**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №3.

Системы управления, информатика и электроэнергетика.

Кафедра 308.

**Отчет по лабораторной работе №3**

**по учебной дисциплине «Информатика»**

Группа: *М3О-112Б-19*

Выполнил:

*Степаненко Д.О.*

Принял:

*Корольков Р.С.*

Москва • 2019

**Содержание.**

1. Постановка задачи…………….......................………………………………………………………………………….…….……3
2. Исходный код……………………………………………………………………………………………………………………..……….…4
3. Вывод…………………………………………………………………………………………….....………….……………………………….9

**Постановка задачи.**

Необходимо разработать программу управления базой данных студентов. Под базой данных понимается файл с представлением списка студентов. Программа должна иметь текстовое меню, состоящее как минимум из следующих пунктов:  1. Add student 2. Remove student 3. Print students. При вызове “Add student” подается строка следующего формата: “<name> <age> <gender (M or F)>”. При “Remove student” на вход подается только имя студента, которого надо удалить. При “Print students” на экран выводится текущий массив студентов. Гарантируется, что имя будет уникальным для каждого студента. Число студентов не ограничено. При изменении необходимо сохранять всю информацию о студентах в отдельный файл (в бинарном виде), загружая его при старте программы.   **Исходный код.**

/\*

============================================================================

\*Project Name :thirdlab \*

\*File name :thirdlab.c \*

\*Language :C \*

\*Programmer :Давиденко Екатерина Валерьевна, М3О-114Б-19 \*

\*Created :10.12.2019 \*

============================================================================

\*/

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <stdbool.h>

const char DATABASE\_FILE[255] = "base.bin"; // База данных

const char TRAN\_LOG\_FILE[255] = "change.bin"; // файл для изменений

typedef struct student\_t //создаем структуру с информацией о студентах

{

char name[100]; // Строка имени.

int age; // Возраст - целое число

char gender[2]; // Пол - единственный символ M или F

bool is\_deleted; // Пометка удаления

} student;

size\_t filesize(FILE\* fp) // функция измерения размера файла

{

fseek(fp, 0, SEEK\_END);

long int size = ftell(fp);

fseek(fp, 0, SEEK\_SET);

return size;

}

// Чтение БД или файла изменений.

// Возвращает указатель на массив студентов

// Параметры:

// filename - имя файла

// db\_size - сюда будет записан размер БД в структурах student

// fp\_base - файловый дескриптор.

student\* ReadFile(const char\* filename, size\_t\* db\_size, FILE\*\* fp\_base) {

\*fp\_base = fopen(filename, "a+b"); // открываем файл

if (!(\*fp\_base))

{

printf("Database is invalid!");

return NULL;

}

size\_t db\_volume = filesize(\*fp\_base); // размер файла в байтах

\*db\_size = db\_volume / sizeof(student); // размер БД в структурах "student"

student\* buff = (student\*)malloc(db\_volume); // создаем буфер для всего файла

if (!buff) { // проверяем, нашлась ли память

printf("Memory allocation error!");

return NULL;

}

fread(buff, sizeof(student), \*db\_size, \*fp\_base); // Читаем файл в буфер

return buff;

}

student\* FindStudent(char\* name, student\* buffer, size\_t buffer\_size) {

for (size\_t i = 0; i < buffer\_size; ++i)

{

if (!strcmp(buffer[i].name, name))

{

return &buffer[i];

}

// !strcmp(a,b) = true, когда строки равны

}

return NULL;

}

void ApplyTranLog (student\*\* database, size\_t\* database\_size, FILE\* db\_ptr,

student\* tran\_log, size\_t\* tran\_log\_size, FILE\* tl\_ptr) // Применение накопленных изменений

{

// Выделяем память размером суммы базы и лога транзакций - это максимум, что может быть

// если не было ни одного удаления, а были только добавления

student\* new\_database = (student\*)malloc((\*database\_size + \*tran\_log\_size) \* sizeof(student));

size\_t new\_database\_counter = 0; // Это счётчик записей для объединённой БД

// Записываем содержимое БД за исключением удалений

for (size\_t i = 0; i < \*database\_size; ++i)

{

student\* deleted\_student = FindStudent((\*database)[i].name, tran\_log, \*tran\_log\_size);

if (deleted\_student && deleted\_student->is\_deleted)

{

continue; // Не надо ничего писать. Этот студент удалён.

