МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

Высшего образования

«чувашский государственный университет имении И. Н. Ульянова»

Факультет информатики и вычислительной техники

Кафедра компьютерных технологий

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

По дисциплине «Информатика и программирование»

Выполнила студент гр. КТ-43-21

Трынова Юлия Александровна

Проверил: Алюнов Д.Ю

Вариант № 10

Чебоксары

2021

**Задание РГР.**

Разработать информационно справочную систему для работы с базой данных. Входные данные:  
База данных состоит из записей. (Код пациента, ФИО, Диагноз, Возраст, Лечащий врач в отдельном файле справочная информация о Лечащем враче)  
Обеспечить:  
- введение данных в оперативную память в двунаправленный список:  
а) из текстового файла;  
б) с клавиатуры;  
- просмотр содержимого базы данных;  
- редактирование базы данных:   
а) удаление записи;   
б) добавление записи;  
- выведение данных:  
а) в текстовый файл;  
б) на экран;  
- поиск данных по коду пациента;   
- сортировка данных по возрасту;  
- получение справочной информации о лечащем враче;

Интерфейс консоли.

**Программный код:**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <Windows.h>

#include <algorithm>

#include <string>

#include <list>

using namespace std;

struct Boln

{

int lech;// номер лечащего врача

int kod;// код пациента

int age;// возраст пациента

string diag;// Диагноз

};

const string PName = "bb.txt";

const string lName = "ll.txt";

int Command;

bool needToDisplay = true;

int patientkol;

int patientkod;

int llech;

//пациенты

bool file1()

{

ifstream read; // создаем объект класса ifstream

read.open(PName);

if (!read.is\_open()) // проверяем открыт ли файл

{

cout << "Ошибка открытия файла!" << endl;

exit(0);

}

read.seekg(0, ios::end);

return read.tellg() != 0;

}

//врачи

bool file2()

{

ifstream readl; // создаем объект класса ifstream

readl.open(lName);

if (!readl.is\_open()) // проверяем открыт ли файл

{

cout << "Ошибка открытия файла!" << endl;

exit(0);

}

readl.seekg(0, ios::end);

return readl.tellg() != 0;

}

// (массив структур)

void DisplayOptions() // (новая функция)

{

if (!needToDisplay)

{

return;

} // команды

cout << "Введите действие: " << endl;

cout << " Редактирование базы данных: " << endl;

cout << " - Добавление записи - 1" << endl;

cout << " - Удаление записи - 2" << endl;

cout << " Вывод данных - 3" << endl;

cout << " Поиск даннных по коду пациента - 4" << endl;

cout << " Сортировать данных по возрасту - 5" << endl;

cout << " Получение справочной информации о лечащем враче - 6" << endl;

cout << " Выход - 0" << endl;

cin >> Command;

}

// Сохранение данных

void SaveData(Boln\* (&patients))

{

ofstream record(PName, ios::out);

if (record)

{

if (patientkol != 0)

{

record << patientkol << endl;

}

for (int i = 0; i < patientkol; i++)

{

record << patients[i].kod << endl;

record << patients[i].lech << endl;

record << patients[i].diag << endl;

record << patients[i].age << endl;

}

}

else

{

cout << "Файла не существует" << endl;

exit(0);

}

record.close();

}

void CopyData(Boln\* (&st\_new), Boln\* (&st\_old), int n)

{

for (int i = 0; i < n; i++)

{

st\_new[i] = st\_old[i];

}

}

void CopyData(Boln& st\_new, Boln& st\_old)

{

st\_new.kod = st\_old.kod;

st\_new.lech = st\_old.lech;

st\_new.diag = st\_old.diag;

st\_new.age = st\_old.age;

}

// добавление новой записи

void Addpatient(Boln\* (&patients))

{

Boln\* buf = new Boln[patientkol]; // создание новой переменной для структуры

CopyData(buf, patients, patientkol);// записывает данные (копия)

patientkol++;

patients = new Boln[patientkol];

CopyData(patients, buf, patientkol - 1);

int a = patientkol - 1;

