МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

### федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

(ГУАП)

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

Преподаватель

канд. техн. наук, доцент Л.Н. Бариков

Отчёт

по лабораторной работе №13

по дисциплине ОСНОВЫ ПРОГРАММИРОВАНИЯ

на тему: «Структуры и файлы структур»

Работу выполнил

студент гр. 4941 Н.С. Горбунов

Санкт-Петербург

2020

**Цель лабораторной работы:** изучение структурной организации, способов доступа к элементам и других особенностей структур и файлов структур; изучение стандартных средств языка C/C++ для работы с бинарными файлами; совершенствование навыков процедурного программирования на языке C/C++ при решении задач обработки файлов.

**Задание на программирование:** используя технологию процедурного программирования разработать программу обработки бинарных файлов структур с числом записей не менее пяти в соответствии с индивидуальным заданием.

***Порядок выполнения работы:***

1. Получить у преподавателя индивидуальное задание.

2. Составить описание процесса решения задачи с использованием функций создания, добавления новых элементов, просмотра, сортировки бинарного файла по числовому и строковому полю, поиска данных в файле по ключу.

3. Сформулировать условие поиска данных в файле и организовать поиск по условию с сохранением найденных записей в новом файле.

4. Составить программу на языке *C*/*C*++.

5. Предусмотреть в программе возможность выбора варианта действия с помощью меню.

6. Проверить и продемонстрировать преподавателю работу программы на полном наборе тестов.

7. Оформить отчет о лабораторной работе в составе: постановка задачи, описание процесса решения задачи, текст программы, контрольные примеры (скриншоты).

**Вариант 13**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **13. Поликлиники** |  |  |  |  |
| ФИО | Специализация | № кабинета | Время начала | День |
| врача |  |  | приёма | недели |

**Математическая модель**

При решении задачи, а значит в тексте программы, кроме функции *main*() необходимо реализовать следующие функции:

- функцию проверки существования физического файла с конкретным именем;

- функцию создания исходного файла структур (базы данных) в соответствии с заданием;

- функцию добавления новых записей в существующий файл;

- функцию сортировки содержимого бинарного файла по значению числового поля;

- функцию сортировки содержимого бинарного файла по значению строкового поля;

- функцию поиска записей в файле по значению строкового поля с созданием файла результатов поиска;

- функции вывода содержимого файла базы данных и файла результатов поиска на экран.

Имена исходного файла и файла результатов поиска задаём константными строками. При необходимости у пользователя будет возможность создать файл с любым именем.

Поскольку работа будет идти с бинарными файлами, то для записи в файл будет использоваться функция *fwrite*(), а для чтения – функция *fread*().

Исполнение программы (функция *main*()) начинается с объявления и инициализации символьных массивов (строк), которые будут использоваться для хранения имён исходного *fname\_i* и файла результатов поиска *fname\_r*.

На экран выводится меню программы, и пользователю предлагается сделать выбор вида очередного действия (выбор конкретного пункта меню).

При выборе пункта 1 – «создание базы данных» - начинается создание бинарного файла структур, состав полей которых соответствует заданию. Этот процесс начинается с вызова функции проверки существования физического файла с указанным именем создаваемого исходного файла *proverka\_file*(). В этой функции делается попытка открытия физического файла с этим именем на чтение. Если такой файл уже существует (если он открылся на чтение), функция возвращает значение 1, в противном случае – 0.

Если физический файл с указанным именем уже существует, то пользователю об этом выдаётся сообщение «Файл базы данных врачей с именем [ ] был создан раньше» и предлагается либо задать имя создаваемого файла другой базы данных, либо вернуться в меню программы. Результатом такого диалога является либо имя создаваемого исходного файла, которое хранится в строке *fname\_i*, либо возврат в меню программы.

Если файл создаётся, то после этого следует оператор открытия создаваемого файла с именем *fname\_i* на запись. Затем вызывается функция *dobavka*(), и пользователь вводит значения полей структуры, которая будет очередной записью создаваемого файла, по выводимым на экран подсказкам. После задания значений всех полей очередной структуры с описанием конкретного врача вводится новая запись фиксированного размера (этот размер определяется составом и типами полей структуры) в создаваемый бинарный файл с использованием функции *fwrite*().

Размер создаваемого файла при открытии не оговаривается, поэтому пользователю после ввода очередной записи в файл выводится запрос на продолжение или прекращение процесса создания файла («ввод с переспросом»).

При завершении процесса создания файла он закрывается с использованием функции *fclose*(), и происходит возврат в меню программы с выводом сообщения «Создание файла закончено. База данных готова к работе».

При выборе пункта 2 – «добавление новых записей» - все действия, выполняемые по пункту 1 меню программы, повторяются за исключением того, что добавление новых записей ведется в ранее созданный файл, имя которого задаёт пользователь. В этом случае этот существующий файл открывается на добавление новых записей в конец файла. При завершении процесса добавления новых записей происходит возврат в меню программы с выводом сообщения «Изменение файла закончено».

При выборе пункта 3 – «сортировка по ФИО врача» - пользователь переходит к процессу сортировки записей существующего файла по алфавиту содержимого одного из полей (сортировка по значению строкового поля). Эти действия можно осуществлять только для существующего физического файла. Поэтому вначале вызывается функция проверки существования физического файла с указанным именем *proverka\_file*(). В этой функции делается попытка открытия физического файла с этим именем на чтение и запись. Если файл не существует – возврат в меню программы с выводом сообщения об ошибке открытия файла. Если файл существует – происходит вызов функции *sort\_name*().

В этой функции сортируемый файл открывается на чтение и запись. Открывается цикл сортировки *do..while*(*fl*), выполнение которого идет до тех пор, пока процесс сортировки необходимо продолжать. Файловый указатель устанавливается на запись с номером 0. Переменная-флаг *fl* принимает значение 0 (дальнейшая сортировка не требуется). Открывается цикл попарного считывания и сравнения значений полей двух соседних структур (записей) от начала файла до его конца. Поскольку сортировка выполняется по значению строкового поля, сравнение значений производится с использованием функции *strncmp*().

Если после сравнения значений выявлено, что записи в файле стоят неправильно, файловый указатель поднимается на две записи вверх, и в сортируемый файл идет запись вначале второй считанной записи, а потом – первой. Так записи в файле меняются местами. При этом флаг необходимости сортировки *fl* принимает значение 1 (цикл дальнейшей сортировки необходимо продолжать).

Если после сравнения значений выявлено, что записи в файле стоят правильно, файловый указатель поднимается на одну запись вверх, и считывается новая пара соседних записей. Это продолжается до достижения конца файла.

Если после просмотра и сравнения значений полей структур всех соседних записей файла выявлено, что была хотя бы одна перестановка записей в файле (значение флага необходимости продолжения процесса сортировки равно 1), то цикл сортировки *do..while*(*fl*) выполняется вновь. Если перестановок не было – цикл сортировки успешно завершен.

При завершении процесса сортировки записей происходит возврат в меню программы с выводом сообщения «Сортировка по ФИО врача закончена».

При выборе пункта 4 – «сортировка по номеру кабинета» - выполняются все те же действия, что и в пункте 3 меню. Отличие заключается в том, что в этом случае вызывается функция *sort\_room*() и сравнение значений числовых полей структур осуществляется с использованием операции >.

При завершении процесса сортировки записей происходит возврат в меню программы с выводом сообщения «Сортировка по номеру кабинета закончена».

При выборе пункта 5 – «поиск врачей по специализации» - пользователь переходит к процессу поиска записей существующего файла по значению задаваемого им ключа поиска (названием специализации) и формированию файла выборки записей. Эти действия можно осуществлять только для существующего физического файла с базой туров. Файл результатов выборки создаётся новый.

Поэтому вначале вызывается функция проверки существования физического файла с указанным именем *proverka\_file*(). В этой функции делается попытка открытия физического файла с этим именем на чтение. Если файл не существует – возврат в меню программы с выводом сообщения об ошибке открытия файла на чтение.

Затем делается попытка открытия создаваемого физического файла выборки с указанным именем на запись. Если при этом произошла ошибка (например, задано неверное имя файла) - возврат в меню программы с выводом сообщения об ошибке открытия файла на запись.

В случае успешных проверок вызывается функция *specializationFind* (). Файл с базой данных открывается на чтение, а файл результатов поиска открывается на запись. Пользователю предлагается ввести в виде значения ключа поиска название специализации, информацию о врачах практикующую её он хотел бы просмотреть.

Все записи исходного файла поочерёдно считываются. Значение соответствующего поля считанной структуры сравнивается с ключевым значением с использованием функции *strncmp*(). Если сравнение произошло – запись в файл выборки.

После завершения просмотра всего содержимого исходного файла он закрывается функцией *fclose*(). Аналогично закрывается и файл выборки результатов. Происходит возврат в меню программы с выводом сообщения «Поиск по специализаии закончен».

При выборе пункта 6 – «просмотр базы данных» - выполняется проверка существования физического файла с заданным именем вызовом функции *proverka\_file*(). Если файл существует, вызывается функция *prosmotrbd1*(). Файл открывается на чтение. Выводится заголовок таблицы, в виде которой будут выводиться значения полей записей файла. Все записи файла поочерёдно читаются, и значения их полей выводятся в виде таблицы.

После завершения процесса файл закрывается.

При выборе пункта 7 – «просмотр базы данных поиска врачей по специализации» - все действия полностью аналогичны выбору пункта меню 6. Исключение – столбцы таблицы переставлены таким образом, что столбец со значением поля поиска (специализации) стоит самым левым.

При выборе пункта 8 – завершение работы программы.

**Текст программы**

/\*Бинарные файлы.

Ввод/вывод с использованием функций библиотек языка C

Программа работы с базой данных "Поликлинники"

Создание базы

Добавление новых записей

Просмотр базы

Поиск врачей по специализации с созданием файла выборки

Сортировка по ФИО в алфавитном порядке

Сортировка в порядке возрастания времени начала приема

\*/

#include <stdio.h>

#include <string.h>

#include<windows.h>

#include <iostream>

#define FNAME\_I "F:\\bd15.dat" //имя файла с исходной базой

#define FNAME\_R "F:\\bd25.dat" //имя файла с результатами поиска

#define strlength 125 //длина таблицы

int proverka\_file(const char\*, const char\*); //проверка наличия файла

void addLine(const char\* fname);

void zag();

void prosmotrbd1(const char\* fname);

void prosmotrbd2(const char\* fname);

void specializationFind(const char\* fname1, const char\* fname2);

void sort\_name(const char\* fname);

void sort\_room(const char\* fname);

const char PR\_R[] = "rb"; //признак открытия бинарного файла на чтение

const char PR\_S[] = "r+b"; //признак открытия файла на чтение и запись

const char PR\_W[] = "wb"; //признак открытия файла на запись

const char PR\_A[] = "ab"; //признак открытия файла на добавление

typedef struct {

char name[60]; //Фио врача

char specialization[20]; //Специализация

int room; //номер кабинета

char time[6]; //время начала приема

char weekday[11]; //день недели

}doctors;

//main\_Begin\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251); //меняем кодировку консоли принудительно

SetConsoleOutputCP(1251); //меняем кодировку консоли принудительно на вывод

int var;

char otv;

char fname\_i[20] = FNAME\_I;

char fname\_r[20] = FNAME\_R;

for (; ;)

{//Выбор вида действия

printf("\n Вид действия:\n");

printf(" 1 - создание базы данных\n");

printf(" 2 - добавление новых записей\n");

printf(" 3 - сортировка по ФИО врача\n");

printf(" 4 - сортировка по номеру кабинета\n");

printf(" 5 - поиск врачей по специализации\n");

printf(" 6 - просмотр базы данных\n");

printf(" 7 - просмотр базы данных поиска врачей\n");

printf(" 8 - завершение задачи\n");

printf(" Введите вид действия ->");

scanf("%i", &var);

switch (var)

{

default:break;

case 1: if (proverka\_file(fname\_i, PR\_R))

{

printf(" Файл базы данных врачей с именем ", fname\_i);

printf(" был создан раньше.\n");

printf(" Создаём файл с новым именем? [Y/N] ");

while ((otv = getchar()) == '\n');

if (otv == 'Y' || otv == 'y' || otv == 'Н' || otv == 'н')

{

printf(" Задайте имя создаваемого файла: ");

scanf("%s", fname\_i);

}

else break;

}

if (!proverka\_file(fname\_i, PR\_W))

{

printf("\n Ошибка открытия файла для записи\n");

break;

