МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт№8: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**ПРАКТИКУМ**

По циклу дисциплин «Информатика»

2 семестр

Тема:

«Обработка последовательной файловой структуры на языке Си»

|  |  |
| --- | --- |
| Группа: | М8О-106Б-22 |
| Студент: | Абдисаламов Э. |
| Преподаватель: | Дубинин А. В. |
| Оценка: |  |
| Дата: |  |

Москва, 2023

Содержание

[Введение 4](#_Toc136868368)

[1. Теоретическая часть 5](#_Toc136868369)

[1.1 Системные вызовы ввода-вывода в C 5](#_Toc136868370)

[Важная терминология 5](#_Toc136868371)

[Системные вызовы ввода-вывода 7](#_Toc136868372)

[1. C создать 7](#_Toc136868373)

[Синтаксис create() в C 7](#_Toc136868374)

[Параметр 7](#_Toc136868375)

[Возвращаемое значение 8](#_Toc136868376)

[Как C create() работает в OS 8](#_Toc136868377)

[2. C открыть 8](#_Toc136868378)

[Синтаксис open() в C 8](#_Toc136868379)

[Параметры 8](#_Toc136868380)

[Как C open() работает в OS 9](#_Toc136868381)

[3. C закрыть 9](#_Toc136868382)

[Синтаксис close() в C 9](#_Toc136868383)

[Параметр 10](#_Toc136868384)

[Возвращаемое значение 10](#_Toc136868385)

[Как C close() работает в операционной системе 10](#_Toc136868386)

[4. Чтение на C 10](#_Toc136868387)

[Синтаксис read() в C 10](#_Toc136868388)

[Параметры 10](#_Toc136868389)

[Возвращаемое значение 10](#_Toc136868390)

[Важные моменты 10](#_Toc136868391)

[5. Запись на C 11](#_Toc136868392)

[Синтаксис write() в C 11](#_Toc136868393)

[Параметры 11](#_Toc136868394)

[Возвращаемое значение 11](#_Toc136868395)

[Важные моменты о C write 11](#_Toc136868396)

[1.2 Бианрный файл 12](#_Toc136868397)

[2. Практическая часть 13](#_Toc136868398)

[2.1 Задание 13](#_Toc136868399)

[2.2. Вариант содержимого и структуры файла 13](#_Toc136868400)

[2.3 Действия: 13](#_Toc136868401)

[2.5 Протокол 13](#_Toc136868402)

[2.6 Оценка сложности функций 26](#_Toc136868403)

[Заключение 27](#_Toc136868404)

[Список использованных источников 28](#_Toc136868405)

# Введение

Файловые дескрипторы в языке Си играют важную роль при работе с файлами. Они представляют числовые значения, используемые для идентификации открытых файлов. Файловые дескрипторы позволяют программам взаимодействовать с файловой системой, читать данные из файлов, записывать информацию и выполнять другие операции.

На языке Си существуют функции, такие как `fopen()`, `fread()` и `fwrite()`, которые позволяют открывать файлы, читать и записывать данные. Корректное закрытие файлов с помощью функции `fclose()` также является важным аспектом работы с файлами в C. В данной работе рассмотрим основные операции с файлами на языке Си, используя файловые дескрипторы и функции стандартной библиотеки.

# 1. Теоретическая часть

## 1.1 Системные вызовы ввода-вывода в C

Системные вызовы — это вызовы, которые программа выполняет к системному ядру для предоставления служб, к которым у программы нет прямого доступа. Например, предоставление доступа к устройствам ввода и вывода, таким как мониторы и клавиатуры. Мы можем использовать различные функции, предоставляемые на языке программирования C, для системных вызовов ввода-вывода, таких как create, open, read, write и т.д.

Прежде чем мы перейдем к системным вызовам ввода-вывода, нам нужно знать о нескольких важных терминах.

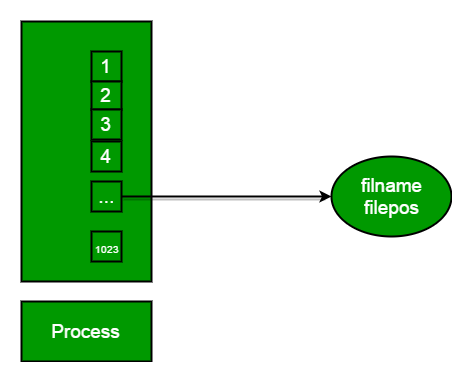
### Важная терминология

**Что такое файловый дескриптор?**

Файловый дескриптор - это целое число, которое однозначно идентифицирует открытый файл процесса.