//требуется ввести данные для добавления новой записи

cout << "Введите код пациента" << endl;

cin >> patients[a].kod;

cout << "Введите возраст пациента" << endl;

cin >> patients[a].age;

cout << "Введите диагноз пациента" << endl;

cin >> patients[a].diag;

cout << "Введите номер лечащего врача" << endl;

cin >> patients[a].lech;

delete[]buf; // удаление новой переменной

}

// Вывод данных

void patients3(Boln\* (&patients))

{

if (!file1()) // проверяем есть ли данные

{

cout << "Записей нет!" << endl; // если их нет выводим соответсвенное сообщение

return;

}

// выводим все имеющиеся данные на экран

for (int i = 0; i < patientkol; i++)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Код пациента " << patients[i].kod << endl; // код пациента

cout << "Врач " << patients[i].lech << endl; // лечащий врач

cout << "Диагноз " << patients[i].diag << endl; // диагноз пациента

cout << "Возраст " << patients[i].age << endl; // возраст пациента

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

void patients4(Boln\* (&patients), int patientkod) // чтение из консоли

{

if (!file1())

{

cout << "Записей нет!" << endl;

return;

}

for (int i = 0; i < patientkol; i++)

{

if (patients[i].kod == patientkod)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Код пациента " << patients[i].kod << endl;

cout << "Врач " << patients[i].lech << endl;

cout << "Диагноз " << patients[i].diag << endl;

cout << "Возраст " << patients[i].age << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

}

// сортировка

bool compareSort(Boln\*& a, Boln\*& b)

{

return a->age < b->age; // сортируем по возрасту

}

void patients5(Boln\* (&patients), bool useSort)

{

if (!file1())

{

cout << "Записей нет" << endl;

return;

}

if (!useSort)

{

return;

}

Boln\* buf = new Boln[patientkol];

CopyData(buf, patients, patientkol);

list<Boln\*> st;

for (int i = 0;

i < patientkol; i++)

{

st.push\_back(&buf[i]);

}

st.sort(compareSort);

for (auto s : st)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "Код пациента " << s->kod << endl;

cout << "Врач " << s->lech << endl;

cout << "Диагноз " << s->diag<< endl;

cout << "Возраст " << s->age << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

// удаление записи

void DeleteBoln(Boln\* (&patients))

{

cout << "Введите id пациента" << endl;

int id;

cin >> id;

id--;

if (id >= 0 && id < patientkol)

{

Boln\* buf = new Boln[patientkol];

CopyData(buf, patients, patientkol);

patientkol--;

patients = new Boln[patientkol];

int q = 0;

for (int i = 0; i <= patientkol; i++)

{

if (i != id)

{

patients[q] = buf[i];

q++;

}

}

delete[]buf;

}

}

// чтение с файла пациентов

void chtenie()

{

if (file1())

{

ifstream read(PName);

if (read)

{

read >> patientkol;

}

}

else

{

patientkol = 0;

}

}

// чтение с файла врачей

void chteniel()

{

if (file2())

{

ifstream readl(lName);

string line;

if (readl)

{

int num = 0;

while (getline(readl, line))

{

++num;

if (num == llech)

{

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << line << endl;

cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

}

}

}

}

else

{

patientkol = 0;

}

}

// вывод по сортировке

void Init(Boln\* (&patients))

{

if (patientkol != 0)

{

ifstream read(PName);

if (read)

{

read >> patientkol;

for (int i = 0; i < patientkol; i++)

{

read >> patients[i].kod;

read >> patients[i].lech;

read >> patients[i].diag;

read >> patients[i].age;

}

}

}

}

int main()

{

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

chtenie();

chteniel();

Boln\* patients = new Boln[patientkol];

Init(patients);

while (true)

{

DisplayOptions();

switch (Command)

{

case 0:

exit(0);

break;

case 1:

needToDisplay = false;

Addpatient(patients);

SaveData(patients);

needToDisplay = true;

break;

case 2:

needToDisplay = false;

DeleteBoln(patients);

