# Аннотация

Представленная курсовая работа состоит из введения, трех глав, заключения, списка литературных источников и приложений.

В первой части работы рассматриваются общие сведения о программе, такие как вариант задания, основные модули программы, а также блок-схемы. Во второй части были описаны условия запуска программы, а также её проверка работоспособности программы.

**СОДЕРЖАНИЕ**

[Аннотация 3](#_Toc26958201)

[Введение 5](#_Toc26958202)

[1 Назначение и область применения программы 6](#_Toc26958203)

[2 Технические характеристики программы 7](#_Toc26958204)

[2.1 Постановка задачи на разработку программы 7](#_Toc26958205)

[2.2 Применяемые математические методы 7](#_Toc26958206)

[2.3 Описание и обоснование выбора метода организации входных, выходных и промежуточных данных 8](#_Toc26958207)

[2.4 Разработка модульной структуры программы 8](#_Toc26958208)

[2.5 Описание алгоритмов функционирования программы 12](#_Toc26958209)

[2.6 Обоснование состава технических и программных средств 32](#_Toc26958210)

[3 Выполнение программы 33](#_Toc26958211)

[3.1 Условия выполнения программы 33](#_Toc26958212)

[3.2 Загрузка и запуск программы 33](#_Toc26958213)

[3.3 Проверка работоспособности программы 33](#_Toc26958214)

[Заключение 42](#_Toc26958215)

[Список использованных источников 43](#_Toc26958216)

[Приложение А 44](#_Toc26958217)

# Введение

Целью данного курсового проекта является разработка программы расчета среднего процента процессорного времени, которая обеспечивает получение оперативных данных об эффективности работы каждого процессора.

Предмет исследования данной темы – обработка данных, представленных в виде таблицы. Темой курсового проекта является разработка программы для обработки данных, представленных в форме таблицы. Любая таблица отражает какую-либо форму человеческой деятельности. С этой точки зрения программа, написанная в результате курсового проектирования, может считаться упрощенным подобием предметно-ориентированной базы данных.

Широко распространенным видом услуг, эффективно реализующихся с помощью компьютеров, является информационно-справочное обслуживание, которое подразумевает хранение сведений, прием новых сведений, обработку сведений, выдачу информации по запросам. Хранимые сведения в общем случае представляются записями. Для представления такого рода услуг создаются автоматизированные информационные системы (АИС) различного назначения.

Основная задача, решаемая в ходе создания АИС, состоит в том, чтобы организовать совместное хранение большого числа различных записей и выдавать по запросу любую из них независимо от того, какие записи и в каком порядке выдавались ранее.

# Назначение и область применения программы

Программное средство должно автоматизировать и упростить работу пользователя.

Областью применения данной программы является получение данных введённых пользователем о времени выполнения заданий на ЭВМ, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям.

Программа предназначена для кампаний, которые хотят систематизировать данные о времени выполнения заданий на ЭВМ, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям.

# Технические характеристики программы

## Постановка задачи на разработку программы

Даны сведения о времени выполнения заданий на ЭВМ (время измеряется в секундах). Структура записи: шифр задания (8 символов), код отдела (3 символа), Ф.И.О. программиста (15 символов), общее время прохождения задания, время центрального процессора − 5 знаков (t\_задания > t\_цп). Получить процент процессорного времени по каждому заданию, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям.

## Применяемые математические методы

В ходе разработки данного программного обеспечения применялась формула расчёта процентного отношения. Для этого использовалось правило пропорции. Принцип работы показан на рисунке 2.1.

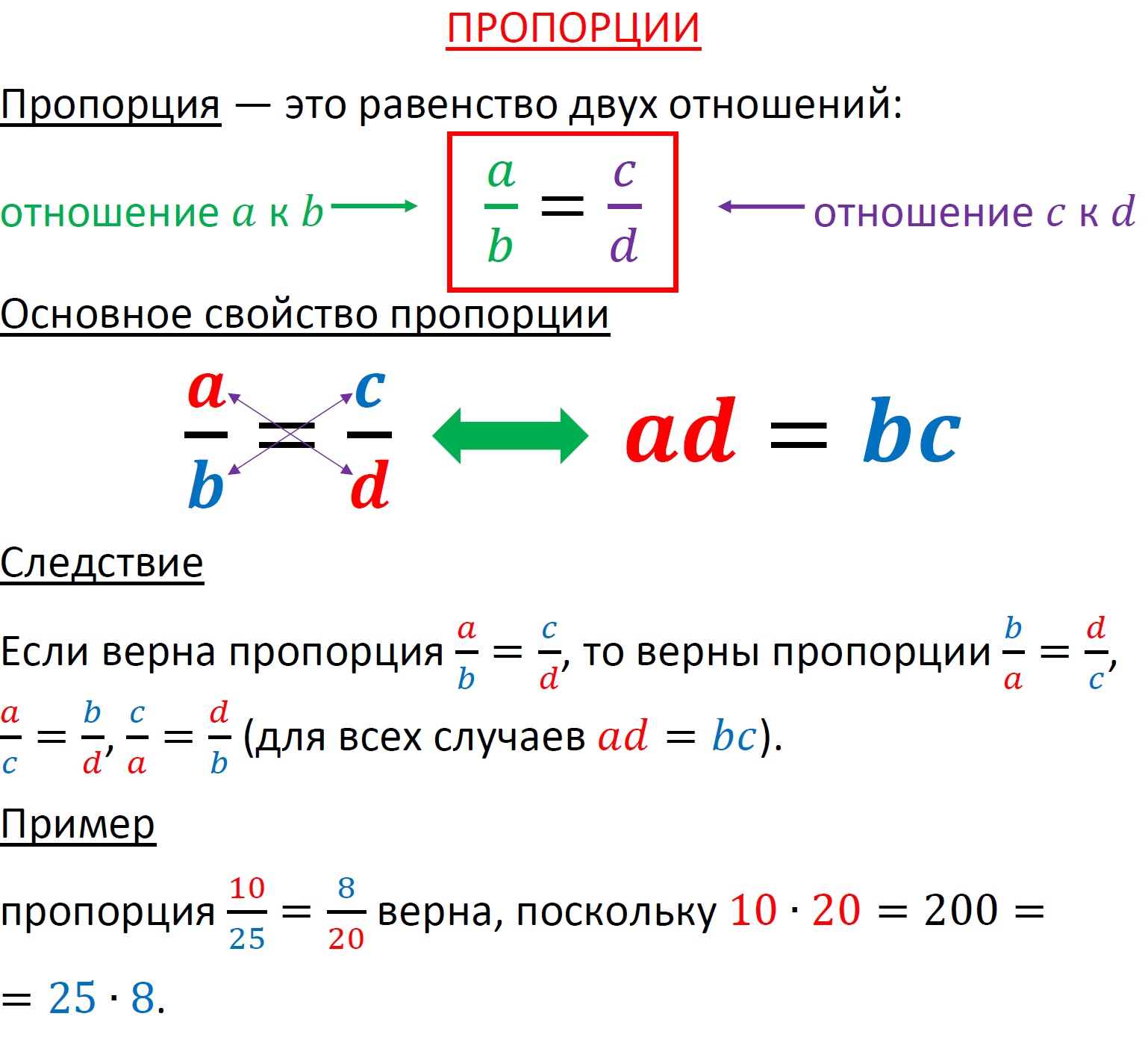


Рисунок 2.1 – Правило пропорции

## Описание и обоснование выбора метода организации входных, выходных и промежуточных данных

Входными данными являются:

1. Шифр задания (8 символов)
2. Код отдела (3 символа)
3. Ф.И.О. программиста (15 символов)
4. общее время прохождения задания (5 знаков)
5. время центрального процессора (5 знаков)

При этом общее время прохождения задания должно быть больше времени центрального процессора.

В качестве организации данных выбран двунаправленный список, так как он более гибок по сравнению с однонаправленным списком.

В качестве выходных данных формируется таблица, в которой можно редактировать, удалять и сортировать данные.

## Разработка модульной структуры программы

Определим необходимые константы:

const int up = 72,

down = 80,

right\_btn = 77,

left\_btn = 75,

enter = 13,

esc = 27,

del = 83;

const int SIZE\_arr\_filename = 50;

// названия пунктов

const string items[7] = {

" Ввод данных ",

" Печать данных ",

" Сохранить данные ",

" Сохранить в другой файл ",

" Поиск ",

" Выбрать другой файл ",

" Выход из программы " };

// названия для сортировки

const string sort\_items[5] = {

"| Шифр задания ",

"Код отдела ",

"ФИО ",

"Общее время ",

"Время ЦП" };

Вывод данных на экран. Параметрами передаются указатель на первый и последний элемент, номер активного пункта меню, номер редактируемого элемента, счетчик для i и номер текущей страницы. Данная функция отвечает за постраничный вывод данных на экран:

time\_task\* print(time\_task\* end, time\_task\* real\_beg ,time\_task\* beg, int active, int edit\_el, int print\_count\_num\_pages, int print\_page);

Чтение из текстового файла. Параметрами передается название файла, а также указатели на начало и конец списка. Данная функция отвечает за чтение из текстового файла:

int read\_file(string filename, time\_task\*\* beg, time\_task\*\* end);

Определение типа файла. Тип файла выбирается пользователем при помощи пункта меню. Параметром мы передаем указатель на начало списка.

int write\_file(time\_task\* temp);

Сохранение в файл. Данная функция несет ответственность за сохранение данных в файл. Параметры функции: указатель на начало списка, название файла, номер элемента массива всех названий файлов, а также тип файла, который выбрал пользователь.

void write\_filetype(time\_task\* temp, string filename, int el, int filetype);

Удаление элемента из списка. В данную функцию передается указатель на начала списка и номер элемента, который мы хотим удалить:

time\_task\* delete\_el(time\_task\* beg, int num\_del);

Чтение из бинарного файла. Параметрами передается название файла, а также указатели на начало и конец списка. Данная функция отвечает за чтение из бинарного файла:

int read\_bin\_file(string filename, time\_task\*\* beg, time\_task\*\* end);

Сортировка. Функция необходима для сортировки данных. Параметры: указатель на начало списка, поле, по которому пользователь хочет произвести сортировку и ее направление:

void sort(time\_task\* beg, int field\_for\_sort, int sort\_direction);

Редактирование. Функция предназначена для корректировки определенного поля, который пользователь может выбрать при просмотре меню. Функция принимает следующие параметры: указатель на начало и конец списка, номер активного пункта меню, элемент для редактирования, счетчик для i, номер текущей страницы и высота всей таблицы:

void edit(time\_task\* end, time\_task\* real\_beg, time\_task\* beg, int active, time\_task\* \_edit\_ob, int edit\_count\_num\_pages, int edit\_page, int edit\_i);

Поиск. Функция нужна для поиска определенного элемента, который захочет найти пользователь. В качестве параметра передается лишь указатель на начало списка:

void find(time\_task\* beg);

Инициализация меню. Данная функция осуществляет вывод пунктов меню на экран при помощи которых пользователь сможет взаимодействовать с программой. Параметрами этой функции будут являться номер активного меню, массив названий пунктов меню и их количество:

int menu(int& active, const string items[], int num\_el);

Расчет процента процессорного времени. Получает процент процессорного времени по каждому заданию, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям. У функции два параметра – общее время выполнения задания и время процессора:

float percent\_time\_cpu(float a, float b);

Выделение памяти. Для этого функции передается указатель на начало списка и конец, а также данные, которые ввел пользователь.

void input(time\_task\*& beg, time\_task\*& end, const time\_task& info);

## Описание алгоритмов функционирования программы

На рисунке 2.2 представлена первая часть структурной схемы функции печати данных. В ней происходит проверка списка на пустоту, чему будет равен temp, а также выводиться шапка таблицы.



Рисунок 2.2 – Структурная схема печати данных

На рисунке 2.3 изображена вторая часть печати данных. В ней запускается цикл, с помощью которого мы проходимся по всем элементам и печатаем их.



Рисунок 2.3 – Структурная схема печати данных

На рисунке 2.4 определены клавиши, по которым происходит навигация по таблице



Рисунок 2.4 – Структурная схема печати данных

На рисунке 2.5 и рисунке 2.6 определены операции, происходящие при нажатии клавиш up и esc. Если пользователь нажал up, то мы должны переместиться по таблице вверх. При нажатии клавиши esc будет происходить следующие действия: выход из таблицы, выход из режима редактирования, а также выход из режима сортировки.



Рисунок 2.5 – Структурная схема печати данных (если нажали клавишу вверх)



Рисунок 2.6 – Структурная схема печати данных (если нажали клавишу esc)

На рисунке 2.7 определена операция, происходящая при нажатии клавиши down.



Рисунок 2.7 – Структурная схема печати данных (если нажали клавишу вниз)

На рисунке 2.8 и рисунке 2.9 определены операции, происходящие при нажатии клавиш вправо и влево.



Рисунок 2.8 – Структурная схема печати данных (если нажали клавишу вправо)



Рисунок 2.9 – Структурная схема печати данных (если нажали клавишу влево)

На рисунке 2.10 определена операция, происходящая при нажатии клавиши delete. Если пользователь согласен, то удаляется элемент из таблицы.



Рисунок 2.10 – Структурная схема печати данных (если нажали клавишу del)

На рисунке 2.11 определена операция, происходящая при нажатии клавиши enter. Если active = -1, то работает режим изменения количества элементов на одной странице, active = 0, то работает режим сортировки, во всех остальных случаях начинается редактирование элемента.



Рисунок 2.11 – Структурная схема печати данных (если нажали клавишу enter)

На рисунке 2.12 и рисунке 2.13 определены операции, происходящие при нажатии клавиш s/ы и n/т.



Рисунок 2.12 – Структурная схема печати данных (если нажали клавишу s/ы)



Рисунок 2.13 – Структурная схема печати данных (если нажали клавишу n/т)

С помощью клавиш h/р можно скрыть/показать горячие клавиши (рисунок 2.14). На рисунке 2.15 представлена сортировка данных.



Рисунок 2.14 – Структурная схема печати данных (если нажали клавишу h/р)



Рисунок 2.15 – Сортировка данных

На рисунке 2.16 изображена структурная схема редактирования элемента. С помощью клавиш вправо/влево пользователь перемещается по полям, при нажатии клавиши enter можно редактировать поле, на котором остановился пользователь.



Рисунок 2.16 – Редактирование полей

На рисунке 2.17 и рисунке 2.18 происходит перерисовка данных для редактирования.



Рисунок 2.17 – Редактирование полей (если нажали вправо)



Рисунок 2.18 – Редактирование полей (если нажали влево)

При нажатии на enter в функции редактирования вызовется блок кода, изображенный на рисунке 2.19. На рисунке 2.20 представлена функция по варианту, она рассчитывается по правилу пропорции.



Рисунок 2.19 – Редактирование полей (если нажали enter)



Рисунок 2.20 – Своя функция

На рисунке 2.21 показана функция удаление элемента. В ней сначала проверяется на один элемент в списке, если условие удовлетворяет, то удаляем этот элемент, иначе ищем удаляемы элемент по шифру задания.



Рисунок 2.21 – Удаление элемента

На рисунке 2.22 представлена функция выделения памяти, а на рисунке 2.23 изображена функция ввод данных с клавиатуры.



Рисунок 2.22 – Выделение памяти



Рисунок 2.23 – Ввод данных

На рисунке 2.24 описана функция записи данных в файл. В файл с названием всех файлов записывается название сохраняемой БД, затем в массив с названиями всех файлов добавляется наша БД. Далее определяется тип сохраняемого файла и происходит запись данных.



Рисунок 2.24 – Запись данных в бинарный и текстовый файл

На рисунке 2.25 изображена структурная схема считывания данных из текстового файла. Если файл существует, то считываем данные используя функцию выделения памяти input().



Рисунок 2.25 – Чтение из текстового файла

На рисунке 2.26 изображена структурная схема считывания данных из бинарного файла. Если файл существует, то считываем данные используя функцию выделения памяти input().



Рисунок 2.26 – Чтение из бинарного файла

На рисунке 2.27 изображена структурная схема поиска. При вводе данных для поиска, запускается цикл, в котором сравнивается каждое поле с введёнными данными, выводим нужный элемент если он найден.



Рисунок 2.27 – Поиск

На рисунке 2.28 описана основная функция, сначала вызывается функция первого запуска, в ней можно выбрать редактируемы файл, затем с помощью меню можно работать с выбранным файлом.



Рисунок 2.28 – Основное меню

## Обоснование состава технических и программных средств

Для написания программы была выбрана операционная система Windows, так как она самая распространённая. В качестве среды разработки было выбрано Visual Studio 2019. В качестве языка программирования был выбран язык высокого уровня C++.

По умолчанию программа расположена на гибком диске 3,5 дюйма. В состав программы входит исполняемый файл Kursach\_IIP.exe. Именно с запуска файла Kursach\_IIP.exe начинается работа приложения. Следующий шаг после запуска исполняемого файла состоит в формировании начальной базы данных, т.к. в исходном варианте файл со сформированной справочной системой отсутствует. Формирование БД исключительная прерогатива конечного пользователя. Использование защиты от несанкционированного использования и копирования не предусмотрено.

# Выполнение программы

## Условия выполнения программы

Проведя исследования, был выявлен следующий минимальный состав аппаратных и программных средств, необходимых для выполнения программы:

* ЦП Pentium IV;
* оперативная память 256Mb;
* минимальная ёмкость диска HDD: 500 Kb;
* стандартный монитор, мышь, клавиатура;
* видеокарта с оперативной памятью не менее 64 Mb;

## Загрузка и запуск программы

Программа может распространятся на флешке, диске или по интернету. В пакет программы входят файлы Kursach\_IIP.exe и MainBD.txt. Запуск производится с помощью файла Kursach\_IIP.exe.

## Проверка работоспособности программы

В начале запуска программы появляется меню с выбором откуда считывать данные (рисунок 3.1).

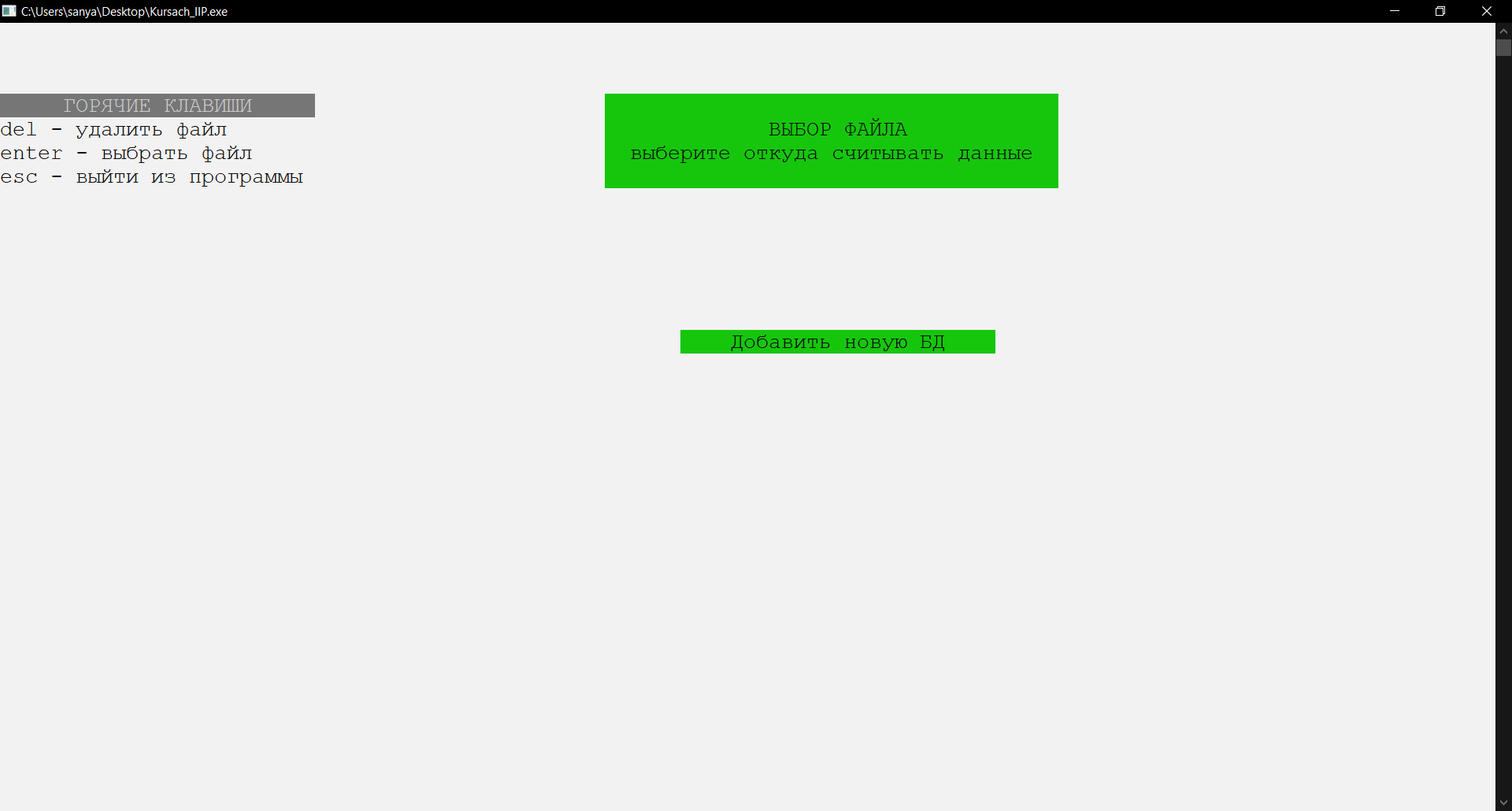


Рисунок 3.1 – Выбор файла

После выбора файла пользователь попадает в меню, где он может работать с конкретным файлом (рисунок 3.2).

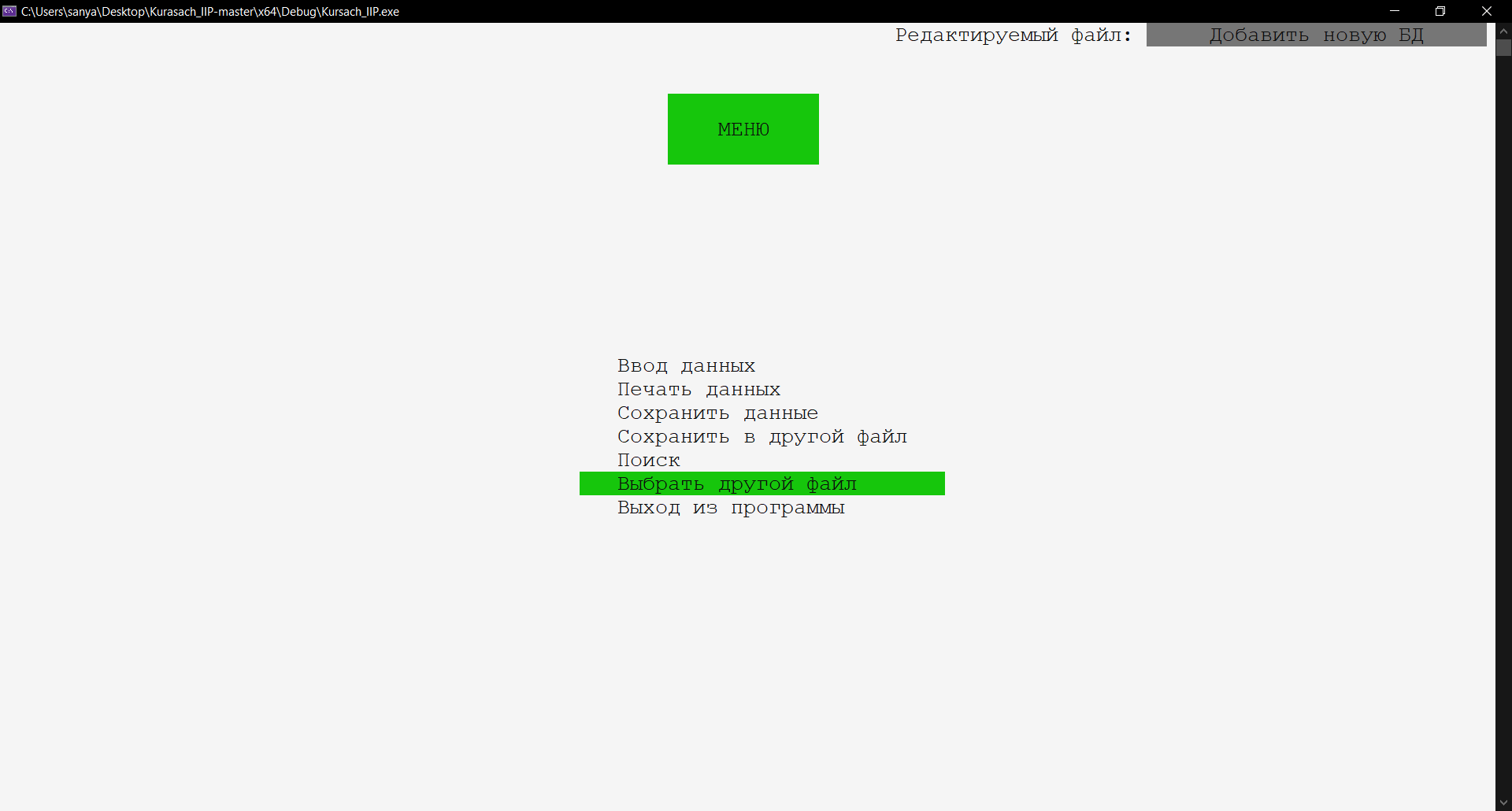


Рисунок 3.2 – Основное меню

При попытке посмотреть, что хранится в конкретном пункте меню будет выведено сообщение о пустом списке (рисунок 3.3).

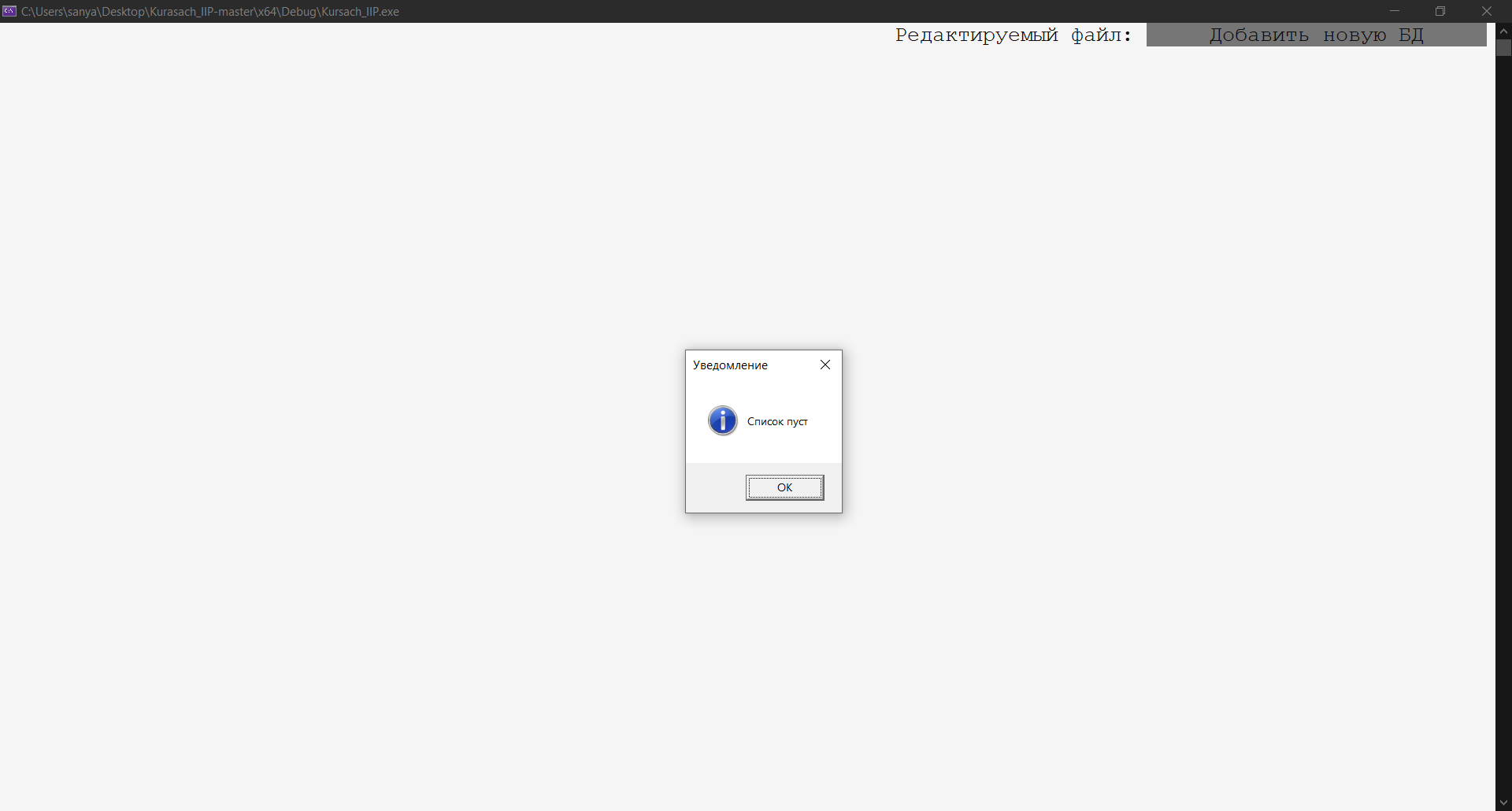


Рисунок 3.3 – Сообщение о пустом списке

Попробуем добавить новые данные в файл (рисунок 3.4).

