person->essay == 0) {

printf("Эссе сдал.");

} else {

printf("Эссе не сдал.");

}

printf("\n");

}

Также хочется отметить, что в данной работе мы также используем операторы «->» , «.», которые по сути означают обращение элемента a к члену структуры b (a -> b). Также File (\*in) и (\*out) это указатели на наши файлы, которые мы будем использовать. Особенностью языка СИ является его связь с памятью компьютера и с командной строкой, что хорошо транслируется в ходе данной работы.

Также нам нужно распечатать нашу таблицу, а за это отвечает файл **print.c**. Уже на данном этапе, чтобы распечатать файл, мы вводим в командной строке «\.print dbfile», а также в ходе работы видно, что именно argv отвечает за количество слов в командной строке. После мы просто считываем бинарный файл и печатаем его с помощью функций, описанных выше.

#include <stdio.h>

#include <stdlib.h>

# include <string.h>

#include "person.h"

#include "io.h"

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

{

if (argc != 2) {

printf("Usage: ./print <binary\_DB\_file>\n");

exit(0);

}

FILE \*in = fopen(argv[1], "r");

if (!in) {

printf("Error: could not open file\n");

exit(1);

}

person s;

while (readbin(&s, in)) {

print(&s);

}

fclose(in);

return 0;

}

Теперь опишем файл **generate.c**. Как и в print.c мы вначале делаем проверку на ввод в командной строке, а позже считываем текстовый файл и пишем его в бинарном формате в другой файл. В ходе данной работы также используются ссылки, то есть мы ссылаемся на какую-то переменную, содержащуюся в памяти компьютера.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "io.h"

#include "person.h"

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

{

if (argc != 3) {

printf("Usage: ./generate <txt\_file\_from> <txt\_file\_to>\n");

exit(0);

}

FILE \*in = fopen(argv[1], "r");

FILE \*out = fopen(argv[2], "w");

if (!in || !out) {

printf("Error: could not open file\n");

exit(1);

}

person current\_person;

while (!feof(in)) {

readtxt(&current\_person, in);

writebin(&current\_person, out);

}

fclose(in);

fclose(out);

return 0;

}

В начале файла **execute.c** проводится проверка командной строки, а также мы считываем с нашей структуры нужные нам переменые. После мы создаем условие и с помощью него анализируем данные в нашем бинарном файле.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include "io.h"

#include "person.h"

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

{

if (argc != 2) {

printf("Usage: ./execute <bin\_DB\_file>\n");

exit(0);

}

FILE \*in = fopen(argv[1], "r");

if (!in) {

printf("Error: could not open file\n");

exit(1);

}

person current\_person;

int avg = 0;

int count = 0;

while (readbin(&current\_person, in)) {

avg += current\_person.literature;

avg += current\_person.russian\_language;

avg += current\_person.informatics;

avg += current\_person.physics;

count++;

}

avg /= count;

fseek(in, 0, SEEK\_SET);

while (readbin(&current\_person, in)) {

if (

(

current\_person.literature

+ current\_person.russian\_language

+ current\_person.informatics

+ current\_person.physics

> avg

) && (

current\_person.medal == 'n'

) && (

current\_person.gender == 'm'

)

) {

print(&current\_person);

}

}

return 0;

}

**Makefile** отвечает за компиляцию. В моём случае он содержит параметры компиляции и параметры clean (отвечает за удаление всех новых файлов).

CC = gcc

CFLAGS = -g -std=c99 -Wextra -pedantic -Werror

all: generate print execute

generate: io.o generate.o

$(CC) $^ -o generate

print: io.o print.o

$(CC) $^ -o print

execute: io.o execute.o

$(CC) $^ -o execute

clean:

rm -f execute generate print \*.bin \*.o

Также в КП6 содержится текстовый файл **students.txt**, в котором прописаны все нужные сведения о нескольких абитуриентах.

Pushkin A.S. m 1099 n 85 89 100 55 pass

Tolstoy L.N. m 1537 y 56 88 45 65 fail

Tsvetaeva M.I. f 1130 n 48 99 54 48 pass

Toyakov A.O. m 1537 y 68 45 98 75 fail

Pershikova M.E. f 1099 n 68 45 47 98 pass

Lomonosov M.V. m 1231 n 100 100 100 100 pass

Kursach N.P. f 1301 y 11 91 83 28 fail

Sidorchuk E.K. f 1211 n 99 89 68 76 pass

# ПРИМЕР РАБОТЫ

artoy@artem-trumpeter  ~/Рабочий стол/kp6   master  make

gcc -g -std=c99 -Wextra -pedantic -Werror -c -o io.o io.c

gcc -g -std=c99 -Wextra -pedantic -Werror -c -o generate.o generate.c

gcc io.o generate.o -o generate

gcc -g -std=c99 -Wextra -pedantic -Werror -c -o print.o print.c

gcc io.o print.o -o print

gcc -g -std=c99 -Wextra -pedantic -Werror -c -o execute.o execute.c

gcc io.o execute.o -o execute

artoy@artem-trumpeter  ~/Рабочий стол/kp6   master ✚  ./generate students.txt database.bin

artoy@artem-trumpeter  ~/Рабочий стол/kp6   master ✚  ./execute database.bin

Фамилия: Pushkin Инициалы: A.S. Пол: m. Есть медаль: n Все оценки: 85 89 100 55 Эссе сдал.

Фамилия: Lomonosov Инициалы: M.V. Пол: m. Есть медаль: n Все оценки: 100 100 100 100 Эссе сдал.

artoy@artem-trumpeter  ~/Рабочий стол/kp6   master ✚  make clean

rm -f execute generate print \*.bin \*.o

# ВЫВОД

Файловая система ANSI С является частью системы ввода/вывода, позволяющей читать или записывать файлы. Она содержит ряд различных функций. В ходе данной работы я познакомился с некоторыми функциями для чтения и записи файлов. Однако, я реализовал достаточно простой алгоритм генерации внешнего, нетекстового файла, что является далеко не плюсом этой программы. Благодаря созданной программе я также на себе прочувствовал основные особенности языка СИ: связь с памятью компьютера и его командной строкой.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гайсарян С.С., Зайцев В.Е. «Курс информатики» Москва, Издательство

МАИ 1993