**CОДЕРЖАНИЕ**

[Введение 5](#_Toc90584409)

[1 Постановка задачи 6](#_Toc90584410)

[2 Обзор методов решения задачи 7](#_Toc90584411)

[3 Структура входных, выходных и промежуточных данных 8](#_Toc90584412)

[3.1 Входные данные 8](#_Toc90584413)

[3.1.1 applications.txt 8](#_Toc90584414)

[3.1.2 accounts.txt 8](#_Toc90584415)

[3.1.3 default.txt 8](#_Toc90584416)

[3.1.4 social.txt 9](#_Toc90584417)

[3.1.5 president.txt 9](#_Toc90584418)

[3.1.6 personal.txt 9](#_Toc90584419)

[3.2 Промежуточные данные 10](#_Toc90584420)

[3.2.1 Шаблон двусвязного списка «List» 10](#_Toc90584421)

[3.3 Выходные данные 10](#_Toc90584422)

[4 Функциональное проектирование 11](#_Toc90584423)

[4.1 Описание классов приложения 11](#_Toc90584424)

[4.1.1 Application 11](#_Toc90584425)

[4.1.2 Account 11](#_Toc90584426)

[4.1.3 Interface 12](#_Toc90584427)

[4.1.4 Menu 12](#_Toc90584428)

[4.1.5 ListError 12](#_Toc90584429)

[4.1.6 MD5 13](#_Toc90584432)

[4.1.7 Student 13](#_Toc90584433)

[4.1.8 Schoolarship 13](#_Toc90584434)

[4.1.9 Personal 14](#_Toc90584435)

[4.1.10 President 15](#_Toc90584436)

[4.1.11 Social 15](#_Toc90584437)

[5 Разработка программных модулей 17](#_Toc90584438)

[5.1 Метод core() классa Menu 17](#_Toc90584439)

[5.2 Метод SchoolarshipAmountManagement() классa Menu 17](#_Toc90584440)

[5.3 Метод calculateKoef() классa Schoolarship 18](#_Toc90584441)

[5.4 Метод AccountManagement(List <Application>& app, List <Account>& acc, Account& current\_acc) классa Menu 18](#_Toc90584442)

[6 Программа и методика испытаний 19](#_Toc90584443)

[6.1 Проверка ввода чисел 19](#_Toc90584444)

[6.2 Проверка на наличие элементов в контейнере 19](#_Toc90584445)

[7 Описание работы программы 20](#_Toc90584446)

[Заключение 25](#_Toc90584447)

[Список используемых источников 26](#_Toc90584448)

[Приложение А 27](#_Toc90584449)

[Приложение Б 28](#_Toc90584450)

[Приложение В 29](#_Toc90584451)

[Приложение Г 71](#_Toc90584452)

## ВВЕДЕНИЕ

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день С++ является одним из самых популярных и распространенных языков.

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. Кроме того, в последнее время набирает ход мобильное направление, где С++ тоже нашел свое применение. И даже в веб-разработке также можно использовать С++ для создания веб-приложений или каких-то вспомогательных сервисов, которые обслуживают веб-приложения. В общем С++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

# ПОСТАНОВКА ЗАДАЧИ

Целью курсового проекта является разработка объектной модели и ее программная реализация на языке C++. Разработать иерархию классов с использованием наследования. Приложение - консольное.

В рамках работы над курсовым проектом должно быть использовано:

Разработка удобного пользовательского меню, включающего следующие опции:

* авторизация пользователя, хранение пароля в зашифрованном виде;
* просмотр информации;
* добавление, редактирование и удаление записей;
* другие опции в зависимости от специфики задачи.

Исходная информация хранится в текстовых файлах.

Предусмотреть использование в программе следующих аспектов:

1) базовые принципы ООП;

2) стандартные и пользовательские функции, дружественные функции;

3) пространства имен;

4) механизмы абстракции (классы и наследование, шаблоны, динамическое выделение памяти);

5) дружественные классы;

6) лямбда-функции;

7) хеширование паролей;

8) динамическую типизацию.

Использовать инкапсуляцияю, перегрузку методов, переопределение методов, абстрактные типы данных (интерфейсы, абстрактные классы), передачу параметров по ссылке и по значению, статические методы.

Разработать и использовать в программе шаблонный класс Список. Производить обработку исключительных ситуаций на языке Си.

Для реализации используется объектно-ориентированный язык программирования С++, среда разработки Microsoft Visual Studio 2019. Приложение написано на ОС Windows 10.

# ОБЗОР МЕТОДОВ РЕШЕНИЯ ЗАДАЧИ

Представленную задачу можно реализовать, используя различные языки программирования.

Программа могла быть написана на языке C#. Это объектно-ориентированный язык программирования, один из наиболее быстро растущих, востребованных и при этом «удобных» языков программирования. Это модификация фундаментального языка С от компании Microsoft, призванная создать наиболее универсальное средство для разработки программного обеспечения для большого количества устройств и операционных систем.

Также данная программа могла бы быть реализована на языке Python. Python - это высокоуровневый, интерпретируемый, интерактивный и объектно-ориентированный скриптовой язык программирования. Python был разработан как "легкочитаемый" язык, часто использующий в качестве ключевых слов слова английского языка. Python является интерпретируемым: исходный код на Python не компилируется в машинный код, а выполняется непосредственно с помощью специальной программы-интерпретатора. Python это интерактивный язык: Это означает, что вы можете писать код прямо в оболочке интерпретатора и вводить новые команды по мере выполнения предыдущих. Python является объектно-ориентированым языком программирования. Python поддерживает принципы ООП, которые подразумевают инкапсуляцию кода в особые структуры, именуемые объектами.

При разработке программы могли быть использованы различные СУБД, например MySQL, SQLite, для хранения и управления данными.

# СТРУКТУРА ВХОДНЫХ, ВЫХОДНЫХ И ПРОМЕЖУТОЧНЫХ ДАННЫХ

После определения требований к функционалу разрабатываемой

программы, ее следует разбить на функциональные блоки. Такой подход упростит понимание кода, а также обеспечит возможность корректировки и добавления различного дополнительного функционала.

## Входные данные

## applications.txt

В файле applications.txt хранятся данные о новых заявках на регистрацию в системе.

Таблица 3.1.1 «Содержимое файла applications.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Логин пользователя |
| string | Пароль пользователя |

## accounts.txt

В файле accounts.txt хранятся данные о всех аккаунтах.

Таблица 3.1.2 «Содержимое файла accounts.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Логин пользователя |
| string | Пароль пользователя |
| bool | Флаг роли  (Администратор/пользователь) |

## default.txt

В файле default.txt хранятся данные о студентах, получающих учебную стипендию.

Таблица 3.1.3 «Содержимое файла default.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Фамилия |
| int | Номер группы |
| bool | Флаг активности  (Активен/неактивен) |
| int | Средний балл |
| double | Стипендия |

## social.txt

В файле social.txt хранятся данные о студентах, получающих социальную стипендию.

Таблица 3.1.4 «Содержимое файла social.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Фамилия |
| int | Номер группы |
| bool | Флаг активности  (Активен/неактивен) |
| int | Средний балл |
| double | Стипендия |
| int | Причина присвоения  (номер константы) |

## president.txt

В файле president.txt хранятся данные о студентах, получающих президентскую стипендию.

Таблица 3.1.5 «Содержимое файла president.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Фамилия |
| int | Номер группы |
| bool | Флаг активности  (Активен/неактивен) |
| int | Средний балл |
| double | Стипендия |
| int | Год присвоения |

## personal.txt

В файле personal.txt хранятся данные о студентах, получающих именную стипендию.

Таблица 3.1.6 «Содержимое файла personal.txt»

|  |  |
| --- | --- |
| Тип данных | Содержимое |
| string | Фамилия |
| int | Номер группы |
| bool | Флаг активности  (Активен/неактивен) |
| int | Средний балл |
| double | Стипендия |
| string | Название стипендии |

## Промежуточные данные

Промежуточными данными в данном курсовом проекте, является шаблон двусвязного списка. Данный шаблон позволяет выводить на консоль, добавлять, редактировать, обменивать местами, удалять данные.

## Шаблон двусвязного списка «List»

template<typename T>

class List

{

private:

template<typename T>

struct Node

{

public:

T Data;

Node<T>\* Next;

Node<T>\* Previous;

};

Node<T>\* First; // Указатель на первый элемент списка

Node<T>\* Last; // Указатель на последний элемент списка

int size = 1;

public:

List() {};

~List() {};

void ListAdd(T data); // добавление объекта в лист

bool isListEmpty(); // проверка на наличие объектов в листе

void clean(); // очистка листа

void del(int pos); // удаление конкретного элемента листа по индексу

T& operator[](int pos); // перегрузка оператора [] для удобного обращения к элементам листа

int Size(); // размер листа

void sort(function<bool(T obj1, T obj2)> comp);

// сортировка через лямбда-функцию

template<typename Attr>

void find(function<bool(T obj, Attr attr)> equalFunc, Attr \_attr);

// поиск через лямбда-функцию

## Выходные данные

После работы пользователя, измененные данные перезаписываются в файлы, указанные в пункте 3.1.

# ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описывается функционирование и структура классов.

## Описание классов приложения

## Application

Класс для заявок.

**Поля:**

* string login; - Логин аккаунта
* string pass; - Пароль аккаунта в хешированном виде

**Методы:**

* static void header(); - Шапка для таблицы
* static void read(List <Application>& list); - Чтение из файла
* static void write(List <Application>& list); - Запись в файл
* string getLogin(); - Геттер для логина
* string getHash(); - Геттер для хеша пароля

**Операторы:**

* friend istream& operator >> (istream& in, Application& a); - Перегрузка ввода
* friend ostream& operator << (ostream& out, Application& a);

- Перегрузка вывода

## Account

Класс для аккаунтов.

**Поля:**

* bool isAdmin; - Параметр роли (true -> Admin / false -> User)

**Методы:**

* static void header(); - Шапка для таблицы
* static string inputPass(); - Функция ввода пароля с маскировкой в виде звездочек
* void changeRole(); - Метод изменения роли аккаунта
* void changePassword(List <Account>& list, Account a); - Метод изменения пароля аккаунта
* static void read(List <Account>& list); - Чтение из файла
* static void write(List <Account>& list); - Запись в файл
* static void newApplication(List <Application>& app,

List <Account>& acc); - Метод регистрации новой заявки

* void setlog(); - Сеттер для логина
* void setlog(string a); - Сеттер для логина
* void setpass(string a); - Сеттер для пароля
* void replacePass(string a); - Сеттер для хеша пароля
* bool getRole(); - Геттер для роли

**Операторы:**

* friend istream& operator >> (istream& in, Account& a); - Перегрузка ввода
* friend ostream& operator << (ostream& out, Account& a); - Перегрузка вывода

## Interface

Класс для интерфейсов.

**Методы:**

* static void AdminUI(); - Интерфейс для администратора
* static void UserUI(); - Интерфейс для пользователя

## Menu

Класс для меню.

**Методы:**

* static void core(); - Главная функция программы
* static void login(List <Application>& app, List <Account>& acc,

Account& current\_acc); - Меню авторизации

* static void save(List <Application>& app, List <Account>& acc);

- Сохранение перед выходом

* static void AdminMenu(List <Application>& app, List <Account>& acc, Account& current\_acc); - Главное меню администратора
* static void UserMenu(List <Account>& acc, Account& current\_acc);

- Главное меню пользователя

* static void AdminMenuOfScholars(); - Меню со стипендиантами для админимтратора
* static void UserMenuOfScholars(); - Меню со стипендиантами для пользователя
* static void SchoolarshipAmountManagement(); - Меню для изменения базовых стипендий
* static void AccountManagement(List <Application>& app,

List <Account>& acc, Account& current\_acc); - Модуль управления аккаунтами

* static void showAccounts(List <Account>& acc, Account& current\_acc);

- Меню с аккаунтами

* static void showNewApplications(List <Application>& app,

List <Account>& acc, Account& current\_acc); - Меню с заявками

## ListError

Класс для вывода ошибок работы со списком.