}

printf(" Создаем базу ", fname\_i, '\n');

getchar();

addLine(fname\_i);

printf("\n Создание файла закончено.\n");

if (proverka\_file(fname\_i, PR\_R))

printf(" База данных готова к работе\n");

else printf("\n база не создана\n");

break;

case 2: if (proverka\_file(fname\_i, PR\_R))

{

printf(" Файл базы данных врачей с именем ", fname\_i);

printf(" был создан раньше.\n");

printf(" Будем добавлять новые записи в него? [Y/N] ");

while ((otv = getchar()) == '\n');

if (otv == 'N' || otv == 'n' || otv == 'Т' || otv == 'т')

{

printf(" Задайте имя файла другой базы: ");

scanf("%s", fname\_i);

if (!proverka\_file(fname\_i, PR\_R))

{

printf(" Такая база данных не создавалась\n");

break;

}

}

}

printf(" Добавляем записи в файл ", fname\_i, '\n');

addLine(fname\_i);

printf("\n Изменение файла закончено.");

break;

case 3: if (!proverka\_file(fname\_i, PR\_S))

{

printf("\n Ошибка открытия файла для чтения и записи\n");

break;

}

sort\_name(fname\_i);

printf("\n Сортировка по ФИО врача закончена.");

break;

case 4: if (!proverka\_file(fname\_i, PR\_S))

{

printf("\n Ошибка открытия файла для чтения и записи\n");

break;

}

sort\_room(fname\_i);

printf("\n Сортировка по номеру кабинета закончена.");

break;

case 5: if (!proverka\_file(fname\_i, PR\_R))

{

printf("\n Ошибка открытия файла для чтения\n");

break;

}

if (!proverka\_file(fname\_r, PR\_W))

{

printf("\n Ошибка открытия файла для записи\n");

break;

}

specializationFind(fname\_i, fname\_r);

printf("\n Поиск по специализации врача закончен.");

break;

case 6: if (!proverka\_file(fname\_i, PR\_R))

{

printf("\n Ошибка открытия файла для чтения\n");

break;

}

prosmotrbd1(fname\_i);

break;

case 7: if (!proverka\_file(fname\_r, PR\_R))

{

printf("\n Ошибка открытия файла для чтения\n");

break;

}

prosmotrbd2(fname\_r);

break;

case 8: return 0;

}

}

}

//Добавление новых элементов в базу данных

void addLine(const char\* fname)

{

char otv;

doctors doctor;

FILE\* baza;

baza = fopen(fname, PR\_A);

rewind(baza);

do

{

char docName[20];

char docFather[20];

printf("\n ФИО врача? ");

scanf("%s %s %s", &doctor.name , &docName, &docFather);

strcat(doctor.name, " ");

strcat(doctor.name, docName);

strcat(doctor.name, " ");

strcat(doctor.name, docFather);

printf(" Специализация? ");

scanf("%s", &doctor.specialization);

printf(" Номер кабинета? ");

scanf("%i", &doctor.room);

printf(" Время начала приема? ");

scanf("%s", &doctor.time);

printf(" День недели? ");

scanf("%s", &doctor.weekday);

fwrite(&doctor, sizeof(doctor), 1, baza);

printf("\n Продолжать?[Y/N]");

while ((otv = getchar()) == '\n');

} while (otv == 'Y' || otv == 'y' || otv == 'Н' || otv == 'н');

fclose(baza);

return;

}

//Вывод заголовка при просмотре исходного файла

void zag()

{

int i;

printf("\n");

for (i = 1; i <= strlength; i++)

printf("-");

printf("\n");

printf("|%60s|%20s|%14s|%14s|%11s|\n", "ФИО ", "Специализация ",

"Номер кабинета", "Начало приема", "День недели");

for (i = 1; i <= strlength; i++)

printf("-");

return;

}

//Вывод заголовка при просмотре файла поиска

void zag2()

{

int i;

printf("\n");

for (i = 1; i <= strlength; i++)

printf("-");

printf("\n");

printf("|%20s|%60s|%14s|%14s|%11s|\n", "Специализация ","ФИО ",

"Номер кабинета", "Начало приема", "День недели");

for (i = 1; i <= strlength; i++)

printf("-");

return;

}

//Просмотр базы данных врачей

void prosmotrbd1(const char\* fname)

{

int i;

doctors doctor;