SaveData(patients);

needToDisplay = true;

break;

case 3:

needToDisplay = false;

patients3(patients);

needToDisplay = true;

break;

case 4:

needToDisplay = false;

cout << "Введите код пациента: " << endl;

cin >> patientkod;

patients4(patients, patientkod);

needToDisplay = true;

break;

case 5:

needToDisplay = false;

patients5(patients, true);

needToDisplay = true;

break;

case 6:

needToDisplay = false;

cout << "Введите номер лечащего врача: " << endl;

cin >> llech;

chteniel();

needToDisplay = true;

break;

default:

cout << "Такой команды нет" << endl;

}

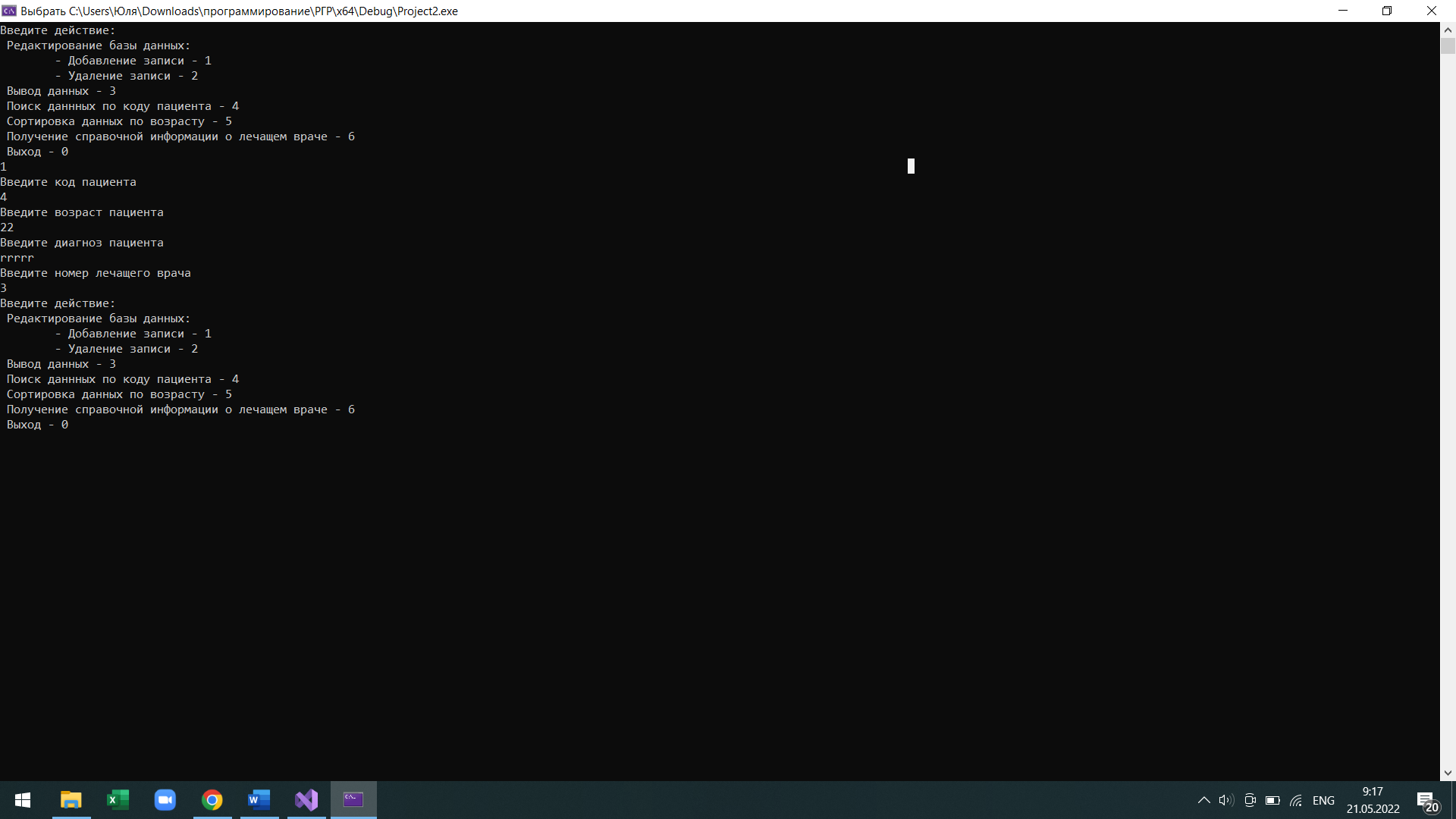
}

}

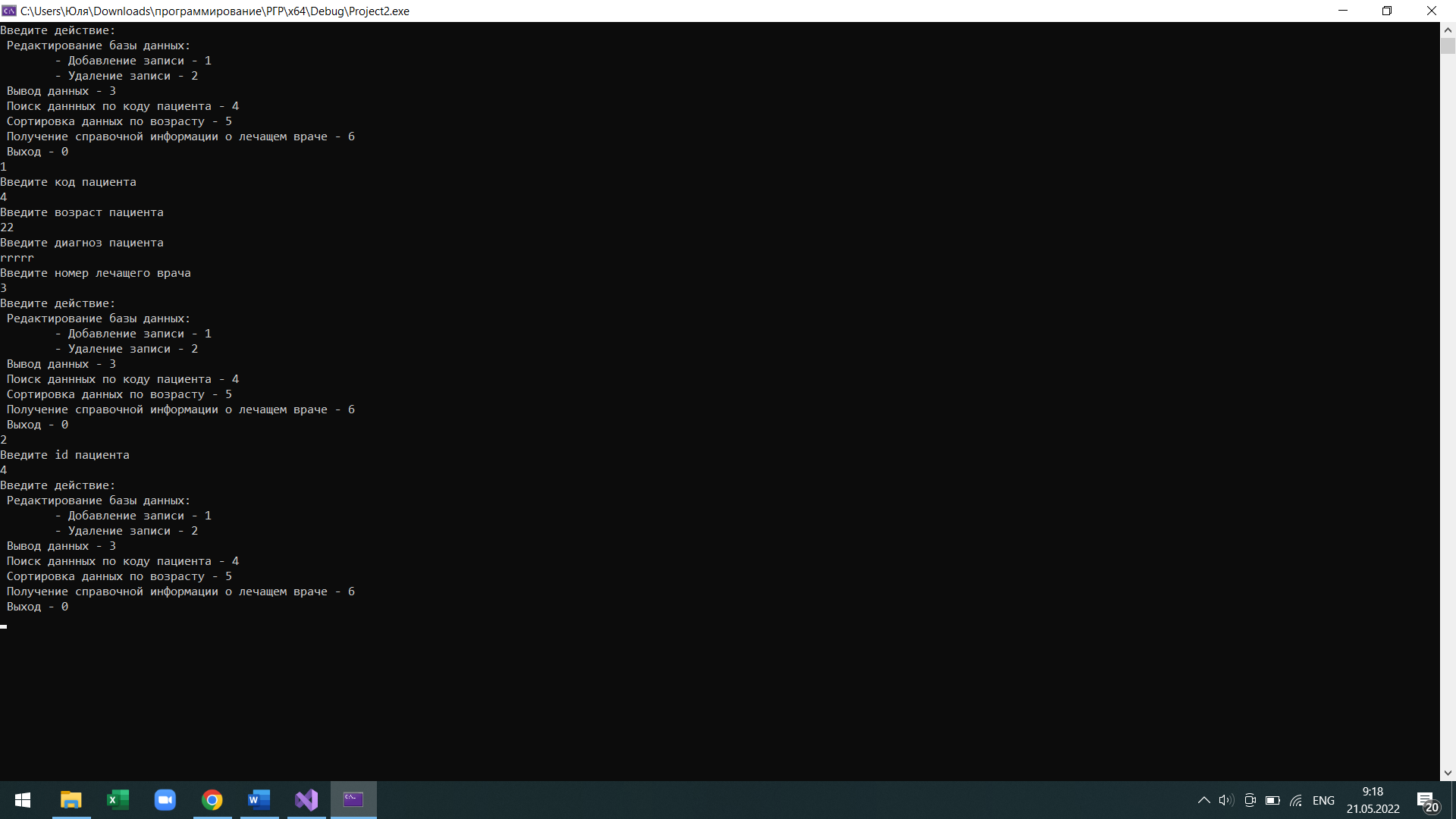
**Скрины**

Редактирование базы данных:

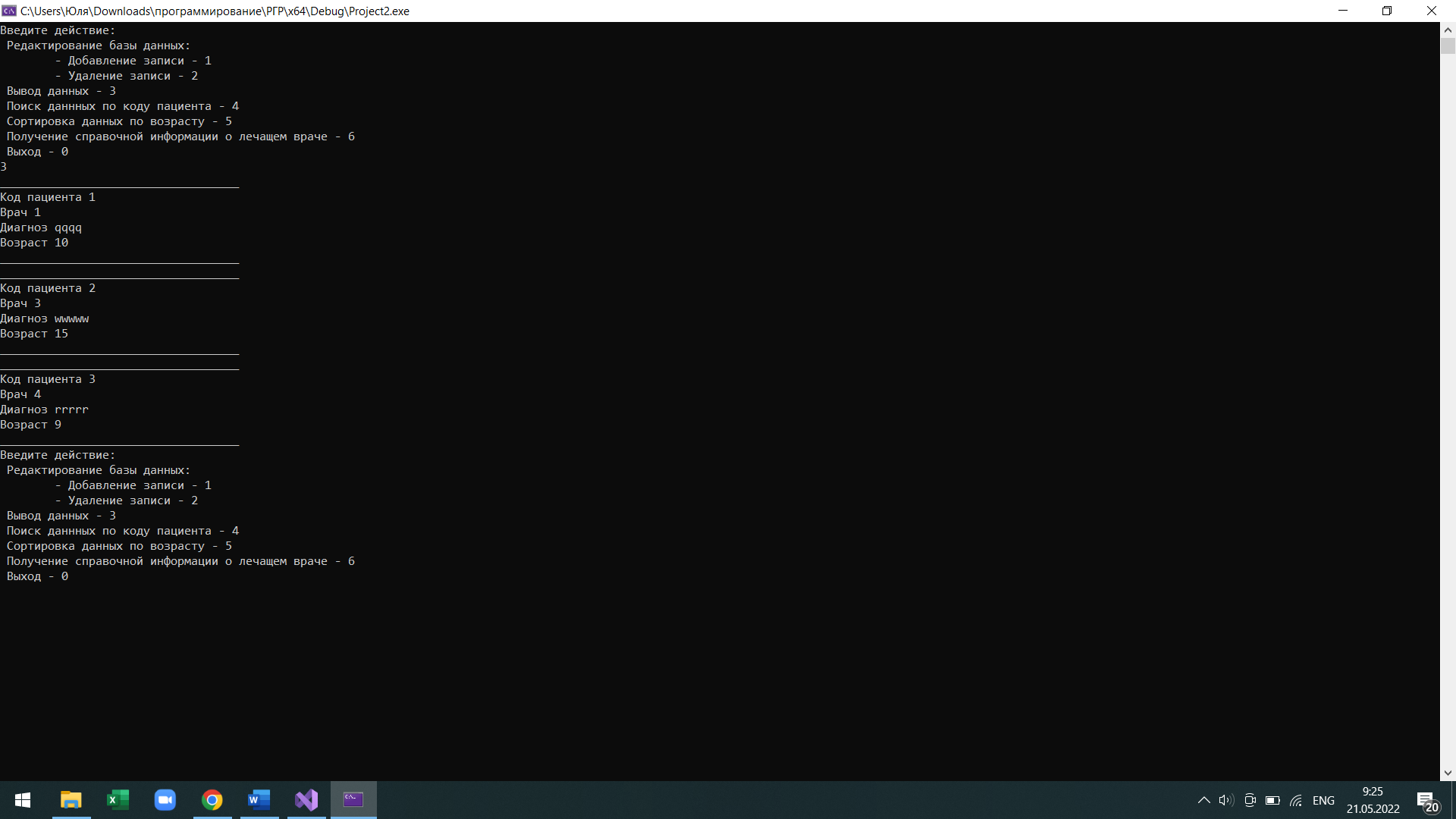
1. Добавление записи:



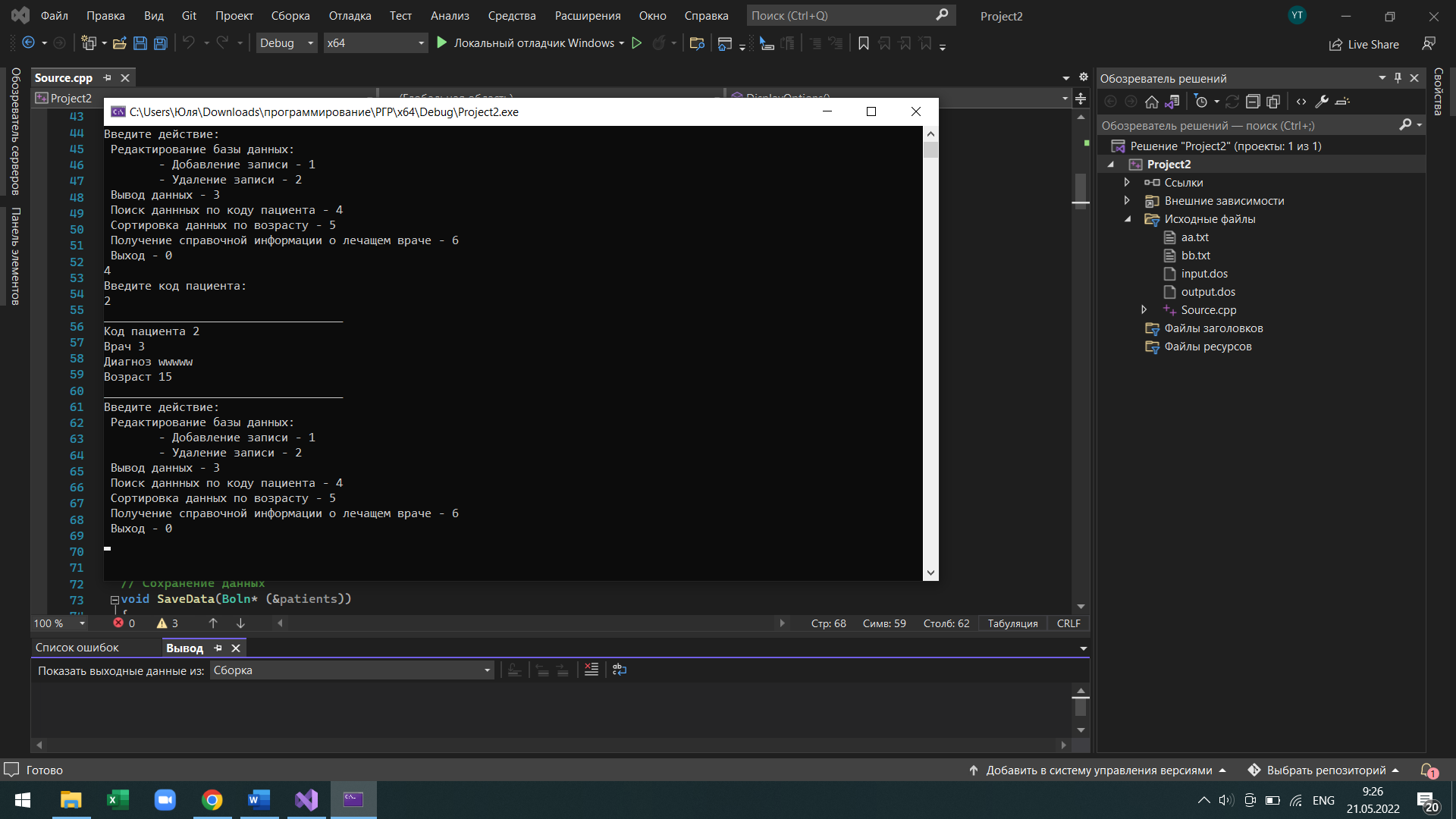
1. Удаление записи:



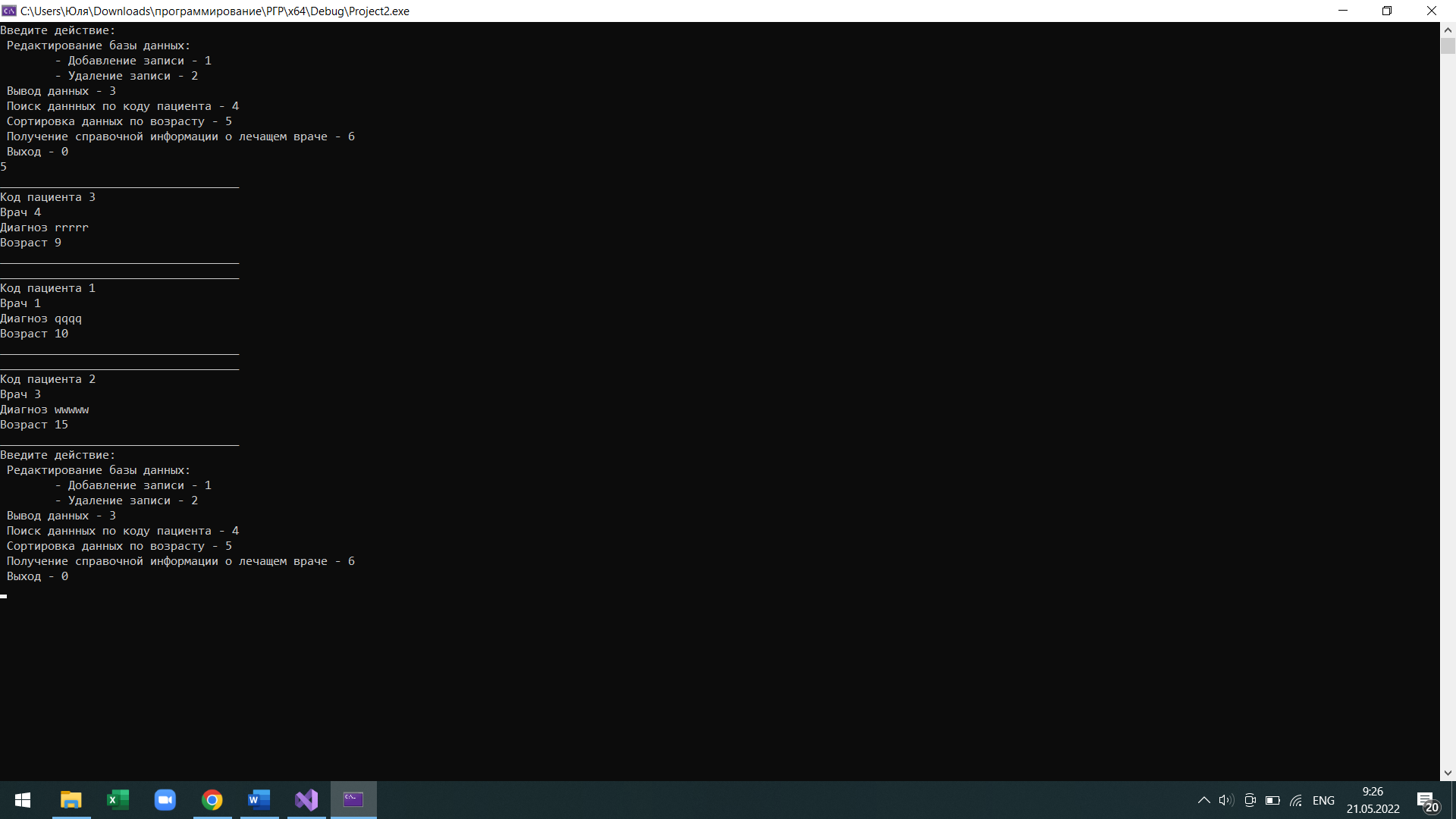
Вывод данных:



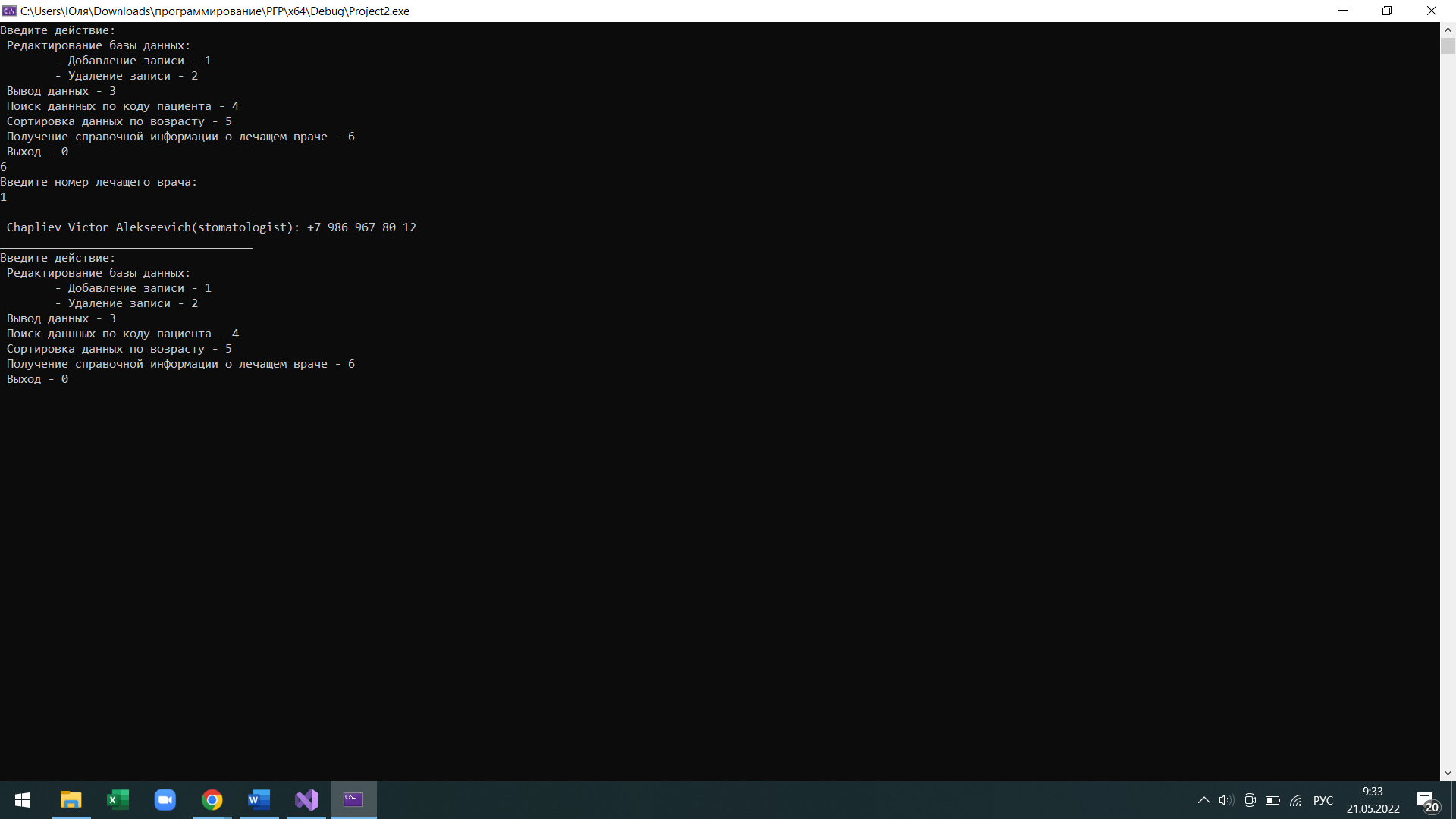
Поиск данных по коду пациента:



Сортировка данных по возрасту:



Получение справочной информации о лечащем враче:



Выход:

