**АННОТАЦИЯ**

Данный документ предназначен для того, чтобы предоставить возможность ознакомления с разработанной в ходе курсового проекта программой лицам, не принимавшим участия в разработке программы ранее либо же лицам, предполагающим возможность дальнейшей разработки данного программного продукта. Также этот документ предназначен для ознакомления лиц, планирующих использовать разработанную в ходе курсового проекта программу в промышленных целях. Его суть состоит в ознакомлении лиц, не принимавших прямого участия в разработке программы, с программой, её возможностями, функциями, а также с тонкостями реализации и функционирования вышеупомянутой программы и её систем.

**СОДЕРЖАНИЕ**

[**1 ВВЕДЕНИЕ** 3](#_Toc154197503)

[**2 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ** 4](#_Toc154197504)

[**3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ** 5](#_Toc154197505)

[**4 ВЫПОЛНЕНИЕ ПРОГРАММЫ** 19](#_Toc154197506)

[**4.1 Условия выполнения программы** 19](#_Toc154197507)

[**4.2 Загрузка и запуск программы** 19](#_Toc154197508)

[**4.3 Проверка работоспособности программы** 19](#_Toc154197509)

[**5 ЗАКЛЮЧЕНИЕ** 19](#_Toc154197510)

[**6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ** 19](#_Toc154197511)

[**7 ПРИЛОЖЕНИЕ** 19](#_Toc154197512)

## **1 ВВЕДЕНИЕ**

Целью данного курсового проекта является разработка программы “Обработка табличной информации”, которая сможет обеспечить получение оперативных точных данных о студентах, пропущенных студентами часах, оправданных студентами часах, а также остальных необходимых для пользователя сведений о студентах.

Разработка вышеупомянутой программы ведётся в соответствии с техническим заданием (8 вариант), полученным в организации “Севастопольский государственный университет”, утверждённым шестнадцатого декабря две тысячи двадцать третьего года.

Дефицит кадров испытывают 42% предприятий, показал июльский опрос Института Гайдара. На момент предыдущего опроса в апреле о нехватке работников заявляли 35% компаний. Эта неоспоримо пугающая тенденция продолжает своё неумолимое движение по всему миру. В настоящий момент времени особенно резко во всех странах мира стоит вопрос о наборе, обучении и выпуске квалифицированных кадров. Можно успешно провести корреляцию между посещаемостью студента, его успехами, достижениями, а также оценками, следовательно, и шансом выпуститься из высшего учебного заведения с дипломом. Однако, даже учитывая всё вышесказанное, во многих учебных заведениях до сих пор посещаемость проставляется и контролируется преподавателем, который, при всём своём опыте и при всех своих неоспоримых навыках и знаниях, всё же человек, и может ошибиться. Для решения подобной проблемы некоторые учебные заведения разрабатывают свою систему по контрою посещаемости занятий студентами. Но, как показала практика, далеко не каждое учебное заведение способно самостоятельно разработать программу с необходимым функционалом. Программа, разработанная в ходе данного курсового проекта, призвана помочь справиться с вышеупомянутым вопросом и дать возможность учебным организациям по всему миру контролировать прилежность их студентов или иных обучающихся.

В задачи, которые необходимо решить программой, разработанной в ходе данного курсового проекта, входят задачи хранения, обработки, добавления, редактирования, удаления, а также произведения иных действий с данными, которые были внесены в программу напрямую или же через файл какого-либо типа.

## **2 НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ ПРОГРАММЫ**

Разработанная в ходе данного курсового проекта программы предназначена для использования учебными заведениями, которым необходима помощь в хранении информации о студентах, отслеживании посещаемости студентов данного учебного заведения, загрузкой и выгрузкой разных наборов данных в виде файлов, хранящих информацию о студентах, а также в произведении иных операций над данными о студентах, содержащимися в том или ином виде. Однако данная программа не только предоставляет широкий перечень функциональных возможностей, но и удобный, дружелюбный к пользователю интерфейс, разработанный специально для консольных приложений подобного типа. Также данная программа предоставляет возможность интерактивной работы с меню посредством клавиш на клавиатуре пользователя. Единственным ограничением по использованию данного приложения является возможность его запуска только на операционных системах Windows от 7 и новее.

## **3 ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ ПРОГРАММЫ**

В качестве языка программирования для реализации программы, решающей поставленные задачи, был выбран язык Си. Язык Си был выбран из-за его легковесности, его быстродействия, а также его лёгкости в освоении и применении. Также язык Си предоставляет возможность работать с памятью напрямую, что решает проблему недетерминированности сборщика мусора в таких языках, как C#, Java и других языках, использующих подобный функционал.

В качестве среды программирования была выбрана интегрированная среда разработки под названием “Visual Studio Code”. Эта среда разработки была выбрана из-за практически нескончаемого количества вспомогательных, для разработки программного обеспечения, средств. Также данная среда разработки имеет небольшой, относительно других сред, вес, и, большую, опять же, относительно других сред, свободу в выборе инструментов разработчика.

В качестве компилятора был выбран GCC (Gnu C Compiler) из-за того, что при использовании GCC для компиляции кода под разные платформы будет использован один и тот же синтаксический анализатор. Поэтому, если удалось собрать программу для одной из целевых платформ, то велика вероятность, что программа нормально соберётся и для других платформ. Также GCC является свободным программным обеспечением и используется как стандартный компилятор для свободных UNIX-подобных операционных систем.

Рассмотрим следующую задачу. Даны сведения за месяц о пропусках занятий студентами групп. Структура записи: порядковый номер, шифр группы (6 символов), ФИО (30 символов), дата рождения, пол (булевская переменная), пропущено часов, оправдано часов. Описание соответствующей структуры данных о студенте на языке Си представим в виде:

struct student {

int number;

WCHAR group[6];

WCHAR FIO[30];

WCHAR birthday[11];

int gender;

int missedHours;

int justifiedHours;

};

Необходимо, чтобы программа могла выполнять следующие c функции с приведённой выше структурой данных:

а) подсчитать количество неоправданных часов по каждому студенту;

б) подсчитать суммарные показатели по каждому виду часов;

в) подсчитать процент неоправданных пропусков в целом;

г) добавление новых студентов в список;

д) удаление студента из списка;

е) корректировка записи о студенте в списке;

ж) сортировка данных;

з) поиск студентов в списке;

и) сохранение, загрузка студентов из бинарных и стандартных файлов;

к) выход из программы.

Была составлена диаграмма прецендентов отражающая отношения между пользователем и программой (Рисунок 3.1).

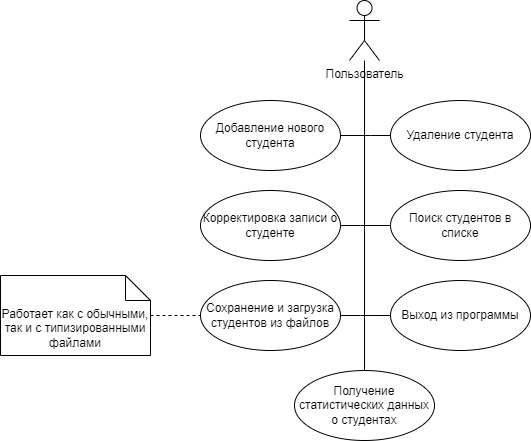


Рисунок 3.1 – Диаграмма прецендентов

Так как в наличии уже имеется минимальное представление о задачах программного продукта можно приступать к проектированию архитектуры программы (Рисунок 3.2). За основу будет взят паттерн разработки MVVM (Model ViewModel View). Это обосновано простотой понимания и реализации, также он поможет сохранить масштабируемость и гибкость кодовой базы. Дальше представлен системный дизайн этой программы, поделённый на три части: Model, View и ViewModel. Каждая из этих частей отвечает за отдельную логику работы программы. Model отвечает непосредственно за бизнес-логику программы, View отвечает за взаимодействие с пользователем и отображение графики, ViewModel отвечает за взаимодействие двух ранее описанных частей, то есть эта часть отвечает за получение команды от пользователя и запуск соответствующей логики модели.

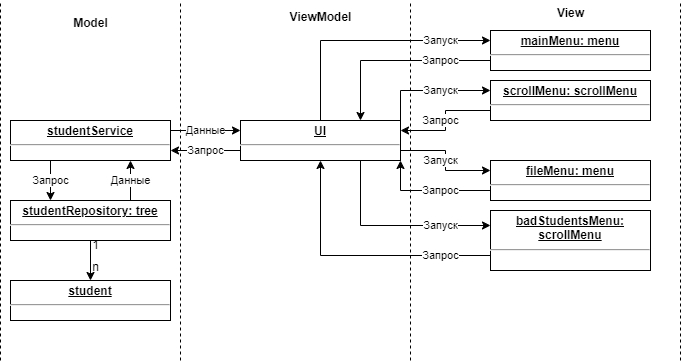


Рисунок 3.2 – Системный дизайн программы

Многие из вышеперечисленных функций требуют возможность взаимодействия со студентами, поэтому необходимо разработать структуру, которая будет предоставлять интерфейс для работы со студентами. Эта структура должна иметь репозиторий (хранилище записей о студентах), а также некоторые вспомогательные поля. Представление данной структуры данных в коде:

struct studentService {

int lastId;

int studentsAmount;

int lastFoundStudentsAmount;

struct node\* studentRepository;

};

Также взаимодействие с программой должно строиться посредством интерактивного меню. Исходя из вышесказанного необходимо разработать структуру пользовательского интерфейса, которая будет хранить все входящие в него элементы, а также будет отвечать за взаимодействие с пользователем. Так как взаимодействие пользователя с программой происходит через эту структуру, она должна иметь доступ к функциям работы со студентами. Представление данной структуры на языке Си:

struct UI {

struct studentService\* studentService;

struct menu \*mainMenu;

struct scrollMenu \*scrollMenu;

struct menu \*fileMenu;

struct scrollMenu \*badStudentsMenu;

};

Далее необходимо реализовать стандартное меню, которое будет содержать выбранный пользователем элемент, количество пунктов в меню, координаты начала и конца меню, а также сами пункты меню, это всё понадобится в дальнейшем для реализации функций отображения меню, выбора пользователя, очистки меню и т.д. Описание соответствующей структуры данных на языке Си представим в виде:

struct menu {

int choice;

int pointsAmount;

COORD start, end;

WCHAR \*\*points;

};

Следующим шагом будет написание структур данных, которые также будут реализовывать функционал меню, но с некоторыми изменениями. Первой из таких является структура меню с возможностью скроллинга. Эта структура будет содержать в себе вышеупомянутую структуру menu и buffer, который будет хранить в себе все элементы меню, которые будут подгружаться или выгружаться в зависимости от действий пользователя. Также в данной структуре должны присутствовать поля, которые понадобятся для реализации её некоторых функций. Представление данной структуры на языке Си:

struct scrollMenu {

struct menu menu;

struct student\* buffer;

int bufferPointsAmount;

int maxPoints;

int page;

int printStudentsMode;

};

Второй подобной структурой будет studentMenu. Она станет отвечать за отображение студента на экране и взаимодействие пользователя со студентом. Кроме структуры menu она должна также содержать указатель на студента, подвергающегося взаимодействию в данный момент. Описание соответствующей структуры данных на языке Си:

struct studentMenu {

struct menu menu;

struct student\* student;

};

Теперь можно показать отношение структур меню через диаграмму классов (Рисунок 3.2). Такая схема взаимодействия была выбрана для максимального переиспользования ранее написанного кода, а также для возможности легче и быстрее исправлять ошибки и добавлять новый функционал.