FILE\* baza;

baza = fopen(fname, PR\_R);

printf("\n База данных врачей поликлинники");

zag();

while (fread(&doctor, sizeof(doctor), 1, baza) > 0)

{

printf("\n|%60s|%20s|%14d|%14s|%11s|", doctor.name, doctor.specialization, doctor.room, doctor.time, doctor.weekday);

}

printf("\n");

for (i = 1; i <= strlength; i++)

printf("-");

fclose(baza);

return;

}

//Просмотр базы данных поиска врачей по специализации

void prosmotrbd2(const char\* fname)

{

int i;

doctors doctor;

FILE\* baza;

baza = fopen(fname, PR\_R); //открываем файл на чтение

printf("\n База данных поиска поиска врачей по специализации");

zag2();

while (fread(&doctor, sizeof(doctor), 1, baza) > 0)

{

printf("\n|%20s|%60s|%14d|%14s|%11s|", doctor.specialization, doctor.name, doctor.room, doctor.time, doctor.weekday);

}

printf("\n");

for (i = 1; i <= strlength; i++)

printf("-");

fclose(baza);

return;

}

//Поиск врачей по специализации

void specializationFind(const char\* fname1, const char\* fname2)

{

char spec[20];

doctors doctor;

FILE\* in, //исходный файл

\* out; //файл результатов поиска

in = fopen(fname1, PR\_R); //открываем файл на чтение

out = fopen(fname2, PR\_W); //открываем файл на запись

printf("\n Название специализации для поиска? ");

scanf("%s", &spec);

while (fread(&doctor, sizeof(doctor), 1, in) > 0)

if (strncmp(doctor.specialization, spec, 20) == 0)

fwrite(&doctor, sizeof(doctor), 1, out);

fclose(out);

fclose(in);

return;

}

//Сортировка по ФИО по алфавиту

void sort\_name(const char\* fname)

{

int i;

int fl;

doctors ppp, doctor;

FILE\* baza;

baza = fopen(fname, PR\_S); //открываем файл на чтение и запись

do {

rewind(baza);

fl = 0;

for (i = 0; fread(&doctor, sizeof(doctor), 1, baza) > 0; i += sizeof(doctor),

fseek(baza, i, SEEK\_SET)) //позиция i от НАЧАЛА файла

{

if (fread(&ppp, sizeof(doctor), 1, baza) > 0)

{

if (strncmp(doctor.name, ppp.name, 60) > 0)

{

fseek(baza, i, SEEK\_SET); //позиция i от НАЧАЛА файла

fwrite(&ppp, sizeof(doctor), 1, baza);

fwrite(&doctor, sizeof(doctor), 1, baza);

fl = 1;

}

}

}

} while (fl);

fclose(baza);

return;

}

//Сортировка по возрастанию номера кабинета

void sort\_room(const char\* fname)

{

int i;

int fl;

doctors ppp, doctor;

FILE\* baza;

baza = fopen(fname, PR\_S); //открываем файл на чтение и запись

do {

rewind(baza);

fl = 0;

for (i = 0; fread(&doctor, sizeof(doctor), 1, baza) > 0; i += sizeof(doctor),

fseek(baza, i, SEEK\_SET))

{

if (fread(&ppp, sizeof(doctor), 1, baza) > 0)

{

if (doctor.room > ppp.room)

{

fseek(baza, i, SEEK\_SET); //позиция i от НАЧАЛА файла

fwrite(&ppp, sizeof(doctor), 1, baza);

fwrite(&doctor, sizeof(doctor), 1, baza);

fl = 1;

}

}

}

} while (fl);

fclose(baza);

return;

}

//проверка наличия физического файла

int proverka\_file(const char\* fname, const char\* pr)