}

new\_database[new\_database\_counter++] = (\*database)[i];

}

// Дописываем в конец БД новые записи из лога транзакций

for (size\_t i = 0; i < \*tran\_log\_size; i++)

{

if (tran\_log[i].is\_deleted)

{

continue; // Удалённых писать не надо

}

new\_database[new\_database\_counter++] = tran\_log[i];

}

// Освобождаем лишнюю память:

// 1. Сжимаем новую БД до фактического размера

new\_database = (student\*)realloc(new\_database, (new\_database\_counter + 1)\* sizeof(student));

// 2. Старая, считанная в память БД более не нужна. Освобождаем её память

free(\*database);

\*database = new\_database; // Перенацеливаем укзатель

\*database\_size = new\_database\_counter; // Фиксируем размер массива

// Записываем изменённую БД в файл

freopen(DATABASE\_FILE, "wb", db\_ptr);

for (size\_t i = 0; i < new\_database\_counter; i++)

{

fwrite(&new\_database[i], sizeof(student), 1, db\_ptr); //записываем структуру в файл

}

// Очищаем лог транзакций и открываем файл только на запись

freopen(TRAN\_LOG\_FILE, "wb", tl\_ptr);

free(tran\_log);

\*tran\_log\_size = 0;

// Закрываем файл базы данных - БД в памяти

fclose(db\_ptr);

}

int main()

{

size\_t\* db\_size = (size\_t\*)malloc(sizeof(size\_t)); // Создаём указатель на размер считанной базы в studentах;

size\_t\* tl\_size = (size\_t\*)malloc(sizeof(size\_t)); // Создаём указатель на размер лога транзакций в studentах;

FILE\* db\_ptr = NULL; // Сюда мы запишем указатель на файловый дескриптор БД

FILE\* tl\_ptr = NULL; // Здесь будет указатель на файловый дескриптор лога транзакций

student\* database = ReadFile(DATABASE\_FILE, db\_size, &db\_ptr); // Читаем базу данных

student\* tran\_log = ReadFile(TRAN\_LOG\_FILE, tl\_size, &tl\_ptr); // Читаем лог транзакций

ApplyTranLog(&database, db\_size, db\_ptr, tran\_log, tl\_size, tl\_ptr);

while (1)

{

printf("0.Exit\n1.Add Student\n2.Remove Student\n3.Print students\n"); // вывод меню

int i;

student person; // Объявлем перемененую (экземпляр) структуры student

scanf("%d", &i); //считываем действие

if (i == 0) return 0; //выходим из программы, если нажата клавиша 0

if (i == 1) //если нажата 1, добавляем студента

{

while (i != 0)

{

printf("\nEnter name, age and gender(M or F)\n"); //предлагаем ввести имя, возраст и пол

scanf("%s%d%s", person.name, &person.age, person.gender); // cчитываем данные

person.is\_deleted = false;

fwrite(&person, sizeof(student), 1, tl\_ptr); //записываем структуру в файл

if (fflush(tl\_ptr)) { // сбрасываем в файл студента на случай сбоя программы

printf("\nError writing student\n");

return -1;

}

//AddStudentToMemory(database, db\_size, person);

database = (student\*)realloc(database, (\*db\_size + 1) \* sizeof(student)); // Выдеяем память (лучше это делать блоками, но это выходит за рамки задачи)

database[(\*db\_size)++] = person;

printf("\nThe student was added\n\n");

printf("0.Back to program menu\n1.Add another one\n"); //выбор дальнейших действий

scanf("%d", &i);

if (i == 0) break;

}

}

else if (i == 2) //если нажата 2, удаляем студента

{

printf("Enter the student name\n");

char name[100]; // Переменная для поиска

scanf("%s", name);

student\* cur1 = FindStudent(name, database, \*db\_size); // указатель на студента

if (!cur1) //если студент не найден, выходим из цикла

{

printf("\nThe student was not found.\n");

i = 0;

continue;

}

(\*cur1).is\_deleted = true;

fwrite(cur1, sizeof(student), 1, tl\_ptr); //записываем структуру в файл

if (fflush(tl\_ptr)) { // сбрасываем в файл запись об удалении на случай сбоя программы

printf("\nError writing student\n");

return -2;

}

} else if (i == 3)

{

for (size\_t i = 0; i < \*db\_size; i++)

{

if (!database[i].is\_deleted) {

printf("Name: %s \t Age: %d \t Gender: %s\n", database[i].name, database[i].age, database[i].gender);

}

}

i = 0;

}

}

fclose(tl\_ptr);

return 0;

}

**Вывод:**

Для создания программы на языке Си использовалась среда разработки codeblocks. Во время написания программы изучила работу с файлами на языке си, работу с указателями, написание функций.