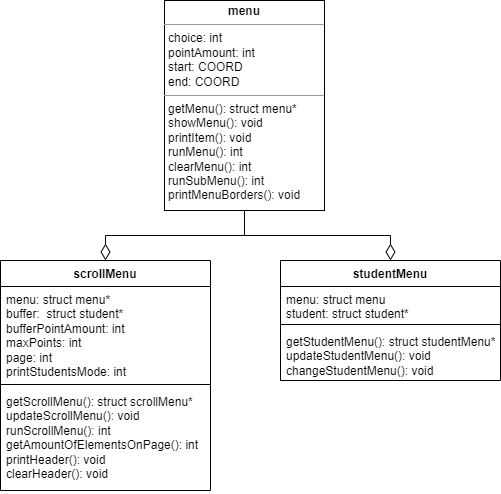


Рисунок 3.3 – Отношения меню

Также необходимо реализовать структуру, которая будет хранить все записи о студентах. В качестве структуры данных для этой задачи было выбрано АВЛ-дерево, которое, как известно, имеет сложность поиска . Описание данной структуры на языке Си:

struct node {

struct student student;

size\_t height;

struct node\* left;

struct node\* right;

};

Теперь, когда в наличие имеются все необходимые структуры данных, можно приступать к реализации функций, необходимых для работы программы. Программа будет запускаться функцией main, чьей задачей является запустить основные модули, необходимые для функционирования программы (Рисунок 3.4).

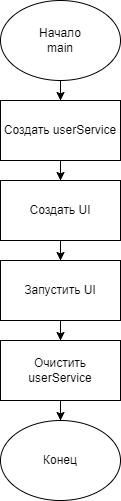


Рисунок 3.4 – Функция main

Функцией, которая отвечает за запуск пользовательского интерфейса будет runUI, чьей единственной задачей является запустить работу основного меню и дожидаться логического окончания его работы (Рисунок 3.5).

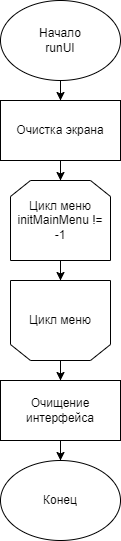


Рисунок 3.5 – Функция runUI

За работу меню будет отвечать функция runMenu, которая будет принимать указатель на структуру данных типа menu. Структурна схема функции runMenu показана на рисунке 3.6. В данной функции также используется функция showMenu (Вывести меню), но её реализацию можно опустить, так как единственное, что она делает, это очищает область меню и выводит его построчно заново в определённом формате. Функция runMenu вернёт номер выбранного пользователем элемента. Эта функция будет применятся для главного меню, потому что для остальных меню будет разработана функция runSubMenu, чьё единственное отличие будет заключаться в замене проверки нажатия на клавишу ESC на проверку нажатия на клавишу стрелки влево.

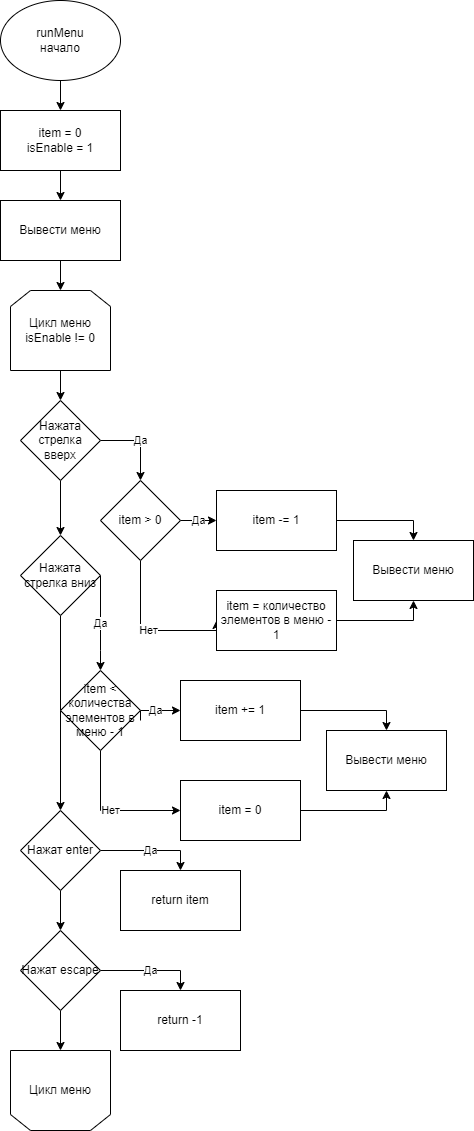


Рисунок 3.6 – функция runMenu

Так как с помощью функций runMenu и runSubMenu будут получены выбранные пользователем пункты меню, нужно определить, как и когда будет происходить вызов функций, предоставляемых пользователю. За это будет отвечать функция initMainMenu.

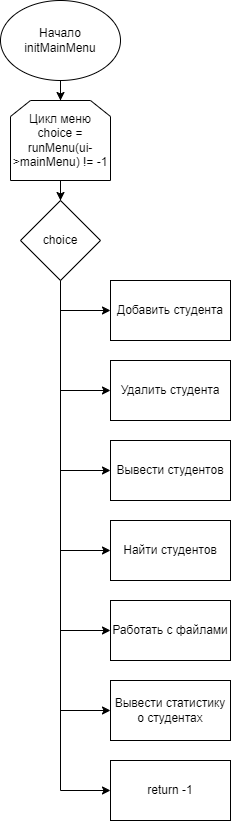


Рисунок 3.7 – Функция initMainMenu

Функции инициализации всех остальных меню также сводятся к вызову заданных заранее функций, так что их реализации могут быть опущены без потери смысла.

Далее необходимо создать функции для работы со студентами. Необходимо отметить, что хоть вся работа со студентами будет происходить через studentService, его функциями можно пренебречь, так как они не несут в себе никакой сложной логики, однако далее будут приведены обязательные функции функционирования АВЛ-дерева, такие как: добавление элемента, удаление элемента, поиск элемента (рисунки 3.8 – 3.10).

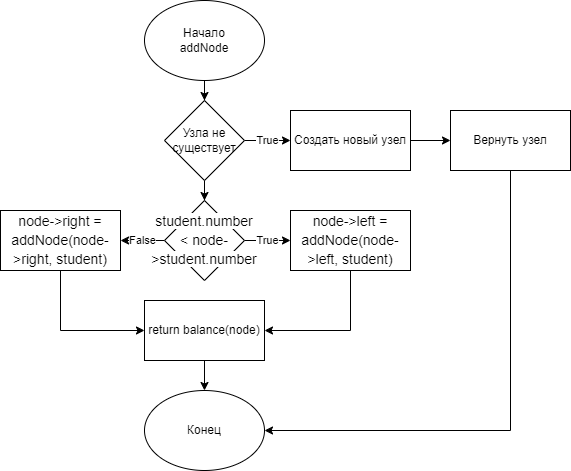


Рисунок 3.8 – Функция addNode

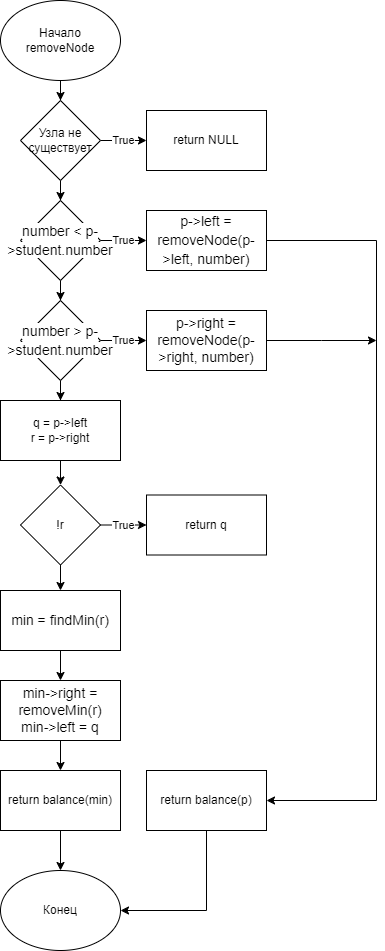


Рисунок 3.9 – Функция removeNode

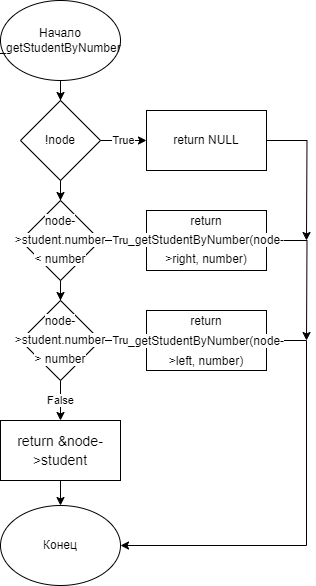


Рисунок 3.10 – Функция \_getStudentByNumber

Следует отметить, что функция balance является функцией балансировки дерева, то есть делает так, чтобы разница высот двух поддеревьев любого узла не превышала единицу. Также необходимо отметить, что p на рисунке 3.9 означает вершину, в которой мы находимся на данный момент.