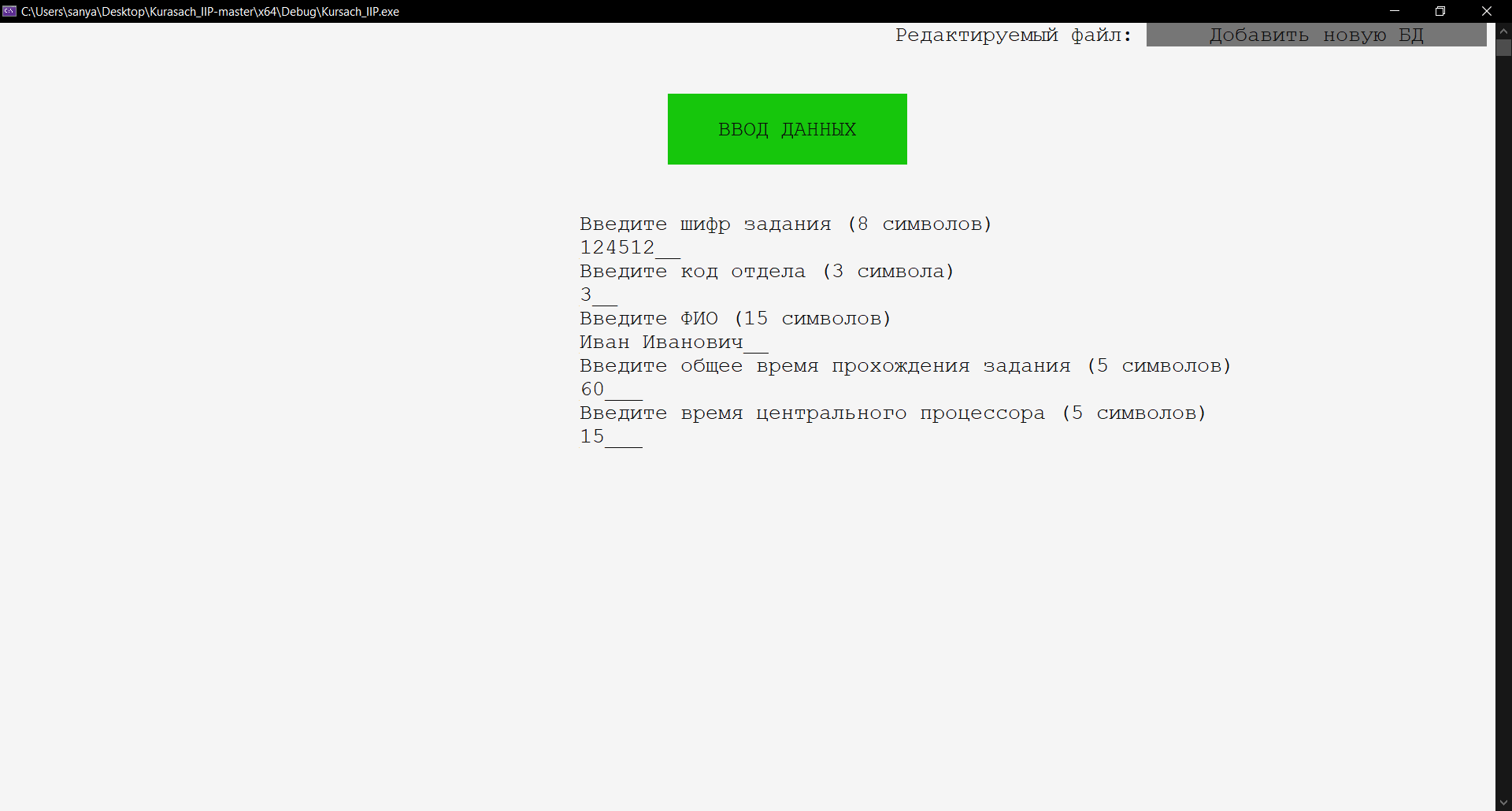


Рисунок 3.4 – Добавляем новые данные в файл

Теперь мы можем распечатать данные (рисунок 3.5)

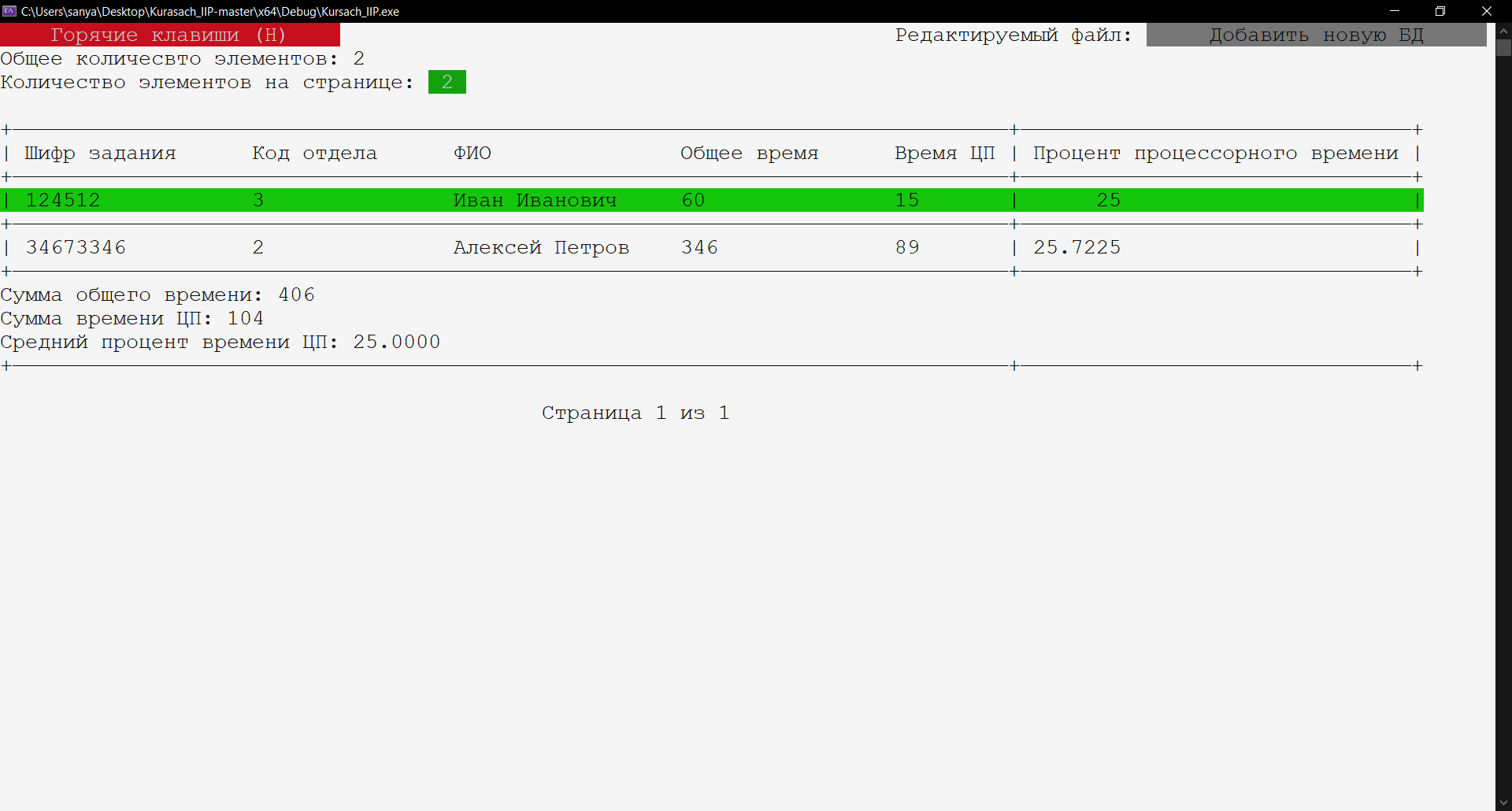


Рисунок 3.5 – Печать данных

Нажав на клавишу H/Р откроется блок с горячими клавишами (рисунок 3.6).

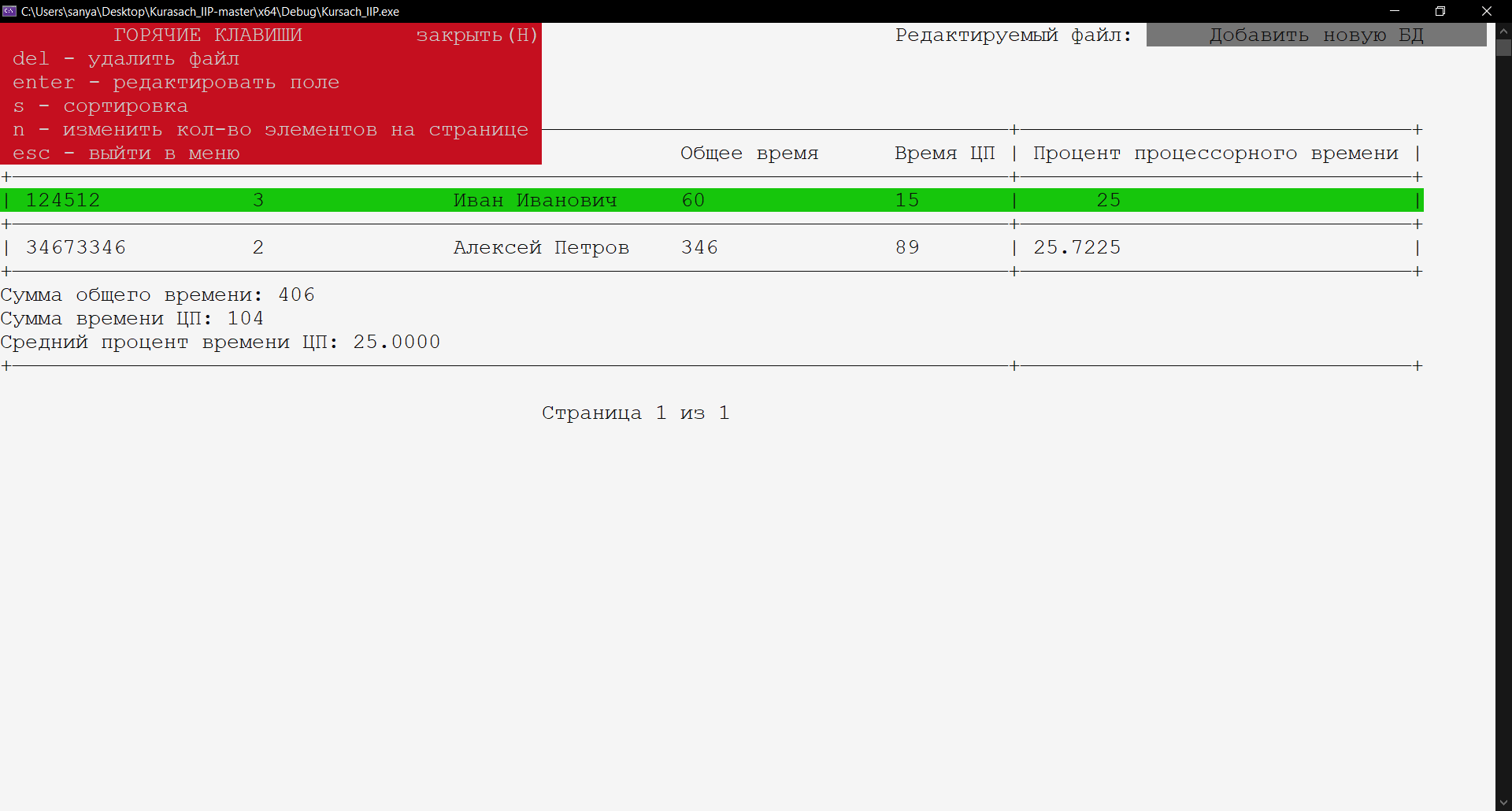


Рисунок 3.6 – Горячие клавиши

Выбрав поле и нажав клавишу enter, мы сможем отредактировать поле (рисунок 3.7).

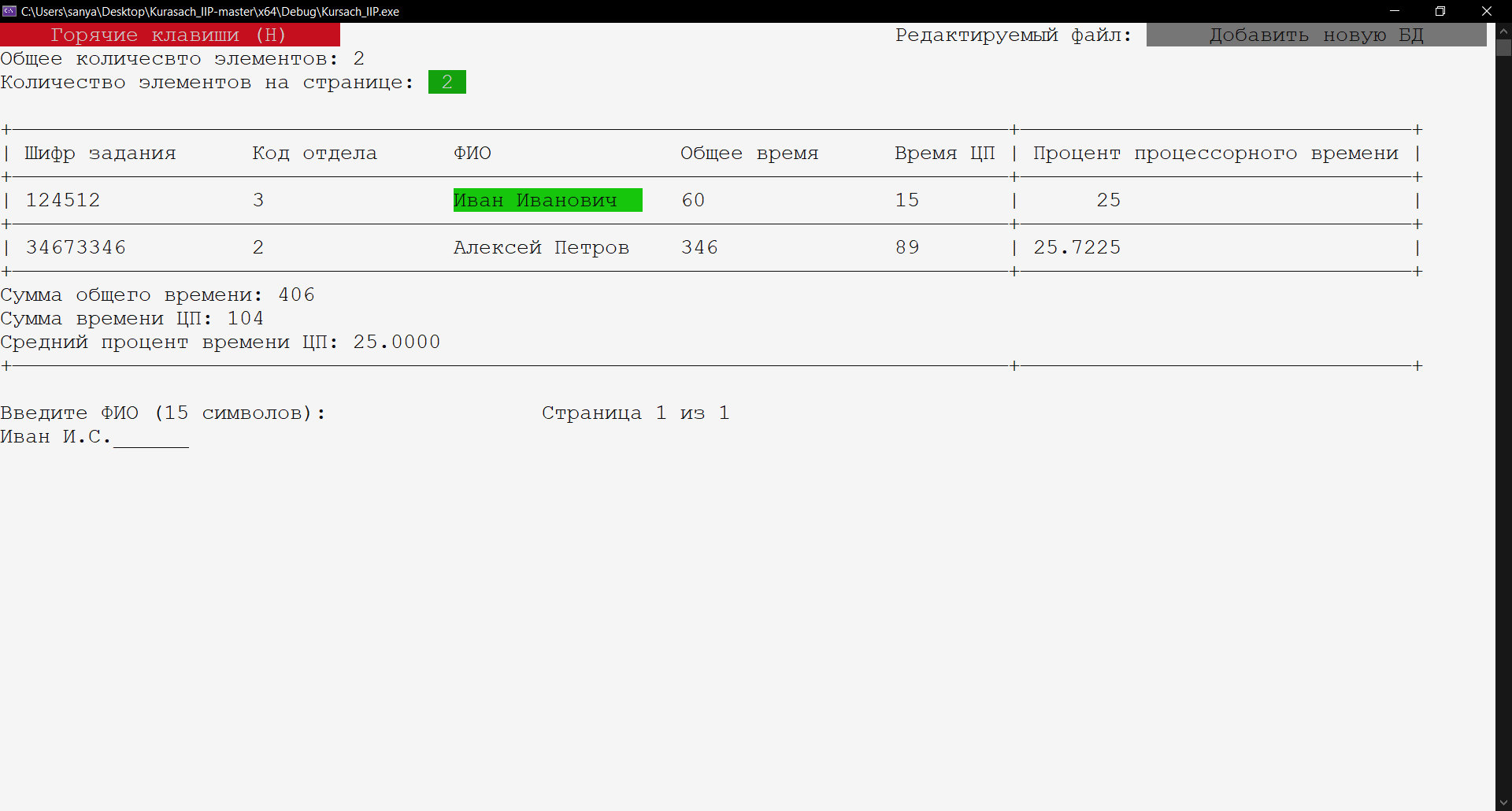


Рисунок 3.7 – Редактирование поля

Нажав на клавишу delete, мы сможем удалить поле из списка (рисунок 3.8)