**Таблица файловых дескрипторов: Таблица файловых** дескрипторов представляет собой набор индексов массива целых чисел, которые являются файловыми дескрипторами, элементы которых являются указателями на записи файловой таблицы. В операционной системе для каждого процесса предусмотрена одна уникальная таблица файловых дескрипторов.

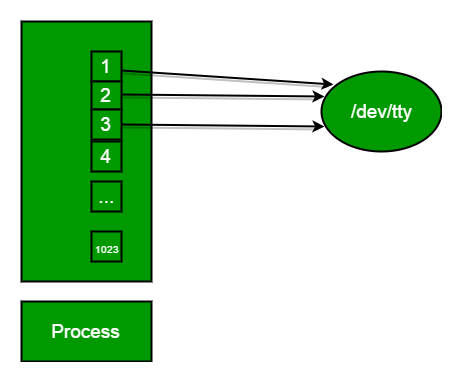
**Запись в файловой таблице:** Записи файловой таблицы представляют собой структуру в памяти, заменяющую открытый файл, которая создается при обработке запроса на открытие файла, и эти записи сохраняют положение файла.



**Стандартные файловые дескрипторы**: при запуске любого процесса автоматически открываются fd (файловый дескриптор) таблицы файловых дескрипторов этого процесса 0, 1, 2 (по умолчанию), каждый из этих 3-х fd ссылается на запись файловой таблицы для файла с именем **/dev/tty**

**/dev/tty**: суррогат терминала в памяти.

**Терминал**: комбинированная клавиатура / видеоэкран.



**Чтение из stdin => чтение из fd 0**: Всякий раз, когда мы вводим какой-либо символ с клавиатуры, он считывается из stdin через fd 0 и сохраняется в файл с именем /dev/tty.  
**Запись в стандартный вывод => запись в fd 1**: Всякий раз, когда мы видим какой-либо вывод на видеоэкран, он из файла с именем /dev / tty и записывается в стандартный вывод на экране через fd 1.  
**Запись в stderr => запись в fd 2**: Мы видим любую ошибку на экране видео, она также из этого файла, записываемого в stderr на экране через fd 2.

## **Системные вызовы ввода-вывода**

В принципе, существует всего 5 типов системных вызовов ввода-вывода:

## C создать

Функция create() используется для создания нового пустого файла на C. Мы можем указать разрешение и имя файла, который мы хотим создать, используя функцию create(). Он определен внутри **файла заголовка<unistd.h>**, а флаги, которые передаются в качестве аргументов, определены внутри **файла заголовка<fcntl.h>**.

### **Синтаксис create() в C**

int **create**(char \**filename*, mode\_t *mode*);

### **Параметр**

* **filename:** имя файла, который вы хотите создать
* **режим:** указывает разрешения для нового файла.

### **Возвращаемое значение**

* возвращают первый неиспользуемый файловый дескриптор (обычно 3 при первом создании используются в процессе, потому что зарезервированы 0, 1, 2 fd)
* возвращают -1 при ошибке

### **Как C create() работает в OS**

* Создайте новый пустой файл на диске.
* Создайте запись в файловой таблице.
* Установите первый неиспользуемый дескриптор файла так, чтобы он указывал на запись файловой таблицы.
* Возвращает используемый файловый дескриптор, -1 при сбое.

## **C открыть**

Функция open() в C используется для открытия файла для чтения, записи или для того и другого. Он также способен создать файл, если он не существует. Он определен внутри **файла заголовка<unistd.h>**, а флаги, которые передаются в качестве аргументов, определены внутри **файла заголовка<fcntl.h>**.