## Поля:

## string error; - Текст ошибки

**Методы:**

* ListError(const char\* error); - Конструктор
* const char\* error\_msg(); - Отображение ошибки

## MD5

Класс для хеширования.

**Методы:**

* string md5(const string str); - Преобразование строки в хеш

## Student

Класс для структуризации информации о студенте.

**Поля:**

* int mark; - Средний балл
* string surname; - Фамилия
* bool isActive; - Активность студента
* int group; - Номер группы

**Методы:**

* virtual void header() = 0; - Шапка для таблицы
* void setGroup(); - Сеттер для номера группы
* int getGroup(); - Геттер для номера группы
* void setActivity(); - Сеттер для активности
* string getActivity(); - Геттер для активности
* void setSurname(); - Сеттер для фамилии
* string getSurname(); - Геттер для фамилии
* void setMark(); - Сеттер для среднего балла
* int getMark(); - Геттер для среднего балла

## Schoolarship

Класс для учебной стипендии.

**Поля:**

* double sum; - Размер стипендии
* static int base; - Базовая стипендия
* double koef ; - Коэффициент для расчета стипендии (относительно среднего балла)

**Методы:**

* void header(); - Шапка для таблицы
* virtual void edit(); - Метод изменения объекта
* virtual void calculateSum(); - Метод пересчета стипендии
* void calculateKoef(); - Метод пересчета коэффициента
* static void find(List <Schoolarship>& list); - Поиск по различным параметрам
* static void sort(List <Schoolarship>& list); - Сортировка по различным параметрам
* static void read(List <Schoolarship>& list); - Чтение из файла
* static void write(List <Schoolarship>& list); - Запись в файл
* static void setBase(); - Сеттер для базовой учебной стипендии
* static int getBase(); - Геттер для базовой учебной стипендии
* double getSum(); - Сеттер для коеффициента
* double getKoef(); - Геттер для коеффициента

**Операторы:**

* friend istream& operator >> (istream& in, Schoolarship& student);

- Перегрузка ввода

* friend ostream& operator << (ostream& out, Schoolarship& student);

- Перегрузка вывода

## Personal

Класс для именной стипендии.

**Поля:**

* static int amount\_pers; - Базовый размер именной стипендии
* string author; - Название стипендии

**Методы**

* void header(); - Шапка для таблицы
* void edit(); - Метод изменения объекта
* void calculateSum(); - Метод пересчета стипендии
* static void find(List <Personal>& list); - Поиск по различным параметрам
* static void sort(List <Personal>& list); - Сортировка по различным параметрам
* static void read(List <Personal>& list); - Чтение из файла
* static void write(List <Personal>& list); - Запись в файл
* static void setAmountPersonal(); - Сеттер для базовой именной стипендии
* static int getAmountPersonal(); - Геттер для базовой именной стипендии
* void setAuthor(); - Сеттер для названия
* string getAuthor(); - Геттер для названия

**Операторы:**

* friend istream& operator >> (istream& in, Personal& student);

- Перегрузка ввода

* friend ostream& operator << (ostream& out, Personal& student);

- Перегрузка вывода

## President

Класс для президентской стипендии.

**Поля:**

* static int amount\_p; - Базовый размер президентской стипендии
* int year; - Год присвоения

**Методы:**

* void header(); - Шапка для таблицы
* void edit(); - Метод изменения объекта
* void calculateSum(); - Метод пересчета стипендии
* static void find(List <President>& list); - Поиск по различным параметрам
* static void sort(List <President>& list); - Сортировка по различным параметрам
* static void read(List <President>& list); - Чтение из файла
* static void write(List <President>& list); - Запись в файл
* static void setAmountPresident(); - Сеттер для базовой президентской стипендии
* static int getAmountPresident(); - Геттер для базовой президентской стипендии
* void setYear(); - Сеттер для года присвоения
* int getYear(); - Геттер для года присвоения

**Операторы:**

* friend istream& operator >> (istream& in, President& student);

- Перегрузка ввода

* friend ostream& operator << (ostream& out, President& student);

- Перегрузка вывода

## Social

Класс для социальной стипендии.

**Поля:**

* static int amount\_s; - Базовый размер социальной стипендии
* int cause; - Причина присвоения социальной стипендии (номер константы)

**Методы:**

* void header(); - Шапка для таблицы
* void edit(); - Метод изменения объекта
* void calculateSum(); - Метод пересчета стипендии
* static void find(List <Social>& list); - Поиск по различным параметрам
* static void sort(List <Social>& list); - Сортировка по различным параметрам
* static void read(List <Social>& list); - Чтение из файла
* static void write(List <Social>& list); - Запись в файл
* static void setAmountSocial(); - Сеттер для базовой социальной стипендии
* static int getAmountSocial(); - Геттер для базовой социальной стипендии
* void setCause(); - Сеттер для причины
* string getCause(); - Геттер для причины

**Операторы:**

* friend istream& operator >> (istream& in, Social& student); - Перегрузка ввода
* friend ostream& operator << (ostream& out, Social& student); - Перегрузка вывода

# РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## Метод core() классa Menu

1. Начало.
2. Объявление списка app для типа данных Application.
3. Объявление списка acc для типа данных Account.
4. Вызов метода Application::read(app) для чтения из файла fin данных о заявках в список app.
5. Вызов метода Account::read(acc) для чтения из файла fin данных о аккаунтах в список acc.
6. Создание переменной current\_account типа Account для хранения информации о текущем аккаунте.
7. Создание и инициализация переменной core\_flag=true типа bool для хранения состояния программы. При значении false, завершается работа.
8. Начало цикла while (core\_flag == true) в качестве основного цикла программы.
9. Очистка консоли.
10. Установка цветов консоли color 8E.
11. Вывод сообщения с вариантами действий.
12. Создание переменной user\_choice типа int для хранения выбора пользователя.
13. Присвоение user\_choice значения, возвращаемого функцией inputNumber(0,2).
14. Очистка консоли.
15. Если user\_choice равняется 0, core\_flag присваивается значение false для выхода из цикла и завершения работы программы.
16. Если user\_choice равняется 1, вызывается функция Menu::login(app,acc,current\_acc) для перехода в меню авторизации.
17. Если user\_choice равняется 2, вызывается функция Account::newApplication(app,acc) для создания новой заявки на регистрацию в системе.
18. Конец цикла while.
19. Вызов функции Menu::save(app,acc) для сохранения данных об аккаунтах и заявках.
20. Конец.

## Метод SchoolarshipAmountManagement() классa Menu

1. Начало.
2. Создание переменной choice типа int для хранения выбора пользователя.
3. Создание и инициализация переменной flag=true типа bool для последующего выхода из цикла, при необходимости.
4. Начало цикла while (flag == true) в качестве основного цикла в модуле по изменению размеров базовых стипендий.
5. Очистка консоли.
6. Вывод информации об актуальных стипендиях.
7. Вывод сообщения с вариантами действий.
8. Присвоение choice значения, возвращаемого функцией inputNumber(0,4).
9. Если choice равняется 0, flag присваивается значение false для выхода из цикла.
10. Если choice равняется 1, устанавливаем значение для базовой учебной стипендии.
11. Если choice равняется 2, устанавливаем значение для базовой именной стипендии.
12. Если choice равняется 3, устанавливаем значение для базовой президентской стипендии.
13. Если choice равняется 4, устанавливаем значение для базовой социальной стипендии.
14. Конец цикла while.
15. Конец.

## Метод calculateKoef() классa Schoolarship

Алгоритм данного метода изображен на блок-схеме в приложении А.

## Метод AccountManagement(List <Application>& app, List <Account>& acc, Account& current\_acc) классa Menu

Алгоритм данного метода изображен на блок-схеме в приложении А.

## 

# ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

## Проверка ввода чисел

Обработаны ситуации некорректного ввода целых чисел на всех этапах программы.

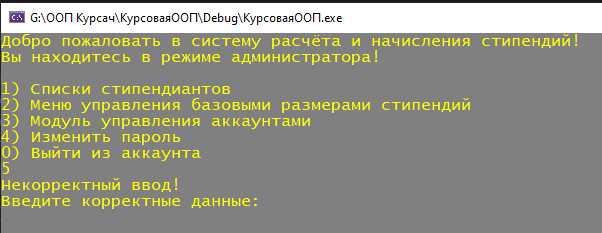


Рисунок 6.1 – Ввод некорректного числа

## Проверка на наличие элементов в контейнере

При отображении элементов пустого списка, появляется сообщение «Список пуст»

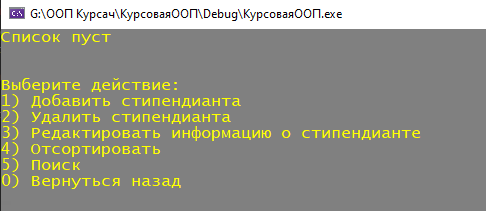


Рисунок 6.2 – Обращение к пустому списку

# ОПИСАНИЕ РАБОТЫ ПРОГРАММЫ

В начале авторизации программа предлагает пользователю войти, зарегистрировать новый аккаунт или выйти из системы (рисунок 7.1).

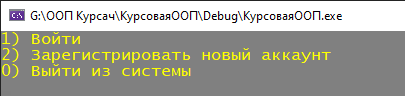


Рисунок 7.1

Если у вас еще нет аккаунта в системе, необходимо зарегистрироваться. После одобрения администратором вашей заявки, вам будет открыт доступ к системе (рисунок 7.2).

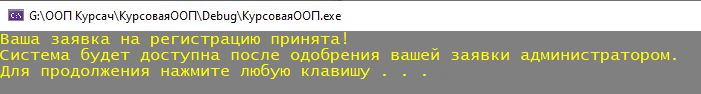


Рисунок 7.2 – Регистрация нового аккаунта

Когда ваша заявка одобрена, можно осуществить авторизацию. После этого вас приветствует главное меню программы. У администратора есть 3 модуля для работы, а так же возможность изменения пароля своего аккаунта (рисунок 7.3).

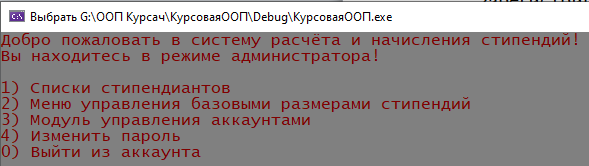


Рисунок 7.3 – Главное меню администратора

В первом меню ведется основная работа системе. Оно предлагает 4 списка со стипендиантами (рисунок 7.4).

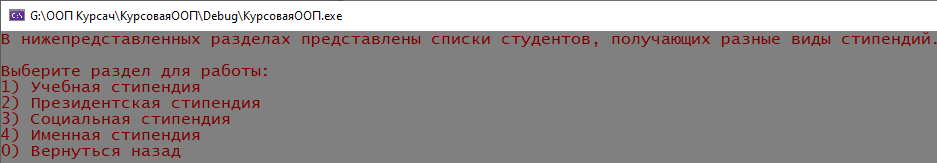


Рисунок 7.4 – Выбор списка

На примере первого списка можно рассмотрим функционал нашей системы (рисунок 7.5)

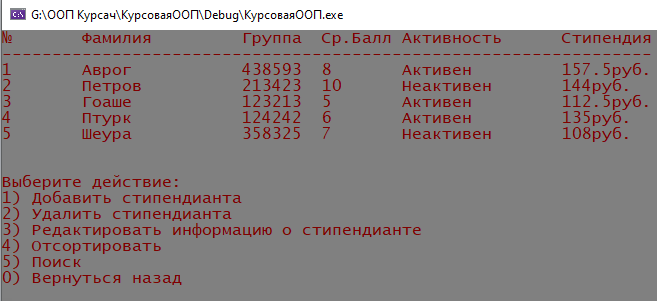


Рисунок 7.5 – Функционал администратора

Все вариации сортировок и поиска можно увидеть, нажав на 4 или 5, соответственно (рисунок 7.6 и рисунок 7.7).