{//Открываем файл

if (!(fopen(fname, pr))) //файл не существует

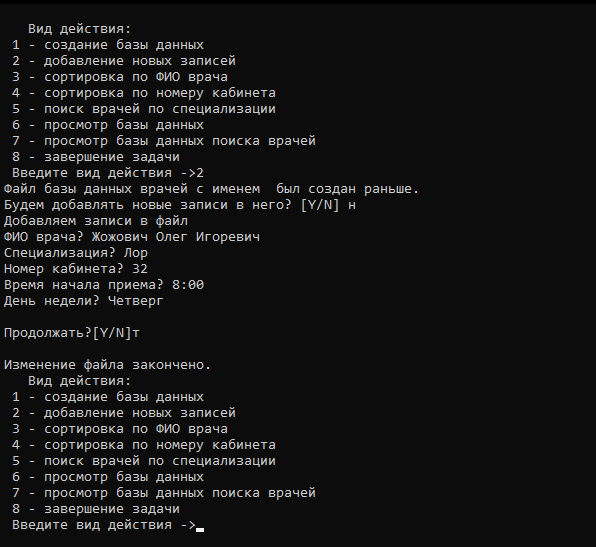
return(0);

return(1);

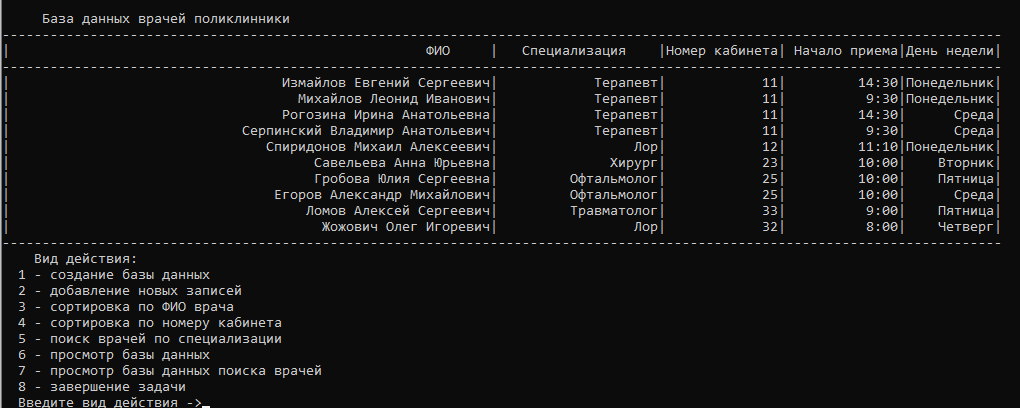
}

**Примеры работы программы**

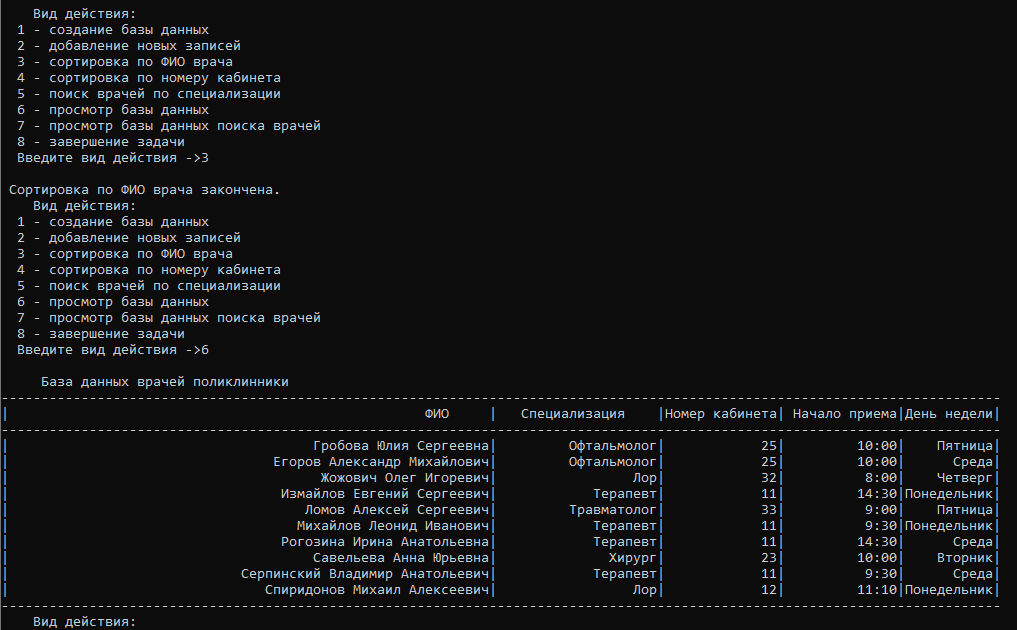
1. Добавление нового врача.

****

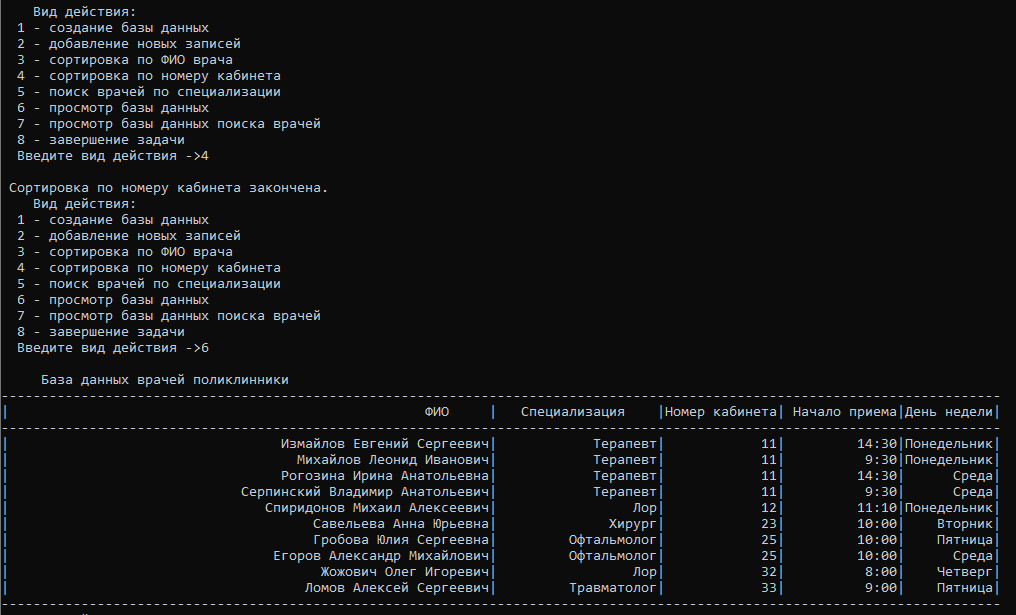
1. **Ввод БД**

****

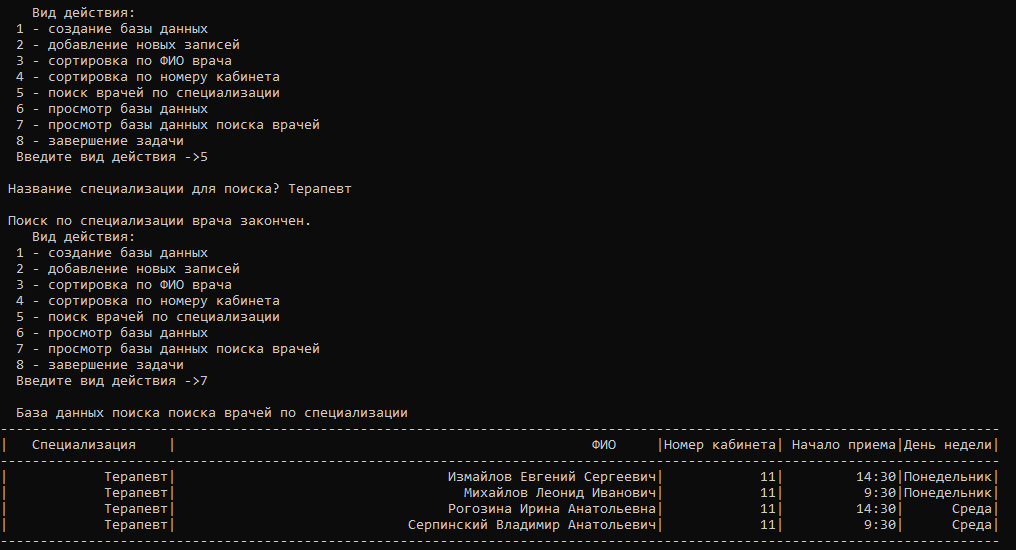
1. **Сортировка по ФИО**

****

1. **Сортировка по номеру кабинета**

****

1. **Поиск по специализации**

****

**Вывод:** используя технологию процедурного программирования разработал программу обработки бинарных файлов структур с числом записей не менее пяти в соответствии с индивидуальным заданием.