### **Синтаксис open() в C**

int **open** (const char\* *Path*, int *flags*);

### **Параметры**

* **Path:** Путь к файлу, который мы хотим открыть.
  + Используйте **абсолютный путь**, начинающийся с “/”, когда вы **не** **работаете в том же каталоге**, что и исходный файл C.
  + Используйте**относительный путь**, который является только именем файла с расширением, когда вы**работаете в том же каталоге**, что и исходный файл C.
* **флаги:**используется для указания того, как вы хотите открыть файл. Мы можем использовать следующие флаги.

| **Флаги** | **Описание** |
| --- | --- |
| **O\_RDONLY** | Открывает файл в режиме только для чтения. |
| **O\_WRONLY** | Открывает файл в режиме только для записи. |
| **O\_RDWR** | Открывает файл в режиме чтения и записи. |
| **O\_CREAT** | Создайте файл, если он не существует. |
| **O\_EXCL** | Запретить создание, если оно уже существует. |
| **O\_ ДОБАВИТЬ** | Открывает файл и помещает курсор в конец содержимого. |
| **O\_ASYNC** | Включить управление вводом и выводом с помощью сигнала. |
| **O\_CLOEXEC** | Включите режим закрытия при выполнении для открытого файла. |
| **O\_NONBLOCK** | Отключает блокировку открытого файла. |
| **O\_TMPFILE** | Создайте неназванный временный файл по указанному пути. |

### **Как C open() работает в OS**

* Найдите существующий файл на диске.
* Создайте запись в файловой таблице.
* Установите первый неиспользуемый дескриптор файла так, чтобы он указывал на запись файловой таблицы.
* Возвращает используемый файловый дескриптор, -1 при сбое.

## **C закрыть**

Функция close() в C сообщает операционной системе, что работа с файловым дескриптором завершена, и закрывает файл, на который указывает файловый дескриптор. Он определен внутри**файла заголовка <unistd.h>**.

### Синтаксис close() в C

int close(int fd);

### **Параметр**

* **fd:**дескриптор файла, который вы хотите закрыть.

### **Возвращаемое значение**

* **0** в случае успеха.
* **-1** при ошибке.

### **Как C close() работает в операционной системе**

* Уничтожить запись файловой таблицы, на которую ссылается элемент fd таблицы файловых дескрипторов  
  – Пока ни один другой процесс не указывает на это!
* Установите для элемента fd таблицы файловых дескрипторов значение **NULL**

## **Чтение на C**

Из файла, указанного файловым дескриптором fd, функция read() считывает указанное количество байт, **cnt** входных данных, в область памяти, указанную с помощью **buf**. Успешное чтение () обновляет время доступа к файлу. Функция read() также определена внутри файла заголовка <unistd.h>.

### Синтаксис read() в C

size\_t **read** (int *fd*, void\* *buf*, size\_t *cnt*);

### **Параметры**

* **fd:** файловый дескриптор файла, из которого должны быть считаны данные.
* **buf:** буфер для чтения данных из
* **cnt:** длина буфера

### **Возвращаемое значение**

* возвращают количество байтов, прочитанных при успешном выполнении
* возвращают 0 при достижении конца файла
* возвращает -1 при ошибке
* возвращает -1 при прерывании сигнала

### **Важные моменты**

* **buf** должен указывать на допустимую ячейку памяти длиной не меньше указанного размера из-за переполнения.
* **fd** должен быть допустимым файловым дескриптором, возвращаемым из open() для выполнения операции чтения, потому что если fd равно нулю, то чтение должно выдать ошибку.
* **cnt** - это запрошенное количество прочитанных байт, в то время как возвращаемое значение - это фактическое количество прочитанных байт. Кроме того, в некоторых случаях системный вызов read должен считывать меньше байтов, чем cnt.

## **Запись на C**

Записывает байты cnt из buf в файл или сокет, связанный с fd. значение cnt не должно быть больше INT\_MAX (определено в пределах.файл заголовка h). Если cnt равно нулю, write() просто возвращает 0, не предпринимая никаких других действий.

Функция write() также определена внутри **файла заголовка <unistd.h>**.

### Синтаксис write() в C

size\_t **write** (int *fd*, void\* *buf*, size\_t *cnt*);

### **Параметры**

* **fd:** дескриптор файла
* **buf:** буфер для записи данных.
* **cnt:** длина буфера.

### **Возвращаемое значение**

* возвращает количество байтов, записанных при успешном выполнении.
* возвращают 0 при достижении конца файла.
* возвращают -1 при ошибке.
* возвращают значение -1 при прерывании сигнала.

### **Важные моменты о C write**

* Файл необходимо открыть для операций записи
* **Длина буфера**должна быть как минимум такой же, как указано в cnt, потому что если размер буфера меньше cnt, то buf приведет к состоянию переполнения.
* **cnt** - это запрошенное количество байт для записи, в то время как возвращаемое значение - это фактическое количество записанных байт. Это происходит, когда у **fd** меньше байтов для записи, чем у cnt.
* Если write() прерывается сигналом, результатом является одно из следующих:
  + Если write() еще не записала никаких данных, она возвращает -1 и присваивает errno значение EINTR.
  + Если функция write() успешно записала некоторые данные, она возвращает количество байтов, которое было записано до того, как оно было прервано.