Добавление записи о студенте происходит с помощью вызова addStudent, которая вызывает функцию addNode для studentRepository, указатель на который находится в studentService.

Удаление записи о студенте производится функцией removeStudent, которая в свою очередь вызывает функцию removeNode. Так как единственным уникальным полем у студента является его идентификационный номер удаление производится именно по нему.

Любое взаимодействие со студентом происходить через меню студента схему которого можно увидеть на рисунке 3.3. Пункты этого меню представляют собой ничто иное как все параметры студента, поэтому когда пользователю потребуется произвести поиск по какому-либо полю либо же изменить какое-либо поле, это взаимодействие будет происходить через меню студента.

## **4 ВЫПОЛНЕНИЕ ПРОГРАММЫ**

# **4.1 Условия выполнения программы**

В результате компиляции программы был получен исполняемый файл с расширением EXE размером 68 Кбайт.

Для выполнения программы необходим минимальный состав программных и аппаратных средств:

-монитор;

-клавиатура;

-центральный процессор, с производительностью от 500 МГц;

-видеокарта, с объемом памяти от 128Мб.

# **4.2 Загрузка и запуск программы**

Запуск программы производится с любого носителя открытием файла prg.exe.

# **4.3 Проверка работоспособности программы**

Управление осуществляется с помощью клавиш вверх и вниз. Для выбора пункта меню необходимо нажать ENTER. Чтобы вернуться на предыдущий шаг необходимо нажать стрелку влево. Для выхода из программы нужно использовать последний пункт главного меню. Главное меню программы представлено на рисунке 4.1.

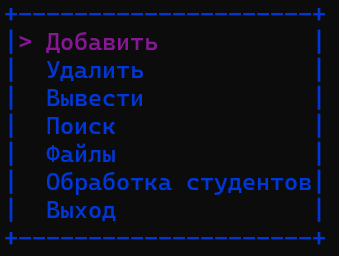


Рисунок 4.1 – Главное меню

На рисунке 4.2 можно представлено меню студента куда в любом порядке путём нажатия клавиши ENTER при выбранном нужном пункте меню можно начать вводить данные. Данные будут считаны после следующего нажатия ENTER.

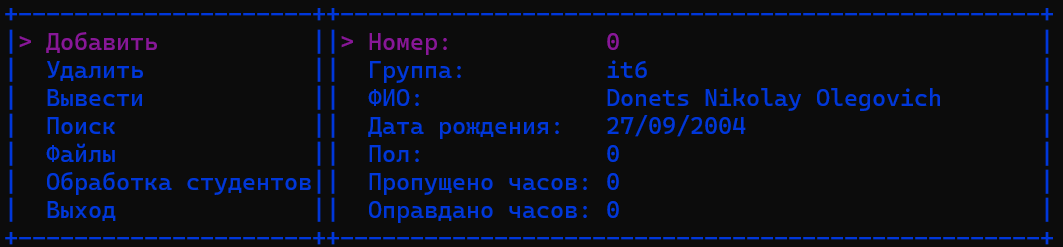


Рисунок 4.2 – Меню студента

На рисунке 4.3 показано как в программе представлен список всех студентов. При нажатии ENTER будет открыто меню студента выбранного студента.

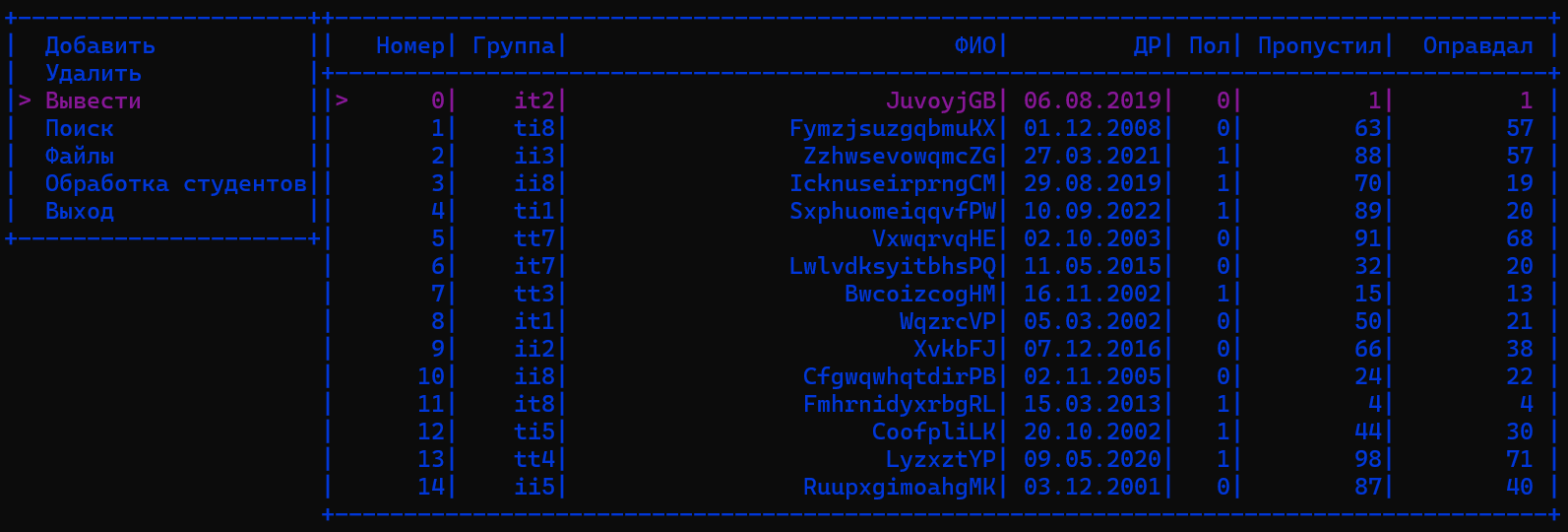


Рисунок 4.3 – Список студентов

Поиск студентов происходит путём введения в меню студента нужного параметра. После этого будет выведен список всех подходящих студентов, которых можно открыть нажав кнопку ENTER

Удаление происходит через поиск. То есть сначала будет вызвана функция поиск студентов, однако, если нажать на выбранного студента ENTER, запись о нём будет удалена.

На рисунках 4.4 – 4.5 представлен интерфейс взаимодействия пользователя с файлами. Сначала пользователю необходимо выбрать пункт меню, соответствующий его желанию, после чего необходимо ввести имя файла, разрешение файла будет выставлено автоматически в зависимости от выбора пользователя.

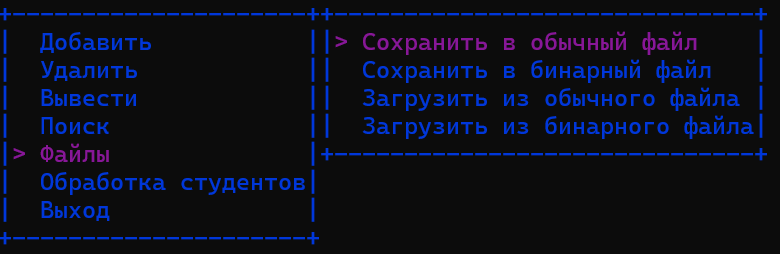


Рисунок 4.4 – Файловое меню

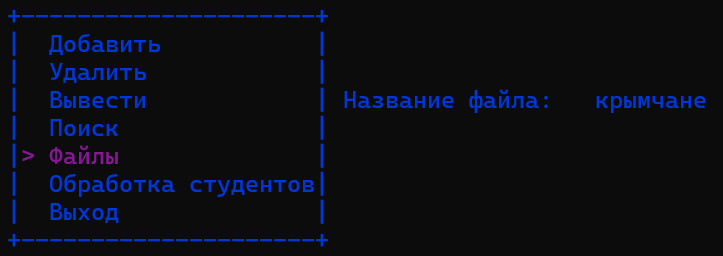


Рисунок 4.5 – Выбор названия файла

На рисунке 4.6 показано представление статистических данных о студентах.

В этих данных находятся такие сведения, как:

а) список студентов имеющих неоправданные пропуски;

б) общее количество пропущенных часов;

в) общее количество оправданных часов;

г) общее количество неоправданных часов;

д) процент неоправданных пропусков.