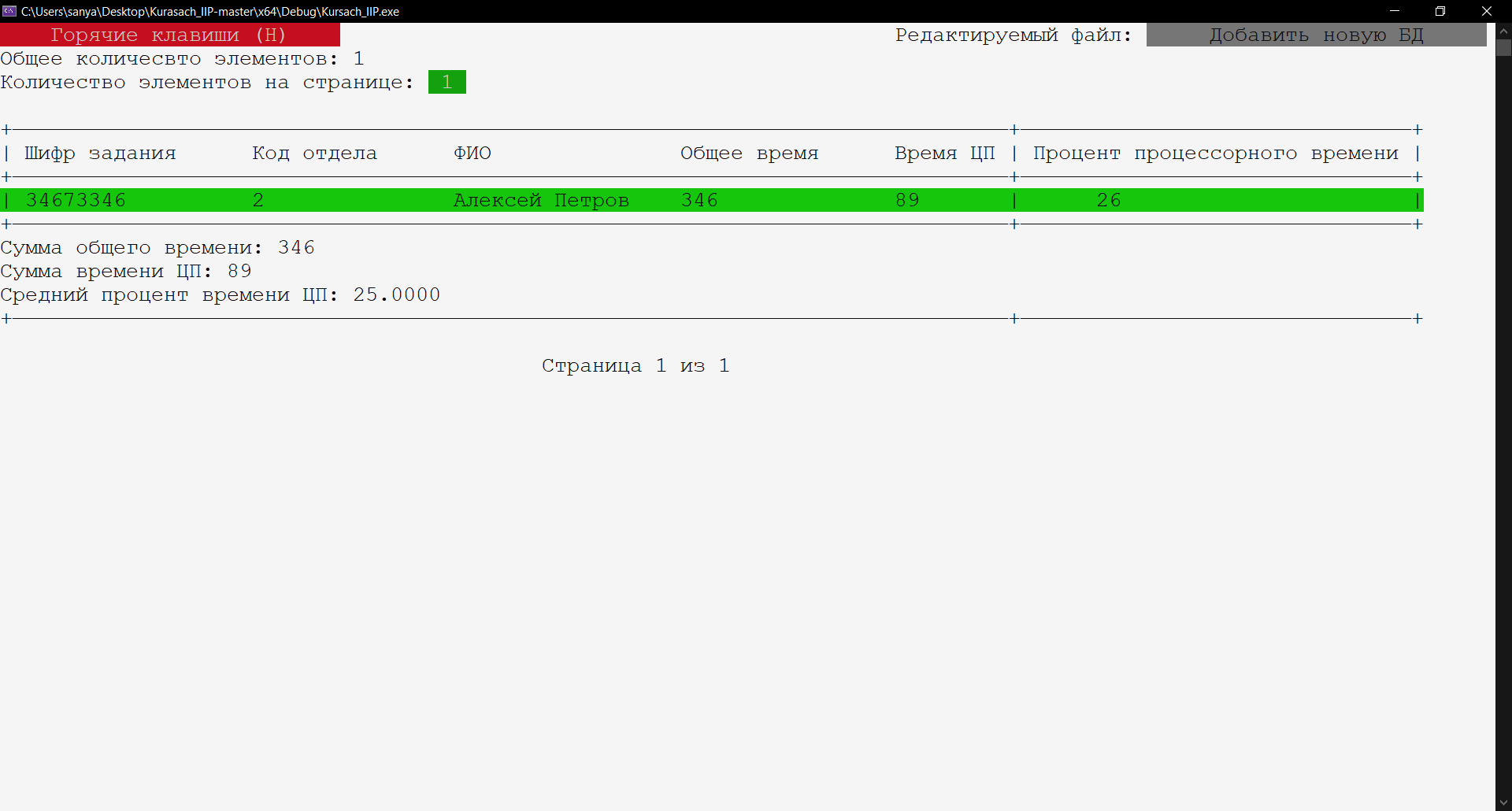


Рисунок 3.8 – Удаление поля

Данные можно отсортировать в две стороны, нажав клавишу s/ы. Введём больше пользователей и отсортируем их по общему времени (рисунок 3.9).

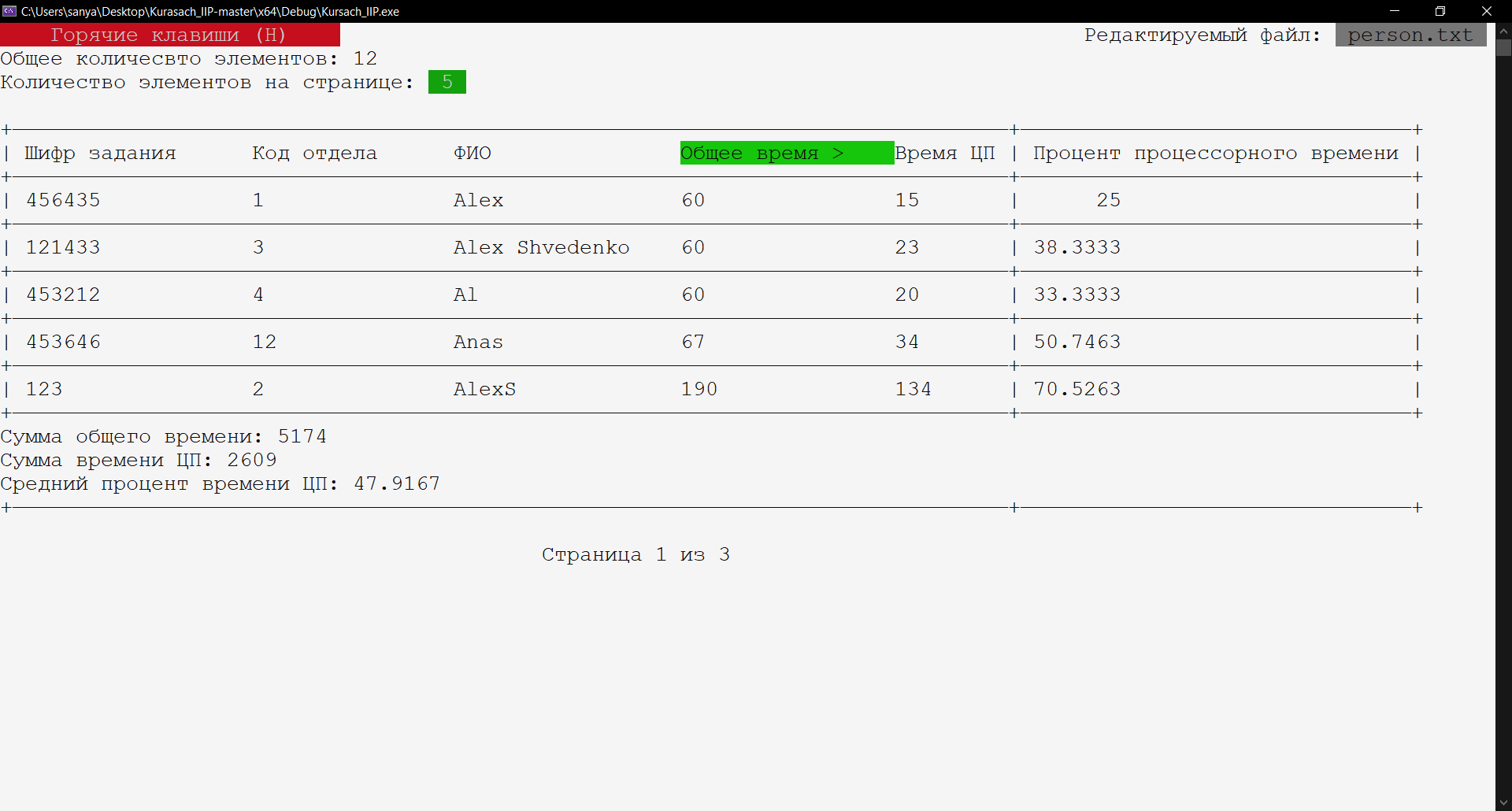


Рисунок 3.9 – Сортировка данных

Также можно изменять количество выводимых пользователей на одной странице (рисунок 3.10). Это делается с помощью клавиши n/т. Значения должны быть в диапазоне от 2 до 10.

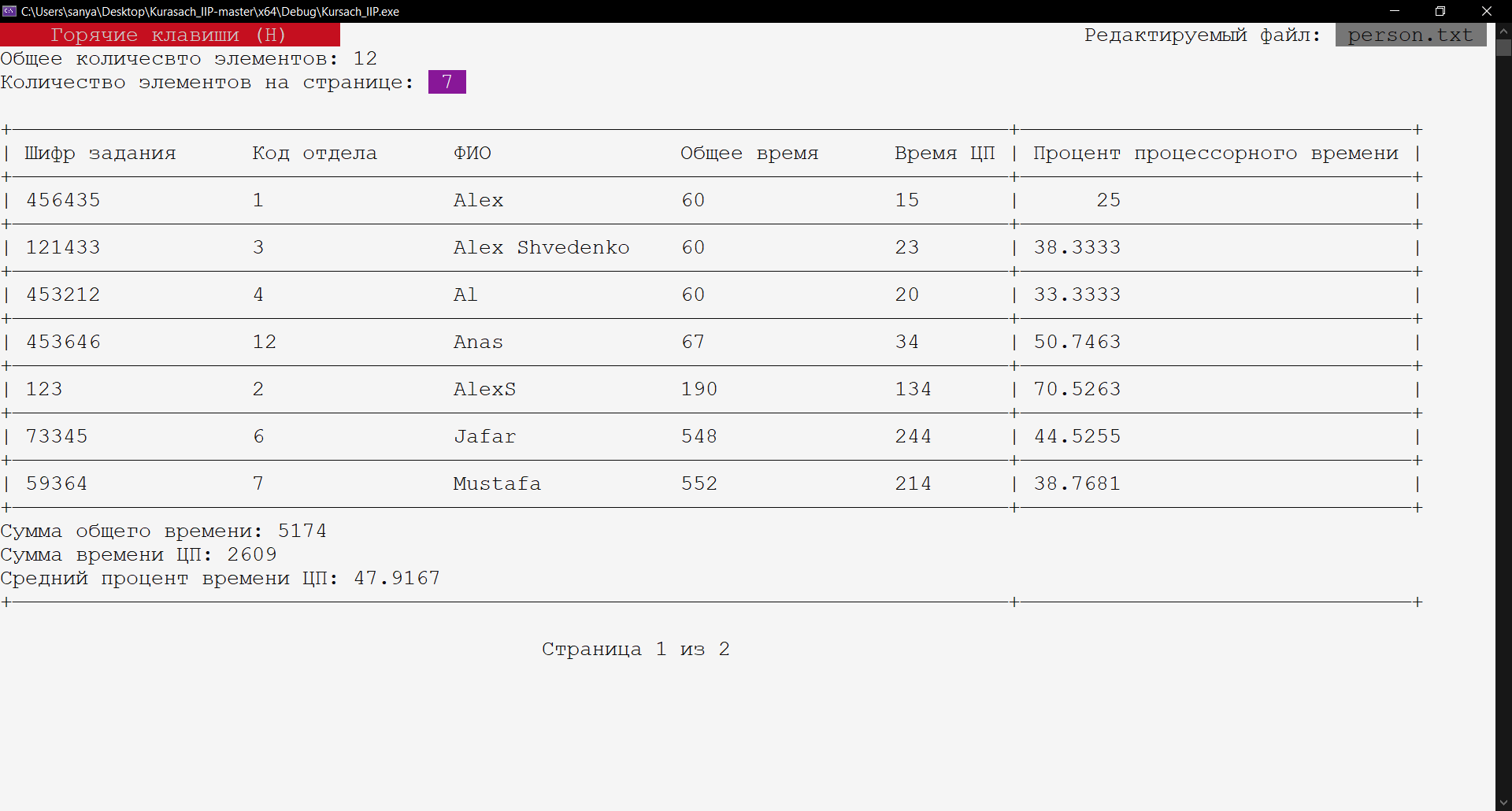


Рисунок 3.10 – Изменение количество выводимых пользователей на одной странице

Для нахождения каких-либо данных можно воспользоваться поиском (рисунок 3.11). В результате поиска сформируется таблица с найденными данными (рисунок 3.12).

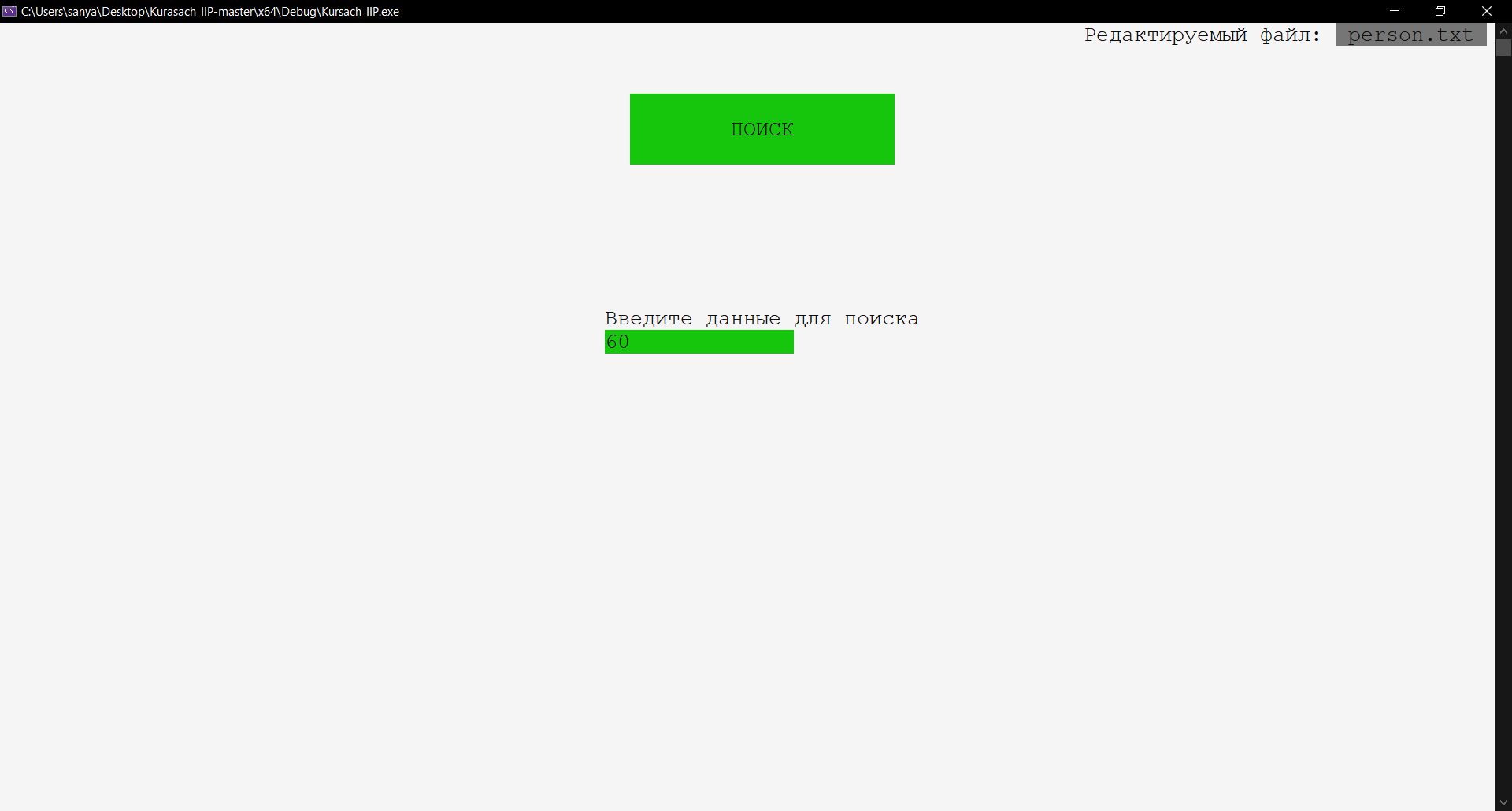


Рисунок 3.11 – Ввод данных для поиска

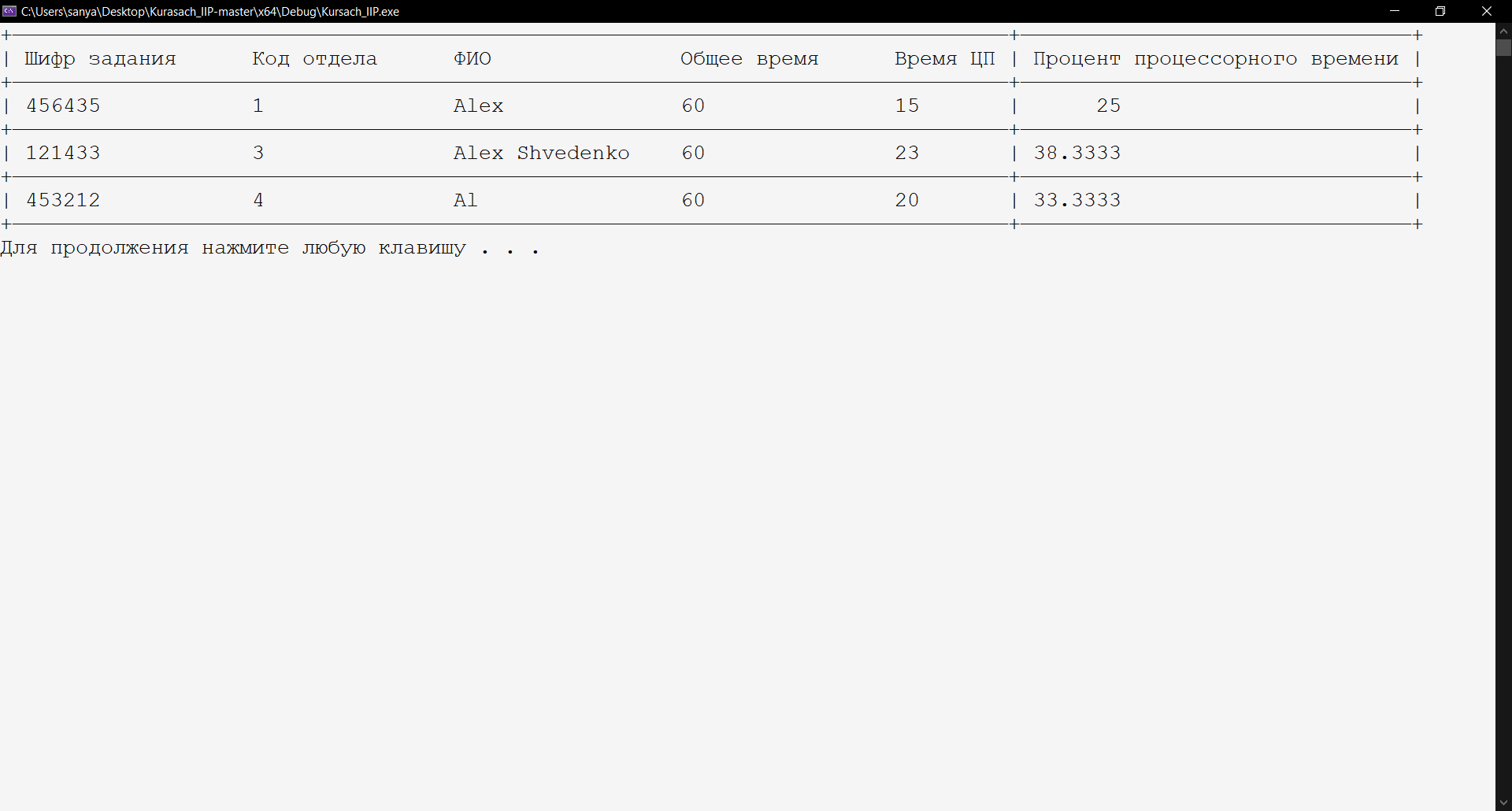


Рисунок 3.12 – Результат поиска

С помощью пункта меню «Сохранить данные» можно сохранить результаты работы с таблицей в текущий файл (рисунок 3.13).

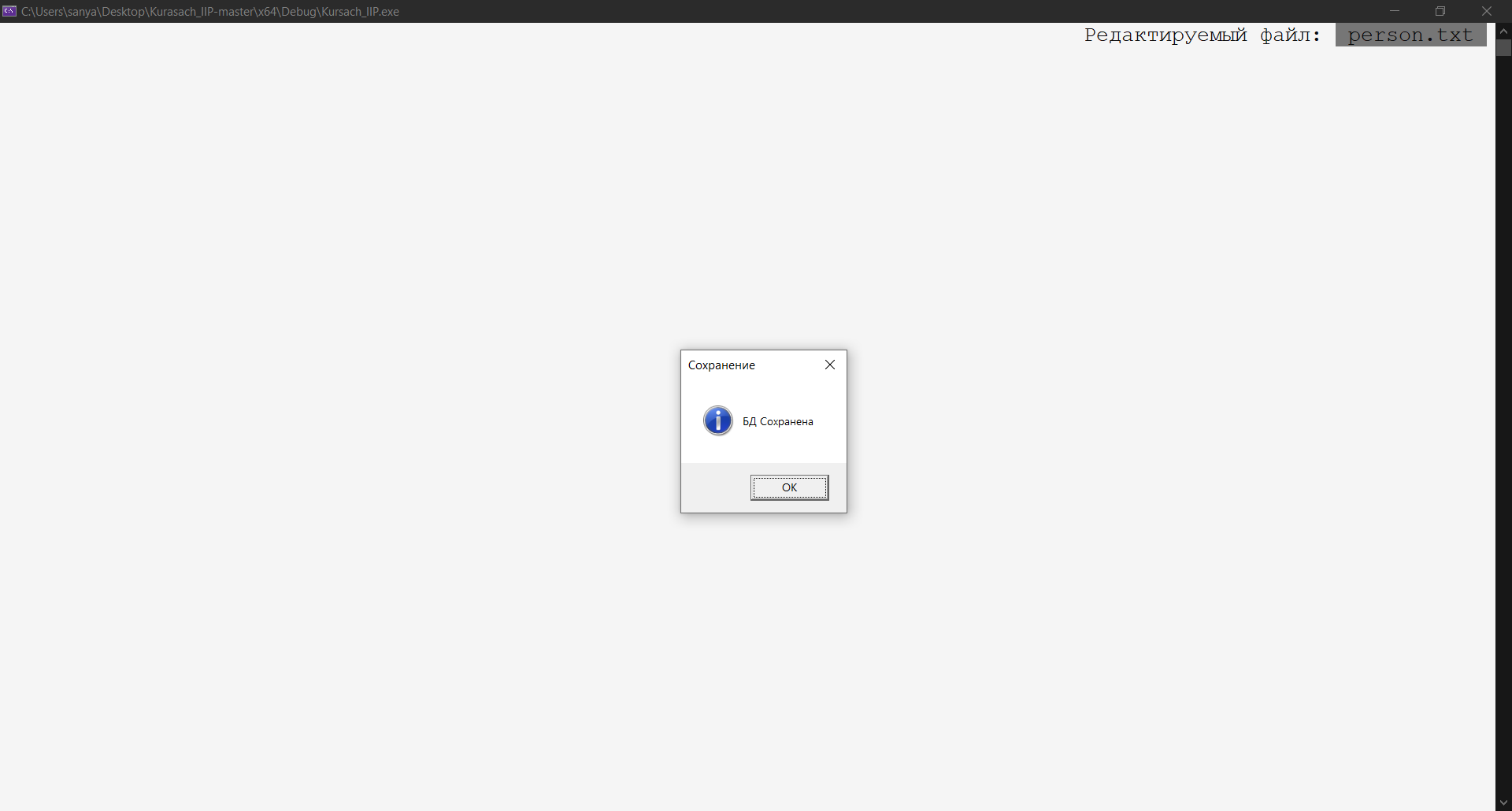


Рисунок 3.13 – Сохранение данных в текущий файл

Все данные можно сохранить в другой файл выбрав пункт меню «Сохранить данные в другой файл». На экран будет выведено меню с выбором типа файла (рисунок 3.14)

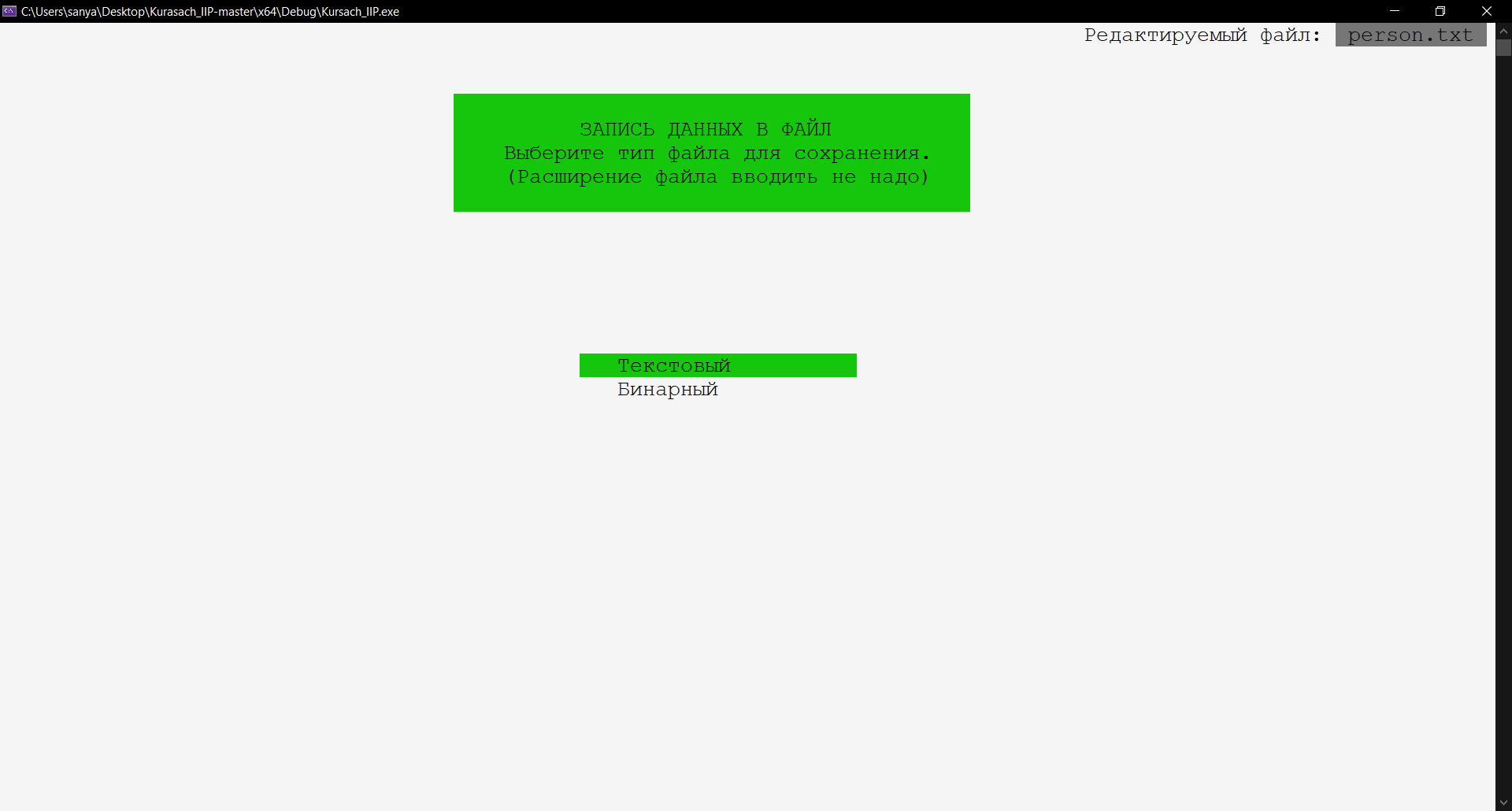


Рисунок 3.14 – Выбор типа файла для сохранения

При выборе одно из типов будет предложено ввести имя файла, оно может быть любым. Расширение файла после имени вводить не надо (.txt, .bin).

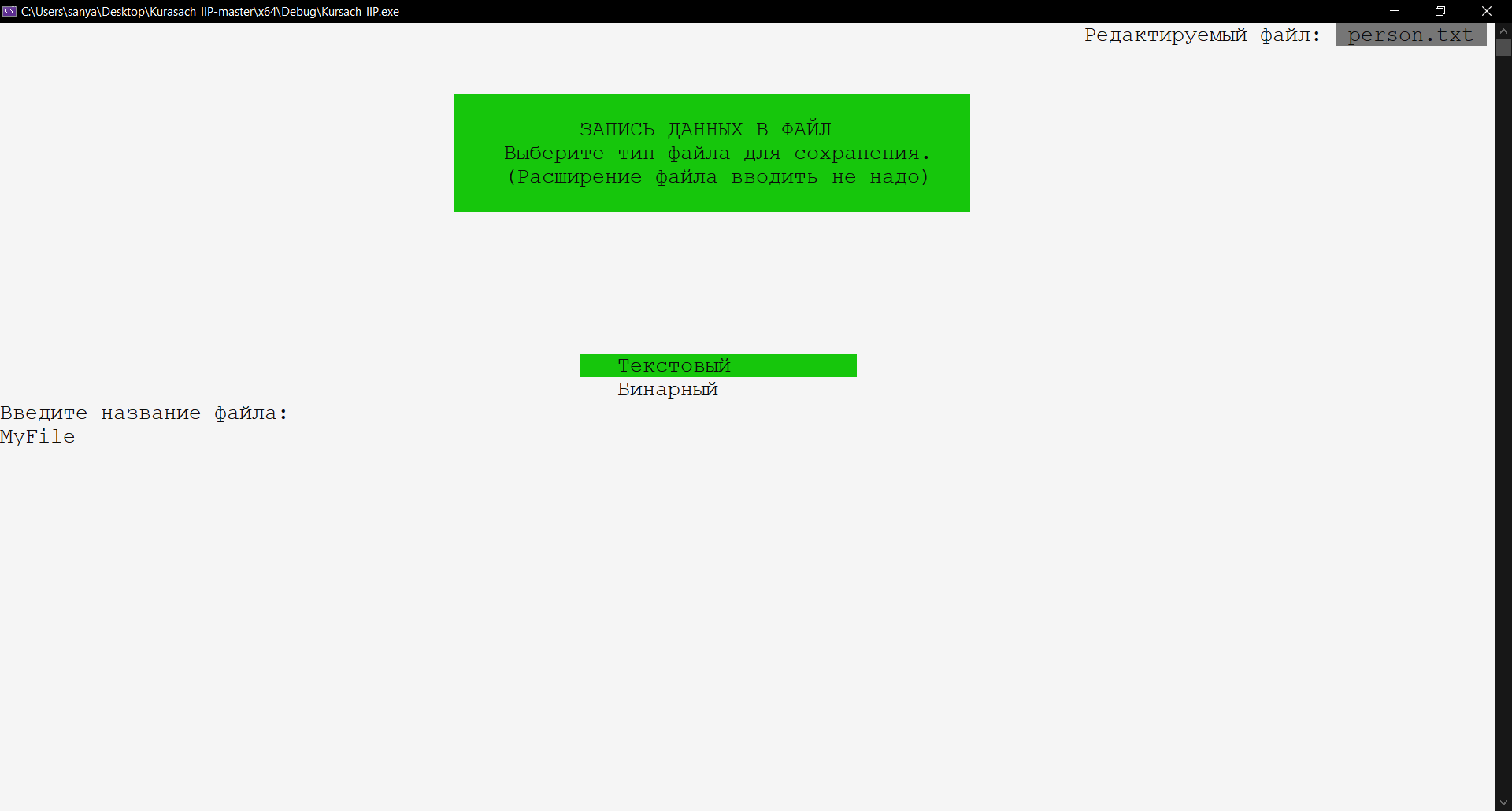


Рисунок 3.15 – Ввод названия файла для сохранения

# Заключение

В результате работы была написана программа, в основе алгоритма которой положена структура данных в виде двунаправленного списка, позволяющая выполнять просмотр данных в двух направлениях. База данных способна хранить большое количество записей и выдавать по запросу любую из них. Реализовано меню, дающее возможность пользователю взаимодействовать с программой и выполнять определенные действия над ней, такие как: удаление, просмотр данных, редактирование, поиск, запись в бинарный и текстовый файл, а также чтения с них.   
Таким образом цель курсового проекта была достигнута.   
Программное средство должно автоматизировать и упростить работу пользователя. Она может применяться не только индивидуальными пользователями, но и предпринимателями малых и средних корпораций, которые хотят систематизировать данные о времени выполнения заданий на ЭВМ, а также суммы по видам времени по всем заданиям и средний процент времени центрального процессора по всем заданиям, после соответствующей доработки.

# Список использованных источников

1. ГОСТ 2.105-95 Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам (с Изменением N 1, с Поправками), ГОСТ от 08 августа 1995 года №2.105-95 <http://docs.cntd.ru/document/1200001260>
2. Методические указания к курсовому проектированию по дисциплине «Информатика и программирование» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика»
3. Вирт, Н. Алгоритмы и структуры данных. Новая версия для Оберона [Электронный ресурс]: учебное пособие / Н. Вирт. — Электрон. дан. — Москва: ДМК Пресс, 2010. — 272 с. — Режим доступа: https://e.lanbook.com/book/1261. — Загл. с экрана.

# Приложение А

**Текст программы**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <string>

#include <windows.h>

#include <conio.h>

#include <stdio.h>

#include <cwchar>

#include <winuser.h>

#include <tchar.h>

using namespace std;

//======================

// ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

//======================

string filename;

string All\_bd = "mainBD.txt";

float total\_el = 0;

int num\_pages = 5, // Кол-во элементов на одной странице

width = 0, // Ширина окна

height = 0, // Высота окна

sum\_all\_time = 0, // Сумма общего времени

sum\_time\_cpu = 0, // Сумма времени ЦП

average\_percent\_time\_cpu = 0; // Средний процент процессорного времени

//===================

// КОНСТАНТЫ

//===================

// коды клавиш

const int up = 72,

down = 80,

right\_btn = 77,

left\_btn = 75,

enter = 13,

esc = 27,

del = 83;

const int SIZE\_arr\_filename = 50;

// названия пунктов

const string items[7] = {

" Ввод данных ",

" Печать данных ",

" Сохранить данные ",

" Сохранить в другой файл ",

" Поиск ",

" Выбрать другой файл ",

" Выход из программы " };

// названия для сортировки

const string sort\_items[5] = {

"| Шифр задания ",

"Код отдела ",

"ФИО ",

"Общее время ",

"Время ЦП" };

// массив всех названий файлов

string arr\_filename[SIZE\_arr\_filename] = {

" Добавить новую БД "

};

//===================

// ДАННЫЕ

//===================

struct info {

string cipher;

string department\_code;

string fio;

string all\_time;

string time\_cpu;

};

//===================

// СПИСОК

//===================

struct time\_task {

info d;

time\_task\* next = 0;

time\_task\* prev = 0;

};

//===================

// ИНТЕРФЕЙС

//===================

time\_task\* print(time\_task\* end, time\_task\* real\_beg ,time\_task\* beg, int active, int edit\_el, int print\_count\_num\_pages, int print\_page); // ВЫВОД ДАННЫХ

void print\_info(const time\_task& t, int active); // ПЕЧАТЬ СОДЕРЖИМОГО

void print\_menu(int sym, const string items[], const int N\_ITEMS); // ШАБЛОН ПЕЧАТИ МЕНЮ

void input(time\_task\*& beg, time\_task\*& end, const time\_task& info); // ВЫДЕЛЕНИЕ ПАМЯТИ

time\_task input\_info(time\_task\* beg); // ВВОД ДАННЫХ

time\_task\* delete\_el(time\_task\* beg, int num\_del); // УДАЛЕНИЕ

int read\_file(string filename, time\_task\*\* beg, time\_task\*\* end); // ЧТЕНИЕ ИЗ ФАЙЛА

int read\_bin\_file(string filename, time\_task\*\* beg, time\_task\*\* end); // ЧТЕНИЕ ИЗ БИНАРНОГО ФАЙЛА

int write\_file(time\_task\* temp); // ЗАПИСЬ В ФАЙЛ

void write\_filetype(time\_task\* temp, string filename, int el, int filetype); // ОПРЕДЕЛЕНИЕ ТИПА ДЛЯ ЗАПИСИ

int menu(int& active, const string items[], int num\_el); // МЕНЮ

void SetColor(int text, int bg); // установка цвета текста и фона

void find(time\_task\* beg); // ПОИСК

void edit(time\_task\* end, time\_task\* real\_beg, time\_task\* beg, int active, time\_task\* \_edit\_ob, int edit\_count\_num\_pages, int edit\_page, int edit\_i); // РЕДАКТИРОВАНИЕ ЭЛЕМЕНТА

void cls(); // ОЧИСТКА ЭКРАНА БЕЗ МЕРЦАНИЯ

void sort(time\_task\* beg, int field\_for\_sort, int sort\_direction); // СОРТИРОВКА

int compare(time\_task\* t\_i, time\_task\* t\_j, int num, int compare\_direction); // СРАВНЕНИЕ ДАННЫХ ДЛЯ СОРТИРОВКИ

void gotoxy(int xpos, int ypos); // ПЕРЕМЕЩЕНИЕ КУРСОРА НА ВЫБРАННУЮ ПОЗИЦИЮ

time\_task\* first\_start(time\_task\*\* beg, time\_task\*\* end); // ВЫБОР БД

string sets(size\_t size); // АНАЛОГ setw()

float percent\_time\_cpu(float a, float b); // РАСЧЁТ ПРОЦЕНТА ПРОЦЕССОРНОГО ВРЕМЕНИ

void show\_filename(size\_t posX, size\_t posY); // ПОКАЗЫВАЕМ КАКОЙ РЕДАКТИРУЕТСЯ ФАЙЛ

void show\_cursor(bool show); // ВИДИМОСТЬ КУРСОРА

string check\_num(string field, int posX, int posY, int max\_length, int is\_text); // ПРОВЕРКИ НА СИМВОЛ И ДЛИНУ СТРОКИ

//===================

// ОСНОВНАЯ ПРОГРАММА

//===================

int main() {

//========================

//========================

//========================

HANDLE hCon;

// вытаскиваем ширину и высоту

hCon = GetStdHandle(-12);

CONSOLE\_SCREEN\_BUFFER\_INFO consoleInfo;

if (GetConsoleScreenBufferInfo(hCon, &consoleInfo))

{

width = consoleInfo.srWindow.Right - consoleInfo.srWindow.Left + 1;

height = consoleInfo.srWindow.Bottom - consoleInfo.srWindow.Top + 1;

}

// меняем размер шрифта

CONSOLE\_FONT\_INFOEX cfi;

cfi.cbSize = sizeof(cfi);

cfi.nFont = 0;

cfi.dwFontSize.X = 0; // Width of each character in the font

cfi.dwFontSize.Y = 24; // Height

cfi.FontFamily = FF\_DONTCARE;

cfi.FontWeight = FW\_NORMAL;

SetCurrentConsoleFontEx(GetStdHandle(STD\_OUTPUT\_HANDLE), FALSE, &cfi);

//========================

//========================

//========================

SetColor(0, 15); // устанавливаем цвет текста и заднего фона чёрным

ShowWindow(GetConsoleWindow(), SW\_MAXIMIZE); // полноэкранный режим

show\_cursor(FALSE); // убираем курсор

int item = 0,

current = 1;

time\_task\* beg = 0,

\* end = 0;

first\_start(&beg, &end);

while (1) {

show\_cursor(FALSE); // убираем курсор

system("cls");

show\_filename(width - filename.length() - 9, 0);

// выводим название раздела

SetColor(0, 10);

gotoxy(width / 2 + 1, 3);

cout << " ";

gotoxy(width / 2 + 1, 4);

cout << " МЕНЮ ";

gotoxy(width / 2 + 1, 5);

cout << " ";

switch (menu(current, items, 7)) {

// Добавление элемента в список

case 1:

system("cls");

show\_filename(width - filename.length() - 9, 0);

input(beg, end, input\_info(beg));

break;

// Печать элементов

case 2:

system("cls");

show\_filename(width - filename.length() - 9, 0);

beg = print(end, beg, beg, 1, 0, 1, 0);

break;

// Запись в файл

case 3:

system("cls");

show\_filename(width - filename.length() - 9, 0);

write\_filetype(beg, filename, -1, filename[filename.length() - 3] == 't' ? 1 : 0);

break;

// Запись в другой файл

case 4:

system("cls");

show\_filename(width - filename.length() - 9, 0);

write\_file(beg);

break;

// Поиск элемента

case 5:

system("cls");

show\_filename(width - filename.length() - 9, 0);

find(beg);

break;

// выбор другого файла

case 6:

if (beg) {

if (MessageBox(0, L"Хотите сохранить данные?", L"Сохранение", MB\_ICONQUESTION | MB\_SETFOREGROUND | MB\_YESNO) == 6) {

write\_filetype(beg, filename, -1, filename[filename.length() - 3] == 't' ? 1 : 0);

}

}

beg = end = first\_start(&beg, &end);

break;

case 7:

case -1:

if (MessageBox(0, L"Вы уверены, что хотите выйти?", L"Уведомление", MB\_ICONQUESTION | MB\_SETFOREGROUND | MB\_YESNO) == 6) {

return 0;

} else break;

}

}

}

//===================

// ФУНКЦИИ

//===================

// =========ПОКАЗЫВАЕМ КАКОЙ РЕДАКТИРУЕТСЯ ФАЙЛ=========

void show\_filename(size\_t posX, size\_t posY) {

gotoxy(posX, posY);

cout << "Редактируемый файл: ";

SetColor(0, 8);

cout << " " << filename << " ";

SetColor(0, 15);

gotoxy(0, 0);

}

// =========ВИДИМОСТЬ КУРСОРА=========

void show\_cursor(bool show) {

HANDLE handle = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO structCursorInfo;

GetConsoleCursorInfo(handle, &structCursorInfo);

structCursorInfo.bVisible = show;

SetConsoleCursorInfo(handle, &structCursorInfo);

}

// ==========ВЫБОР БД (первый запуск)==========

time\_task\* first\_start(time\_task\*\* beg, time\_task\*\* end) {

int k = 1,

current = 1,

fl = 0,

new\_line = 0;

num\_pages = 5;

total\_el = 0;

while (1) {

k = 1;

int exit\_fl = 0;

// +++++++ОТКРЫВАЕМ ФАЙЛ СО ВСЕМИ БД+++++++

ifstream fin;

fin.open(All\_bd);

if (!fin) {

MessageBox(0, L"Невозможно открыть файл!", L"Ошибка", MB\_ICONERROR | MB\_SETFOREGROUND);

return 0;

}

// записываем считанные данные в массив

while (getline(fin, arr\_filename[k++])) {

if (k >= 50) {

MessageBox(0, L"Выведены первые 50 файлов!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

}

}

fin.close();

// ++++++++++++++++++++++++++++++++++++++++++

system("cls");

gotoxy(0, 3);

SetColor(7, 8);

cout << " ГОРЯЧИЕ КЛАВИШИ " << endl;

SetColor(0, 15);

cout << "del - удалить файл" << endl;

cout << "enter - выбрать файл" << endl;

cout << "esc - выйти из программы" << endl;

// выводим название раздела

SetColor(0, 10);

gotoxy(width / 2 - 12, 3);

cout << sets(36);

gotoxy(width / 2 - 12, 4);

cout << " ВЫБОР ФАЙЛА ";

gotoxy(width / 2 - 12, 5);

cout << " выберите откуда считывать данные ";

gotoxy(width / 2 - 12, 6);

cout << sets(36);

size\_t pos = 0;

int main\_bd = menu(current, arr\_filename, k - 1);

// если нажали delete

if (main\_bd <= -2) {

exit\_fl = 1;

if (MessageBox(0, L"Вы уверены, что хотите удалить?", L"Удаление", MB\_ICONQUESTION | MB\_YESNO | MB\_SETFOREGROUND) == 6) {

ofstream file\_All\_bd(All\_bd);

main\_bd = abs(main\_bd);

if (main\_bd == k - 1) current--; // если удаляется последний элемент

remove(arr\_filename[main\_bd - 1].c\_str());

// удаляем элемент из массива

for (int i = 1; i < k; i++) {

if (i == main\_bd - 1) {

for (int j = i; j < k - 1; j++) arr\_filename[j] = arr\_filename[j + 1];

k--;

break;

}

}

// запись новых данных в файл

for (int i = 1; i < k - 1; i++) {

file\_All\_bd << arr\_filename[i] << endl;

}

file\_All\_bd.close();

}

} else if (main\_bd == -1) { // если нажали esc

if (MessageBox(0, L"Вы уверены, что хотите выйти?", L"Уведомление", MB\_ICONQUESTION | MB\_SETFOREGROUND | MB\_YESNO) == 6) {

exit(0);

} else exit\_fl = 1;

}

else {

filename = arr\_filename[main\_bd - 1];

if (main\_bd - 1 != 0) {

pos = filename.find(".txt", filename.length() - 4); // ищем .txt в названии файла, если вернёт -1, то это бинарный

if (pos != -1) {

read\_file(filename, beg, end);

}

else {

read\_bin\_file(filename, beg, end);

}

// считаем сумму общего времени и сумму времени ЦП

sum\_all\_time = sum\_time\_cpu = average\_percent\_time\_cpu = 0;

for (time\_task\* temp = \*beg; temp; temp = temp->next) {

sum\_all\_time += stoi(temp->d.all\_time);

sum\_time\_cpu += stoi(temp->d.time\_cpu);

average\_percent\_time\_cpu += percent\_time\_cpu(stof(temp->d.all\_time), stof(temp->d.time\_cpu));

}

return \*beg;

}

else return 0;

}

if (exit\_fl == 0) return 0;

}

}

// ==========ПЕРЕМЕЩЕНИЕ КУРСОРА НА ВЫБРАННУЮ ПОЗИЦИЮ==========

void gotoxy(int xpos, int ypos)

{

COORD scrn;

HANDLE hOuput = GetStdHandle(STD\_OUTPUT\_HANDLE);

scrn.X = xpos; scrn.Y = ypos;

SetConsoleCursorPosition(hOuput, scrn);

}

// ==========ОЧИСТКА СТРОКИ==========

void clearRow(int row)

{

DWORD a;

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE); // получаем хэндл окна консоли\*/

COORD coord = { 0, row - 1 }; // получаем координаты строки для очистки

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

GetConsoleScreenBufferInfo(hStdOut, &csbi); // получаем данные из буфера вывода консоли

FillConsoleOutputCharacter(hStdOut, ' ', width, coord, &a); // заполняем строку пробелами

}

// ==========ВЫДЕЛЕНИЕ ПАМЯТИ==========

void input(time\_task\*& beg, time\_task\*& end, const time\_task& info) {

if (info.d.cipher == "-1" || info.d.all\_time == "-1" || info.d.department\_code == "-1" || info.d.fio == "-1" || info.d.time\_cpu == "-1") return;

time\_task\* newel = new time\_task;

newel->next = NULL;

newel->prev = NULL;

newel->d = info.d;

if (!beg) {

beg = end = newel;

}

else {

end->next = newel;

newel->prev = end;

end = newel;

num\_pages = 5;

}

}

// ==========ПРОВЕРКИ НА СИМВОЛ И ДЛИНУ СТРОКИ==========

string check\_num(string field, int posX, int posY, int max\_length, int is\_text) {

/\*

если is\_text = 0 - ввод чисел

если is\_text = 1 - ввод символов

если is\_text = 2 - ввод чисел и ввод символов

\*/

int length = 0;

int pospos = 0;

field = "";

int\* posarrays = new int[(\_\_int64)max\_length + 1];

for (int i = 0; i < max\_length; i++) {

posarrays[i] = i;

if(max\_length != i) field += (is\_text == 2 ? " " : "\_"); // создаём маску

}

int pos = posarrays[pospos];

cout << field;

while (true) {

int ch = \_getch();

if (ch == enter && length >= 1) {

if (is\_text == 0 && stoi(field) <= 0) { // если поле <= 0

MessageBox(0, L"Данные должны быть >= 1!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