## 1.2 Бианрный файл

Бинарный файл — это вид файлов, которые хранятся внутри компьютера. В таких файлах находится информация о часто используемых программах. Таким образом удается быстрее запускать эти программы, потому что нет необходимости полностью их преобразовывать. Напомним, что компьютер прекрасно понимает только бинарный формат файлов, поэтому каждая программа компилируется и преобразуется в бинарный формат, чтобы компьютер смог ее запустить.

Компьютер понимает бинарный формат, однако люди — нет, поэтому они придумали себе текстовые форматы. Их легче понять, ими легче манипулировать и легче переносить. Как мы уже писали, каждая компьютерная программа, написанная на любом языке программирования, представляет собой текстовый файл.

2. Практическая часть

2.1 Задание

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, содержащего представительный набор записей (не менее 20). Распечатать содержимое сгенерированного файла в виде таблицы и выполнить над ним заданное действие для 3–5 значений параметров запроса p и распечатать результат.

2.2. Вариант содержимого и структуры файла

Сведения о вступительных экзаменах абитуриентов: №, ID, общая сумма баллом, балл по химии ДВИ, балл по химии ЕГЭ, балл по математике ЕГЭ, балл по биологии ЕГЭ, балл по русскому языку, индивидуальные достижения, право на общежитие.

2.3 Действия:

1. Список абитуриентов с правом на общежитие, средняя оценка которых находится в указанном диапазоне.

2. Сортировка по баллам по указанному предмету.

## 2.5 Протокол

* Data\_funcs.h:

#pragma once

#include <stdbool.h>

typedef struct{

    int idx;

    \_\_int64\_t id;

    int total\_sum;

    int chemistry\_dvi;

    int chemistry\_ege;

    int bio;

    int math;

    int rus;

    int ind\_adv;

    bool dormitory\_rights;

} database\_line;

void print\_header();

void print\_db\_line(database\_line\* db\_line);

void print\_end\_table();

void print\_table(FILE\* database);

void print\_dorm\_eligible\_students\_by\_avg\_grade\_in\_range(FILE\* database, int a, int b);

void print\_sorted\_by\_math(FILE\* database);

void print\_sorted\_by\_rus(FILE\* database);

void print\_sorted\_by\_him\_dvi(FILE\* database);

void print\_sorted\_by\_him\_ege(FILE\* database);

void print\_sorted\_by\_bio(FILE\* database);

* Data\_funcs.c:

#include <stdio.h>

#include <string.h>

#include "data\_funcs.h"

#include "hoar\_sort.h"

// Найти абитуриентов с средной оценкой в указанном диапозоне с правом на общежитие

// Сортировка по значениям

void print\_header(){

    printf("%-2s\t%-12s\t%-10s\t%-9s\t%-9s\t%-12s\t%-14s\t%-16s\t%-25s\t%-17s\n",

    "№", "ID", "Общий балл", "Химия ДВИ", "Химия ЕГЭ", "Биология ЕГЭ", "Математика ЕГЭ", "Русский язык ЕГЭ", "Индивидуальные достижения", "Право на общежитие");

}

void print\_db\_line(database\_line\* db\_line){

    char dormitory\_rights[20] = "Нет"; // Право на общежитие

    if (db\_line->dormitory\_rights) strcpy(dormitory\_rights, "Да");

    printf("%-2d\t%-12ld\t%-10d\t%-9d\t%-9d\t%-12d\t%-14d\t%-16d\t%-25d\t%-17s\n",

    db\_line->idx, db\_line->id, db\_line->total\_sum, db\_line->chemistry\_dvi, db\_line->chemistry\_ege, db\_line->bio, db\_line->math, db\_line->rus, db\_line->ind\_adv, dormitory\_rights);

}

void print\_end\_table(){

    printf("%-2s\t%-12s\t%-10s\t%-9s\t%-9s\t%-12s\t%-14s\t%-16s\t%-25s\t%-17s\n",

    "--", "------------", "----------", "---------", "---------", "------------", "-------------", "---------------", "------------------------", "----------------");

}

void print\_table(FILE\* database){

    fseek(database, 0, SEEK\_SET);

    print\_header();

    database\_line cur\_db\_line;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        print\_db\_line(&cur\_db\_line);

    }

    print\_end\_table();

}

void print\_dorm\_eligible\_students\_by\_avg\_grade\_in\_range(FILE\* database, int a, int b){

    bool found = false;

    fseek(database, 0, SEEK\_SET);

    database\_line cur\_db\_line;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        int avg\_grade = cur\_db\_line.total\_sum / 5;

        if ((avg\_grade >= a) && (avg\_grade <= b) && (cur\_db\_line.dormitory\_rights)) {

            print\_db\_line(&cur\_db\_line);

            if(!found) found = true; // возможно лучше убрать if?

        }

    }

    if (!found) printf("Таких не найдено\n");

}

void print\_database\_by\_arr(FILE\* database, elem\* arr, int len\_arr){

    print\_header();

    database\_line cur\_db\_line;

    elem cur\_elem;

    for (int i = 0; i < len\_arr; i++){

        cur\_elem = arr[i];

        // printf("###%d - %d\n", cur\_elem.number, cur\_elem.key);

        int target\_rows\_number = cur\_elem.number - 1;

        // Перемещаем указатель файла к нужной позиции

        fseek(database, target\_rows\_number \* sizeof(database\_line), SEEK\_SET);

        // Считываем число из файла

        fread(&cur\_db\_line, sizeof(database\_line), 1, database);

        print\_db\_line(&cur\_db\_line);

    }

    print\_end\_table();

}

int count\_rows\_in\_file(FILE\* file, size\_t record\_size) {

    fseek(file, 0, SEEK\_END);

    long file\_size = ftell(file);

    rewind(file);

    int count = file\_size / record\_size;

    return count;

}

void print\_arr\_elem(elem \*arr, int len\_arr){

    elem cur\_elem;

    for (int i = 0; i < len\_arr; i++){

        cur\_elem = arr[i];

        printf("N%-2d - %-2d\n", cur\_elem.number, cur\_elem.key);

    }

}

void print\_sorted\_by\_math(FILE\* database)

{

    fseek(database, 0, SEEK\_SET);

    elem arr\_elem[count\_rows\_in\_file(database, sizeof(database\_line))];

    int len\_arr = 0;

    database\_line cur\_db\_line;

    elem cur\_elem;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        cur\_elem = (elem){cur\_db\_line.math ,cur\_db\_line.idx};

        arr\_elem[len\_arr] = cur\_elem;

        len\_arr++;

    }

    hoar\_sort(arr\_elem, len\_arr);

    print\_database\_by\_arr(database, arr\_elem, len\_arr);

}

void print\_sorted\_by\_rus(FILE\* database)

{

    fseek(database, 0, SEEK\_SET);

    elem arr\_elem[count\_rows\_in\_file(database, sizeof(database\_line))];

    int len\_arr = 0;

    database\_line cur\_db\_line;

    elem cur\_elem;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        cur\_elem = (elem){cur\_db\_line.rus ,cur\_db\_line.idx};

        arr\_elem[len\_arr] = cur\_elem;

        len\_arr++;

    }

    hoar\_sort(arr\_elem, len\_arr);

    print\_database\_by\_arr(database, arr\_elem, len\_arr);

}

void print\_sorted\_by\_him\_dvi(FILE\* database)

{

    fseek(database, 0, SEEK\_SET);

    elem arr\_elem[count\_rows\_in\_file(database, sizeof(database\_line))];

    int len\_arr = 0;

    database\_line cur\_db\_line;

    elem cur\_elem;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        cur\_elem = (elem){cur\_db\_line.chemistry\_dvi ,cur\_db\_line.idx};

        arr\_elem[len\_arr] = cur\_elem;

        len\_arr++;

    }

    hoar\_sort(arr\_elem, len\_arr);

    print\_database\_by\_arr(database, arr\_elem, len\_arr);

}

void print\_sorted\_by\_him\_ege(FILE\* database)

{

    fseek(database, 0, SEEK\_SET);

    elem arr\_elem[count\_rows\_in\_file(database, sizeof(database\_line))];

    int len\_arr = 0;

    database\_line cur\_db\_line;

    elem cur\_elem;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        cur\_elem = (elem){cur\_db\_line.chemistry\_ege ,cur\_db\_line.idx};

        arr\_elem[len\_arr] = cur\_elem;

        len\_arr++;

    }

    hoar\_sort(arr\_elem, len\_arr);

    print\_database\_by\_arr(database, arr\_elem, len\_arr);

}

void print\_sorted\_by\_bio(FILE\* database)

{

    fseek(database, 0, SEEK\_SET);

    elem arr\_elem[count\_rows\_in\_file(database, sizeof(database\_line))];

    int len\_arr = 0;

    database\_line cur\_db\_line;

    elem cur\_elem;

    while (fread(&cur\_db\_line, sizeof(database\_line), 1, database)){

        cur\_elem = (elem){cur\_db\_line.bio ,cur\_db\_line.idx};

        arr\_elem[len\_arr] = cur\_elem;

        len\_arr++;

    }

    hoar\_sort(arr\_elem, len\_arr);

    print\_database\_by\_arr(database, arr\_elem, len\_arr);

}

* kp6.c:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "data\_funcs.h"

#include "to\_binfile.h"

#define HELPTEXT "\

Данная программа работает с данными из таблицы с данными поступающих.\n\

Для получения желаемого результата нужно запускать с ключом.\n\

Основные ключи:\n\

    \033[35m--help\033[m, получить справку по программе\n\

    \033[35m--tobin binfilepath+name\033[m, преобразовать файл в бинарный\n\

    \033[35m--sort math\033[m, получить сортированный список по баллам математика ЕГЭ\n\

    \033[35m       him dvi\033[m,                            по баллам химия ДВИ\n\

    \033[35m       him ege\033[m,                            по баллам химия ЕГЭ\n\

    \033[35m       bio\033[m,                                по баллам биология ЕГЭ\n\

    \033[35m       rus\033[m,                                по баллам русский язык ЕГЭ\n\

    \033[35m--range a b\033[m, вывести список абитуриентов с правом на общежитие, средний балл которых находиться в [a;b].\n\

            a b - задаваемые числа числа\n\

    \n\

    Все ключи используются по отдельности\n\

    Передаваемый файл может быть тектовым или бинарным. Если файл не бинарный, то создатся бинарный в рабочей директории.\n\

    Чтобы запустить программу, Вам необходимо написать: \033[35m./kp6 --help\033[m и вы получите данное сообщение.\n\

    Чтобы просто преобразовать файл в бинарный без взаимодейтвия с данными, используйте ключ --tobin, например: \033[35m./kp6 data.txt --tobin data.bin\033[m\n\

    Для работы с данными из таблицы используйте ключи, например: \033[35m./kp6 data.txt --sort math\033[m\n\

    \n\

    Формат таблицы: \"№\" \"ID\" \"Общий балл\" \"Химия ДВИ\" \"Химия ЕГЭ\" \"Биология ЕГЭ\" \"Математика ЕГЭ\" \"Русский язык ЕГЭ\" \"Индивидуальные достижения\" \"Право на общежитие(\"Да\"/\"Нет\")\"\n\

"

#define tmp\_database\_name "database.bin"

bool is\_binary\_file(const char\* file\_path);

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

    if ((argc <= 1) || (argc > 5))

    {

        printf("Ошибка запуска, используйте ключ --help, чтобы получить справку о программе\n");

        exit(1);

    }

    else if (strcmp(argv[1], "--help") == 0)

    {

        printf(HELPTEXT);

        exit(0);

    }

    if ((argv[2]) == NULL)// чтобы не было дальнейшех сравнений, при котором будет Segmentation fault

    {

        printf("Ошибка запуска, используйте ключ --help или запустите без ключей, чтобы получить справку о программе\n");

        exit(1);

    }

    else if (strcmp(argv[2], "--tobin") == 0)

    {

        if (argc > 4) {

            printf("Ошибка запуска, используйте ключ --help или запустите без ключей, чтобы получить справку о программе\n");

            exit(1);

        }

        file\_to\_bin(argv[1], argv[3]);

        exit(0);

    }

    else if ((strcmp(argv[2], "--range") == 0) || (strcmp(argv[2], "--sort") == 0))

    {

        FILE\* database = NULL;

        if (!is\_binary\_file(argv[1])) {

            file\_to\_bin(argv[1], tmp\_database\_name);

            database = fopen(tmp\_database\_name, "rb");

        }

        else {

            database = fopen(argv[1], "rb");

        }

        if (database == NULL) {

            printf("Cannot open database file\n");

            fclose(database);

            exit(2);

        }

        print\_table(database);

        if (strcmp(argv[2], "--range") == 0)

        {

            if (argv[3] == NULL || argv[4] == NULL){

                printf("Error: No such parameters keys --range\n");

                exit(3);

            }

            printf("Результат работы программы:\n");

            print\_dorm\_eligible\_students\_by\_avg\_grade\_in\_range(database, atoi(argv[3]), atoi(argv[4]));

        }

        else

        {

            if (argv[3] == NULL){

                printf("Error: No such parameters keys --sort\n");

                exit(4);

            }

            printf("Результат сортировки:\n");

            if(strcmp(argv[3], "math") == 0){

                print\_sorted\_by\_math(database);

            }

            else if (strcmp(argv[3], "rus") == 0){

                print\_sorted\_by\_rus(database);

            }

            else if (strcmp(argv[3], "bio") == 0){

                print\_sorted\_by\_bio(database);

            }

            else if (strcmp(argv[3], "him") == 0){

                if (strcmp(argv[4], "dvi") == 0){

                    print\_sorted\_by\_him\_dvi(database);

                }

                else if (strcmp(argv[4], "ege") == 0) print\_sorted\_by\_him\_ege(database);

                else {

                    printf("Error: No such parameters keys --sort\n");

                    exit(4);

                }

            }

            else {

                printf("Error: No such parameters keys --sort\n");

                exit(4);

            }

        }

        fclose(database);

    }

    else

    {

        printf("Ошибка запуска, используйте ключ --help или запустите без ключей, чтобы получить справку о программе\n");

        exit(0);

    }

    return 0;

}

bool is\_binary\_file(const char\* file\_path) {

    FILE\* file = fopen(file\_path, "rb");

    if (file == NULL) {

        printf("Error open %s\n", file\_path);

        exit(1);

    }

    bool is\_binary = false;

    int ch;

    while ((ch = fgetc(file)) != EOF) {

        if (ch == '\0') {

            is\_binary = true;

            break;

        }

    }

    fclose(file);

    return is\_binary;

}

* to\_binfile.h

#pragma once

int file\_to\_bin(const char\* input\_file, const char\* output\_file);

* to\_binfile.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include "data\_funcs.h"

void file\_to\_bin(const char\* input\_file, const char\* output\_file)

{

    FILE \*input = fopen(input\_file, "rb");

    FILE \*output = fopen(output\_file, "wb");

    if ((input == NULL) || (output == NULL)) // Ошибка открытия

    {

        printf("Error open\n");

        exit(1);

    }

    database\_line db\_line;

    char dormitory\_rights[20]; // Право на общежитие

    while (fscanf(input, "%d  %ld  %d  %d  %d  %d  %d  %d  %d  %s\n",

     &db\_line.idx, &db\_line.id, &db\_line.total\_sum, &db\_line.chemistry\_dvi, &db\_line.chemistry\_ege,

     &db\_line.bio, &db\_line.math, &db\_line.rus, &db\_line.ind\_adv, dormitory\_rights) != EOF){

        db\_line.dormitory\_rights = false;

        if (strcmp(dormitory\_rights, "Да") == 0){

            db\_line.dormitory\_rights = true;

        }

        int count\_successfully\_written = fwrite(&db\_line, sizeof(database\_line), 1, output);

        if (count\_successfully\_written != 1) {

            printf("Error writing\n");

            exit(2);

        }

    }

    fclose(input);

    fclose(output);

}

2.6 Оценка сложности функций

n – кол-во строк в рабочем файле

file\_to\_bin – O(n)

print\_dorm\_eligible\_students\_by\_avg\_grade\_in\_range – O(n)

count\_rows\_in\_file – O(1)

print\_sorted\_by\_math – O(nlogn)

# Заключение

В данной работе мы изучили процесс перевода обычного файла в бинарный формат и выполнения различных действий с ним на языке C. Мы использовали файловые дескрипторы и функции стандартной библиотеки для работы с файлами.

В целом, работа с бинарными файлами на языке C может быть мощным инструментом для сохранения и обработки данных. Понимание работы с файловыми дескрипторами и функциями работы с файлами позволяет программистам эффективно работать с файловой системой и выполнять различные операции над файлами

# Список использованных источников

1. <https://translated.turbopages.org/proxy_u/en-ru.ru.71c9bba8-647db39f-fe66ae6b-74722d776562/https/www.geeksforgeeks.org/input-output-system-calls-c-create-open-close-read-write/>
2. https://learnc.info/c/binary\_files.html