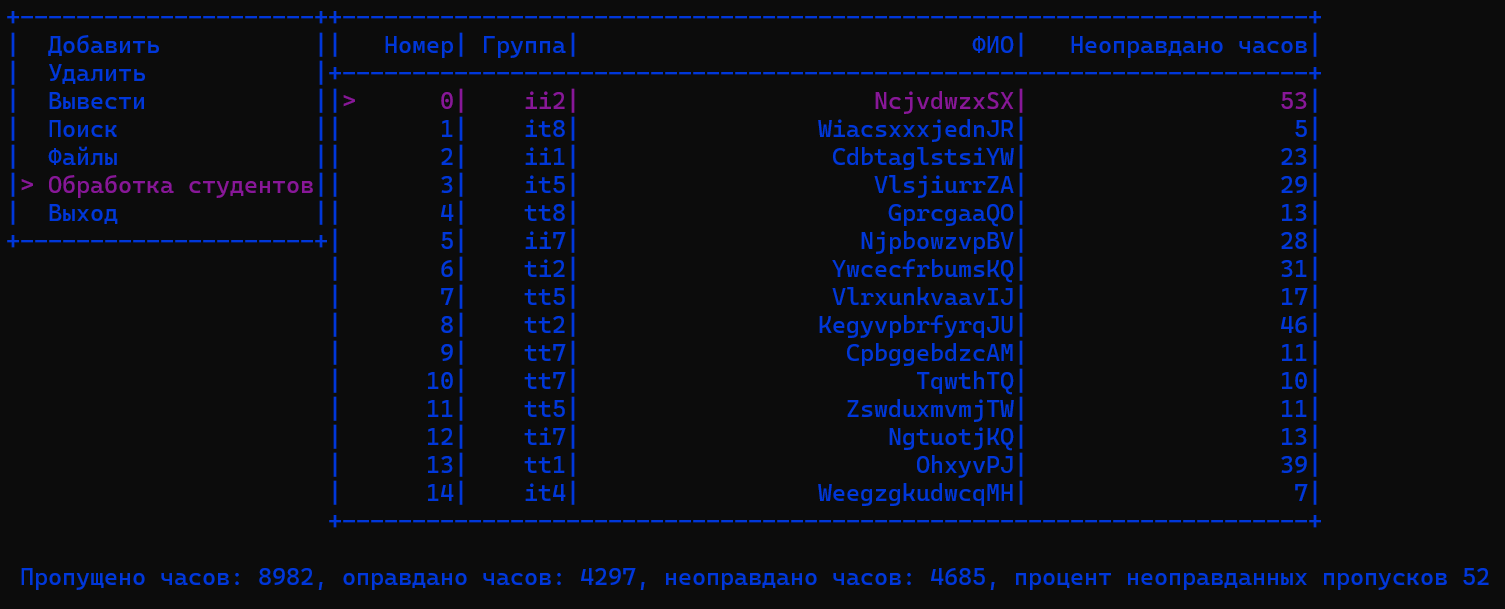


Рисунок 4.6 – Статистические данные о студентах

## **5 ЗАКЛЮЧЕНИЕ**

Программа была составлена в соответствии с постановкой задачи на курсовое проектирование по дисциплине «Объектно-ориентированное программирование».

Программа работает в виде интерактивного меню, что минимизирует сложности её использования для пользователя.

Проведены тестовые примеры для подтверждения работоспособности программы. Тестирование подтвердило, что программа работает корректно, верно выводит и подсчитывает значения, введенные пользователем. Программа предоставляет широкую свободу использования пользователю в том числе выбор типа файла в котором хранить или из которого считывать информацию. Программа также отличается высокой интерактивностью по сравнению с другими программами подобного типа. Пользователь может напрямую взаимодействовать со всеми элементами в списке, что облегчает работу с программой, а также её восприятие пользователем. Также программа использует АВЛ-дерево, что позволяет оптимизировать поиск записи о студенте. Пользователь может работать со списком, а именно добавлять/удалять элементы, редактировать их и получать статистику о недобросовестных студентах.

Тестирование подтвердило, что программа работает корректно и готова к применению в учебных заведениях.

## **6 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Алгоритмы и программы. Язык С++ [Электронный ресурс] / Конова Е. А., Поллак Г. А.– 3-е изд., стер. – : Лань, 2018 .– 384 с. Книга из коллекции Лань - Информатика .– ISBN 978-5-8114-2020-9 . – URL: https://e.lanbook.com/book/103905

2. Алгоритмы и программы. Язык С++ [Электронный ресурс]: учебное пособие / Конова Е. А., Поллак Г. А. – 4-е изд., стер. – Санкт-Петербург: Лань, 2019 .– 384 с. Книга из коллекции Лань - Информатика .– ISBN 978-5-8114-4039-9 .– URL: https://e.lanbook.com/book/114696

3. Искусство алгоритмизации [Электронный ресурс] / В. В. Потопахин. – Москва: ДМК Пресс, 2011 . – 320 с. : ил. + CD . – . – Библиогр. : с. 319. ISBN 978-5-94074-621-8

4. Язык Си: введение для программистов: пер. с англ. / Р. Берри, Б. Микинз .– М. : Финансы и статистика, 1988 .– 191 с .– ISBN 5-279-00280-1.

5. Программирование на языке Си: Учеб. пособие для студентов вузов, обучающихся по направлениям: "Прикладная математика и информатика", "Информатика и вычислительная техника". / В.В. Подбельский, С.С. Фомин. – 2-е изд., доп. – М.: Финансы и статистика, 2003. – 600с. – Библиогр.:с.578 .– ISBN 5-279-02180-6.

6. Ефимова, И.Ю. Лабораторный практикум по структурному программированию: учебное пособие / И.Ю. Ефимова, Т.Н. Варфоломеева. – 3-е изд., стер. – Москва: ФЛИНТА, 2019. – 112 с. – ISBN 978-5-9765-2041-7. – Текст: электронный // Электронно-библиотечная система «Лань»: [сайт]. – URL: https://e.lanbook.com/book/125416

7. 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. – URL: http://docs.cntd.ru/document/1200001260

8. Методические указания к курсовому проектированию по дисциплине «Информатика и вычислительная техника» для студентов дневной и заочной форм обучения направлений 09.03.02 – «Информационные системы и технологии» и 09.03.03 – «Прикладная информатика»

9. 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.

10. dpereverza, Зачем нам UML? Или как сохранить себе нервы и время. – URL: <https://habr.com/ru/articles/458680/>

11.zesetup, Отношения классов — от UML к коду. – URL: <https://habr.com/ru/articles/150041/>

12. Smart-talent, 9 реальных причин дефицита персонала в России. – URL: <https://habr.com/ru/articles/778784/>

## **7 ПРИЛОЖЕНИЕ**

**Код программы**

Файл generateRandomStudents.c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

#include <wchar.h>

#include "student.h"

WCHAR\* random\_string(WCHAR\* set, int length) {

WCHAR\* result = (WCHAR\*)malloc((length + 1) \* sizeof(WCHAR));

int set\_length = wcslen(set);

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

int index = rand() % set\_length;

result[i] = set[index];

}

result[length] = L'\0';

return result;

}

WCHAR\* random\_date() {

WCHAR\* result = (WCHAR\*)malloc(11 \* sizeof(WCHAR));

int day = rand() % 31 + 1;

int month = rand() % 12 + 1;

int year = rand() % 23 + 2000;

wsprintfW(result, L"%02d.%02d.%04d\0", day, month, year);

return result;

}

struct student random\_student() {

struct student result;

result.number = rand() % 100 + 1;

wcscpy(result.group, random\_string(L"it", 2));

wcscat(result.group, random\_string(L"12345678", 1));

wcscpy(result.FIO, random\_string(L"ABCDEFGHIJKLMNOPQRSTUVWXYZ", 1));

wcscat(result.FIO, random\_string(L"abcdefghijklmnopqrstuvwxyz", rand() % 10 + 3));

wcscat(result.FIO, random\_string(L"ABCDEFGHIJKLMNOPQRSTUVWXYZ", 1));

wcscat(result.FIO, random\_string(L"ABCDEFGHIJKLMNOPQRSTUVWXYZ", 1));

wcscpy(result.birthday, random\_date());

result.gender = rand() % 2;

result.missedHours = rand() % 100;

result.justifiedHours = rand() % (result.missedHours + 1);

return result;

}

struct student\* generateStudents(int amount) {

srand(time(NULL));

struct student\* students = (struct student\*)malloc(amount\*sizeof(struct student));

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

students[i] = random\_student();

}

return students;

}

Файл menu.c

#include "menu.h"

#include "studentService.h"

struct menu\* getMenu(int \_pointsAmount, COORD \_start, COORD \_end, WCHAR \*\*\_points) {

struct menu \*menu = (struct menu\*)malloc(sizeof(struct menu));

menu->choice = 0;

menu->pointsAmount = \_pointsAmount;

menu->start = \_start;

menu->end = \_end;

menu->points = \_points;

return menu;

}

void showMenu(struct menu \*menu, int choice) {

printMenuBorders(menu->start, menu->end);

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

int x = menu->start.X;

int y = menu->start.Y;

for (int i = 0; i < menu->pointsAmount; i++) {

gotoxy(x, y++);

if (i == choice) {

SetConsoleTextAttribute(hConsole, 5);

wprintf(L"> ");

printItem(menu, i);

SetConsoleTextAttribute(hConsole, 1);

}

else {

wprintf(L" ");

printItem(menu, i);

}

}

}

void printItem(struct menu \*menu, int item) {

wprintf(menu->points[item]);

}

int runMenu(struct menu \*menu) {

int iItem = 0;

int isEnable = 1;

showMenu(menu, iItem);

while(isEnable) {

if(GetAsyncKeyState(VK\_UP))

{

keybd\_event(VK\_UP, 0, KEYEVENTF\_KEYUP, 0);

if(iItem > 0)

iItem -= 1;

else

iItem = menu->pointsAmount-1;

showMenu(menu, iItem);

}

if(GetAsyncKeyState(VK\_DOWN))

{

keybd\_event(VK\_DOWN, 0, KEYEVENTF\_KEYUP, 0);

if(iItem < menu->pointsAmount-1)

iItem += 1;

else

iItem = 0;

showMenu(menu, iItem);

}

if(GetAsyncKeyState(VK\_RETURN))

{

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

return iItem;

}

if(GetAsyncKeyState(VK\_ESCAPE)) {

keybd\_event(VK\_ESCAPE, 0, KEYEVENTF\_KEYUP, 0);

return -1;

}

}

}

int runSubMenu(struct menu \*menu) {

int iItem = 0;

int isEnable = 1;

showMenu(menu, iItem);

while(isEnable) {

if(GetAsyncKeyState(VK\_UP))

{

keybd\_event(VK\_UP, 0, KEYEVENTF\_KEYUP, 0);

if(iItem > 0)

iItem -= 1;

else

iItem = menu->pointsAmount-1;

showMenu(menu, iItem);

}

if(GetAsyncKeyState(VK\_DOWN))

{

keybd\_event(VK\_DOWN, 0, KEYEVENTF\_KEYUP, 0);

if(iItem < menu->pointsAmount-1)

iItem += 1;

else

iItem = 0;

showMenu(menu, iItem);

}

if(GetAsyncKeyState(VK\_RETURN))

{

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

clearMenu(menu);

return iItem;

}

if(GetAsyncKeyState(VK\_LEFT)) {

keybd\_event(VK\_LEFT, 0, KEYEVENTF\_KEYUP, 0);

clearMenu(menu);

return -1;