} else break; // выходим, если нажали enter

}

if (ch == esc) return "-1";

if (ch == 8 && length >= 1) { // если нажали backspace

length--;

gotoxy(posX, posY);

pospos--;

pos = posarrays[pospos];

field[pos] = (is\_text == 2 ? ' ' : '\_');

cout << field;

}

else if (length != max\_length && ch != esc && ch != enter && is\_text == 2 && ch != 8) {

length++;

field[pos] = ch;

gotoxy(posX, posY);

pospos++;

pos = posarrays[pospos];

cout << field;

} else if (length != max\_length && ch != esc && ch != enter && ch != 8 && (is\_text == 1 ? !(ch >= '0' && ch <= '9') : (ch >= '0' && ch <= '9'))) {

length++;

field[pos] = ch;

gotoxy(posX, posY);

pospos++;

pos = posarrays[pospos];

cout << field;

}

}

delete[] posarrays;

return field.substr(0, length);

}

// ==========ВВОД ДАННЫХ==========

time\_task input\_info(time\_task \* beg) {

time\_task t;

time\_task\* temp = beg;

int fl = 0;

// выводим название раздела

SetColor(0, 10);

gotoxy(width / 2 + 1, 3);

cout << " ";

gotoxy(width / 2 + 1, 4);

cout << " ВВОД ДАННЫХ ";

gotoxy(width / 2 + 1, 5);

cout << " ";

SetColor(0, 15);

gotoxy(width / 2 - 6, 8);

cout << "Введите шифр задания (8 символов)" << endl;

gotoxy(width / 2 - 6, 9);

t.d.cipher = check\_num(t.d.cipher, width / 2 - 6, 9, 8, 0);

if (t.d.cipher == "-1") return t;

gotoxy(width / 2 - 6, 10);

cout << "Введите код отдела (3 символа)" << endl;

gotoxy(width / 2 - 6, 11);

t.d.department\_code = check\_num(t.d.department\_code, width / 2 - 6, 11, 3, 0);

if (t.d.department\_code == "-1") return t;

gotoxy(width / 2 - 6, 12);

cout << "Введите ФИО (15 символов)" << endl;

gotoxy(width / 2 - 6, 13);

t.d.fio = check\_num(t.d.fio, width / 2 - 6, 13, 15, 1);

if(t.d.fio == "-1") return t;

do {

gotoxy(width / 2 - 6, 14);

cout << "Введите общее время прохождения задания (5 символов)" << endl;

gotoxy(width / 2 - 6, 15);

t.d.all\_time = check\_num(t.d.all\_time, width / 2 - 6, 15, 5, 0);

if (t.d.all\_time == "-1") return t;

gotoxy(width / 2 - 6, 16);

cout << "Введите время центрального процессора (5 символов)" << endl;

gotoxy(width / 2 - 6, 17);

t.d.time\_cpu = check\_num(t.d.time\_cpu, width / 2 - 6, 17, 5, 0);

if (t.d.time\_cpu == "-1") return t;

if (stoi(t.d.all\_time) < stoi(t.d.time\_cpu)) {

MessageBox(0, L"Общее время должно быть больше времени центрального процессора!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

clearRow(15);

clearRow(16);

clearRow(17);

clearRow(18);

gotoxy(0, 14);

}

else break;

} while (1);

total\_el++;

return t;

}

// ==========РАСЧЁТ ПРОЦЕНТА ПРОЦЕССОРНОГО ВРЕМЕНИ==========

float percent\_time\_cpu(float a, float b) {

return (b \* 100) / a;

}

// ==========АНАЛОГ setw()==========

string sets(size\_t size) {

string res\_s;

for (int i = 0; i < size; i++) {

res\_s += " ";

}

return res\_s;

}

// ==========ОЧИСТКА ЭКРАНА БЕЗ МЕРЦАНИЯ==========

void cls() {

HANDLE hd = GetStdHandle(STD\_OUTPUT\_HANDLE);

COORD cd;

cd.X = 0;

cd.Y = 0;

SetConsoleCursorPosition(hd, cd);

}

// ==========ПЕЧАТЬ СОДЕРЖИМОГО==========

void print\_info(const time\_task & t, int active) {

if (active) SetColor(0, 15);

if (active == 1) {

SetColor(0, 15);

SetColor(0, 10);

cout << "| " << t.d.cipher << sets(12 - t.d.cipher.length());

SetColor(0, 15);

cout << setw(6 + t.d.department\_code.length());

}

else {

cout << "| " << t.d.cipher << sets((13 - t.d.cipher.length()) + t.d.department\_code.length()) << sets(5 - t.d.department\_code.length());

}

if (active == 2) {

SetColor(0, 15);

SetColor(0, 10);

cout << t.d.department\_code << sets(10 - t.d.department\_code.length());

SetColor(0, 15);

cout << setw(6 + t.d.fio.length());

}

else {

cout << t.d.department\_code << sets(16 - t.d.department\_code.length());

}

if (active == 3) {

SetColor(0, 15);

SetColor(0, 10);

cout << t.d.fio << sets(15 - t.d.fio.length());

SetColor(0, 15);

cout << setw(3 + t.d.all\_time.length());

}

else {

cout << t.d.fio << sets(18 - t.d.fio.length());

}

if (active == 4) {

SetColor(0, 15);

SetColor(0, 10);

cout << t.d.all\_time << sets(11 - t.d.all\_time.length());

SetColor(0, 15);

cout << setw(6 + t.d.time\_cpu.length());

}

else {

cout << t.d.all\_time << sets(17 - t.d.all\_time.length());

}

if (active == 5) {

SetColor(0, 15);

SetColor(0, 10);

cout << t.d.time\_cpu << sets(9 - t.d.time\_cpu.length()) << "|";

SetColor(0, 15);

}

else {

cout << t.d.time\_cpu << setw(10 - t.d.time\_cpu.length()) << "|";

}

cout << setfill(' ') << setw(8) << percent\_time\_cpu(stof(t.d.all\_time), stof(t.d.time\_cpu)) << setfill(' ') << setw(24) << setprecision(4) << fixed << "|";

}

// ==========ВЫВОД ДАННЫХ==========

// если active = 0, то это режим сортировки

// если active = -1, то это изменение кол-ва элементов на странице

time\_task\* print(time\_task\* end, time\_task\* real\_beg, time\_task \* beg, int active, int edit\_el, int print\_count\_num\_pages, int print\_page) {

wint\_t buf;

time\_task\* temp = beg,

\* buf\_el = beg,

\* first\_buf\_el = beg,

\* edit\_ob = beg;

int i = 1, // номер текущего элемента

first\_i = 0, // первый элемент в каждой странице

count\_num\_pages = print\_count\_num\_pages, // счётчик для i (название не соответствует применению)

page = print\_page, // текущая страница

np = 0, // новая страница

sort\_field = 1, // поле для сортировки

remember\_active = 0, // запоминающая переменная для active

direction = 0, // направление сортировки (0 - от меньшего к большему, 1 - наоборот)

fl = 0,

show\_hotkey = 0; // видимость горячих клавиш

// пересчитываем сумму общего времени и сумму времени ЦП

if (edit\_el == 0) {

sum\_all\_time = sum\_time\_cpu = average\_percent\_time\_cpu = 0;

for (temp = beg; temp; temp = temp->next) {

sum\_all\_time += stoi(temp->d.all\_time);

sum\_time\_cpu += stoi(temp->d.time\_cpu);

average\_percent\_time\_cpu += percent\_time\_cpu(stof(temp->d.all\_time), stof(temp->d.time\_cpu));

}

}

do {

float total\_pages = ceil(total\_el / num\_pages); // общее кол-во страниц

cls();

show\_filename(width - filename.length() - 9, 0);

// если пустой список

if (!beg) {

MessageBox(0, L"Список пуст", L"Уведомление", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

return beg;

}

// +++++ОСНОВНОЙ ВЫВОД+++++

int num\_del = 0; // номер для удаления

if (total\_el < num\_pages) {

num\_pages = total\_el;

}

if (active > num\_pages || active == 0 || active == -1) {

temp = buf\_el;

edit\_ob = buf\_el;

} else {

temp = beg;

edit\_ob = beg;

}

gotoxy(0, 0);

cout << "\nОбщее количесвто элементов: " << total\_el << setw(20) << endl;

cout << "Количество элементов на странице: ";

if (active == -1) {

SetColor(15, 5);

} else SetColor(7, 2);

cout << " " << num\_pages << " " << endl << endl;

SetColor(0, 15);

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

if (active == 0) {

for (int i = 1; i <= 5; i++) {

if (i == sort\_field) {

SetColor(0, 10);

}

cout << sort\_items[i - 1];

if (i == sort\_field) {

if (direction == 0) {

cout << "<";

fl = 0;

}

else {

fl = 1;

cout << ">";

}

switch (i) {

case 1:

case 2:

case 4:

cout << sets(4);

break;

case 3:

cout << sets(13);

break;

case 5: // если это последний пункт

cout << "|";

break;

}

} else {

switch (i) {

case 1:

case 2:

case 4:

cout << sets(5);

break;

case 3:

cout << sets(14);

break;

case 5: // если это последний пункт

cout << " |";

break;

}

}

SetColor(0, 15);

}

} else {

cout << "| Шифр задания Код отдела ФИО Общее время Время ЦП |";

}

cout << " Процент процессорного времени |" << endl;

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

for (i = count\_num\_pages; temp; temp = temp->next, i++) {

// достаём первые элементы

if (i % num\_pages == 1 && page != 0) {

first\_i = i;

buf\_el = temp;

for (int j = 0; j < num\_pages; j++, temp = temp->prev) {

first\_buf\_el = temp->prev;

}

temp = buf\_el;

}

if (i % num\_pages != 0) {

np = 0;

}

// разукрашивание выбранного элемента

if (i == active) {

SetColor(0, 10);

num\_del = stoi(temp->d.cipher);

edit\_ob = temp; // редактируемый объект

print\_info(\*temp, edit\_el);

}

else {

print\_info(\*temp, 0);

}

SetColor(0, 15);

cout << endl;

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

if (i % num\_pages == 0 && np == 0) {

np = 1;

break;

}

}

cout << "Сумма общего времени: " << sum\_all\_time << endl;

cout << "Сумма времени ЦП: " << sum\_time\_cpu << endl;

cout << "Средний процент времени ЦП: " << average\_percent\_time\_cpu / total\_el << endl;

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl << endl;

cout << setw(width / 2) << "Страница " << page + 1 << " из " << setprecision(0) << total\_pages;

// +++++++++++++++++

// горячие клавиши

gotoxy(0, 0);

SetColor(7, 4);

if (show\_hotkey == 0) {

cout << " Горячие клавиши (H) " << endl;

}

else {

cout << " ГОРЯЧИЕ КЛАВИШИ закрыть(H)" << endl;

cout << " del - удалить файл " << endl;

cout << " enter - редактировать поле " << endl;

cout << " s - сортировка " << endl;

cout << " n - изменить кол-во элементов на странице " << endl;

cout << " esc - выйти в меню " << endl;

}

SetColor(0, 15);

int height\_el\_num\_pages = 0; // высота элементов

if ((page + 1 != total\_pages)) {

height\_el\_num\_pages = i - count\_num\_pages;

}

else {

height\_el\_num\_pages = i - count\_num\_pages - 1;

}

if (edit\_el) {

gotoxy(0, 14 + height\_el\_num\_pages + (height\_el\_num\_pages + 1));

return beg; // если редактируется какой-нибудь элемент, то выходим из ф-ии чтобы не было рекурсии

} else {

show\_cursor(FALSE);

}

// считывание клавиш

buf = \_getwch();

switch (buf) {

case up:

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

}

if (active == 0 || active == -1 || total\_el == 1) break;

if (active > 1) active--;

if (active % num\_pages == 0 || page == total\_pages) {

page--;

buf\_el = first\_buf\_el;

count\_num\_pages = first\_i - num\_pages; // формируем i

system("cls");

}

break;

case down:

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

}

if (active == -1 || total\_el == 1) break;

if (active != 0) {

if (active % (num\_pages) == 0) {

if (i == total\_el) { // если это самый последний элемент

break;

}

page++;

buf\_el = temp->next;

count\_num\_pages = i + 1; // формируем i

system("cls");

}

if (active < total\_el) active++;

if (page == 0) {

if (active % (num\_pages + 1) == 0 && temp != 0) {

page++;

buf\_el = temp->next;

}

}

}

break;

case right\_btn:

if (show\_hotkey == 1) break; // если включен блок горячих клавиш

if (active == -1 || total\_el == 1) break;

if (i + 1 <= total\_el && active != 0) {

buf\_el = temp->next;

count\_num\_pages += num\_pages;

active = i + 1;

page++;

system("cls");

}

if (active % num\_pages == 0) {

if (sort\_field < 5) {

sort\_field++;

}

break;

}

break;

case left\_btn:

if (show\_hotkey == 1) break; // если включен блок горячих клавиш

if (active == -1 || total\_el == 1) break;

if (active == 0) {

if (sort\_field > 1) {

sort\_field--;

}

break;

}

if (active - num\_pages >= 1) {

buf\_el = first\_buf\_el;

count\_num\_pages -= num\_pages;

active = first\_i - num\_pages;

page--;

system("cls");

}

break;

case del:

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

}

if (MessageBox(0, L"Вы уверены, что хотите удалить?", L"Удаление", MB\_ICONQUESTION | MB\_YESNO | MB\_SETFOREGROUND) == 6) {

beg = delete\_el(beg, num\_del);

if (active != 1) {

active--;

}

total\_el--;

if (active % num\_pages == 0) {

buf\_el = first\_buf\_el;

count\_num\_pages -= num\_pages;

page--;

} else if (active < num\_pages) buf\_el = beg;

// пересчитываем сумму общего времени и сумму времени ЦП

sum\_all\_time = sum\_time\_cpu = average\_percent\_time\_cpu = 0;

for (temp = beg; temp; temp = temp->next) {

sum\_all\_time += stoi(temp->d.all\_time);

sum\_time\_cpu += stoi(temp->d.time\_cpu);

average\_percent\_time\_cpu += percent\_time\_cpu(stof(temp->d.all\_time), stof(temp->d.time\_cpu));

}

}

system("cls");

break;

case esc:

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

break;

}

if (active == 0 || active == -1) {

active = remember\_active;

break;

} else {

return beg;

}

case enter:

if (show\_hotkey == 1) break; // если включен блок горячих клавиш

if (active == -1) { // изменение кол-ва элементов на одной странице

show\_cursor(TRUE);

do {

string str;

int fl = 0;

// установка курсора в правильное место

gotoxy(34, 2);

SetColor(0, 15);

cout << sets(50);

gotoxy(35, 2);

// =====================================

str = to\_string(num\_pages);

cin >> str;

// проверки на ошибки

try {

if ( !(str[0] >= '0' && str[0] <= '9') ) { // если это символ

throw 1;

} else if (stoi(str) >= 2 && stoi(str) <= 10) {

num\_pages = stoi(str);

break;

} else throw 2; // если число не попало в диапозон от 2 до 10

}

catch (int ex) {

if (ex == 1) {

MessageBox(0, L"Нельзя вводить символы!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

} else if (ex == 2) {

MessageBox(0, L"Значение должно быть в диапозоне от 2 до 10!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

}

}

} while (1);

// делаем активным первый элемент

i = 1;

first\_i = 0;

count\_num\_pages = 1;

page = 0;

temp = buf\_el = first\_buf\_el = beg;

active = 1;

system("cls");

show\_cursor(FALSE);

break;

}

if (active == 0) {

sort(beg, sort\_field, direction);

if (direction == 0) direction = sort\_field; else direction = 0;

break;

}

edit(end, beg, buf\_el, active, edit\_ob, count\_num\_pages, page, height\_el\_num\_pages);

system("cls");

break;

case 115: // символ s

case 1099: // символ ы

if (active == -1) break;