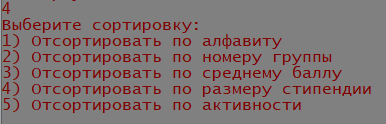


Рисунок 7.6 – Сортировки

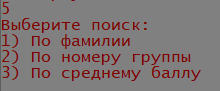


Рисунок 7.7 – Поиск

Модуль управления аккаунтами разделяется на два подмодуля: управление аккаунтами и заявками (рисунок 7.8).

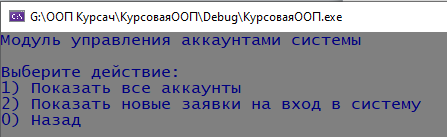


Рисунок 7.8 – Модуль управления аккаунтами

В данных подмодулях реализован функционал для обработки аккаунтов и заявок на регистрацию новых аккаунтов (рисунок 7.9 и

рисунок 7.10).

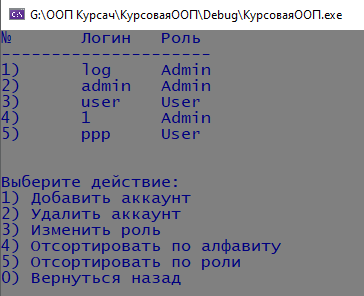


Рисунок 7.9 – Подмодуль управления аккаунтами

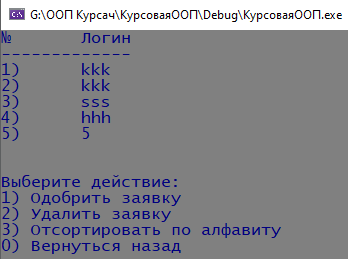


Рисунок 7.10 – Подмодуль управления заявками

На рисунке 7.11 изображено меню изменения размеров базовых стипендий.

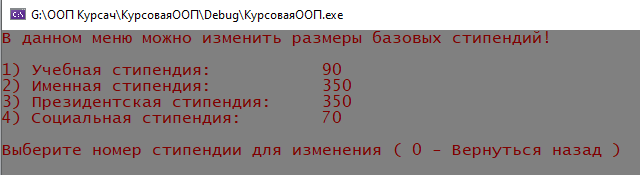


Рисунок 7.11 – Меню изменения стипендий

Главное меню пользователя и его ограниченный функционал в системе изображено на рисунках 7.12-7.13.

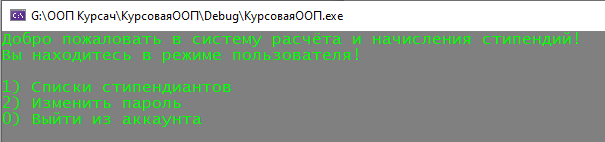


Рисунок 7.12 – Главное меню пользователя

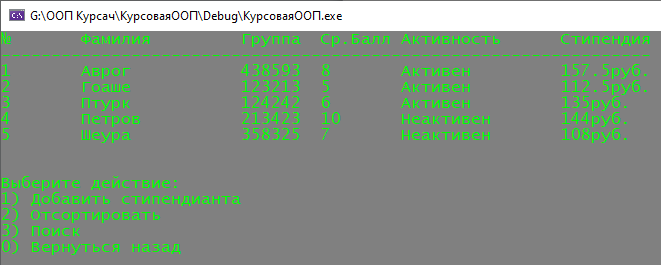


Рисунок 7.13 – Функционал пользователя

У всех аккаунтов есть возможность изменения пароля аккаунта. Данный процесс изображен на рисунке 7.14.

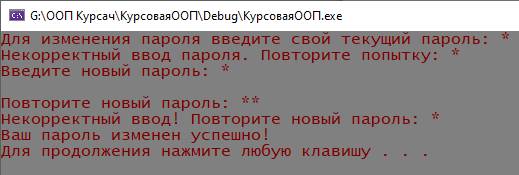


Рисунок 7.14 – Изменение пароля

# ЗАКЛЮЧЕНИЕ

В результате работы над данным курсовым проектом была разработана работоспособная информационная система со своим набором системных функций и возможностей. Данный курсовой проект был разработан в соответствии с поставленными задачами, весь функционал был реализован в полном объеме.

Для создания программного продукта были подробно исследованы и применены принципы ООП.

В ходе разработки были изучены возможности языка программирования C++, а конкретно объектно-ориентированного программирования, получены навыки работы в отладчике Visual Studio 2019, опыт написания классов и много другое.

Работа была разделена на такие этапы, как анализ существующих аналогов, литературных источников, постановка требований к проектируемому программному средству, системное и функциональное проектирование, конструирование программного средства, разработка программных модулей и тестирование проекта. После последовательного выполнения вышеперечисленных этапов разработки была получена исправно работающая программа.

В дальнейшем планируется усовершенствование текущего функционала, путем усовершенствования графического интерфейса, добавления новых функций, новых режимов работы.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Шилдт, Г. Полный справочник по C++ / Г. Шилдт – Москва: Вильямс, 2007 – 800 с.
2. Объектно-ориентированное программирование на языке С++/Ю.А. Луцик, В.Н. Комличенко – Минск: БГУИР, 2008, - 266 с.

# ПРИЛОЖЕНИЕ А

(обязательное)

**Блок-схемы алгоритмов**

# ПРИЛОЖЕНИЕ Б

(обязательное)

**Диаграмма классов**

# ПРИЛОЖЕНИЕ В

(обязательное)

**Листинг кода**

Account.h

#pragma once

#include "Application.h"

#include <conio.h>

class Account : public Application

{

friend class Menu;

protected:

bool isAdmin = false; // Параметр роли (true -> Admin / false -> User)

public:

Account() {

login = "0";

pass = "0";

isAdmin = false;

}

~Account() {};

static void header(); // Шапка для таблицы

static string inputPass(); // Функция ввода пароля с маскировкой в виде звездочек

void changeRole(); // Метод изменения роли аккаунта

void changePassword(List <Account>& list, Account a); // Метод изменения пароля аккаунта

static void read(List <Account>& list); // Чтение из файла

static void write(List <Account>& list); // Запись в файл

static void newApplication(List <Application>& app, List <Account>& acc); // Метод регистрации новой заявки

friend istream& operator >> (istream& in, Account& a); // Перегрузка ввода

friend ostream& operator << (ostream& out, Account& a); // Перегрузка вывода

void setlog();

void setlog(string a);

void setpass(string a);

void replacePass(string a);

bool getRole();

};

Application.h

#pragma once

#include <iostream>

#include <fstream>

#include <string>

#include "List.h"

#include "md5.h"

const string FILE\_WITH\_APPLICATIONS = "applications.txt";

const string FILE\_WITH\_ACCOUNTS = "accounts.txt";

class Application

{

friend class Account;

protected:

string login; // Логин аккаунта

string pass; // Пароль аккаунта в хешированном виде

public:

Application() {

login = "0";

pass = "0";

};

~Application() {};

static void header(); // Шапка для таблицы

static void read(List <Application>& list); // Чтение из файла

static void write(List <Application>& list); // Запись в файл

friend istream& operator >> (istream& in, Application& a); // Перегрузка ввода

friend ostream& operator << (ostream& out, Application& a); // Перегрузка вывода

string getLogin();

string getHash();

};

Interface.h

#pragma once

#include "Account.h"

#include "President.h"

#include "Personal.h"

#include "Social.h"

#include "List.h"

#include "validation.h"

#include <iostream>

#include <algorithm>

using namespace std;

template<typename T>

class Interface

{

public:

static void AdminUI(); // Интерфейс для администратора

static void UserUI(); // Интерфейс для пользователя

};

template<typename T>

void Interface<T>::AdminUI() {

List<T> list;

T::read(list);

for (int i = 0; i < list.Size(); i++) list[i].calculateSum();

T temp;

bool flag = true;

int choice, del\_choice, confirm, edit\_choice;

while (flag) {

if (list.isListEmpty())

{

system("cls");

cout << "Список пуст" << endl;

}

else

{

system("cls");

list[0].header();

for (int i = 0; i < list.Size(); i++) {

cout << i + 1 << "\t" << list[i] << endl;

}

}

cout << "\n\nВыберите действие:\n1) Добавить стипендианта\n2) Удалить стипендианта\n3) Редактировать информацию о стипендианте\n";

cout << "4) Отсортировать\n5) Поиск\n0) Вернуться назад\n";

choice = inputNumber(0, 5);

switch (choice)

{

case 0:

flag = false;

break;

case 1:

cin >> temp;

list.ListAdd(temp);

break;

case 2:

if (list.isListEmpty())

{

cout << "Список пуст" << endl;

system("pause");

}

else

{

cout << "Введите номер стипендианта для удаления:\n";

del\_choice = inputNumber(1, list.Size());

system("cls");

cout << "Вы действительно хотите удалить стипендианта с фамилией " << list[del\_choice - 1].getSurname() << " ?\n1) Да\n2) Нет\n";

confirm = inputNumber(1, 2);

if (confirm == 1) list.del(del\_choice - 1);

}

break;

case 3:

if (list.isListEmpty())

{

cout << "Список пуст" << endl;

system("pause");

}

else

{

cout << "Введите номер стипендианта для редактирования:";

edit\_choice = inputNumber(1, list.Size());

list[edit\_choice - 1].edit();

}

break;

case 4:

T::sort(list);

break;

case 5:

T::find(list);

break;

}

}

T::write(list);

}

template<typename T>

void Interface<T>::UserUI() {

List<T> list;

T::read(list);

T temp;

bool flag = true;

int choice;

while (flag) {

if (list.isListEmpty())

{

system("cls");

cout << "Список пуст" << endl;

}

else

{

system("cls");

list[0].header();

for (int i = 0; i < list.Size(); i++) {

cout << i + 1 << "\t" << list[i] << endl;

}

}

cout << "\n\nВыберите действие:\n1) Добавить стипендианта\n";

cout << "2) Отсортировать\n3) Поиск\n0) Вернуться назад\n";

choice = inputNumber(0, 3);

switch (choice)

{

case 0:

flag = false;

break;

case 1:

cin >> temp;

list.ListAdd(temp);

break;

case 2:

T::sort(list);

break;

case 3:

T::find(list);

break;

}

}

T::write(list);

}

List.h

#pragma once

#include <iostream>

#include <functional>

#include "ListError.h"

using namespace std;

template<typename T>

class List

{

private:

template<typename T>

struct Node

{

public:

T Data;

Node<T>\* Next;

Node<T>\* Previous;

};

Node<T>\* First; // Указатель на первый элемент списка

Node<T>\* Last; // Указатель на последний элемент списка

int size = 1;

public:

List();

~List() {

clean();

};

void ListAdd(T data); // добавление объекта в лист

bool isListEmpty(); // проверка на наличие объектов в листе

void clean(); // очистка листа

void del(int pos); // удаление конкретного элемента листа по индексу

T& operator[](int pos); // перегрузка оператора [] для удобного обращения к элементам листа

int Size(); // размер листа

void sort(function<bool(T obj1, T obj2)> comp); // сортировка через

лямбда-функцию

template<typename Attr>

void find(function<bool(T obj, Attr attr)> equalFunc, Attr \_attr); // поиск через лямбда-функцию

};

template<typename T>

List<T>::List()

{

First = NULL;

Last = NULL;

size = 0;

}

template<typename T>

bool List<T>::isListEmpty() {

Node<T>\* Token = this->First;

if (this->First == NULL) return true;

else return false;

}

template<typename T>

void List<T>::ListAdd(T data)

{

Node<T>\* Item = new Node<T>();

Item->Data = data;

Item->Next = NULL;

Item->Previous = Last;

Last = Item;

if (First == NULL)

{

First = Item;

}

else

{

Node<T>\* Token = this->First;

while ((Token->Next) != NULL)

{

Token = Token->Next;

}

Token->Next = Item;

}

}

template<typename T>

int List<T>::Size()

{

int num = 0;

Node<T>\* Token = this->First;

if (this->First == NULL)

{

return 0;

}

while (Token->Next != NULL)

{

Token = Token->Next;

num++;

}

num++;

return num;

}

template<typename T>

void List<T>::clean()

{

Node<T>\* p, \* q;

p = First;

while (p != NULL)

{

q = p->Next;

delete p;

p = q;

}

First = NULL;

Last = NULL;

}

template<typename T>

T& List<T>::operator[](int location)

{

T obj = T();

try {

if (this->Size() == 0) throw ListError("Список пуст!");

if (location + 1 > this->Size() || location < 0) throw ListError("Неправильный индекс!");

}

catch(ListError& e){

cout << e.error\_msg() << endl;

}

if (location == 0)

{

return First->Data;

}

Node<T>\* Token = this->First;

if (location == 0)

{

cout << First->Data;

}

else

{

for (int k = 0; k < location; k++)

{

Token = Token->Next;

}

}

return Token->Data;

}

template<typename T>

void List<T>::del(int pos)

{

Node<T>\* Token = First;

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

{

if (Token == First && size == 1 && i == pos)

{

Token->Next = NULL;

Token->Previous = NULL;

First = NULL;

Last = NULL;

size = 0;

return;

}

if (Token == First && i == pos)

{

Token->Next->Previous = NULL;

First = Token->Next;

size--;

continue;

}

else

{

if (Token == Last && i == pos)

{

Last = Token->Previous;

Token->Previous->Next = NULL;

size--;

return;

}

else

if (size == 3 && i == pos)

{

First->Next = Last;

Last->Previous = First;

size--;

continue;

}

else if (i == pos)

{

Token->Next->Previous = Token->Previous;

Token->Previous->Next = Token->Next;

delete Token;

size--;

continue;

}

}

Token = Token->Next;

}

}

template <typename T>

void List<T>::sort(function<bool(T obj1, T obj2)> comp)

{

Node<T>\* i, \* j;

for (i = this->First; i->Next != nullptr; i = i->Next)

{

for (j = i->Next; j != nullptr; j = j->Next)

{

if (comp(i->Data, j->Data))

{

swap(i->Data, j->Data);

}

}

}

}

template<typename T>

template<typename Attr>

void List<T>::find(function<bool(T obj, Attr attr)> equalFunc, Attr \_attr)

{

Node<T>\* tempNode = First;

int i = 0;

while (tempNode != nullptr)

{

if (equalFunc(tempNode->Data, \_attr)) {

i++;

if (i == 1) tempNode->Data.header();

cout <<i <<")\t" <<tempNode->Data<<endl;

}

tempNode = tempNode->Next;

}

if (i == 0) cout << "По вашему запросу ничего не найдено!\n";

}

ListError.h

#pragma once

#include <stdexcept>

using namespace std;

class ListError : public exception

{

public:

ListError(const char\* error) : exception(error) { this->error == error; }

const char\* error\_msg() { return error.c\_str(); }

private:

string error;

};

md5.h

#ifndef BZF\_MD5\_H

#define BZF\_MD5\_H

#include <cstring>

#include <iostream>

// a small class for calculating MD5 hashes of strings or byte arrays

// it is not meant to be fast or secure

//

// usage: 1) feed it blocks of uchars with update()

// 2) finalize()

// 3) get hexdigest() string

// or

// MD5(std::string).hexdigest()

//

// assumes that char is 8 bit and int is 32 bit

class MD5

{

public:

typedef unsigned int size\_type; // must be 32bit

MD5();

MD5(const std::string& text);

void update(const unsigned char\* buf, size\_type length);

void update(const char\* buf, size\_type length);

MD5& finalize();

std::string hexdigest() const;

friend std::ostream& operator<<(std::ostream&, MD5 md5);

private:

void init();

typedef unsigned char uint1; // 8bit

typedef unsigned int uint4; // 32bit

enum { blocksize = 64 }; // VC6 won't eat a const static int here

void transform(const uint1 block[blocksize]);

static void decode(uint4 output[], const uint1 input[], size\_type len);

static void encode(uint1 output[], const uint4 input[], size\_type len);

bool finalized;

uint1 buffer[blocksize]; // bytes that didn't fit in last 64 byte chunk

uint4 count[2]; // 64bit counter for number of bits (lo, hi)

uint4 state[4]; // digest so far

uint1 digest[16]; // the result

// low level logic operations

static inline uint4 F(uint4 x, uint4 y, uint4 z);

static inline uint4 G(uint4 x, uint4 y, uint4 z);

static inline uint4 H(uint4 x, uint4 y, uint4 z);

static inline uint4 I(uint4 x, uint4 y, uint4 z);

static inline uint4 rotate\_left(uint4 x, int n);

static inline void FF(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void GG(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void HH(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

static inline void II(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac);

};

std::string md5(const std::string str);

#endif

Menu.h

#pragma once

#include "Interface.h"

using namespace std;

class Menu

{

public:

Menu() {

}

static void core(); // Главная функция программы

static void login(List <Application>& app, List <Account>& acc, Account& current\_acc); // Меню авторизации

static void save(List <Application>& app, List <Account>& acc);

// Сохранение перед выходом

static void AdminMenu(List <Application>& app, List <Account>& acc, Account& current\_acc); // Главное меню администратора

static void UserMenu(List <Account>& acc, Account& current\_acc);

// Главное меню пользователя

static void AdminMenuOfScholars(); // Меню со стипендиантами для админимтратора

static void UserMenuOfScholars(); // Меню со стипендиантами для пользователя

static void SchoolarshipAmountManagement(); // Меню для изменения базовых стипендий

static void AccountManagement(List <Application>& app, List <Account>& acc, Account& current\_acc); // Модуль управления аккаунтами

static void showAccounts(List <Account>& acc, Account& current\_acc);

// Меню с аккаунтами

static void showNewApplications(List <Application>& app,List <Account>& acc, Account& current\_acc); // Меню с заявками

};

Personal.h

#pragma once

#include "Schoolarship.h"

#include "validation.h"

#include <iostream>

class Personal : public Schoolarship

{

protected:

static int amount\_pers; // Базовый размер именной стипендии

string author; // Название стипендии

public:

Personal () {};

~Personal() {};

void header(); // Шапка для таблицы

void edit(); // Метод изменения объекта

void calculateSum(); // Метод пересчета стипендии

static void find(List <Personal>& list); // Поиск по различным параметрам

static void sort(List <Personal>& list); // Сортировка по различным параметрам

static void read(List <Personal>& list); // Чтение из файла

static void write(List <Personal>& list); // Запись в файл

friend istream& operator >> (istream& in, Personal& student);

// Перегрузка ввода

friend ostream& operator << (ostream& out, Personal& student);

// Перегрузка вывода

static void setAmountPersonal();

static int getAmountPersonal();

void setAuthor();

string getAuthor();

};

President.h

#pragma once

#include "Schoolarship.h"

class President : public Schoolarship

{

protected:

static int amount\_p; // Базовый размер президентской стипендии

int year = 2021; // Год присвоения

public:

President() {};

~President() {};

void header(); // Шапка для таблицы

void edit(); // Метод изменения объекта

void calculateSum(); // Метод пересчета стипендии

static void find(List <President>& list); // Поиск по различным параметрам

static void sort(List <President>& list); // Сортировка по различным параметрам

static void read(List <President>& list); // Чтение из файла

static void write(List <President>& list); // Запись в файл

friend istream& operator >> (istream& in, President& student);

// Перегрузка ввода

friend ostream& operator << (ostream& out, President& student);

// Перегрузка вывода

static void setAmountPresident();

static int getAmountPresident();

void setYear();

int getYear();

};

Schoolarship.h

#pragma once

#include "Student.h"

#include "List.h"

#include <fstream>

const string FILE\_WITH\_DEFAULT = "default.txt";

const string FILE\_WITH\_PRESIDENT = "president.txt";

const string FILE\_WITH\_SOCIAL = "social.txt";

const string FILE\_WITH\_PERSONAL = "personal.txt";

class Schoolarship : public Student

{

protected:

double sum = 0; // Размер стипендии

static int base; // Базовая стипендия

double koef = 1; // Коэффициент для расчета стипендии (относительно среднего балла)

public:

Schoolarship() {}

~Schoolarship() {}

void header(); // Шапка для таблицы

virtual void edit(); // Метод изменения объекта

virtual void calculateSum(); // Метод пересчета стипендии

void calculateKoef(); // Метод пересчета коэффициента

static void find(List <Schoolarship>& list); // Поиск по различным параметрам

static void sort(List <Schoolarship>& list); // Сортировка по различным параметрам

static void read(List <Schoolarship>& list); // Чтение из файла

static void write(List <Schoolarship>& list); // Запись в файл

friend istream& operator >> (istream& in, Schoolarship& student);

// Перегрузка ввода

friend ostream& operator << (ostream& out, Schoolarship& student);

// Перегрузка вывода

static void setBase();

static int getBase();

double getSum();

double getKoef();

};

Social.h

#pragma once

#include "Schoolarship.h"

// Константы (причины присвоения)

const string cause1 = "Инвалидность";

const string cause2 = "Малообеспеченная семья";

const string cause3 = "Проживание в зоне АЭС";

const string cause4 = "Сирота";

class Social : public Schoolarship

{

protected:

static int amount\_s; // Базовый размер социальной стипендии

int cause = 0; // Причина присвоения социальной стипендии (номер константы)

public:

Social() {};

~Social() {};

void header(); // Шапка для таблицы

void edit(); // Метод изменения объекта

void calculateSum(); // Метод пересчета стипендии

static void find(List <Social>& list); // Поиск по различным параметрам

static void sort(List <Social>& list); // Сортировка по различным параметрам

static void read(List <Social>& list); // Чтение из файла

static void write(List <Social>& list); // Запись в файл

friend istream& operator >> (istream& in, Social& student); // Перегрузка ввода

friend ostream& operator << (ostream& out, Social& student); // Перегрузка вывода

static void setAmountSocial();

static int getAmountSocial();

void setCause();

string getCause();

};

Student.h

#pragma once

#include "validation.h"

#include <iostream>

#include <string>

using namespace std;

const string NOT\_ACTIVE = "Не активен";

const string ACTIVE = "Активен";

class Student

{

protected:

int mark = 0; // Средний балл

string surname; // Фамилия

bool isActive = false; // Активность студента

int group = 111111; // Номер группы

public:

Student() {};

~Student() {};

virtual void header() = 0;

void setGroup();

int getGroup();

void setActivity();

string getActivity();

void setSurname();

string getSurname();

void setMark();

int getMark();

};

validation.h

#pragma once

#include <iostream>

using namespace std;

// ПРОВЕРКА НА ВВОД

// Ввод чисел

int inputNumber(int left\_range, int right\_range);

// Проверка на число

bool isNumberNumeric();

// Проверка на ввод числа из корректного промежутка

bool isNumberRangeCorrect(int number, int left\_range, int right\_range);

Account.cpp

#include "Account.h"

bool Account::getRole() {

return isAdmin;

}

void Account::setlog() {

cout << "Введите новый логин пользователя:\n";

cin >> login;

}

void Account::setlog(string a) {

login = a;

}

void Account::setpass(string a) {

// Используется, когда в строке а находится пароль в ОБЫЧНОМ виде

pass = md5(a);

}

void Account::replacePass(string a) {

// Используется, когда в строке а находится пароль в ХЕШИРОВАННОМ виде

pass = a;

}

void Account::changeRole() {

if (isAdmin == false) isAdmin = true;

else isAdmin = false;

}

void Account::header() {

cout << "№\tЛогин\tРоль\n";

for (int i = 0; i < 21; i++) cout << "-";

cout << endl;

}

string Account::inputPass() {

string pass;

char input[30], c;

int counter = 0;

while ((c = \_getch()) != '\r')

{

input[counter++] = c;

\_putch('\*');

}

pass.assign(input, 0, counter);

cout << endl;

return pass;

}

void Account::changePassword(List <Account>& list, Account a) {

system("cls");

string pass\_comparison;

bool first\_attempt = true;

int fail\_counter = 0;

while ((a.pass != md5(pass\_comparison)) && (fail\_counter != 3)) {

if (first\_attempt) cout << "Для изменения пароля введите свой текущий пароль: ";

else cout << "Некорректный ввод пароля. Повторите попытку: ";

pass\_comparison = inputPass();

fail\_counter++;

first\_attempt = false;

}

if (fail\_counter != 3) {

string new\_pass, new\_pass\_copy;

cout << "Введите новый пароль: ";

new\_pass = inputPass();

cout << endl;

bool first\_attempt = true, match = false;

int fails = 0;

while (!match && fails != 3) {

if (first\_attempt) cout << "Повторите новый пароль: ";

else cout << "Некорректный ввод! Повторите новый пароль: ";

new\_pass\_copy = inputPass();

if (new\_pass == new\_pass\_copy) match = true;

fails++;

first\_attempt = false;

}

if (fails != 3) {

a.pass = new\_pass;

for (int i = 0; i < list.Size(); i++) {

if (list[i].login == a.login) {

list[i].pass = md5(new\_pass); break;

}

}

cout << "Ваш пароль изменен успешно!" << endl;

system("pause");

}

else {

cout << "Повторный ввод пароля не совпал трижды. Операция изменения пароля завершена. Пароль не изменен!" << endl;

system("pause");

}

}

else {

cout << "Неправильный пароль был введен трижды. Изменение пароля недоступно!" << endl;

system("pause");

system("cls");

}

}

void Account::read(List <Account>& list) {

ifstream fin(FILE\_WITH\_ACCOUNTS, ios::in); // Открыли файл для чтения

int size;

Account temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.login

>> temp.pass

>> temp.isAdmin;

list.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void Account::write(List <Account>& list) {

ofstream fout(FILE\_WITH\_ACCOUNTS, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].login << ' ' << list[i].pass << ' ' << list[i].isAdmin << '\n';

fout.close();

}

istream& operator >> (istream& in, Account& a) {

cout << "Введите логин нового пользователя:\n";

in >> a.login;

string temp;

cout << "Введите новый пароль:\n";

in >> temp;

a.pass = md5(temp);

return in;

}

ostream& operator << (ostream& out, Account& a) {

cout << a.login << "\t";

if (a.isAdmin) cout << "Admin\n";

else cout << "User\n";

return out;

}

void Account::newApplication(List <Application>& app, List <Account>& acc) {

Application new\_acc;

cout << "Создание нового аккаунта в системе" << endl;

bool flag = false;

while (!flag) {

flag = true;

cout << "Введите логин: ";

cin >> new\_acc.login;

for (int i = 0; i < acc.Size(); i++) {

if (acc[i].getLogin() == new\_acc.login) {

flag = false;

cout << "Аккаунт с таким логином уже зарегистрирован! Пожалуйста, придумайте другой." << endl;

}

}

}

cout << "Введите пароль: ";

cin >> new\_acc.pass;

new\_acc.pass = md5(new\_acc.pass);

app.ListAdd(new\_acc);

system("cls");

cout << "Ваша заявка на регистрацию принята!" << endl << "Система будет доступна после одобрения вашей заявки администратором." << endl;

system("pause");

}

Application.cpp

#include "Application.h"

string Application::getLogin() {

return login;

}

string Application::getHash() {

return pass;

}

void Application::header() {

cout << "№\tЛогин\n";

for (int i = 0; i < 13; i++) cout << "-";

cout << endl;

}

void Application::read(List <Application>& list) {

ifstream fin(FILE\_WITH\_APPLICATIONS, ios::in); // Открыли файл для чтения

int size;

Application temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.login

>> temp.pass;

list.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void Application::write(List <Application>& list) {

ofstream fout(FILE\_WITH\_APPLICATIONS, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].login << ' ' << list[i].pass << '\n';

fout.close();

}

istream& operator >> (istream& in, Application& a) {

cout << "Введите логин нового пользователя:\n";

in >> a.login;

string temp;

cout << "Введите новый пароль:\n";

in >> temp;

a.pass = md5(temp);

return in;

}

ostream& operator << (ostream& out, Application& a) {

cout << a.login << "\n";

return out;

}

main.cpp

#include "Menu.h"

#include <windows.h>

using namespace std;

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Menu::core();

return 0;

}

md5.cpp

#include "md5.h"

/\* system implementation headers \*/

#include <cstdio>

// Constants for MD5Transform routine.

#define S11 7

#define S12 12

#define S13 17

#define S14 22

#define S21 5

#define S22 9

#define S23 14

#define S24 20

#define S31 4

#define S32 11

#define S33 16

#define S34 23

#define S41 6

#define S42 10

#define S43 15

#define S44 21

///////////////////////////////////////////////

// F, G, H and I are basic MD5 functions.

inline MD5::uint4 MD5::F(uint4 x, uint4 y, uint4 z) {

return x & y | ~x & z;

}

inline MD5::uint4 MD5::G(uint4 x, uint4 y, uint4 z) {

return x & z | y & ~z;

}

inline MD5::uint4 MD5::H(uint4 x, uint4 y, uint4 z) {

return x ^ y ^ z;

}

inline MD5::uint4 MD5::I(uint4 x, uint4 y, uint4 z) {

return y ^ (x | ~z);

}

// rotate\_left rotates x left n bits.

inline MD5::uint4 MD5::rotate\_left(uint4 x, int n) {

return (x << n) | (x >> (32 - n));

}

// FF, GG, HH, and II transformations for rounds 1, 2, 3, and 4.

// Rotation is separate from addition to prevent recomputation.

inline void MD5::FF(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + F(b, c, d) + x + ac, s) + b;

}

inline void MD5::GG(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + G(b, c, d) + x + ac, s) + b;

}

inline void MD5::HH(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + H(b, c, d) + x + ac, s) + b;

}

inline void MD5::II(uint4& a, uint4 b, uint4 c, uint4 d, uint4 x, uint4 s, uint4 ac) {

a = rotate\_left(a + I(b, c, d) + x + ac, s) + b;

}

//////////////////////////////////////////////

// default ctor, just initailize

MD5::MD5()

{

init();

}

//////////////////////////////////////////////

// nifty shortcut ctor, compute MD5 for string and finalize it right away

MD5::MD5(const std::string& text)

{

init();

update(text.c\_str(), text.length());

finalize();

}

//////////////////////////////

void MD5::init()

{

finalized = false;

count[0] = 0;

count[1] = 0;

// load magic initialization constants.

state[0] = 0x67452301;

state[1] = 0xefcdab89;

state[2] = 0x98badcfe;

state[3] = 0x10325476;

}

//////////////////////////////

// decodes input (unsigned char) into output (uint4). Assumes len is a multiple of 4.

void MD5::decode(uint4 output[], const uint1 input[], size\_type len)

{

for (unsigned int i = 0, j = 0; j < len; i++, j += 4)

output[i] = ((uint4)input[j]) | (((uint4)input[j + 1]) << 8) |

(((uint4)input[j + 2]) << 16) | (((uint4)input[j + 3]) << 24);

}

//////////////////////////////

// encodes input (uint4) into output (unsigned char). Assumes len is

// a multiple of 4.

void MD5::encode(uint1 output[], const uint4 input[], size\_type len)

{

for (size\_type i = 0, j = 0; j < len; i++, j += 4) {

output[j] = input[i] & 0xff;

output[j + 1] = (input[i] >> 8) & 0xff;

output[j + 2] = (input[i] >> 16) & 0xff;

output[j + 3] = (input[i] >> 24) & 0xff;

}

}

//////////////////////////////

// apply MD5 algo on a block

void MD5::transform(const uint1 block[blocksize])

{

uint4 a = state[0], b = state[1], c = state[2], d = state[3], x[16];

decode(x, block, blocksize);

/\* Round 1 \*/

FF(a, b, c, d, x[0], S11, 0xd76aa478); /\* 1 \*/

FF(d, a, b, c, x[1], S12, 0xe8c7b756); /\* 2 \*/

FF(c, d, a, b, x[2], S13, 0x242070db); /\* 3 \*/

FF(b, c, d, a, x[3], S14, 0xc1bdceee); /\* 4 \*/

FF(a, b, c, d, x[4], S11, 0xf57c0faf); /\* 5 \*/

FF(d, a, b, c, x[5], S12, 0x4787c62a); /\* 6 \*/

FF(c, d, a, b, x[6], S13, 0xa8304613); /\* 7 \*/

FF(b, c, d, a, x[7], S14, 0xfd469501); /\* 8 \*/

FF(a, b, c, d, x[8], S11, 0x698098d8); /\* 9 \*/

FF(d, a, b, c, x[9], S12, 0x8b44f7af); /\* 10 \*/

FF(c, d, a, b, x[10], S13, 0xffff5bb1); /\* 11 \*/

FF(b, c, d, a, x[11], S14, 0x895cd7be); /\* 12 \*/

FF(a, b, c, d, x[12], S11, 0x6b901122); /\* 13 \*/

FF(d, a, b, c, x[13], S12, 0xfd987193); /\* 14 \*/

FF(c, d, a, b, x[14], S13, 0xa679438e); /\* 15 \*/

FF(b, c, d, a, x[15], S14, 0x49b40821); /\* 16 \*/

/\* Round 2 \*/

GG(a, b, c, d, x[1], S21, 0xf61e2562); /\* 17 \*/

GG(d, a, b, c, x[6], S22, 0xc040b340); /\* 18 \*/

GG(c, d, a, b, x[11], S23, 0x265e5a51); /\* 19 \*/

GG(b, c, d, a, x[0], S24, 0xe9b6c7aa); /\* 20 \*/

GG(a, b, c, d, x[5], S21, 0xd62f105d); /\* 21 \*/

GG(d, a, b, c, x[10], S22, 0x2441453); /\* 22 \*/

GG(c, d, a, b, x[15], S23, 0xd8a1e681); /\* 23 \*/

GG(b, c, d, a, x[4], S24, 0xe7d3fbc8); /\* 24 \*/

GG(a, b, c, d, x[9], S21, 0x21e1cde6); /\* 25 \*/

GG(d, a, b, c, x[14], S22, 0xc33707d6); /\* 26 \*/

GG(c, d, a, b, x[3], S23, 0xf4d50d87); /\* 27 \*/

GG(b, c, d, a, x[8], S24, 0x455a14ed); /\* 28 \*/

GG(a, b, c, d, x[13], S21, 0xa9e3e905); /\* 29 \*/

GG(d, a, b, c, x[2], S22, 0xfcefa3f8); /\* 30 \*/

GG(c, d, a, b, x[7], S23, 0x676f02d9); /\* 31 \*/

GG(b, c, d, a, x[12], S24, 0x8d2a4c8a); /\* 32 \*/

/\* Round 3 \*/

HH(a, b, c, d, x[5], S31, 0xfffa3942); /\* 33 \*/

HH(d, a, b, c, x[8], S32, 0x8771f681); /\* 34 \*/

HH(c, d, a, b, x[11], S33, 0x6d9d6122); /\* 35 \*/

HH(b, c, d, a, x[14], S34, 0xfde5380c); /\* 36 \*/

HH(a, b, c, d, x[1], S31, 0xa4beea44); /\* 37 \*/

HH(d, a, b, c, x[4], S32, 0x4bdecfa9); /\* 38 \*/

HH(c, d, a, b, x[7], S33, 0xf6bb4b60); /\* 39 \*/

HH(b, c, d, a, x[10], S34, 0xbebfbc70); /\* 40 \*/

HH(a, b, c, d, x[13], S31, 0x289b7ec6); /\* 41 \*/

HH(d, a, b, c, x[0], S32, 0xeaa127fa); /\* 42 \*/

HH(c, d, a, b, x[3], S33, 0xd4ef3085); /\* 43 \*/

HH(b, c, d, a, x[6], S34, 0x4881d05); /\* 44 \*/

HH(a, b, c, d, x[9], S31, 0xd9d4d039); /\* 45 \*/

HH(d, a, b, c, x[12], S32, 0xe6db99e5); /\* 46 \*/

HH(c, d, a, b, x[15], S33, 0x1fa27cf8); /\* 47 \*/

HH(b, c, d, a, x[2], S34, 0xc4ac5665); /\* 48 \*/

/\* Round 4 \*/

II(a, b, c, d, x[0], S41, 0xf4292244); /\* 49 \*/

II(d, a, b, c, x[7], S42, 0x432aff97); /\* 50 \*/

II(c, d, a, b, x[14], S43, 0xab9423a7); /\* 51 \*/

II(b, c, d, a, x[5], S44, 0xfc93a039); /\* 52 \*/

II(a, b, c, d, x[12], S41, 0x655b59c3); /\* 53 \*/

II(d, a, b, c, x[3], S42, 0x8f0ccc92); /\* 54 \*/

II(c, d, a, b, x[10], S43, 0xffeff47d); /\* 55 \*/

II(b, c, d, a, x[1], S44, 0x85845dd1); /\* 56 \*/

II(a, b, c, d, x[8], S41, 0x6fa87e4f); /\* 57 \*/

II(d, a, b, c, x[15], S42, 0xfe2ce6e0); /\* 58 \*/

II(c, d, a, b, x[6], S43, 0xa3014314); /\* 59 \*/

II(b, c, d, a, x[13], S44, 0x4e0811a1); /\* 60 \*/

II(a, b, c, d, x[4], S41, 0xf7537e82); /\* 61 \*/

II(d, a, b, c, x[11], S42, 0xbd3af235); /\* 62 \*/

II(c, d, a, b, x[2], S43, 0x2ad7d2bb); /\* 63 \*/

II(b, c, d, a, x[9], S44, 0xeb86d391); /\* 64 \*/

state[0] += a;

state[1] += b;

state[2] += c;

state[3] += d;

// Zeroize sensitive information.

memset(x, 0, sizeof x);

}

//////////////////////////////

// MD5 block update operation. Continues an MD5 message-digest

// operation, processing another message block

void MD5::update(const unsigned char input[], size\_type length)

{

// compute number of bytes mod 64

size\_type index = count[0] / 8 % blocksize;

// Update number of bits

if ((count[0] += (length << 3)) < (length << 3))

count[1]++;

count[1] += (length >> 29);

// number of bytes we need to fill in buffer

size\_type firstpart = 64 - index;

size\_type i;

// transform as many times as possible.

if (length >= firstpart)

{

// fill buffer first, transform

memcpy(&buffer[index], input, firstpart);

transform(buffer);

// transform chunks of blocksize (64 bytes)

for (i = firstpart; i + blocksize <= length; i += blocksize)

transform(&input[i]);

index = 0;

}

else

i = 0;

// buffer remaining input

memcpy(&buffer[index], &input[i], length - i);

}

//////////////////////////////

// for convenience provide a verson with signed char

void MD5::update(const char input[], size\_type length)

{

update((const unsigned char\*)input, length);

}

//////////////////////////////

// MD5 finalization. Ends an MD5 message-digest operation, writing the

// the message digest and zeroizing the context.

MD5& MD5::finalize()

{

static unsigned char padding[64] = {

0x80, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0

};

if (!finalized) {

// Save number of bits

unsigned char bits[8];

encode(bits, count, 8);

// pad out to 56 mod 64.

size\_type index = count[0] / 8 % 64;

size\_type padLen = (index < 56) ? (56 - index) : (120 - index);

update(padding, padLen);

// Append length (before padding)

update(bits, 8);

// Store state in digest

encode(digest, state, 16);

// Zeroize sensitive information.

memset(buffer, 0, sizeof buffer);

memset(count, 0, sizeof count);

finalized = true;

}

return \*this;

}

//////////////////////////////

// return hex representation of digest as string

std::string MD5::hexdigest() const

{

if (!finalized)

return "";

char buf[33];

for (int i = 0; i < 16; i++)

sprintf(buf + i \* 2, "%02x", digest[i]);

buf[32] = 0;

return std::string(buf);

}

//////////////////////////////

std::ostream& operator<<(std::ostream& out, MD5 md5)

{

return out << md5.hexdigest();

}

//////////////////////////////

std::string md5(const std::string str)

{

MD5 md5 = MD5(str);

return md5.hexdigest();

}

Menu.cpp

#include "Menu.h"

void Menu::core() {

List<Application> app;

List<Account> acc;

Application::read(app);

Account::read(acc);

Account current\_acc;

bool core\_flag = true;

while (core\_flag) {

system("cls");

system("color 8E");

cout << "1) Войти" << endl << "2) Зарегистрировать новый аккаунт"

<< endl << "0) Выйти из системы" << endl;

int user\_choice = inputNumber(0, 2);

system("cls");

switch (user\_choice) {

case 0: core\_flag = false; break;

case 1:

Menu::login(app, acc, current\_acc);

break;

case 2:

Account::newApplication(app,acc);

break;

}

}

Menu::save(app, acc);

}

void Menu::AdminMenuOfScholars() {

bool flag = true;

int choice;

while (flag) {

system("cls");

cout << "В нижепредставленных разделах представлены списки студентов, получающих разные виды стипендий.\n\nВыберите раздел для работы:\n";

cout << "1) Учебная стипендия\n2) Президентская стипендия\n3) Социальная стипендия\n4) Именная стипендия\n0) Вернуться назад\n";

choice = inputNumber(0, 4);

switch (choice)

{

case 0:

flag = false;

break;

case 1:

Interface<Schoolarship>::AdminUI();

break;

case 2:

Interface<President>::AdminUI();

break;

case 3:

Interface<Social>::AdminUI();

break;

case 4:

Interface<Personal>::AdminUI();

break;

}

}

}

void Menu::UserMenuOfScholars() {

bool flag = true;

int choice;

while (flag) {

system("cls");

cout << "В нижепредставленных разделах представлены списки студентов, получающих разные виды стипендий.\n\nВыберите раздел для работы:\n";

cout << "1) Учебная стипендия\n2) Президентская стипендия\n3) Социальная стипендия\n4) Именная стипендия\n0) Вернуться назад\n";

choice = inputNumber(0, 4);

switch (choice)

{

case 0:

flag = false;

break;

case 1:

Interface<Schoolarship>::UserUI();

break;

case 2:

Interface<President>::UserUI();

break;

case 3:

Interface<Social>::UserUI();

break;

case 4:

Interface<Personal>::UserUI();

break;

}

}

}

void Menu::SchoolarshipAmountManagement() {

int choice;

bool flag = true;

while (flag) {

system("cls");

cout << "В данном меню можно изменить размеры базовых стипендий!\n\n";

cout << "1) Учебная стипендия:\t\t" << Schoolarship::getBase()

<< endl;

cout << "2) Именная стипендия:\t\t" << Personal::getAmountPersonal() << endl;

cout << "3) Президентская стипендия:\t" << President::getAmountPresident() << endl;

cout << "4) Социальная стипендия:\t" << Social::getAmountSocial()

<< endl << endl;

cout << "Выберите номер стипендии для изменения ( 0 - Вернуться назад ) \n";

choice = inputNumber(0, 4);

switch (choice)

{

case 0:

flag = false;

break;

case 1:

Schoolarship::setBase();

break;

case 2:

Personal::setAmountPersonal();

break;

case 3:

President::setAmountPresident();

break;

case 4:

Social::setAmountSocial();

break;

}

}

}

void Menu::showAccounts(List <Account>& acc, Account& current\_acc) {

bool flag = true;

int choice, ch\_choice, del\_choice, confirm;

Account temp;

while (flag) {

system("cls");

Account::header();

for (int i = 0; i < acc.Size(); i++) {

cout << i+1<< ")\t" << acc[i];

}

cout << "\n\nВыберите действие:\n1) Добавить аккаунт\n2) Удалить аккаунт\n3) Изменить роль\n";

cout << "4) Отсортировать по алфавиту\n5) Отсортировать по роли\n0) Вернуться назад\n";

choice = inputNumber(0, 5);

switch (choice) {

case 0:

flag = false;

break;

case 1:

cin >> temp;

acc.ListAdd(temp);

break;

case 2:

if (acc.isListEmpty())

{

cout << "Список пуст" << endl;

system("pause");

}

else

{

cout << "Введите номер аккаунта для удаления:\n";

del\_choice = inputNumber(1, acc.Size());

if (current\_acc.getLogin() == acc[del\_choice - 1].getLogin()) {

cout << "Вы не можете удалить свой аккаунт!" << endl;

system("pause");

}

else {

cout << "Вы действительно хотите удалить аккаунт с логином "

<< acc[del\_choice - 1].getLogin() << " ?\n1) Да\n2) Нет\n";

confirm = inputNumber(1, 2);

if (confirm == 1) acc.del(del\_choice - 1);

}

}

break;

case 3:

cout << "Введите номер аккаунта:\n";

ch\_choice = inputNumber(1, acc.Size());

if (current\_acc.getLogin() == acc[ch\_choice - 1].getLogin()) {

сout << "Вы не можете изменить роль своего аккаунта!" << endl;

system("pause");

}

else {

acc[ch\_choice - 1].changeRole();

}

break;

case 4:

acc.sort(

[](Account ob1, Account ob2) -> bool

{

return ob1.getLogin() > ob2.getLogin();

});

break;

case 5:

acc.sort(

[](Account ob1, Account ob2) -> bool

{

return ob1.getRole() > ob2.getRole();

});

break;

}

}

}

void Menu::showNewApplications(List <Application>& app, List <Account>& acc, Account& current\_acc) {

bool flag = true;

int choice, ch\_choice, del\_choice, confirm;

Account temp;

while (flag) {

system("cls");

Application::header();

for (int i = 0; i < app.Size(); i++) {

cout << i + 1 << ")\t" << app[i];

}

cout << "\n\nВыберите действие:\n1) Одобрить заявку\n2) Удалить заявку\n3) Отсортировать по алфавиту\n0) Вернуться назад\n";

choice = inputNumber(0, 3);

switch (choice) {

case 0:

flag = false;

break;

case 1:

cout << "Введите номер заявки для одобрения:\n";

ch\_choice = inputNumber(1, acc.Size());

temp.setlog(app[ch\_choice-1].getLogin());

temp.replacePass(app[ch\_choice - 1].getHash());

acc.ListAdd(temp);

app.del(ch\_choice - 1);

break;

case 2:

if (app.isListEmpty())

{

cout << "Список пуст" << endl;

system("pause");

}

else

{

cout << "Введите номер аккаунта для удаления:\n";

del\_choice = inputNumber(1, app.Size());

system("cls");

cout << "Вы действительно хотите удалить аккаунт с логином " << app[del\_choice - 1].getLogin() << " ?\n1) Да\n2) Нет\n";

confirm = inputNumber(1, 2);

if (confirm == 1) app.del(del\_choice - 1);

}

break;

case 3:

app.sort(

[](Application ob1, Application ob2) -> bool

{

return ob1.getLogin() > ob2.getLogin();

});

break;

}

}

}

void Menu::AccountManagement(List <Application>& app, List <Account>& acc, Account& current\_acc) {

bool acc\_menu = true;

int choice;

while (acc\_menu) {

system("cls");

system("color 81");

cout << "Модуль управления аккаунтами системы\n\nВыберите действие:\n1) Показать все аккаунты\n2) Показать новые заявки на вход в систему\n0) Назад\n";

choice = inputNumber(0, 2);

system("cls");

switch (choice) {

case 0: acc\_menu = false; break;

case 1: Menu::showAccounts(acc, current\_acc); break;

case 2: Menu::showNewApplications(app, acc,current\_acc); break;

}

}

}

void Menu::AdminMenu(List <Application>& app, List <Account>& acc, Account& current\_acc) {

int choice;

bool flag = true;

while (flag) {

system("cls");

system("color 84");

cout << "Добро пожаловать в систему расчёта и начисления стипендий!\n"

<< "Вы находитесь в режиме администратора!\n\n"

<< "1) Списки стипендиантов\n2) Меню управления базовыми размерами стипендий\n3) Модуль управления аккаунтами\n4) Изменить пароль\n0) Выйти из аккаунта" << endl;

choice = inputNumber(0, 4);

system("cls");

switch (choice)

{

case 0:

flag = false;

break;

case 1:

Menu::AdminMenuOfScholars();

break;

case 2:

Menu::SchoolarshipAmountManagement();

break;

case 3:

Menu::AccountManagement(app,acc, current\_acc);

break;

case 4:

current\_acc.changePassword(acc,current\_acc);

break;

}

}

}

void Menu::UserMenu(List <Account>& acc, Account& current\_acc) {

int choice;

bool flag = true;

while (flag) {

system("cls");

system("color 8A");

cout << "Добро пожаловать в систему расчёта и начисления стипендий!\n"

<< "Вы находитесь в режиме пользователя!\n\n"

<< "1) Списки стипендиантов\n2) Изменить пароль\n0) Выйти из аккаунта" << endl;

choice = inputNumber(0, 2);

system("cls");

switch (choice)

{

case 0:

flag = false;

break;

case 1:

Menu::UserMenuOfScholars();

break;

case 2:

current\_acc.changePassword(acc, current\_acc);

break;

}

}

}

void Menu::login(List <Application>& app, List <Account>& acc, Account& current\_acc) {

string written\_login, written\_password;

bool similar = false;

cout << "\*\*\*Авторизируйтесь в системе\*\*\*" << endl;

while (!similar) {

cout << "Введите логин: "; cin >> written\_login;

cout << "Введите пароль: ";

written\_password = Account::inputPass();

cout << endl;

for (int i = 0; i < acc.Size(); i++) {

if (acc[i].login == written\_login && acc[i].pass == md5(written\_password)) {

current\_acc = acc[i];

similar = true;

}

}

if (!similar) cout << "Данные введены некорректно! Повторите попытку. " << endl;

}

system("cls");

if (current\_acc.isAdmin) Menu::AdminMenu(app, acc, current\_acc);

else Menu::UserMenu(acc,current\_acc);

}

void Menu::save(List <Application>& app, List <Account>& acc) {

Application::write(app);

Account::write(acc);

}

Personal.cpp

#include "Personal.h"

int Personal::amount\_pers = 350;

void Personal::calculateSum() {

calculateKoef();

sum = base \* koef;

if (isActive) sum \*= 1.25;

sum += amount\_pers;

}

void Personal::setAmountPersonal() {

cout << "Введите размер Именной стипендии:\n";

amount\_pers = inputNumber(50, 1500);

}

int Personal::getAmountPersonal() {

return amount\_pers;

}

void Personal::setAuthor() {

cout << "Какая именная стипендия у данного студента?:\n";

cin >> author;

}

string Personal::getAuthor() {

return author;

}

void Personal::header() {

cout << "№\tФамилия\t\tГруппа\tСр.Балл\tАктивность\tСтипендия\tНазвание\n";

for (int i = 0; i < 80; i++) cout << "-";

cout << endl;

}

void Personal::find(List <Personal>& list) {

cout << "Выберите поиск:\n1) По фамилии\n2) По номеру группы\n3) По среднему баллу\n";

int choice = inputNumber(1, 3);

string temp\_surn;

int temp\_int;

switch (choice)

{

case 1:

cout << "Введите фамилию для поиска:\n";

cin >> temp\_surn;

list.find<string>(

[](Personal ob, string temp\_surn) -> bool

{

return ob.getSurname() == temp\_surn;

}, temp\_surn);

system("pause");

break;

case 2:

cout << "Введите номер группы для поиска:\n";

temp\_int = inputNumber(111111, 999999);

list.find<int>(

[](Personal ob, int temp\_int) -> bool

{

return ob.getGroup() == temp\_int;

}, temp\_int);

system("pause");

break;

case 3:

cout << "Введите средний балл для поиска:\n";

temp\_int = inputNumber(0, 10);

list.find<int>(

[](Personal ob, int temp\_int) -> bool

{

return ob.getMark() == temp\_int;

}, temp\_int);

system("pause");

break;

}

}

void Personal::sort(List <Personal>& list) {

cout << "Выберите сортировку:\n1) Отсортировать по алфавиту\n2) Отсортировать по номеру группы\n3) Отсортировать по среднему баллу\n";

cout << "4) Отсортировать по размеру стипендии\n5) Отсортировать по активности\n";

int choice = inputNumber(1, 5);

switch (choice)

{

case 1:

list.sort(

[](Personal ob1, Personal ob2) -> bool

{

return ob1.getSurname() > ob2.getSurname();

});

break;

case 2:

list.sort(

[](Personal ob1, Personal ob2) -> bool

{

return ob1.getGroup() > ob2.getGroup();

});

break;

case 3:

list.sort(

[](Personal ob1, Personal ob2) -> bool

{

return ob1.getMark() < ob2.getMark();

});

break;

case 4:

list.sort(

[](Personal ob1, Personal ob2) -> bool

{

return ob1.getSum() < ob2.getSum();

});

break;

case 5:

list.sort(

[](Personal ob1, Personal ob2) -> bool

{

return ob1.getActivity() > ob2.getActivity();

});

break;

}

}

void Personal::edit() {

bool flag = true;

int choice;

while (flag) {

system("cls");

this->header();

cout << "x)\t" << \*this << "\n\n";

cout << "Выберите поле для изменения:\n1) Фамилия\n2) Номер группы\n3) Активность\n4) Средний балл\n5) Название стипендии\n\n0) Завершить редактирование\n";

choice = inputNumber(0, 5);

switch (choice) {

case 0:

flag = false;

break;

case 1:

this->setSurname();

break;

case 2:

this->setGroup();

break;

case 3:

this->setActivity();

this->calculateSum();

break;

case 4:

this->setMark();

this->calculateKoef();

this->calculateSum();

break;

case 5:

this->setAuthor();

break;

}

}

}

istream& operator >> (istream& in, Personal& student) {

system("cls");

in >> dynamic\_cast<Schoolarship&>(student);

cout << "Какую именную стипендию получает студент? (Например: им.Франциска Скорины) \n";

in >> student.author;

student.calculateSum();

return in;

}

ostream& operator << (ostream& out, Personal& student) {

cout << dynamic\_cast<Schoolarship&>(student);

cout << "\t\t" << student.author;

return out;

}

void Personal::read(List <Personal>& v) {

ifstream fin(FILE\_WITH\_PERSONAL, ios::in); // Открыли файл для чтения

int size;

Personal temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.surname

>> temp.group

>> temp.isActive

>> temp.mark

>> temp.sum

>> temp.author;

v.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void Personal::write(List <Personal>& list) {

ofstream fout(FILE\_WITH\_PERSONAL, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].surname << ' ' << list[i].group << ' ' << list[i].isActive << ' ' << list[i].mark << ' ' << list[i].sum << ' ' << list[i].author << '\n';

fout.close();

}

President.cpp

#include "President.h"

int President::amount\_p = 350;

void President::calculateSum() {

calculateKoef();

sum = base \* koef;

if (isActive) sum \*= 1.25;

sum += amount\_p;

}

void President::setAmountPresident(){

cout << "Введите размер Президентской стипендии:\n";

amount\_p = inputNumber(50, 1500);

}

int President::getAmountPresident() {

return amount\_p;

}

void President::setYear() {

cout << "Введите год присвоения студенту Президентской стипендии:\n";

year = inputNumber(2000, 2021);

}

int President::getYear() {

return year;

}

void President::header() {

cout << "№\tФамилия\t\tГруппа\tСр.Балл\tАктивность\tСтипендия\tГод присвоения\n";

for (int i = 0; i < 86; i++) cout << "-";

cout << endl;

}

void President::find(List <President>& list) {

cout << "Выберите поиск:\n1) По фамилии\n2) По номеру группы\n3) По среднему баллу\n4) По году присвоения\n";

int choice = inputNumber(1, 4);

string temp\_surn;

int temp\_int;

switch (choice)

{

case 1:

cout << "Введите фамилию для поиска:\n";

cin >> temp\_surn;

list.find<string>(

[](President ob, string temp\_surn) -> bool

{

return ob.getSurname() == temp\_surn;

}, temp\_surn);

system("pause");

break;

case 2:

cout << "Введите номер группы для поиска:\n";

temp\_int = inputNumber(111111, 999999);

list.find<int>(

[](President ob, int temp\_int) -> bool

{

return ob.getGroup() == temp\_int;

}, temp\_int);

system("pause");

break;

case 3:

cout << "Введите средний балл для поиска:\n";

temp\_int = inputNumber(0, 10);

list.find<int>(

[](President ob, int temp\_int) -> bool

{

return ob.getMark() == temp\_int;

}, temp\_int);

system("pause");

break;

case 4:

cout << "Введите год присвоения для поиска:\n";

temp\_int = inputNumber(2010, 2021);

list.find<int>(

[](President ob, int temp\_int) -> bool

{

return ob.getYear() == temp\_int;

}, temp\_int);

system("pause");

break;

}

}

void President::sort(List <President>& list) {

cout << "Выберите сортировку:\n1) Отсортировать по алфавиту\n2) Отсортировать по номеру группы\n3) Отсортировать по среднему баллу\n";

cout << "4) Отсортировать по размеру стипендии\n5) Отсортировать по активности\n6) Отсортировать по году присвоения\n";

int choice = inputNumber(1, 6);

switch (choice)

{

case 1:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getSurname() > ob2.getSurname();

});

break;

case 2:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getGroup() > ob2.getGroup();

});

break;

case 3:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getMark() < ob2.getMark();

});

break;

case 4:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getSum() < ob2.getSum();

});

break;

case 5:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getActivity() > ob2.getActivity();

});

break;

case 6:

list.sort(

[](President ob1, President ob2) -> bool

{

return ob1.getYear() > ob2.getYear();

});

break;

}

}

void President::edit() {

bool flag = true;

int choice;

while (flag) {

system("cls");

this->header();

cout << "x)\t" << \*this << "\n\n";

cout << "Выберите поле для изменения:\n1) Фамилия\n2) Номер группы\n3) Активность\n4) Средний балл\n5) Год присвоения\n\n0) Завершить редактирование\n";

choice = inputNumber(0, 5);

switch (choice) {

case 0:

flag = false;

break;

case 1:

this->setSurname();

break;

case 2:

this->setGroup();

break;

case 3:

this->setActivity();

this->calculateSum();

break;

case 4:

this->setMark();

this->calculateKoef();

this->calculateSum();

break;

case 5:

this->setYear();

break;

}

}

}

istream& operator >> (istream& in, President& student) {

system("cls");

in >> dynamic\_cast<Schoolarship&>(student);

cout << "В каком году студента наградили Президентской стипендией?\n";

in >> student.year;

student.calculateSum();

return in;

}

ostream& operator << (ostream& out, President& student) {

cout << dynamic\_cast<Schoolarship&>(student);

cout << "\t\t" << student.year << "г.";

return out;

}

void President::read(List <President>& v) {

ifstream fin(FILE\_WITH\_PRESIDENT, ios::in); // Открыли файл для чтения

int size;

President temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.surname

>> temp.group

>> temp.isActive

>> temp.mark

>> temp.sum

>> temp.year;

v.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void President::write(List <President>& list) {

ofstream fout(FILE\_WITH\_PRESIDENT, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].surname << ' ' << list[i].group << ' ' << list[i].isActive << ' ' << list[i].mark << ' ' << list[i].sum << ' ' << list[i].year << '\n';

fout.close();

}

Schoolarship.cpp

#include "Schoolarship.h"

int Schoolarship::base = 90;

void Schoolarship::setBase() {

cout << "Введите размер базовой стипендии (в рублях):\n";

base = inputNumber(50, 500);

}

int Schoolarship::getBase() {

return base;

}

void Schoolarship::calculateSum() {

calculateKoef();

sum = base \* koef\*1.00;

if (isActive) sum \*= 1.25;

}

void Schoolarship::calculateKoef() {

if (mark < 5) {

koef = 0;

return;

}

switch (mark)

{

case 5: koef = 1; break;

case 6: case 7: koef = 1.2; break;

case 8: koef = 1.4; break;

case 9: case 10:koef = 1.6; break;

}

}

double Schoolarship::getKoef() {

return koef;

}

double Schoolarship::getSum() {

return sum;

}

void Schoolarship::find(List <Schoolarship>& list) {

cout << "Выберите поиск:\n1) По фамилии\n2) По номеру группы\n3) По среднему баллу\n";

int choice = inputNumber(1, 3);

string temp\_surn;

int temp\_int;

switch (choice)

{

case 1:

cout << "Введите фамилию для поиска:\n";

cin >> temp\_surn;

list.find<string>(

[](Schoolarship ob, string temp\_surn) -> bool

{

return ob.getSurname() == temp\_surn;

}, temp\_surn);

system("pause");

break;

case 2:

cout << "Введите номер группы для поиска:\n";

temp\_int = inputNumber(111111, 999999);

list.find<int>(

[](Schoolarship ob, int temp\_int) -> bool

{

return ob.getGroup() == temp\_int;

}, temp\_int);

system("pause");

break;

case 3:

cout << "Введите средний балл для поиска:\n";

temp\_int = inputNumber(0, 10);

list.find<int>(

[](Schoolarship ob, int temp\_int) -> bool

{

return ob.getMark() == temp\_int;

}, temp\_int);

system("pause");

break;

}

}

void Schoolarship::sort(List <Schoolarship>& list) {

cout << "Выберите сортировку:\n1) Отсортировать по алфавиту\n2) Отсортировать по номеру группы\n3) Отсортировать по среднему баллу\n";

cout <<"4) Отсортировать по размеру стипендии\n5) Отсортировать по активности\n";

int choice = inputNumber(1, 5);

switch (choice)

{

case 1:

list.sort(

[](Schoolarship ob1, Schoolarship ob2) -> bool

{

return ob1.getSurname() > ob2.getSurname();

});

break;

case 2:

list.sort(

[](Schoolarship ob1, Schoolarship ob2) -> bool

{

return ob1.getGroup() > ob2.getGroup();

});

break;

case 3:

list.sort(

[](Schoolarship ob1, Schoolarship ob2) -> bool

{

return ob1.getMark() < ob2.getMark();

});

break;

case 4:

list.sort(

[](Schoolarship ob1, Schoolarship ob2) -> bool

{

return ob1.getSum() < ob2.getSum();

});

break;

case 5:

list.sort(

[](Schoolarship ob1, Schoolarship ob2) -> bool

{

return ob1.getActivity() > ob2.getActivity();

});

break;

}

}

void Schoolarship::edit() {

bool flag = true;

int choice;

while (flag) {

system("cls");

this->header();

cout << "x)\t"<< \*this << "\n\n";

cout << "Выберите поле для изменения:\n1) Фамилия\n2) Номер группы\n3) Активность\n4) Средний балл\n\n0) Завершить редактирование\n";

choice = inputNumber(0, 4);

switch (choice) {

case 0:

flag = false;

break;

case 1:

this->setSurname();

break;

case 2:

this->setGroup();

break;

case 3:

this->setActivity();

this->calculateSum();

break;

case 4:

this->setMark();

this->calculateKoef();

this->calculateSum();

break;

}

}

}

void Schoolarship::header() {

cout << "№\tФамилия\t\tГруппа\tСр.Балл\tАктивность\tСтипендия\n";

for (int i = 0; i < 65; i++) cout << "-";

cout << endl;

}

istream& operator >> (istream& in, Schoolarship& student) {

system("cls");

cout << "\*\*\*Создание нового студента\*\*\*\nВведите фамилию студента: \n";

in >> student.surname;

cout << "Введите номер группы студента: \n";

in >> student.group;

cout << "Введите средний балл студента:\n";

in >> student.mark;

cout << "Если студент активен, нажмите 1\nЕсли нет, нажмите 0 \n";

in >> student.isActive;

student.calculateSum();

return in;

}

ostream& operator << (ostream& out, Schoolarship& student) {

cout << student.surname << "\t\t" << student.group << "\t" << student.mark << "\t";

if (student.isActive) cout << "Активен \t";

else cout << "Неактивен\t";

cout << student.sum << "руб.";

return out;

}

void Schoolarship::read(List <Schoolarship>& list) {

ifstream fin(FILE\_WITH\_DEFAULT, ios::in); // Открыли файл для чтения

int size;

Schoolarship temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.surname

>> temp.group

>> temp.isActive

>> temp.mark

>> temp.sum;

list.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void Schoolarship::write(List <Schoolarship>& list) {

ofstream fout(FILE\_WITH\_DEFAULT, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].surname << ' ' << list[i].group << ' ' << list[i].isActive << ' ' << list[i].mark << ' ' << list[i].sum << ' ' << '\n';

fout.close();

}

Social.cpp

#include "Social.h"

int Social::amount\_s = 70;

void Social::calculateSum() {

calculateKoef();

sum = base \* koef;

if (isActive) sum \*= 1.25;

sum += amount\_s;

}

void Social::setAmountSocial() {

cout << "Введите размер Социальной стипендии:\n";

amount\_s = inputNumber(50, 250);

}

int Social::getAmountSocial() {

return amount\_s;

}

void Social::setCause() {

cout << "По какой причине студент получает социальную стипендию?\n1) "<< cause1<<"\n2) "<<cause2 << "\n3) " << cause3 << "\n4) " << cause4 << "\n";

cause = inputNumber(1, 4);

}

string Social::getCause() {

switch (cause)

{

case 1: return cause1; break;

case 2: return cause2; break;

case 3: return cause3; break;

case 4: return cause4; break;

}

}

void Social::header() {

cout << "№\tФамилия\t\tГруппа\tСр.Балл\tАктивность\tСтипендия\tПричина присвоения\n";

for (int i = 0; i < 94; i++) cout << "-";

cout << endl;

}

void Social::find(List <Social>& list) {

cout << "Выберите поиск:\n1) По фамилии\n2) По номеру группы\n3) По среднему баллу\n";

int choice = inputNumber(1, 3);

string temp\_surn;

int temp\_int;

switch (choice)

{

case 1:

cout << "Введите фамилию для поиска:\n";

cin >> temp\_surn;

list.find<string>(

[](Social ob, string temp\_surn) -> bool

{

return ob.getSurname() == temp\_surn;

}, temp\_surn);

system("pause");

break;

case 2:

cout << "Введите номер группы для поиска:\n";

temp\_int = inputNumber(111111, 999999);

list.find<int>(

[](Social ob, int temp\_int) -> bool

{

return ob.getGroup() == temp\_int;

}, temp\_int);

system("pause");

break;

case 3:

cout << "Введите средний балл для поиска:\n";

temp\_int = inputNumber(0, 10);

list.find<int>(

[](Social ob, int temp\_int) -> bool

{

return ob.getMark() == temp\_int;

}, temp\_int);

system("pause");

break;

}

}

void Social::sort(List <Social>& list) {

cout << "Выберите сортировку:\n1) Отсортировать по алфавиту\n2) Отсортировать по номеру группы\n3) Отсортировать по среднему баллу\n";

cout << "4) Отсортировать по размеру стипендии\n5) Отсортировать по активности\n";

int choice = inputNumber(1, 5);

switch (choice)

{

case 1:

list.sort(

[](Social ob1, Social ob2) -> bool

{

return ob1.getSurname() > ob2.getSurname();

});

break;

case 2:

list.sort(

[](Social ob1, Social ob2) -> bool

{

return ob1.getGroup() > ob2.getGroup();

});

break;

case 3:

list.sort(

[](Social ob1, Social ob2) -> bool

{

return ob1.getMark() < ob2.getMark();

});

break;

case 4:

list.sort(

[](Social ob1, Social ob2) -> bool

{

return ob1.getSum() < ob2.getSum();

});

break;

case 5:

list.sort(

[](Social ob1, Social ob2) -> bool

{

return ob1.getActivity() > ob2.getActivity();

});

break;

}

}

void Social::edit() {

bool flag = true;

int choice;

while (flag) {

system("cls");

this->header();

cout << "x)\t" << \*this << "\n\n";

cout << "Выберите поле для изменения:\n1) Фамилия\n2) Номер группы\n3) Активность\n4) Средний балл\n5) Причина присвоения\n\n0) Завершить редактирование\n";

choice = inputNumber(0, 5);

switch (choice) {

case 0:

flag = false;

break;

case 1:

this->setSurname();

break;

case 2:

this->setGroup();

break;

case 3:

this->setActivity();

this->calculateSum();

break;

case 4:

this->setMark();

this->calculateKoef();

this->calculateSum();

break;

case 5:

this->setCause();

break;

}

}

}

istream& operator >> (istream& in, Social& student) {

system("cls");

in >> dynamic\_cast<Schoolarship&>(student);

student.setCause();

student.calculateSum();

return in;

}

ostream& operator << (ostream& out, Social& student) {

cout << dynamic\_cast<Schoolarship&>(student);

cout << "\t\t" << student.getCause();

return out;

}

void Social::read(List <Social>& v) {

ifstream fin(FILE\_WITH\_SOCIAL, ios::in); // Открыли файл для чтения

int size;

Social temp;

if (fin.is\_open()) {

fin >> size;

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

fin >> temp.surname

>> temp.group

>> temp.isActive

>> temp.mark

>> temp.sum

>> temp.cause; //may trouble

v.ListAdd(temp);

}

}

fin.close(); //Закрыли файл

}

void Social::write(List <Social>& list) {

ofstream fout(FILE\_WITH\_SOCIAL, ios::out);

fout << list.Size() << '\n';

for (int i = 0; i < list.Size(); i++)

fout << list[i].surname << ' ' << list[i].group << ' ' << list[i].isActive << ' ' << list[i].mark << ' ' << list[i].sum << ' ' << list[i].cause << '\n';

fout.close();

}

Student.cpp

#include "Student.h"

void Student::setGroup() {

cout << "Введите номер учебной группы:\n";

group = inputNumber(111111, 999999);

}

int Student::getGroup() {

return group;

}

void Student::setMark() {

cout << "Введите средний балл студента:\n";

mark = inputNumber(0, 10);

}

int Student::getMark() {

return mark;

}

void Student::setActivity() {

cout << "Студент принимает активное участие в жизни университета?\n1) Да\n2) Нет\n";

int temp = inputNumber(1, 2);

if (temp == 1) isActive = true;

else isActive = false;

}

string Student::getActivity() {

if (!isActive) return NOT\_ACTIVE;

return ACTIVE;

}

void Student::setSurname() {

cout << "Введите фамилию студента:\n";

cin >> surname;

}

string Student::getSurname() {

return surname;

}

validation.cpp

#include "validation.h"

bool isNumberNumeric() {

if (cin.get() == '\n') return true;

else

{

cin.clear();

cin.ignore(256, '\n');

return false;

}

}

bool isNumberRangeCorrect(int number, int left\_range, int right\_range) {

if ((number >= left\_range) && (number <= right\_range))return true;

else return false;

}

int inputNumber(int left\_range, int right\_range) {

int number;

while (true) {

cin >> number;

if (isNumberNumeric() && isNumberRangeCorrect(number, left\_range, right\_range)) {

return number;

}

else {

cout << "Некорректный ввод!" << endl;

cout << "Введите корректные данные: ";

}

}

}

# ПРИЛОЖЕНИЕ Г

(обязательное)

**Ведомость документов**