}

}

}

void clearMenu(struct menu\* menu) {

for (int y = menu->start.Y-1; y < menu->end.Y+1; y++) {

gotoxy(menu->start.X-1, y);

for (int x = menu->start.X-1; x < menu->end.X+2; x++) {

wprintf(L" ");

}

}

}

void printMenuBorders(COORD start, COORD end) {

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

SetConsoleTextAttribute(hConsole, 1);

gotoxy(start.X-1, start.Y-1);

wprintf(L"+");

for(int i = start.X-1; i < end.X; i++) {

wprintf(L"-");

}

wprintf(L"+");

for(int i = start.Y; i < end.Y; i++) {

gotoxy(start.X - 1, i);

wprintf(L"|");

gotoxy(end.X + 1, i);

wprintf(L"|");

}

gotoxy(start.X-1, end.Y);

wprintf(L"+");

for(int i = start.X-1; i < end.X; i++) {

wprintf(L"-");

}

wprintf(L"+");

}

Файл menu.h

#ifndef MENU

#define MENU

#include <stdio.h>

#include <windows.h>

#include <tchar.h>

#include <locale.h>

#define ESC "\033"

#define home() wprintf(ESC L"[H") //Move cursor to the indicated row, column (origin at 1,1)

#define clrscr() wprintf(ESC L"[2J") //lear the screen, move to (1,1)

#define gotoxy(x,y) printf(ESC "[%d;%dH", y, x);

#define visible\_cursor() wprintf(ESC L"[?251");

#define resetcolor() wprintf(ESC L"[0m")

#define set\_display\_atrib(color) wprintf(ESC L"[%dm",color)

struct menu {

int choice;

int pointsAmount;

COORD start, end;

WCHAR \*\*points;

};

struct menu\* getMenu(int \_pointsAmount, COORD \_start, COORD \_end, WCHAR \*\*\_points);

void showMenu(struct menu \*menu, int choice);

void printItem(struct menu \*menu, int item);

int runMenu(struct menu \*menu);

void clearMenu(struct menu\* menu);

int runSubMenu(struct menu \*menu);

void printMenuBorders(COORD start, COORD end);

#endif

Файл program.c

#include <windows.h>

#include <stdio.h>

#include <locale.h>

#include <tchar.h>

#include "UI.h"

#include "studentService.h"

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

setlocale(LC\_ALL, "Russian");

struct studentService \*studentService = getStudentService();

fillStudentRepository(studentService);

printf("Student service loaded\n");

struct UI \*ui = getUI(studentService);

printf("UI loaded");

runUI(ui);

freeStudents(studentService);

return 0;

}

Файл scrollMenu.c

#include "scrollMenu.h"

#include "studentMenu.h"

struct scrollMenu\* getScrollMenu(struct student\* \_buffer, int \_pointsAmount, int \_bufferPointsAmount, COORD \_start, COORD \_end, int \_printStudentsMode) {

struct scrollMenu \*smenu = (struct scrollMenu\*)malloc(sizeof(struct scrollMenu));

int maxPoints = 15;

smenu->page = 0;

smenu->buffer = \_buffer;

smenu->bufferPointsAmount = \_bufferPointsAmount;

smenu->maxPoints = maxPoints;

smenu->printStudentsMode = \_printStudentsMode;

WCHAR \*\*points;

points = (WCHAR\*\*)malloc(maxPoints\*sizeof(WCHAR\*));

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

points[i] = (WCHAR\*)malloc(15\*sizeof(WCHAR));

}

smenu->menu = getMenu(\_pointsAmount, \_start, \_end, points);

return smenu;

}

void updateScrollMenu(struct scrollMenu\* smenu) {

clearMenu(smenu->menu);

if (!smenu->buffer) {

smenu->menu->points[0] = L"Нет таких студентов";

smenu->menu->pointsAmount = 1;

return;

}

int start = smenu->page \* smenu->maxPoints;

smenu->menu->pointsAmount = getAmountOfElementsOnPage(smenu);

if (smenu->printStudentsMode == 0) {

for(int i = 0; i < smenu->menu->pointsAmount; i++) {

smenu->menu->points[i] = studentToString(smenu->buffer[start+i]);

}

}

else {

for(int i = 0; i < smenu->menu->pointsAmount; i++) {

smenu->menu->points[i] = studentWithUnjustifiedHoursToString(smenu->buffer[start+i]);

}

}

}

int getAmountOfElementsOnPage(struct scrollMenu \*smenu) {

int amountOfElementsOnPage;

if ((smenu->bufferPointsAmount - smenu->page \* smenu->maxPoints)/smenu->maxPoints > 0) {

amountOfElementsOnPage = smenu->maxPoints;

smenu->menu->end.Y = smenu->menu->start.Y + amountOfElementsOnPage;

}

else {

amountOfElementsOnPage = smenu->bufferPointsAmount - smenu->page \* smenu->maxPoints;

smenu->menu->end.Y = smenu->menu->start.Y + amountOfElementsOnPage;

}

return amountOfElementsOnPage;

}

int runScrollMenu(struct scrollMenu \*smenu) {

updateScrollMenu(smenu);

int iItem = 0;

int isEnable = 1;

printHeader(smenu);

showMenu(smenu->menu, 0);

while(isEnable) {

if(GetAsyncKeyState(VK\_UP))

{

keybd\_event(VK\_UP, 0, KEYEVENTF\_KEYUP, 0);

if(iItem > 0)

iItem -= 1;

else {

if (smenu->page > 0) {

smenu->page--;

iItem = smenu->maxPoints-1;

updateScrollMenu(smenu);

}

}

showMenu(smenu->menu, iItem);

}

if(GetAsyncKeyState(VK\_DOWN))

{

keybd\_event(VK\_DOWN, 0, KEYEVENTF\_KEYUP, 0);

if(iItem < smenu->menu->pointsAmount-1)

iItem += 1;

else {

if (smenu->page < (smenu->bufferPointsAmount-1)/smenu->maxPoints) {

smenu->page++;

iItem = 0;

updateScrollMenu(smenu);

}

}

showMenu(smenu->menu, iItem);

}

if(GetAsyncKeyState(VK\_RETURN))

{

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

clearMenu(smenu->menu);

clearHeader(smenu);

return iItem;

}

if(GetAsyncKeyState(VK\_LEFT)) {

keybd\_event(VK\_LEFT, 0, KEYEVENTF\_KEYUP, 0);

isEnable = 0;

clearMenu(smenu->menu);

clearHeader(smenu);

return -1;

}

}

}

void printHeader(struct scrollMenu\* smenu) {

gotoxy(smenu->menu->start.X - 1, smenu->menu->start.Y - 3);

wprintf(L"+");

for (int i = smenu->menu->start.X + 1; i < smenu->menu->end.X + 2; i++) wprintf(L"-");

wprintf(L"+");

gotoxy(smenu->menu->start.X - 1, smenu->menu->start.Y - 2);

if (smenu->printStudentsMode == 0) {

wprintf(L"| %6s|%7s|%31s|%11s|%4s|%10s|%10s |", L"Номер", L"Группа", L"ФИО", L"ДР", L"Пол", L"Пропустил", L"Оправдал");

} else {

wprintf(L"| %6s|%7s|%31s|%20s|", L"Номер", L"Группа", L"ФИО", L"Неоправдано часов");

}

}

void clearHeader(struct scrollMenu\* smenu) {

for (int y = smenu->menu->start.Y - 3; y < smenu->menu->end.Y + 1; y++) {

gotoxy(smenu->menu->start.X - 1, y);

for (int x = smenu->menu->start.X - 1; x < smenu->menu->end.X+2; x++) {

wprintf(L" ");

}

}

}

Файл scrollMenu.h

#ifndef SCROLLMENU

#define SCROLLMENU

#include "menu.h"

#include "student.h"

#include "studentService.h"

struct scrollMenu {

struct menu\* menu;

struct student\* buffer;

int bufferPointsAmount;

int maxPoints;

int page;

int printStudentsMode;

};

struct scrollMenu\* getScrollMenu(struct student\* buffer, int \_pointsAmount, int bufferPointsAmount, COORD \_start, COORD \_end, int \_printStudentsMode);

void updateScrollMenu(struct scrollMenu\* smenu);

int runScrollMenu(struct scrollMenu \*menu);

int getAmountOfElementsOnPage(struct scrollMenu \*smenu);

void printHeader(struct scrollMenu\* smenu);

void clearHeader(struct scrollMenu\* smenu);

#endif

Файл student.c

#include "student.h"

WCHAR\* studentToString(struct student student) {

WCHAR\* str = (WCHAR\*)malloc(256\*sizeof(WCHAR));

wsprintfW(str, L"%6d|%7s|%31s|%11s|%4d|%10d|%10d", student.number, student.group, student.FIO, student.birthday, student.gender, student.missedHours, student.justifiedHours);

return str;

}

WCHAR\* studentWithUnjustifiedHoursToString(struct student student) {

WCHAR\* str = (WCHAR\*)malloc(256\*sizeof(WCHAR));

wsprintfW(str, L"%6d|%7s|%31s|%20d", student.number, student.group, student.FIO, student.missedHours - student.justifiedHours);

return str;

}

struct student \*newStudent() {

struct student\* student = (struct student\*)malloc(sizeof(struct student));

student->number = 0;

wsprintfW(student->group, L"");

wsprintfW(student->FIO, L"");

wsprintfW(student->birthday, L"");

student->gender = 0;

student->missedHours = 0;

student->justifiedHours = 0;

return student;

}

Файл student.h

#ifndef STUDENT

#define STUDENT

#include <windows.h>

#include <stdio.h>

struct student {

int number;

WCHAR group[6];

WCHAR FIO[30];

WCHAR birthday[11];

int gender;

int missedHours;

int justifiedHours;

};

WCHAR\* studentToString(struct student student);

WCHAR\* studentWithUnjustifiedHoursToString(struct student student);

struct student\* newStudent();

#endif

Файл studentMenu.c

#include "studentMenu.h"

struct studentMenu\* getStudentMenu(struct student \*\_student) {

struct studentMenu \*menu = (struct studentMenu\*)malloc(sizeof(struct studentMenu));

menu->menu.choice = 0;

menu->menu.pointsAmount = 7;