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

}

if (active == 0) {

active = remember\_active;

} else {

remember\_active = active;

active = 0;

}

break;

case 110: // символ n

case 1090: // символ т

if (show\_hotkey == 1) {

system("cls");

show\_hotkey = 0;

}

if (active == -1) {

active = remember\_active;

}

else {

remember\_active = active;

active = -1;

}

break;

case 104: // символ h

case 1088: // символ р

if (show\_hotkey == 0) {

show\_hotkey = 1;

} else show\_hotkey = 0;

system("cls");

break;

}

} while (1);

}

// ==========РЕДАКТИРОВАНИЕ ЭЛЕМЕНТА==========

void edit(time\_task\* end, time\_task\* real\_beg ,time\_task \* beg, int active, time\_task \* \_edit\_ob, int edit\_count\_num\_pages, int edit\_page, int edit\_i) {

int edit\_el = 1;

string remember\_field;

do {

print(end, real\_beg, beg, active, edit\_el, edit\_count\_num\_pages, edit\_page);

switch (\_getwch()) {

case right\_btn:

if (edit\_el >= 1 && edit\_el < 5) print(end, real\_beg, beg, active, ++edit\_el, edit\_count\_num\_pages, edit\_page);

break;

case left\_btn:

if (edit\_el <= 5 && edit\_el > 1) print(end, real\_beg, beg, active, --edit\_el, edit\_count\_num\_pages, edit\_page);

break;

case esc:

return;

break;

case enter:

switch (edit\_el) {

case 1:

cout << "\nВведите новый шифр задания (8 символов):" << endl;

remember\_field = \_edit\_ob->d.cipher;

\_edit\_ob->d.cipher = check\_num(\_edit\_ob->d.cipher, 0, 17 + edit\_i \* 2, 8, 0);

if (\_edit\_ob->d.cipher == "-1") {

\_edit\_ob->d.cipher = remember\_field;

break;

}

break;

case 2:

cout << "\nВведите новый код отдела (3 символа):" << endl;

remember\_field = \_edit\_ob->d.department\_code;

\_edit\_ob->d.department\_code = check\_num(\_edit\_ob->d.department\_code, 0, 17 + edit\_i \* 2, 3, 0);

if (\_edit\_ob->d.department\_code == "-1") {

\_edit\_ob->d.department\_code = remember\_field;

break;

}

break;

case 3:

cout << "\nВведите ФИО (15 символов):" << endl;

remember\_field = \_edit\_ob->d.fio;

\_edit\_ob->d.fio = check\_num(\_edit\_ob->d.fio, 0, 17 + edit\_i \* 2, 15, 1);

if (\_edit\_ob->d.fio == "-1") {

\_edit\_ob->d.fio = remember\_field;

break;

}

break;

case 4:

cout << "\nВведите новое общее время (5 символов):" << endl;

do {

remember\_field = \_edit\_ob->d.all\_time;

\_edit\_ob->d.all\_time = check\_num(\_edit\_ob->d.all\_time, 0, 17 + edit\_i \* 2, 5, 0);

if (\_edit\_ob->d.all\_time == "-1") {

\_edit\_ob->d.all\_time = remember\_field;

break;

}

if (stoi(\_edit\_ob->d.all\_time) < stoi(\_edit\_ob->d.time\_cpu)) {

MessageBox(0, L"Общее время должно быть больше времени центрального процессора!", L"Ошибка", MB\_ICONWARNING | MB\_SETFOREGROUND);

clearRow(16 + edit\_i \* 2);

gotoxy(0, 17 + edit\_i \* 2);

} else break;

} while (1);

break;

case 5:

cout << "\nВведите новое время ЦП (5 символов):" << endl;

do {

remember\_field = \_edit\_ob->d.time\_cpu;

\_edit\_ob->d.time\_cpu = check\_num(\_edit\_ob->d.time\_cpu, 0, 17 + edit\_i \* 2, 5, 0);

if (\_edit\_ob->d.time\_cpu == "-1") {

\_edit\_ob->d.time\_cpu = remember\_field;

break;

}

if (stoi(\_edit\_ob->d.all\_time) < stoi(\_edit\_ob->d.time\_cpu)) {

MessageBox(0, L"Общее время должно быть больше времени центрального процессора!", L"Ошибка", MB\_ICONWARNING | MB\_SETFOREGROUND);

clearRow(16 + edit\_i \* 2);

gotoxy(0, 17 + edit\_i \* 2);

} else break;

} while (1);

break;

}

// пересчитываем сумму общего времени и сумму времени ЦП

sum\_all\_time = sum\_time\_cpu = average\_percent\_time\_cpu = 0;

for (time\_task\* temp = real\_beg; temp; temp = temp->next) {

sum\_all\_time += stoi(temp->d.all\_time);

sum\_time\_cpu += stoi(temp->d.time\_cpu);

average\_percent\_time\_cpu += percent\_time\_cpu(stof(temp->d.all\_time), stof(temp->d.time\_cpu));

}

system("cls");

print(end, real\_beg, beg, active, edit\_el, edit\_count\_num\_pages, edit\_page);

return;

}

} while (1);

}

// ==========УДАЛЕНИЕ==========

time\_task\* delete\_el(time\_task \* beg, int num\_del) {

time\_task\* temp;

time\_task\* buf;

if (!beg) {

MessageBox(0, L"Список пуст", L"Уведомление", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

return 0;

}

temp = beg;

// если один элемент в списке

if (beg->next == 0) {

delete temp;

return 0;

}

while (temp) {

if (num\_del == stoi(temp->d.cipher)) { // если введённый номер совпал с шифром задания

buf = temp->next;

// если удаляется второй элемент

if (temp->prev == 0) {

buf->prev = 0;

delete temp;

return buf;

}

if (buf != 0) buf->prev = temp->prev; // если следующий элемент не 0

buf = temp->prev;

buf->next = temp->next;

delete temp;

return beg;

}

temp = temp->next;

}

MessageBox(0, L"Произошла ошибка!", L"Ошибка", MB\_ICONERROR | MB\_SETFOREGROUND);

return beg;

}

// ==========СРАВНЕНИЕ ДАННЫХ ДЛЯ СОРТИРОВКИ==========

int compare(time\_task\* t\_i, time\_task\* t\_j, int num, int compare\_direction) {

switch (num) {

case 1:

if (compare\_direction == 0) {

return stoi(t\_i->d.cipher) > stoi(t\_j->d.cipher);

} else return stoi(t\_i->d.cipher) < stoi(t\_j->d.cipher);

case 2:

if (compare\_direction == 0) {

return stoi(t\_i->d.department\_code) > stoi(t\_j->d.department\_code);

} else return stoi(t\_i->d.department\_code) < stoi(t\_j->d.department\_code);

case 3:

if (compare\_direction == 0) {

return t\_i->d.fio > t\_j->d.fio;

} else return t\_i->d.fio < t\_j->d.fio;

case 4:

if (compare\_direction == 0) {

return stoi(t\_i->d.all\_time) > stoi(t\_j->d.all\_time);

} else return stoi(t\_i->d.all\_time) < stoi(t\_j->d.all\_time);

case 5:

if (compare\_direction == 0) {

return stoi(t\_i->d.time\_cpu) > stoi(t\_j->d.time\_cpu);

} else return stoi(t\_i->d.time\_cpu) < stoi(t\_j->d.time\_cpu);

}

return 0;

}

// ==========СОРТИРОВКА==========

void sort(time\_task\* beg, int field\_for\_sort, int sort\_direction){

time\_task \*temp\_i = beg,

\*temp\_j = beg;

for (; temp\_i; temp\_i = temp\_i->next) {

for (temp\_j = temp\_i; temp\_j; temp\_j = temp\_j->next) {

if (compare(temp\_i, temp\_j, field\_for\_sort, sort\_direction)) {

swap(temp\_i->d, temp\_j->d);

}

}

}

}

// ==========ПОИСК ЭЛЕМЕНТА ПО ФАМИЛИИ==========

void find(time\_task \* beg) {

time\_task\* temp = beg;

string find\_el;

bool fl = 0;

string str;

if (!beg) {

MessageBox(0, L"Список пуст", L"Уведомление", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

return;

}

// выводим название раздела

SetColor(0, 10);

gotoxy(width / 2 - 2, 3);

cout << sets(21);

gotoxy(width / 2 - 2, 4);

cout << " ПОИСК ";

gotoxy(width / 2 - 2, 5);

cout << sets(21);

SetColor(0, 15);

gotoxy(width / 2 - 4, height / 2 - 4);

cout << "Введите данные для поиска" << endl;

gotoxy(width / 2 - 4, height / 2 - 3);

SetColor(0, 10);

find\_el = check\_num(find\_el, width / 2 - 4, height / 2 - 3, 15, 2);

SetColor(0, 15);

if (find\_el == "-1") return;

system("cls");

while (temp) {

if (find\_el == temp->d.all\_time || find\_el == temp->d.cipher || find\_el == temp->d.department\_code || find\_el == temp->d.time\_cpu || find\_el == temp->d.fio) {

if (fl == 0) {

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

cout << "| Шифр задания Код отдела ФИО Общее время Время ЦП | Процент процессорного времени |" << endl;

cout << "+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

}

fl = 1;

print\_info(\*temp, 0);

cout << "\n+———————————————————————————————————————————————————————————————————————————————+———————————————————————————————+" << endl;

}

temp = temp->next;

}

if (fl == 0) {

MessageBox(0, L"Сотрудник с такими данными не найден", L"Уведомление", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

return;

} else system("pause");

}

// ==========ЧТЕНИЕ ИЗ БИНАРНОГО ФАЙЛА==========

int read\_bin\_file(string filename, time\_task\*\* beg, time\_task\*\* end) {

total\_el = 0;

ifstream fin;

fin.open(filename, ios::binary);

if (!fin) {

MessageBox(0, L"Нет файла!", L"Ошибка", MB\_ICONERROR | MB\_SETFOREGROUND);

return 1;

}

fin.seekg(ios\_base::beg);

time\_task\* t = new time\_task;

t->next = NULL;

t->prev = NULL;

\*beg = 0;

while (fin.read((char\*)& t->d, sizeof(t->d)) ) {

input(\*beg, \*end, \*t);

total\_el++;

}

fin.close();

return 0;

}

// ==========ЧТЕНИЕ ИЗ ФАЙЛА==========

int read\_file(string filename, time\_task \* \*beg, time\_task \* \*end) {

int k = 0;

total\_el = 0;

ifstream fin;

fin.open(filename);

if (!fin) {

MessageBox(0, L"Нет файла!", L"Ошибка", MB\_ICONERROR | MB\_SETFOREGROUND);

return 1;

}

time\_task t;

\*beg = 0;

while (getline(fin, t.d.cipher)) {

getline(fin, t.d.department\_code);

getline(fin, t.d.fio);

getline(fin, t.d.all\_time);

getline(fin, t.d.time\_cpu);

input(\*beg, \*end, t);

total\_el++;

}

fin.close();

return 0;

}

// ==========ЗАПИСЬ В ФАЙЛ==========

int write\_file(time\_task \* temp) {

if (!temp) {

MessageBox(0, L"Список пуст", L"Уведомление", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

return 0;

}

// МЕНЮ ДЛЯ ЗАПИСИ В ФАЙЛ

int active\_file = 1,

k = 1;

string write\_filename;

// выводим название раздела

SetColor(0, 10);

gotoxy(width / 2 - 16, 3);

cout << sets(41);

gotoxy(width / 2 - 16, 4);

cout << " ЗАПИСЬ ДАННЫХ В ФАЙЛ ";

gotoxy(width / 2 - 16, 5);

cout << " Выберите тип файла для сохранения. ";

gotoxy(width / 2 - 16, 6);

cout << " (Расширение файла вводить не надо) ";

gotoxy(width / 2 - 16, 7);

cout << sets(41);

const string items[2] = {

" Текстовый ",

" Бинарный "};

int is\_text = menu(active\_file, items, 2);

if (is\_text == -1) return 0;

// ======================

show\_cursor(TRUE);

cout << "Введите название файла:" << endl;

do {

try {

write\_filename = check\_num(write\_filename, 0, 17, 20, 2);

if (write\_filename == "-1") return 0;

for (int i = 0; write\_filename[i]; i++) write\_filename[i] = tolower(write\_filename[i]); // приводим к нижнему регистру для правильного сравнения

if (write\_filename == "mainbd") throw 1;

// проверка на повторение названия файла

for (k; k < SIZE\_arr\_filename; k++) {

if (arr\_filename[k].find(write\_filename) != -1) {

throw 2;

}

if (arr\_filename[k] == "") break; // если дальше пустые элементы

}

break;

} catch (int ex) { // если такое название уже есть

if (ex == 1) {

MessageBox(0, L"Недопустимое название файла!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

}

else if (ex == 2) {

MessageBox(0, L"Файл с таким именем уже существует!", L"Предупреждение", MB\_ICONWARNING | MB\_SETFOREGROUND);

}

clearRow(18);

gotoxy(0, 17);

}

} while (1);

if (is\_text == 1) {

write\_filetype(temp, write\_filename + ".txt", k, 1);

} else write\_filetype(temp, write\_filename + ".bin", k, 0);

return 0;

}

// ==========ОПРЕДЕЛЕНИЕ ТИПА ДЛЯ ЗАПИСИ==========

void write\_filetype(time\_task \*temp, string filename, int el, int filetype) {

ofstream fout; // файл в который производится запись данных

ofstream fout\_all\_bd; // файл в который производится запись названия файла

if (el != -1) fout\_all\_bd.open("mainBD.txt", ios\_base::app);

// обработка ошибок

if (!fout || !fout\_all\_bd) {

MessageBox(0, L"Невозможно открыть файл для записи!", L"Ошибка", MB\_ICONERROR | MB\_SETFOREGROUND);

return;

}

fout.open(filename, (filetype == 1) ? ios\_base::out : ios::binary | ios::out);

if (el != -1) fout\_all\_bd << filename << endl; // записываем в общий файл названия БД

if (el != -1) arr\_filename[el++] = filename; // добавялем в массив новое название файла

if (filetype == 1) {

while (temp) {

fout << temp->d.cipher << endl;

fout << temp->d.department\_code << endl;

fout << temp->d.fio << endl;

fout << temp->d.all\_time << endl;

fout << temp->d.time\_cpu << endl;

temp = temp->next;

}

} else {

while (temp) {

fout.write((char\*)& temp->d, sizeof temp->d);

temp = temp->next;

}

}

// закрываем файлы

fout.close();

fout\_all\_bd.close();

MessageBox(0, L"БД Сохранена", L"Сохранение", MB\_ICONINFORMATION | MB\_SETFOREGROUND);

}

// ==========ШАБЛОН ПЕЧАТИ МЕНЮ==========

void print\_menu(int sym, const string items[], const int N\_ITEMS) {

for (int i = 1; i <= N\_ITEMS; i++) {

SetColor(0, 15);

gotoxy((width / 2) - 6, (height / 2) + i - 3); // ставим меню в центр

if (i == sym) {

SetColor(0, 10);

}

cout << items[i - 1] << endl;

SetColor(0, 15);

}

}

// ==========МЕНЮ==========

int menu(int& active, const string items[], int num\_el) {

wint\_t buf;

do {

cls();

print\_menu(active, items, num\_el);

buf = \_getwch();

switch (buf) {

case up: // клавиша вверх

if (active > 1) active--;

break;

case down: // клавиша вниз

if (active < num\_el) active++;

break;

case enter: // клавиша enter

return active;

case esc: // клавиша escape

return -1;

case del:

return -active;

}

} while (1);

}

// ==========УСТАНОВКА ЦВЕТА ТЕКСТА И ФОНА==========

void SetColor(int text, int bg) {

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((bg << 4) | text));

}