COORD start, end;

start.X = 25; start.Y = 2;

end.X = start.X + 19 + 30; end.Y = start.Y + menu->menu.pointsAmount;

menu->menu.start = start;

menu->menu.end = end;

menu->student = \_student;

WCHAR \*\*points;

points = (WCHAR\*\*)malloc(menu->menu.pointsAmount\*sizeof(WCHAR\*));

for (int i = 0; i < menu->menu.pointsAmount; i++) {

points[i] = (WCHAR\*)malloc(50\*sizeof(WCHAR));

}

menu->menu.points = points;

updateStudentMenu(menu);

return menu;

}

void changeStudentMenu(struct studentMenu\* menu, int item) {

clearMenu(&menu->menu);

switch(item) {

case 0:

wsprintfW(menu->menu.points[0], L"Номер: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR number[10];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(number);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

menu->student->number = \_wtoi(number);

updateStudentMenu(menu);

break;

case 1:

wsprintfW(menu->menu.points[1], L"Группа: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR group[6];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(group);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

wcscpy(menu->student->group, group);

updateStudentMenu(menu);

break;

case 2:

wsprintfW(menu->menu.points[2], L"ФИО: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR FIO[30];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(FIO);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

wcscpy(menu->student->FIO, FIO);

updateStudentMenu(menu);

break;

case 3:

wsprintfW(menu->menu.points[3], L"Дата рождения: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR birthday[11];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(birthday);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

wcscpy(menu->student->birthday, birthday);

updateStudentMenu(menu);

break;

case 4:

wsprintfW(menu->menu.points[4], L"Пол: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR gender[10];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(gender);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

menu->student->gender = \_wtoi(gender);

updateStudentMenu(menu);

break;

case 5:

wsprintfW(menu->menu.points[5], L"Пропущено часов: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR missedHours[10];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(missedHours);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

menu->student->missedHours = \_wtoi(missedHours);

updateStudentMenu(menu);

break;

case 6:

wsprintfW(menu->menu.points[6], L"Оправдано часов: ");

clearMenu(&menu->menu);

showMenu(&menu->menu, item);

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

WCHAR justifiedHours[10];

while (!GetAsyncKeyState(VK\_RETURN)) {

gotoxy(menu->menu.start.X + 19, menu->menu.start.Y + item);

\_getws(justifiedHours);

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

}

menu->student->justifiedHours = \_wtoi(justifiedHours);

updateStudentMenu(menu);

break;

}

clearMenu(&menu->menu);

}

void updateStudentMenu(struct studentMenu\* menu) {

wsprintfW(menu->menu.points[0], L"Номер: %d", menu->student->number);

wsprintfW(menu->menu.points[1], L"Группа: %s", menu->student->group);

wsprintfW(menu->menu.points[2], L"ФИО: %s", menu->student->FIO);

wsprintfW(menu->menu.points[3], L"Дата рождения: %s", menu->student->birthday);

wsprintfW(menu->menu.points[4], L"Пол: %d", menu->student->gender);

wsprintfW(menu->menu.points[5], L"Пропущено часов: %d", menu->student->missedHours);

wsprintfW(menu->menu.points[6], L"Оправдано часов: %d", menu->student->justifiedHours);

clearMenu(&menu->menu);

}

Файл studentMenu.h

#ifndef STUDENTMENU

#define STUDENTMENU

#include "menu.h"

#include "student.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <wchar.h>

#include <windows.h>

struct studentMenu {

struct menu menu;

struct student\* student;

};

struct studentMenu\* getStudentMenu(struct student \*student);

void updateStudentMenu(struct studentMenu\* menu);

void changeStudentMenu(struct studentMenu\* menu, int item);

#endif

Файл studentService.c

#include "studentService.h"

#include "generateRandomStudents.c"

void addStudent(struct studentService\* studentService, struct student student) {

student.number = studentService->lastId++;

studentService->studentsAmount++;

studentService->studentRepository = addNode(studentService->studentRepository, student);

return;

}

struct student\* getAllStudents(struct studentService\* studentService) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc((studentService->studentsAmount)\*sizeof(struct student));

getStudents(studentService->studentRepository, students, 0);

return students;

}

struct studentService\* getStudentService() {

struct studentService\* studentService = (struct studentService\*)malloc(sizeof(struct studentService));

studentService->studentsAmount = 0;

studentService->lastFoundStudentsAmount = 0;

studentService->lastId = 0;

studentService->studentRepository = NULL;

return studentService;

}

void fillStudentRepository(struct studentService\* studentService) {

int amount = 180;

struct student\* students = generateStudents(amount);

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

addStudent(studentService, students[i]);

}

printf("Test students loaded\n");

}

struct student\* getStudentsByFIO(struct studentService\* studentService, WCHAR FIO[30]) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByFIO(studentService->studentRepository, students, 0, FIO);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentByNumber(struct studentService\* studentService, int number) {

struct student\* student = \_getStudentByNumber(studentService->studentRepository, number);

studentService->lastFoundStudentsAmount = student?1:0;

return student;

}

struct student\* getStudentsByGroup(struct studentService\* studentService, WCHAR group[6]) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByGroup(studentService->studentRepository, students, 0, group);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentsByBirthday(struct studentService\* studentService, WCHAR birthday[11]) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByBirthday(studentService->studentRepository, students, 0, birthday);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentsByGender(struct studentService\* studentService, int gender) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByGender(studentService->studentRepository, students, 0, gender);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentsByJustifiedHours(struct studentService\* studentService, int justifiedHours) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByJustifiedHours(studentService->studentRepository, students, 0, justifiedHours);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentsByMissedHours(struct studentService\* studentService, int missedHours) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListByMissedHours(studentService->studentRepository, students, 0, missedHours);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) return NULL;

return students;

}

struct student\* getStudentsWithUnjustifiedHours(struct studentService\* studentService) {

if (!studentService->studentRepository) return NULL;

struct student\* students = (struct student\*)malloc(studentService->studentsAmount \* sizeof(struct student));

int foundStudents = getStudentsListWithUnjustifiedHours(studentService->studentRepository, students, 0);

studentService->lastFoundStudentsAmount = foundStudents;

if (!foundStudents) {

studentService->lastFoundStudentsAmount = 0;

return NULL;

}

return students;

}

void removeStudent(struct studentService\* studentService, int id) {

studentService->studentRepository = removeNode(studentService->studentRepository, id);

studentService->studentsAmount--;

}

void loadStudentsFromFile(struct studentService\* studentService, char \*filename) {

FILE\* file = fopen(filename, "r");

if (!file) {

return;

}

int \*i = (int\*)malloc(sizeof(int));

studentService->studentRepository = NULL;

studentService->studentRepository = loadFromFile(studentService->studentRepository, file, i);

studentService->studentsAmount=\*i;

studentService->lastId=\*i;

fclose(file);

}

void saveStudentsToFile(struct studentService\* studentService, char \*filename) {

FILE\* file = fopen(filename, "w");

if (!file) {

return;

}

saveToFile(studentService->studentRepository, file);

fclose(file);

}

void loadStudentsFromBinaryFile(struct studentService\* studentService, char \*filename) {

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

if (!file) {

return;

}

int \*i = (int\*)malloc(sizeof(int));

studentService->studentRepository = NULL;

studentService->studentRepository = loadFromFile(studentService->studentRepository, file, i);

studentService->studentsAmount=\*i;

studentService->lastId=\*i;

fclose(file);

}

void saveStudentsToBinaryFile(struct studentService\* studentService, char \*filename) {

FILE\* file = fopen(filename, "wb");

if (!file) {

return;

}

saveToFile(studentService->studentRepository, file);

fclose(file);

}

void freeStudents(struct studentService\* studentService) {

freeTree(studentService->studentRepository);

free(studentService);

}

WCHAR\* getStats(struct studentService\* studentService) {

int missedHours = getMissedHours(studentService->studentRepository);

int justifiedHours = getJustifiedHours(studentService->studentRepository);

float unjustifiedHoursPercent = 0.0;

int unjustifiedHours = missedHours - justifiedHours;

if (justifiedHours == 0 && missedHours == 0) {

unjustifiedHoursPercent = 0;

}

else if (justifiedHours == 0) {

unjustifiedHoursPercent = 100;

}

else {

unjustifiedHoursPercent = (unjustifiedHours \* 1.0 / missedHours)\*100;

}

WCHAR\* str = (WCHAR\*)malloc(120\*sizeof(WCHAR));

wsprintfW(str, L"Пропущено часов: %d, оправдано часов: %d, неоправдано часов: %d, процент неоправданных пропусков %d", missedHours, justifiedHours, unjustifiedHours, (int)unjustifiedHoursPercent);

return str;

}

Файл studentService.h

#ifndef STUDENT\_SERVICE

#define STUDENT\_SERVICE

#include "tree.h"

struct studentService {

int lastId;

int studentsAmount;

int lastFoundStudentsAmount;

struct node\* studentRepository;

};

void addStudent(struct studentService\* studentService, struct student student);

void removeStudent(struct studentService\* studentService, int id);

struct student\* getAllStudents(struct studentService\* studentService);

struct studentService\* getStudentService();

struct student\* getStudentsByFIO(struct studentService\* studentService, WCHAR FIO[30]);

struct student\* getStudentByNumber(struct studentService\* studentService, int number);

struct student\* getStudentsByGroup(struct studentService\* studentService, WCHAR group[6]);

struct student\* getStudentsByBirthday(struct studentService\* studentService, WCHAR birthday[11]);

struct student\* getStudentsByGender(struct studentService\* studentService, int gender);

struct student\* getStudentsByJustifiedHours(struct studentService\* studentService, int justifiedHours);

struct student\* getStudentsByMissedHours(struct studentService\* studentService, int missedHours);

struct student\* getStudentsWithUnjustifiedHours(struct studentService\* studentService);

void loadStudentsFromFile(struct studentService\* studentService, char \*filename);

void saveStudentsToFile(struct studentService\* studentService, char \*filename);

void loadStudentsFromBinaryFile(struct studentService\* studentService, char \*filename);

void saveStudentsToBinaryFile(struct studentService\* studentService, char \*filename);

void freeStudents(struct studentService\* studentService);

WCHAR\* getStats(struct studentService\* studentService);

void fillStudentRepository(struct studentService\* studentService);

#endif

Файл tree.c

#include "tree.h"

struct node\* addNode(struct node\* node, struct student student) {

if(!node) {

node = (struct node\*)malloc(sizeof(struct node));

node->student = student;

node->height = 1;

node->right = NULL;

node->left = NULL;

return node;

}

else if (student.number < node->student.number) {

node->left = addNode(node->left, student);

}

else {

node->right = addNode(node->right, student);

}

return balance(node);

}

int getStudents(struct node\* node, struct student\* students, int i) {

if (!node) return i;

i = getStudents(node->left, students, i);

students[i++] = node->student;

i = getStudents(node->right, students, i);

return i;

}

int height(struct node\* node) {

return node?node->height:0;

}

int balanceFactor(struct node\* p) {

return height(p->right)-height(p->left);

}

void fixHeight(struct node\* node) {

int hl = height(node->left);

int hr = height(node->right);

node->height = (hl>hr?hl:hr)+1;

}

struct node\* rotateRight(struct node\* p) {

struct node\* q = p->left;

p->left = q->right;

q->right = p;

fixHeight(p);

fixHeight(q);

return q;

}

struct node\* rotateLeft(struct node\* q) {

struct node\* p = q->right;

q->right = p->left;

p->left = q;

fixHeight(q);

fixHeight(p);

return p;

}

struct node\* balance(struct node\* p) {

fixHeight(p);

if( balanceFactor(p)==2 ) {

if( balanceFactor(p->right) < 0 )

p->right = rotateRight(p->right);

return rotateLeft(p);

}

if( balanceFactor(p)==-2 ) {

if( balanceFactor(p->left) > 0 )

p->left = rotateLeft(p->left);

return rotateRight(p);

}

return p;

}

struct node\* findMin(struct node\* node) {

return node->left?findMin(node->left):node;

}

struct node\* removeMin(struct node\* node) {

if(!node->left)

return node->right;

node->left = removeMin(node->left);

return balance(node);

}

struct node\* removeNode(struct node\* p, int number) {

if( !p ) return NULL;

if( number < p->student.number )

p->left = removeNode(p->left, number);

else if( number > p->student.number )

p->right = removeNode(p->right, number);

else

{

struct node\* q = p->left;

struct node\* r = p->right;

free(p);

if( !r ) return q;

struct node\* min = findMin(r);

min->right = removeMin(r);

min->left = q;

return balance(min);

}

return balance(p);

}

int getStudentsListByFIO(struct node\* node, struct student\* students, int i, WCHAR FIO[30]) {

if (!node) return i;

i = getStudentsListByFIO(node->left, students, i, FIO);

if (!wcscmp(node->student.FIO, FIO)) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByFIO(node->right, students, i, FIO);

return i;

}

struct student\* \_getStudentByNumber(struct node\* node, int number) {

if (!node) return NULL;

if (node->student.number < number) {

return \_getStudentByNumber(node->right, number);

} else if (node->student.number > number) {

return \_getStudentByNumber(node->left, number);

} else {

return &node->student;

}

}

int getStudentsListByGroup(struct node\* node, struct student\* students, int i, WCHAR group[6]) {

if (!node) return i;

i = getStudentsListByGroup(node->left, students, i, group);

if (!wcscmp(node->student.group, group)) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByGroup(node->right, students, i, group);

return i;

}

int getStudentsListByBirthday(struct node\* node, struct student\* students, int i, WCHAR birthday[11]) {

if (!node) return i;

i = getStudentsListByBirthday(node->left, students, i, birthday);

if (!wcscmp(node->student.birthday, birthday)) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByBirthday(node->right, students, i, birthday);

return i;

}

int getStudentsListByGender(struct node\* node, struct student\* students, int i, int gender) {

if (!node) return i;

i = getStudentsListByGender(node->left, students, i, gender);

if (node->student.gender == gender) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByGender(node->right, students, i, gender);

return i;

}

int getStudentsListByJustifiedHours(struct node\* node, struct student\* students, int i, int justifiedHours) {

if (!node) return i;

i = getStudentsListByJustifiedHours(node->left, students, i, justifiedHours);

if (node->student.justifiedHours == justifiedHours) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByJustifiedHours(node->right, students, i, justifiedHours);

return i;

}

int getStudentsListByMissedHours(struct node\* node, struct student\* students, int i, int missedHours) {

if (!node) return i;

i = getStudentsListByMissedHours(node->left, students, i, missedHours);

if (node->student.missedHours == missedHours) {

students = (struct student\*)realloc(students, sizeof(struct student)\*(i+1));

students[i++] = node->student;

}

i = getStudentsListByMissedHours(node->right, students, i, missedHours);

return i;

}

int getStudentsListWithUnjustifiedHours(struct node\* node, struct student\* students, int i) {

if (!node) return i;

i = getStudentsListWithUnjustifiedHours(node->left, students, i);

if (node->student.missedHours > node->student.justifiedHours) {

students = (struct student\*)realloc(students, (i+1) \* sizeof(struct student));

students[i++] = node->student;

}

i = getStudentsListWithUnjustifiedHours(node->right, students, i);

return i;

}

void saveToFile(struct node\* node, FILE\* file) {

if (!node) return;

fwrite(node, sizeof(struct node), 1, file);

saveToFile(node->left, file);

saveToFile(node->right, file);

}

struct node\* loadFromFile(struct node\* node, FILE\* file, int \*i) {

if (!feof(file)) {

node = (struct node\*)malloc(sizeof(struct node));

fread(node, sizeof(struct node), 1, file);

if (node->left)

node->left = loadFromFile(node->left, file, i);

node->student.number = \*i;

(\*i)++;

if (node->right)

node->right = loadFromFile(node->right, file, i);

}

else {

return NULL;

}

return node;

}

void freeTree(struct node\* node) {

if(!node) return;

freeTree(node->left);

freeTree(node->right);

free(node);

}

int getJustifiedHours(struct node\* node) {

if (!node) return 0;

return getJustifiedHours(node->left) + getJustifiedHours(node->right) + node->student.justifiedHours;

}

int getMissedHours(struct node\* node) {

if (!node) return 0;

return getMissedHours(node->left) + getMissedHours(node->right) + node->student.missedHours;

}

Файл tree.h

#ifndef TREE

#define TREE

#include <stdio.h>

#include <windows.h>

#include "student.h"

struct node {

struct student student;

int height;

struct node\* left;

struct node\* right;

};

struct node\* addNode(struct node\* node, struct student student);

int getStudents(struct node\* node, struct student\* students, int i);

int height(struct node\* node);

int balanceFactor(struct node\* p);

void fixHeight(struct node\* node);

struct node\* rotateRight(struct node\* p);

struct node\* rotateLeft(struct node\* q);

struct node\* balance(struct node\* p);

struct node\* findMin(struct node\* node);

struct node\* removeMin(struct node\* node);

struct node\* removeNode(struct node\* p, int number);

struct student\* \_getStudentByNumber(struct node\* node, int number);

int getStudentsListByFIO(struct node\* node, struct student\* students, int i, WCHAR FIO[30]);

int getStudentsListByGroup(struct node\* node, struct student\* students, int i, WCHAR group[6]);

int getStudentsListByGender(struct node\* node, struct student\* students, int i, int gender);

int getStudentsListByBirthday(struct node\* node, struct student\* students, int i, WCHAR birthday[11]);

int getStudentsListByJustifiedHours(struct node\* node, struct student\* students, int i, int justifiedHours);

int getStudentsListByMissedHours(struct node\* node, struct student\* students, int i, int missedHours);

int getStudentsListWithUnjustifiedHours(struct node\* node, struct student\* students, int i);

void saveToFile(struct node\* node, FILE\* file);

struct node\* loadFromFile(struct node\* node, FILE\* file, int \*i);

void freeTree(struct node\* node);

int getJustifiedHours(struct node\* node);

int getMissedHours(struct node\* node);

#endif

Файл UI.c

#include "UI.h"

struct UI\* getUI(struct studentService \*studentService) {

struct UI \*ui = (struct UI\*)malloc(sizeof(struct UI));

ui->studentService = studentService;

//Main menu

int mainMenuPointsAmount = 7;

COORD mainMenuStart, mainMenuEnd;

mainMenuStart.X = 2; mainMenuStart.Y = 2;

mainMenuEnd.X = mainMenuStart.X + 20; mainMenuEnd.Y = mainMenuStart.Y + mainMenuPointsAmount;

WCHAR \*\*mainMenuPoints;

mainMenuPoints = (WCHAR\*\*)malloc(mainMenuPointsAmount\*sizeof(WCHAR\*));

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

mainMenuPoints[i] = (WCHAR\*)malloc(15\*sizeof(WCHAR));

}

mainMenuPoints[0] = L"Добавить";

mainMenuPoints[1] = L"Удалить";

mainMenuPoints[2] = L"Вывести";

mainMenuPoints[3] = L"Поиск";

mainMenuPoints[4] = L"Файлы";

mainMenuPoints[5] = L"Обработка студентов";

mainMenuPoints[6] = L"Выход";

ui->mainMenu = getMenu(

mainMenuPointsAmount,

mainMenuStart,

mainMenuEnd,

mainMenuPoints

);

printf("Main menu loaded\n");

//Scroll menu

int scrollMenuPointsAmount = 15;

int scrollMenuBufferPointsAmount = studentService->studentsAmount;

COORD scrollMenuStart, scrollMenuEnd;

scrollMenuStart.X = mainMenuEnd.X + 3; scrollMenuStart.Y = 4;

scrollMenuEnd.X = scrollMenuStart.X + 87; scrollMenuEnd.Y = scrollMenuStart.Y + scrollMenuPointsAmount;

struct student\* buffer = getAllStudents(studentService);

printf("Scroll menu buffer filled\n");

ui->scrollMenu = getScrollMenu (buffer,

scrollMenuPointsAmount,

scrollMenuBufferPointsAmount,

scrollMenuStart,

scrollMenuEnd,

0

);

printf("Scroll menu loaded\n");

//File menu

int fileMenuPointsAmount = 4;

COORD fileMenuStart;

fileMenuStart.X = scrollMenuStart.X;

fileMenuStart.Y = mainMenuStart.Y;

COORD fileMenuEnd;

fileMenuEnd.X = fileMenuStart.X + 29;

fileMenuEnd.Y = fileMenuStart.Y + fileMenuPointsAmount;

WCHAR \*\*fileMenuPoints;

fileMenuPoints = (WCHAR\*\*)malloc(fileMenuPointsAmount\*sizeof(WCHAR\*));

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

fileMenuPoints[i] = (WCHAR\*)malloc(15\*sizeof(WCHAR));

}

fileMenuPoints[0] = L"Сохранить в обычный файл";

fileMenuPoints[1] = L"Сохранить в бинарный файл";

fileMenuPoints[2] = L"Загрузить из обычного файла";

fileMenuPoints[3] = L"Загрузить из бинарного файла";

ui->fileMenu = getMenu(

fileMenuPointsAmount,

fileMenuStart,

fileMenuEnd,

fileMenuPoints

);

printf("File menu loaded\n");

//Bad students menu

struct student\* badBuffer = getStudentsWithUnjustifiedHours(studentService);

int badStudentsMenuBufferPointsAmount = studentService->lastFoundStudentsAmount;

printf("Bad students menu buffer filled\n");

COORD badStudentsMenuStart = scrollMenuStart;

COORD badStudentsMenuEnd;

badStudentsMenuEnd.X = scrollMenuStart.X + 68;

badStudentsMenuEnd.Y = scrollMenuEnd.Y;

ui->badStudentsMenu = getScrollMenu (badBuffer,

scrollMenuPointsAmount,

badStudentsMenuBufferPointsAmount,

badStudentsMenuStart,

badStudentsMenuEnd,

1

);

printf("Bad students menu loaded\n");

hidecursor();

return ui;

}

void runUI(struct UI \*ui) {

system("cls");

while(initMainMenu(ui) != -1) {}

freeUI(ui);

return;

}

void freeUI(struct UI \*ui) {

for(int i = 0; i < ui->mainMenu->pointsAmount; i++) {

free(ui->mainMenu->points[i]);

}

free(ui->mainMenu->points);

for(int i = 0; i < ui->scrollMenu->menu->pointsAmount; i++) {

free(ui->scrollMenu->menu->points[i]);

}

free(ui->scrollMenu->menu->points);

free(ui->scrollMenu->buffer);

for(int i = 0; i < ui->fileMenu->pointsAmount; i++) {

free(ui->fileMenu->points[i]);

}

free(ui->fileMenu->points);

for(int i = 0; i < ui->badStudentsMenu->menu->pointsAmount; i++) {

free(ui->badStudentsMenu->menu->points[i]);

}

free(ui->badStudentsMenu->menu->points);

free(ui->badStudentsMenu->buffer);

}

int initMainMenu(struct UI\* ui) {

int choice;

while ((choice = runMenu(ui->mainMenu)) != -1) {

switch(choice) {

case 0:

{

struct student\* student = newStudent();

initStudentMenu(student);

addStudent(ui->studentService, \*student);

}

break;

case 1:

{

struct student\* students = findStudent(ui);

if (!students) {

break;

} else {

ui->scrollMenu->buffer = students;

ui->scrollMenu->bufferPointsAmount = ui->studentService->lastFoundStudentsAmount;

int studentNumber = students[runScrollMenu(ui->scrollMenu)].number;

removeStudent(ui->studentService, studentNumber);

}

updateScrollBuffer(ui);

}

break;

case 2:

initScrollMenu(ui);

updateScrollBuffer(ui);

break;

case 3:

{

struct student\* students = findStudent(ui);

if (!students) {

break;

}

ui->scrollMenu->buffer = students;

ui->scrollMenu->bufferPointsAmount = ui->studentService->lastFoundStudentsAmount;

initScrollMenuForSearch(ui);

}

break;

case 4:

initFileMenu(ui);

break;

case 5:

initBadStudentsMenu(ui);

break;

case 6:

return -1;

}

}

}

struct student\* findStudent(struct UI\* ui) {

struct student\* student = newStudent();

struct studentMenu\* studentMenu = getStudentMenu(student);

int choice = runSubMenu(&studentMenu->menu);

if (choice == -1) return NULL;

changeStudentMenu(studentMenu, choice);

struct student\* students;

switch (choice) {

case 0:

students = getStudentByNumber(ui->studentService, student->number);

break;

case 1:

students = getStudentsByGroup(ui->studentService, student->group);

break;

case 2:

students = getStudentsByFIO(ui->studentService, student->FIO);

break;

case 3:

students = getStudentsByBirthday(ui->studentService, student->birthday);

break;

case 4:

students = getStudentsByGender(ui->studentService, student->gender);

break;

case 5:

students = getStudentsByMissedHours(ui->studentService, student->missedHours);

break;

case 6:

students = getStudentsByJustifiedHours(ui->studentService, student->justifiedHours);

break;

}

return students;

}

void updateScrollBuffer(struct UI\* ui) {

ui->scrollMenu->buffer = getAllStudents(ui->studentService);

ui->scrollMenu->bufferPointsAmount = ui->studentService->studentsAmount;

}

void updateBadBuffer(struct UI\* ui) {

ui->badStudentsMenu->buffer = getStudentsWithUnjustifiedHours(ui->studentService);

ui->badStudentsMenu->bufferPointsAmount = ui->studentService->lastFoundStudentsAmount;

}

int initScrollMenu(struct UI\* ui) {

int choice;

struct scrollMenu\* smenu = ui->scrollMenu;

updateScrollBuffer(ui);

while ((choice = runScrollMenu(smenu)) != -1) {

struct student\* student = &smenu->buffer[choice+smenu->page\*smenu->maxPoints];

initStudentMenu(student);

struct student\* stud = getStudentByNumber(ui->studentService, student->number);

\*stud = \*student;

}

system("cls");

}

int initScrollMenuForSearch(struct UI\* ui) {

int choice;

struct scrollMenu\* smenu = ui->scrollMenu;

while ((choice = runScrollMenu(smenu)) != -1) {

struct student\* student = &smenu->buffer[choice+smenu->page\*smenu->maxPoints];

initStudentMenu(student);

struct student\* stud = getStudentByNumber(ui->studentService, student->number);

\*stud = \*student;

}

}

int initBadStudentsMenu(struct UI\* ui) {

int choice;

struct scrollMenu\* smenu = ui->badStudentsMenu;

updateBadBuffer(ui);

gotoxy(ui->mainMenu->start.X, smenu->menu->end.Y + 2);

wprintf(getStats(ui->studentService));

while ((choice = runScrollMenu(smenu)) != -1) {

struct student\* student = &smenu->buffer[choice+smenu->page\*smenu->maxPoints];

initStudentMenu(student);

struct student\* stud = getStudentByNumber(ui->studentService, student->number);

\*stud = \*student;

}

system("cls");

}

int initStudentMenu(struct student\* student) {

int choice;

struct studentMenu\* studentMenu = getStudentMenu(student);

while ((choice = runSubMenu(&studentMenu->menu)) != -1) {

changeStudentMenu(studentMenu, choice);

}

}

int initFileMenu(struct UI\* ui) {

int choice;

choice = runSubMenu(ui->fileMenu);

switch (choice) {

case 0:

{

char \*fileName = getFileName(ui);

sprintf(fileName, "%s.txt", fileName);

saveStudentsToFile(ui->studentService, fileName);

break;

}

case 1:

{

char \*fileName = getFileName(ui);

sprintf(fileName, "%s.dat", fileName);

saveStudentsToBinaryFile(ui->studentService, fileName);

break;

}

case 2:

{

char \*fileName = getFileName(ui);

sprintf(fileName, "%s.txt", fileName);

loadStudentsFromFile(ui->studentService, fileName);

updateScrollBuffer(ui);

break;

}

case 3:

{

char \*fileName = getFileName(ui);

sprintf(fileName, "%s.dat", fileName);

loadStudentsFromBinaryFile(ui->studentService, fileName);

updateScrollBuffer(ui);

break;

}

}

clearMenu(ui->fileMenu);

}

char\* getFileName(struct UI\* ui) {

gotoxy(ui->scrollMenu->menu->start.X, ui->scrollMenu->menu->start.Y);

wprintf(L"Название файла: ");

char\* fileName = (char\*)malloc(10\*sizeof(char));

while (!GetAsyncKeyState(VK\_RETURN)) {

keybd\_event(VK\_RETURN, 0, KEYEVENTF\_KEYUP, 0);

gotoxy(ui->scrollMenu->menu->start.X + 18, ui->scrollMenu->menu->start.Y);

gets(fileName);

}

Sleep(500);

return fileName;

}

void hidecursor()

{

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

CONSOLE\_CURSOR\_INFO info;

info.dwSize = 100;

info.bVisible = FALSE;

SetConsoleCursorInfo(consoleHandle, &info);

}

Файл UI.h

#ifndef USERINTERFACE

#define USERINTERFACE

#include <stdio.h>

#include <locale.h>

#include "menu.h"

#include "studentService.h"

#include "scrollMenu.h"

#include "studentMenu.h"

struct UI {

struct studentService\* studentService;

struct menu \*mainMenu;

struct scrollMenu \*scrollMenu;

struct menu \*fileMenu;

struct scrollMenu \*badStudentsMenu;

};

struct UI\* getUI(struct studentService \*studentService);

void runUI(struct UI \*ui);

int initMainMenu(struct UI\* ui);

int initScrollMenu(struct UI\* ui);

void updateScrollBuffer(struct UI\* ui);

void updateBadBuffer(struct UI\* ui);

int initStudentMenu(struct student\* student);

struct student\* findStudent(struct UI\* ui);

void hidecursor();

char\* getFileName(struct UI\* ui);

int initFileMenu(struct UI\* ui);

int initBadStudentsMenu(struct UI\* ui);

void freeUI(struct UI \*ui);

int initScrollMenuForSearch(struct UI\* ui);

#endif