СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 3](#_Toc134120754)

[АНОТАЦИЯ 4](#_Toc134120755)

[ВВЕДЕНИЕ 5](#_Toc134120756)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc134120757)

[1.1 Структурное программирование 7](#_Toc134120758)

[1.2 Процедурное программирование 7](#_Toc134120759)

[1.3 Модульное программирование 8](#_Toc134120760)

[1.4 Ссылки 9](#_Toc134120761)

[1.5 Работа с памятью. Указатели 9](#_Toc134120762)

[1.6 Работа с файлами 10](#_Toc134120763)

[1.7 Динамические структуры данных 11](#_Toc134120764)

[1.8 Линейный список 11](#_Toc134120765)

[1.9 Функции 11](#_Toc134120766)

[1.10 Объектно-ориентированное программирование 12](#_Toc134120767)

[1.11 Конструкторы и деструкторы 13](#_Toc134120768)

[1.12 Классы и дружественные функции 14](#_Toc134120769)

[1.12.1 Описание классов и их методов 15](#_Toc134120770)

[1.13 Методы шифрования и дешифрования 15](#_Toc134120771)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 19](#_Toc134120772)

[2.1 Структура программы 19](#_Toc134120773)

[2.2 Алгоритм решения задачи 23](#_Toc134120774)

[2.3 Программная реализация задания 23](#_Toc134120775)

[2.3.1 Пример выполнения задания 23](#_Toc134120776)

[2.3.2 Отображение списка студентов 23](#_Toc134120777)

[2.3.3 Изменение данных о студенте 23](#_Toc134120778)

[2.3.3 Добавление информации о новом студенте 23](#_Toc134120779)

[2.3.4 Удаление данных о студенте 23](#_Toc134120780)

[2.3.5 Шифрование данных \* 23](#_Toc134120781)

[2.4 Руководство пользователя 23](#_Toc134120782)

[2.8 Системные требования 24](#_Toc134120783)

[ЗАКЛЮЧЕНИЕ 25](#_Toc134120784)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 26](#_Toc134120785)

[ПРИЛОЖЕНИЕ А Исходный код программы 27](#_Toc134120786)

АНОТАЦИЯ

Тема курсовой работы: Разработка программы информационного поиска студентов по заданным критериям с возможностью шифрования данных

Выполнил: Заргаров Александр Денисович, БИСО-03-22

Работа состоит из введения, двух глав, заключения и списка используемых источников.

Во введении рассмотрены цели, задачи работы и ее актуальность.

В первой главе, относящейся к теоретической части, рассмотрены и приведены основные термины, понятия и определения из языка программирования С++.

Во второй главе, относящейся к практической части, приведены листинги кода, представлена программа, выполняющая поставленную задачу, а приведены блок схемы и разобраны алгоритмы ее работы.

В заключении приведены основные выводы, полученные в хотя выполнения работы.

Ключевые слова: С++, объектно-ориентированное программирование, шифрование и дешифрование, шифрование данных, проектирование базы информационного поиска студентов, поиск по заданным критериям.

ВВЕДЕНИЕ

Задачей курсовой работы является: «Разбить группу на 2 части, с указанием интервала года рождения:

- хорошисты и отличники;

- троечники;

Каждую часть отсортировать по номерам зачетных книжек»

Цель курсовой работы по дисциплине «Языки программирования» состоит в закреплении и углублении знаний и навыков, полученных при изучении дисциплины. Курсовая работа предполагает выполнение задания повышенной сложности по проектированию, разработке и тестированию программного обеспечения, а также оформлению сопутствующей документации.

В результате изучения дисциплины обучающийся должен:

Знать:

1) языки, системы и инструментальные средства программирования в профессиональной деятельности;

2) математический аппарат, математически пакеты, программные комплексы;

3) общие принципы построения и использования современных языков программирования высокого уровня.

Уметь:

1) использовать языки, системы и инструментальные средства программирования в профессиональной деятельности;

2) строить алгоритм решения задачи, проводить его анализ и реализовывать в современных программных комплексах;

3) работать с интегрированной средой разработки программного обеспечения.

Владеть:

1) языками программирования, системами и инструментальными средствами программирования в профессиональной деятельности;

2) навыками разработки, документирования, тестирования и отладки программ на языке программирования высокого уровня;

3) основными методами разработки алгоритмов и программ;

4) методами создания структур данных, используемые для представления типовых информационных объектов.

Задачи курсовой работы:

1) проанализировать исходные данные, указанные в задании;

2) определить данные, структуры, классы, методы и функции, необходимые для выполнения работы согласно варианту;

3) разработать соответствующей алгоритм решения конкретной задачи;

4) реализовать элементы, описанные в пункте 2;

5) подготовить контрольные данные для тестирования программного обеспечения;

6) отладить разработанное программное обеспечение на основе контрольных данных, подготовленных в предыдущем пункте.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Структурное программирование

Структурное программирование воплощает принципы системного подхода в процессе создания и эксплуатации программного обеспечения ЭВМ. В основу структурного программирования положены следующие достаточно простые положения:

1) алгоритм и программа должны составляться поэтапно;

2) сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход;

3) логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Принципы структурного программирования:

1) программа построена из трёх базовых управляющих конструкций: последовательность, ветвление, цикл;

2) в программе базовые управляющие конструкции вложены друг в друга произвольным образом;

3) повторяющиеся фрагменты программы были оформлены в виде подпрограмм (процедур и функций);

4) перечисленные конструкции должны имеют один вход и один выход;

5) разработка программы ведётся пошагово, методом «сверху вниз».

1.2 Процедурное программирование

Процедурное программирование — программирование на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Процедурное программирование является отражением архитектуры традиционных ЭВМ.

Особенности процедурного программирования:

1) предопределенные функции. Предопределенная функция — это инструкция, идентифицируемая именем. Обычно предопределенные функции встроены в языки программирования более высокого уровня, но они получены из библиотеки или реестра, а не из программы;

2) локальная переменная. Локальная переменная — это переменная, которая объявлена, в основной структуре метода и ограничена локальной областью действия, которую она задает. Локальная переменная может использоваться только в том методе, в котором она определена;

3) глобальная переменная. Глобальная переменная — это переменная, которая объявляется вне любой другой функции, определенной в коде. Благодаря этому глобальные переменные могут использоваться во всех функциях, в отличие от локальной переменной;

4) модульность: Модульность — это когда две разные системы имеют под рукой две разные задачи, но сгруппированы вместе, чтобы сначала выполнить более крупную задачу. В этом случае каждая группа систем будет выполнять свои собственные задачи один за другим, пока все задачи не будут выполнены;

5) передача параметров: Передача параметров — это механизм, используемый для передачи параметров в функции, подпрограммы или процедуры.

1.3 Модульное программирование

Модуль в программировании — это фрагмент кода, имеющий определенное функциональное значение и характеризующийся логической завершенностью.

Модульное программирование — это способ создания программы посредством объединения модулей в единую структуру.

В основе модульного программирования лежат три основных концепции:

Основные концепции модульного программирования:

1) каждый модуль имеет единственную точку входа и выхода;

2) размер модуля по возможности должен быть минимизирован;

3) вся система построена из модулей;

4) каждый модуль не зависит от того, как реализованы другие модули.

Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

1.4 Ссылки

Ссылка — это псевдоним для другой переменной. Они объявляются при помощи символа &. Ссылки должны быть проинициализированы при объявлении, причем только один раз.

Ссылка при определении сразу же инициализируется. Инициализация ссылки производится следующим образом:

int i = 0;

int& iref = i;

Физически iref представляет собой постоянный указатель на int - переменную типа int\* const.

Поскольку ссылка — это псевдоним, то при передаче объекта в функцию по ссылке внутри нее объект можно изменять. Ссылки не могут ссылаться на другие ссылки или на поле битов. Не может быть массивов ссылок или указателей на ссылку. Ссылка может использоваться для возврата результата из функции. Возвратить результат по ссылке — значит возвратить не указатель на объект и не его значение, а сам этот объект.

1.5 Работа с памятью. Указатели

Указатели являются одним из основных понятий языка Си. В такие переменные можно записывать адреса любых участков памяти, на чаще всего – адрес начального элемента динамического массива.

Важно знать, что:

1) указатель – это переменная, в которой записан адрес другой переменной;

2) при объявлении указателя надо указать тип переменных, на которых он будет указывать, а перед именем поставить знак \*;

3) знак & перед именем переменной обозначает ее адрес;

4) знак \* перед указателем в рабочей части программы (не в объявлении) обозначает значение ячейки, на которую указывает указатель;

5) нельзя записывать по указателю, который указывает непонятно куда – это вызывает сбой программы, поскольку что-то стирается в памяти;

6) для обозначения недействительного указателя используется константа NULL;

7) при изменении значения указателя на n он в самом деле сдвигается к n-ому следующему числу данного типа, то есть для указателей на целые числа на n\*sizeof(int) байт;

8) указатель печатаются по формату %p.

1.6 Работа с файлами

Файл – это именованная область ячеек памяти, в которой хранятся данные одного типа. Файл имеет следующие характерные особенности: уникальное имя; однотипность данных; произвольная длина, которая ограничивается только емкостью диска. Для работы с файлом в языке C++ необходима ссылка на файл. Для определения такой ссылки существует структура FILE, описанная в заголовочном файле stdio.h. Данная структура содержит все необходимые поля для управления файлами, например, текущий указатель буфера, текущий счетчик байтов, базовый адрес буфера ввода-вывода, номер файла.

Функция открытия файла.

При открытии файла (потока) в программу возвращается указатель на поток (файловый указатель), являющийся указателем на объект структурного типа FILE. Этот указатель идентифицирует поток во всех последующих операциях.

Функция закрытия файла.

Открытые на диске файлы после окончания работы с ними рекомендуется закрыть явно. Это является хорошим тоном в программировании.

Функция переименования файла.

Функция переименовывает файл; первый параметр – старое имя файла, второй – новое. Возвращает 0 при неудачном выполнении.

Функция контроля конца файла

Для контроля достижения конца файла есть функция feof. int feof(FILE \* filename);

1.7 Динамические структуры данных

Часто в серьезных программах надо использовать данные, размер и структура которых должны меняться в процессе работы. Динамические массивы здесь не выручают, поскольку заранее нельзя сказать, сколько памяти надо выделить – это выясняется только в процессе работы. Например, надо проанализировать текст и определить, какие слова и в каком количество в нем встречаются, причем эти слова нужно расставить по алфавиту. В таких случаях применяют данные особой структуры, которые представляют собой отдельные элементы, связанные с помощью ссылок.

1.8 Линейный список

В простейшем случае каждый узел содержит всего одну ссылку. Для определенности будем считать, что решается задача частотного анализа текста – определения всех слов, встречающихся в тексте и их количества. В этом случае область данных элемента включает строку (длиной не более 40 символов) и целое число.

1.9 Функции

Функция определяет действия, которые выполняет программа. Функции позволяют выделить набор инструкций и придать ему имя. А затем многократно по присвоенному имени вызывать в различных частях программы. По сути функция — это именованный блок кода.

Функцию можно вызвать из любого количества мест в программе. Значения, передаваемые функции, являются аргументами, типы которых должны быть совместимы с типами параметров в определении функции.

Длина функции практически не ограничена, однако для максимальной эффективности кода целесообразно использовать функции, каждая из которых выполняет одиночную, четко определенную задачу. Сложные алгоритмы лучше разбивать на более короткие и простые для понимания функции, если это возможно. Функции могут быть перегружены, что означает, что разные версии функции могут использовать одно и то же имя, если они отличаются числом и типом формальных параметров.

Определение функции состоит из объявления и тела функции, заключенных в фигурные скобки, которые содержат объявления переменных, операторы и выражения. В следующем примере показано полное определение функции.

1.10 Объектно-ориентированное программирование

Объектно-ориентированное программирование – это подход, при котором вся программа рассматривается как набор взаимодействующих друг с другом объектов. При этом нам важно знать их характеристики.

Основные задачи ООП — структурировать код, повысить его читабельность и ускорить понимание логики программы. Косвенно выполняются и другие задачи: например, повышается безопасность кода и сокращается его дублирование.

Такой подход помогает строить сложные системы более просто и естественно благодаря тому, что вся предметная область разбивается на объекты и каждый из них слабо связан с другими объектами. Слабая связанность возникает вследствие соблюдения трех принципов: инкапсуляции, наследования и полиморфизма.

1) инкапсуляция – сокрытие поведения объекта внутри него. Объекту «водитель» не нужно знать, что происходит в объекте «машина», чтобы она ехала. Это ключевой принцип ООП;

2) полиморфизм – это переопределение поведения. Можно снова рассмотреть «человека» и «водителя», но теперь добавить «пешехода». Человек умеет как-то передвигаться, но как именно, зависит от того, водитель он или пешеход. То есть у пешехода и водителя схожее поведение, но реализованное по-разному: один перемещается ногами, другой – на машине.

ООП позволяет упростить сложные объекты, составляя их из более маленьких и простых, поэтому над программой могут работать сотни разработчиков, каждый из которых занят своим блоком. Большинство современных языков программирования — объектно-ориентированные, и, однажды поняв суть, вы сможете освоить сразу несколько языков.

1.11 Конструкторы и деструкторы

Конструктор – это функция-член, имя которой совпадает с именем класса, инициализирующая переменные-члены, распределяющая память для их хранения.

Деструктор – это функция-член, имя которой представляет собой имя класса, предназначенная для уничтожения переменных.

При создании объектов одной из наиболее широко используемых операций которую вы будете выполнять в ваших программах, является инициализация элементов данных объекта. Чтобы упростить процесс инициализации элементов данных класса, C++ использует специальную функцию, называемую конструктором, которая запускается для каждого создаваемого вами объекта. Также C++ обеспечивает функцию, называемую деструктором, которая запускается при уничтожении объекта.

Конструктор представляет собой метод класса, который облегчает вашим программам инициализацию полей при создании объекта класса. Конструктор имеет такое же имя, как и сам класс. Конструктор не имеет возвращаемого значения. Конструкторы относят к интерфейсу класса, чтобы с их помощью можно было создавать объекты данного класса из внешней части программы.

Таким образом, деструктор не может быть перегружен и должен существовать в классе в единственном экземпляре. Деструктор вызывается автоматически при уничтожении объекта.

1.12 Классы и дружественные функции

Класс представляет составной тип, который может использовать другие типы. Классы и объекты в С++ являются основными концепциями объектно-ориентированного программирования — ООП.

Класс предназначен для описания некоторого типа объектов. То есть класс является планом объекта. А объект представляет конкретное воплощение класса, его реализацию. Объекты — конкретное представление абстракции, которые имеют свои свойства и методы. Свойства — это любые данные, которыми можно характеризовать объект класса. Методы — это функции, выполняющие различные действия над данными (свойствами) класса. Поле класса в объектно-ориентированном программировании — переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях.

Класс может определять переменные и константы для хранения состояния объекта и функции для определения поведения объекта.

На объекты классов, как и на объекты других типов, можно определять указатели. Затем через указатель можно обращаться к членам класса - переменным и методам. Однако если при обращении через обычную переменную используется символ точка, то для обращения к членам класса через указатель применяется стрелка (->).

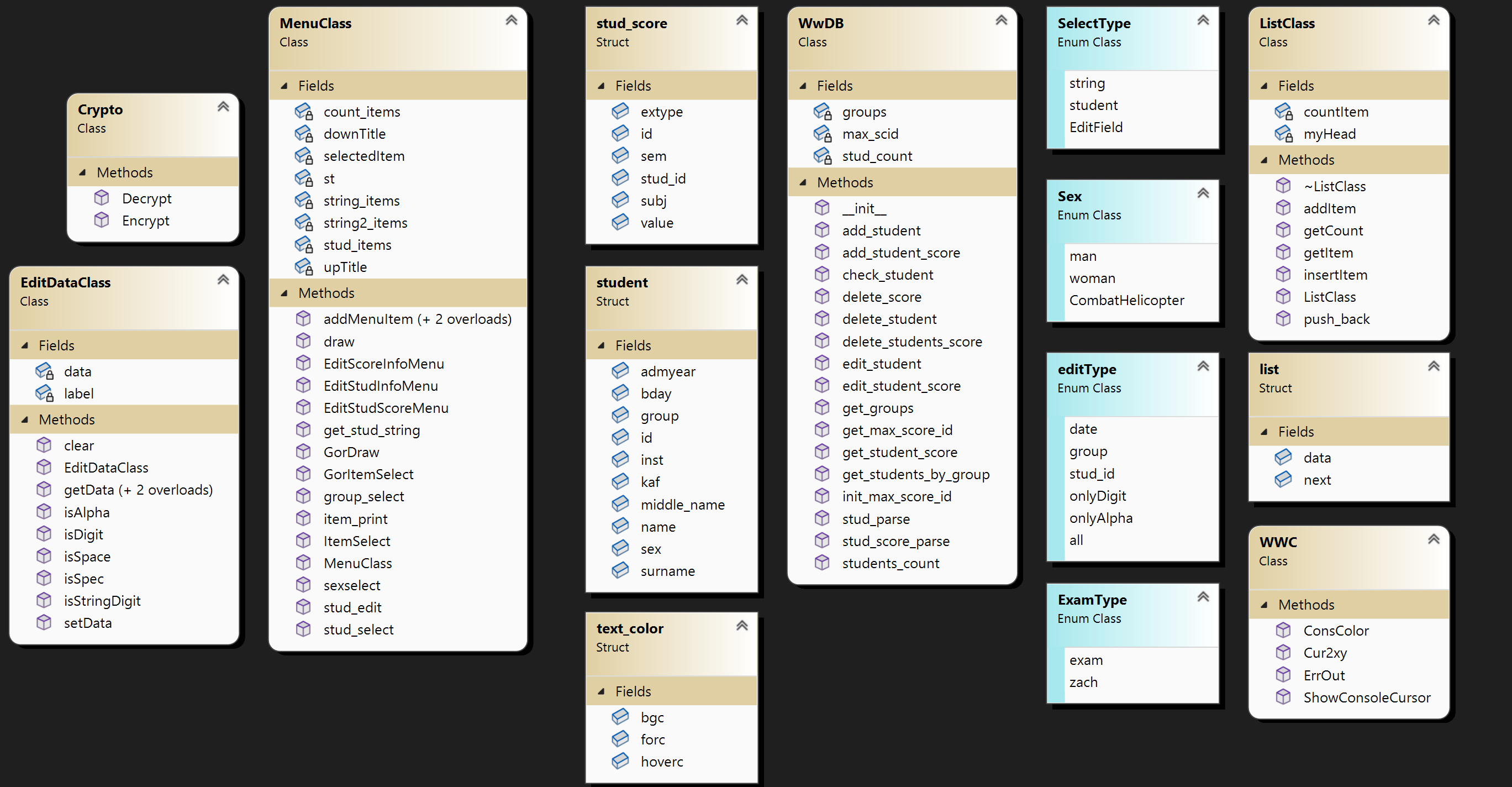
Дружественные функции — это функции, которые не являются членами класса, однако имеют доступ к его закрытым членам - переменным и функциям, которые имеют спецификатор private.

Для определения дружественных функций используется ключевое слово friend.

1.12.1 Описание классов и их методов

Классы в языке программировании С++ — это абстракция, которая описывает методы и свойства, ещё не существующих объектов. Объекты — конкретное представление абстракции, которые имеют свои свойства и методы. Свойства — это любые данные, которыми можно характеризовать объект класса. Методы — это функции, выполняющие различные действия над данными (свойствами) класса. Поле класса в объектно-ориентированном программировании — переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях.

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте.



1.13 Методы шифрования и дешифрования

Шифрование – это процесс кодирования информации с целью предотвращения несанкционированного доступа.

Дешифрование – процесс, обратный процессу шифрованию.

OpenSSL – криптографическая библиотека с открытым исходным кодом. Библиотека поддерживает почти все низкоуровневые алгоритмы хеширования и шифрования, а также реализует большинство популярных криптографических стандартов, в том числе: позволяет создавать ключи RSA, DH, DSA, шифровать данные.

Для начала реализуем функцию Encrypt, для этого поэтапно реализуем следующее:

1) Шифрование алгоритмом AES строки;

2) Запись зашифрованных данных в файл;

3) Чтение данных из файла и его удаление;

Листинг 1.13.1

string command = "echo " + to\_enc + " | openssl\\bin\\openssl.exe aes-128-ecb -pbkdf2 -a -A -pass pass:" + pass + " 1>crptmp 2>NUL";

system(command.c\_str());

|  |  |
| --- | --- |
|  |  |

Листинг 1.13.2

fstream crptmpf("crptmp", fstream::in);

getline(crptmpf, to\_enc);

crptmpf.close();

remove("crptmp");

Для реализации функции Decrypt поэтапно реализуем следующее:

1) Расшифровка зашифрованной строки;

2) Считывание данных из файла;

3) Удаление файла;

Листинг 1.13.3

string command = "echo " + to\_dec + " | openssl\\bin\\openssl.exe aes-128-ecb -pbkdf2 -a -d -pass pass:" + pass + " > crptmp";

system(command.c\_str());

Листинг 1.13.4

fstream crptmpf("crptmp", fstream::in);

getline(crptmpf, to\_dec);

crptmpf.close();

remove("crptmp");

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Структура программы

Исходными данными для программы является информация о группе студентов из N человек, где запись о студенте содержит следующие данные (таблица 2.1.1).

Таблица 2.1.1 – Структура student и типы данных

|  |  |
| --- | --- |
| Основная структура данных | Типы данных |
| Фамилия | string(строковый) |
| Имя | string(строковый) |
| Отчество | string(строковый) |
| Пол студента | Sex(пол) |
| Число, месяц, год рождения | String(строковый) |
| Год поступления в институт | int(целочисленный) |
| Институт | string(строковый) |
| Кафедра | string(строковый) |
| Группа | string(строковый) |
| Номер зачетной книжки | string(строковый) |

Таблица 2.1.2 – Структура stud\_score и типы данных

|  |  |
| --- | --- |
| Основная структура данных | Типы данных |
| Id | int(целочисленный) |
| Id студента (шифр) | string(строковый) |
| Тип экзамена | ExamType(enum) |
| Название предмета | string(строковый) |
| Оценка | short(целочисленный) |
| Семестр | int(целочисленный) |

По условиям курсовой работы, допустимо максимально 9 сессий и 10 предметов в каждом семестре, которые могут быть разными. Все данные являются форматными.

Для реализации соответствующей задачи были разработаны классы. В данных классах описаны методы решения поставленной задачи.

В представленной ниже таблице продемонстрированы роли используемых классов (таблица 2.1.3).

Таблица 2.1.3 – Классы

|  |  |
| --- | --- |
| Класс | Назначение |
| ListClass | Динамический список структур student |
| Crypto | Шифрование данных |
| WwDB | Набор объектов для работы с базой данных |
| WWC | Набор методов для работы с консолью |
| EditDataClass | Набор методов для получения информации с консоли |
| Menu | Набор методов для работы меню |

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте (таблица 2.1.4).

Таблица 2.1.4 – Поля классов

|  |  |  |
| --- | --- | --- |
| Классы | Поле класса | |
| Тип данных | Название и характеристика |
| ListClass | List\* | myHead – начало списка |
| int | countItems – длина списка |
| WwDB | int | stud\_count – количество студентов  max\_scid – максимальный id оценки |
| set<string> | groups - группы |
| EditDataClass | string | label – название ввода  data – строка вводимых данных |
| Menu | int | count\_items – количество пунктов меню  selected\_item – выбранный элемент |
| vector<string> | string\_items – пункты меню |
| vector<student> | stud\_items – пункты меню |
| vector<pair<string, string>> | string2\_items – пункты меню |
| string | upTitle – заголовок меню |
| SelectType | st – тип пунктов меню |

Далее методы классов, представленные в таблице 2.1.5.

Таблица 2.1.5 Методы классов

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Класс | Методы класса | | | | | | | | |
| Название | | | Назначение | | | Аргументы  (их тип) | Тип возвращаемого значения | |
| ListClass | ~ListClass | | | Деструктор класса | | | - | - | |
| addItem | | | Добавить элемент | | | \_data (student) | - | |
| insertItem | | | Вставить элемент | | | Index(int), \_data(student) | - | |
| push\_back | | | Добавить элемент в конец | | | \_data(student) | - | |
| getCount | | | Получить количество элементов | | | - | int | |
| getItem | | | Получить элемент | | | index(int) | student | |
| WWC | ConsColor | | | Установка цвета текста консоли | | | ForC, BackC (int) | - | |
| Cur2xy | | | Установка положения каретки консоли | | | xpos, ypos (int) | - | |
| ShowConsoleCursor | | | Смена отображения курсора | | | showFlag (bool) | - | |
| ErrOut | | | Вывод ошибки | | | Err (string) | - | |
| EditDataClass | clear() | | | Очистка консоли | | | \_data (string) | - | |
| setData | | | Установка значения поля ввода | | | \_data (string) | - | |
| isDigit | | | Проверка на число | | | ch (char) | bool | |
| isStringDigit | | | Проверка строки на число | | | \_str (string) | bool | |
| isAlpha | | | Проверка на букву | | | ch (char) | bool | |
| isSpace | | | Проверка на пробел | | | ch (char) | bool | |
| isSpec | | | Проверка на спец. символ | | | ch (char) | bool | |
| getData | | | Ввод данных из консоли | | | et (editType) | string | |
| getData | | | Ввод данных из консоли | | | et (editType),  min, max (int) | int | |
| getData | | | Ввод данных из консоли | | | et (editType),  len (int) | string | |
| Crypto | Decrypt | | | Дешифровать строку | | | to\_dec(string) | string | |
| Encrypt | | | Зашифровать строку | | | to\_enc(string) | string | |
| WwDB | \_\_init\_\_ | | | Инициализация и подготовка базы данных к работе | | | - | - | |
| get\_groups | | | Получить список групп | | | - | vector<string> | |
| init\_max\_score\_id | | | Инициализировать максимальный id оценки | | | - | - | |
| get\_max\_score\_id | | | Получить максимальный id оценки | | | - | int | |
| students\_count | | | Получить количество студентов | | | - | int | |
| stud\_parse | | | Парсинг строки со студентом | | | st (string), jsr (int) | student | |
| stud\_score\_parse | | | Парсинг строки с оценкой | | | st (string), jsr (int) | stud\_score | |
| check\_student | | | Проверка наличия студента | | | st (student) | bool | |
| add\_student | | | Добавление студента | | | st (student) | - | |
| edit\_student | | | Изменение студента | st (student) | | | - | |
| get\_students\_by\_group | | | Получить студентов группы | | | \_group (string) | vector<student> | |
| get\_student\_score | | | Получить оценки студента | | | \_stud\_id (string) | vector<stud\_score> | |
| add\_student\_score | | | Добавить оценку студенту | | | stud\_sc (stud\_score) | int | |
| edit\_student\_score | | | Изменить оценку студента | | | stud\_sc (stud\_score) | - | |
| delete\_score | | | Удалить оценку | | | \_id (int) | - | |
| delete\_students\_score | | | Удалить все оценки студента | | | stud\_id (string) | - | |
| delete\_student | | | Удалить студента | | | stud\_id (string) | bool | |
| Menu | | removeStudent | Удаление студента | | | zach (string)  (номер зачётки) | | | - |
| addMenuItem | Добавить элемент меню (string) | | | \_item (string) | | | - |
| addMenuItem | Добавить элемент меню (student) | | | \_item (student) | | | - |
| addMenuItem | Добавить элемент меню (pair<string, string>) | | | \_item (pair<string, string>) | | | - |
| get\_stud\_string | Получить строку с информацией о студенте | | | i (int) | | | string |
| item\_print | Вывести элемент меню | | | i (int), fl (bool) | | | - |
| draw | Вывести меню на экран | | | colors (map<int, text\_color>) | | | - |
| ItemSelect | Функция для выбора элемента меню | | | colors (map<int, text\_color>),  fl (short) | | | int |
| group\_select | Меню «Выбор группы» | | | - | | | string |
| stud\_select | Меню «Выбор студента» | | | - | | | student |
| stud\_edit | Меню «Изменение студента» | | | stud\_id (string) | | | int |
| EditStudInfoMenu | Меню «Изменение информации о студенте» | | | stud (student) | | | student |
| EditStudScoreMenu | Меню «Изменение оценок» | | | stud\_id (string) | | | stud\_score |
| EditScoreInfoMenu | Меню «Изменение оценки» | | | stsc (stud\_score) | | | stud\_score |
| GorDraw | Отрисовка меню горизонтального выбора | | | - | | | - |
| GorItemSelect | Выбор элементов горизонтального меню | | | x, y (int) | | | int |
| sexselect | Меню «Выбор пола» | | | x, y (int) | | | string |

2.2 Алгоритм решения задачи

«Разбить группу на 2 части, с указанием интервала года рождения:

- хорошисты и отличники;

- троечники;

Каждую часть отсортировать по номерам зачетных книжек»

Поэтапное описание алгоритма выборки/сортировки для выполнения варианта

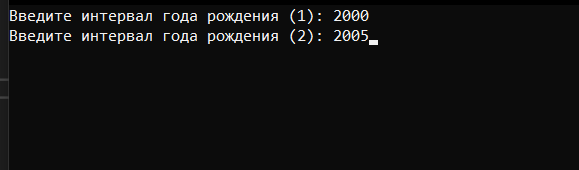
1. Ввод интервала года рождения
2. Инициализация массива студентов
3. Проход по массиву студентов
   1. Проверка года рождения
   2. Получение оценок студента
   3. Проход по массиву оценок
   4. Проверка на «положительность» оценки
   5. Добавление в массив студентов (отл/хор)/троечников в зависимости от оценок
4. Вывод

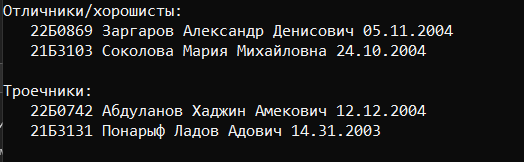
Блок-схема



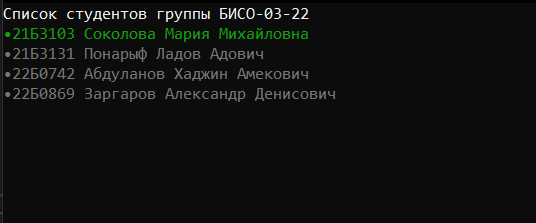
2.3 Программная реализация задания

2.3.1 Пример выполнения задания

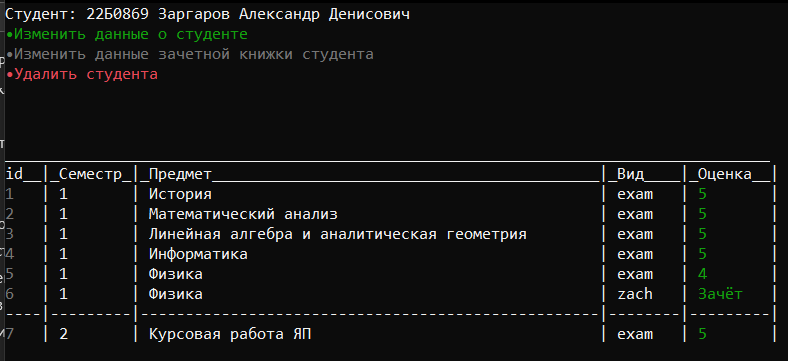


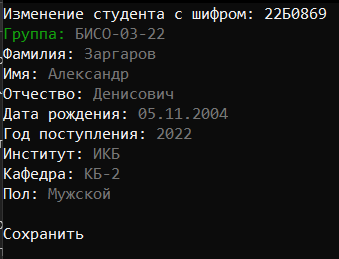


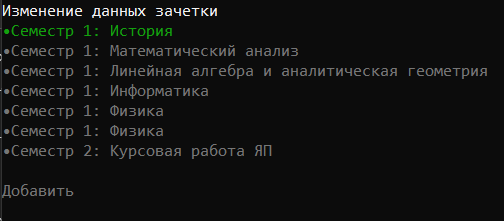
2.3.2 Отображение списка студентов

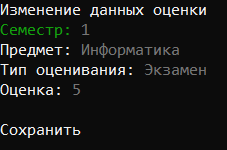


2.3.3 Изменение данных о студенте



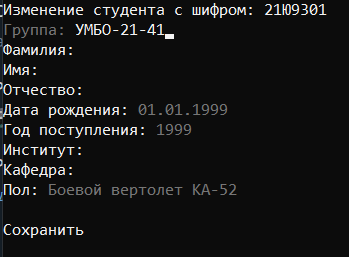




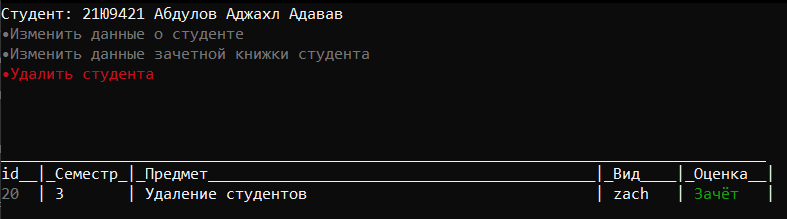


2.3.3 Добавление информации о новом студенте



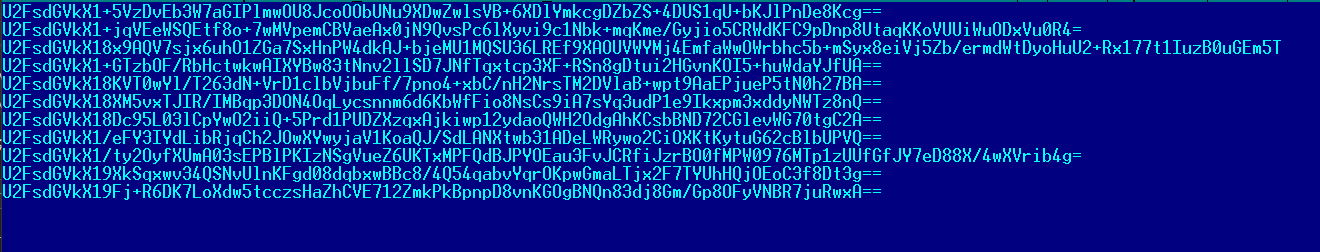


2.3.4 Удаление данных о студенте





2.3.5 Шифрование данных \*



2.4 Руководство пользователя

Управление

Навигация по меню:

* "W" | "↑" - Вверх
* "A" | "←" | "Backspace" - Назад (в начальном меню - выход)
* "S" | "↓" - Вниз
* "D" | "→" | "Enter" - Выбрать/вперед

Дополнительные функции:

* "N" - добавить студента (работает только в начальном меню (с группами))
* "V" - запустить выполнение варианта (работает только в меню выбора студента)
* "Delete" - удалить оценку (только в меню изменения оценки).

Описание работы программы

При запуске программы на экране выводится консоль с меню, где пользователь может выбрать одну из нескольких групп. Из этого же меню можно создать нового студента путем нажатия клавиши «N». При попытке вернуться назад программа предложит выйти из нее.

Далее, после выбора группы, пользователю предоставляется выбор студента. Отсюда можно выполнить вариант, путем нажатия клавиши «V».

После выбора пользователя откроется меню просмотра оценок и выбора действия над пользователем. Таблица оценок отображается снизу. Предметы разделены по семестрам, оценки подсвечиваются соответствующим им цветом. На выбор есть три вида действия над студентом:

1. Посмотреть/изменить информацию о студенте
2. Изменить информацию об оценках
3. Удалить студента из базы данных (при этом также удаляются все его оценки)

При выборе пункта «1» откроется меню с информацией о студенте. Чтобы изменить значение поля, нужно нажать Enter либо стрелку вправо. Чтобы подтвердить ввод, нужно нажать Enter. Для сохранения информации нужно выбрать пункт «Сохранить». Для выхода без сохранения необходимо инициировать возврат назад соответствующими клавишами.

При выборе пункта «2» откроется меню выбора оценки. Для изменения или добавления оценки необходимо выбрать соответствующий пункт меню. После этого откроется меню изменения оценки. Управление аналогичное с изменением студента. Для удаления оценки, необходимо нажать клавишу «Delete».

При выборе пункта «3» из базы данных удаляется вся информация о студенте.

2.8 Системные требования

Язык программирования: С++.

Операционная система: Linux Mint 20.1 выше или аналог, Windows 10.

ОЗУ: 1 Гб и более.

Свободное место на диске: 200 Мб и более.

ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы была написана программа, «Разбить группу на 2 части, с указанием интервала года рождения:

- хорошисты и отличники;

- троечники;

Каждую часть отсортировать по номерам зачетных книжек»

Кроме того, в процессе реализации программы мною были закреплены базовые навыки программирования, полученные при изучении дисциплины Языки программирования и языка программирования C++.

Были подробно рассмотрены теоретические выкладки, использованные в процессе написания программы.

Программа была реализована c использованием технологии ООП. Более того, была внедрена работа с динамической памятью, работа с файлами. Более того, детально были разобраны и применены на практике методы симметричного и ассиметричного шифрования и дешифрования файлов.

Дополнительно были созданы необходимые и достаточные условия для корректного выполнения индивидуального варианта задания.

По результату выполнения тестирования программного продукта, можно сделать вывод о том, что программа работает корректно и справляется с поставленной задачей.

Перспективы разработки данного продукта характеризуются высокой потребностью в универсальном и безопасном программном продукте, предназначенном для внедрения в средства автоматизации и информационные системы.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Мерсов А. А. Основы объектно-ориентированного программирования на языке С++ [Электронный ресурс]: практикум / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2021. — Электрон. опт. диск (ISO)
2. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2022.
3. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2022.
4. Стенли Липпман Язык программирования С++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2023. 1104 c.
5. Страуструп Б. Дизайн и эволюция языка С++. – Litres, 2022.
6. Уильямс Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. – Litres, 2022.
7. Аммерааль Л. STL для программистов на C++. – Litres, 2022.
8. Дейл Н., Уимз Ч., Хедингтон М. Программирование на C++. – Litres, 2022.
9. Нефедов Д. Г., Русяк И. Г., Вавилова Д. Д. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. – 2020.
10. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – " Издательский дом"" Питер""", 2018.
11. Гудлиф П. Ремесло программиста. – 2009.
12. Мерсов А. А. Языки программирования [Электронный ресурс]: методические рекомендации по выполнению курсовой работы / А. А. Мерсов, А. М. Русаков, В. В. Филатов. — М.: РТУ МИРЭА, 2022. — Электрон. опт. диск (ISO).

ПРИЛОЖЕНИЕ А Исходный код программы

Cursach\_StudentsDB.cpp

#include <iostream>

#include <string>

#include <conio.h>

#include "EditDataClass.h"

#include "Menu.h"

#include "WorkWithConsole.h"

#include "Work\_with\_DB.h"

#include "List.h"

#include <thread>

#include <chrono>

using namespace std;

string selectedItem;

void GroupMenu(vector<string> groups) {

MenuClass\* mainMenu = new MenuClass("Главное меню");

for (int i = 0; i < groups.size(); ++i)

mainMenu->addMenuItem(groups[i]);

selectedItem = mainMenu->group\_select();

delete mainMenu;

}

student StudentsListMenu(vector<student> studs) {

MenuClass\* stud\_list = new MenuClass("Список студентов группы "+selectedItem);

//cout << studs[0].id;

for (int i = 0; i < studs.size(); ++i)

stud\_list->addMenuItem(studs[i]);

student stud\_selectedItem = stud\_list->stud\_select();

delete stud\_list;

return stud\_selectedItem;

}

void AddStudent(student& stud) {

MenuClass\* stud\_add\_menu = new MenuClass("Изменение студента с шифром: " + stud.id);

stud = stud\_add\_menu->EditStudInfoMenu(stud);

delete stud\_add\_menu;

return;

}

int StudentEdit(student stud) {

MenuClass\* stud\_edit\_menu = new MenuClass("Студент: " + stud.id + " " + stud.surname + " " + stud.name + " " + stud.middle\_name);

int code = stud\_edit\_menu->stud\_edit(stud.id);

delete stud\_edit\_menu;

return code;

}

student EditStudInfo(student stud) {

MenuClass\* stud\_edit\_menu = new MenuClass("Изменение студента с шифром: " + stud.id);

stud = stud\_edit\_menu->EditStudInfoMenu(stud);

delete stud\_edit\_menu;

return stud;

}

bool ExitMenu() {

MenuClass\* ExMenu = new MenuClass("");

ExMenu->addMenuItem("Да");

ExMenu->addMenuItem("Нет");

cout << "Вы точно хотите выйти?";

if (!ExMenu->ItemSelect({ {0, {FOREGROUND\_INTENSITY, 0, FOREGROUND\_GREEN}}, {1, {FOREGROUND\_INTENSITY, 0, FOREGROUND\_RED}} })) {

system("cls");

exit(1);

}

system("cls");

WWC::ConsColor(15);

return false;

}

stud\_score EditStudScore(string stud\_id) {

MenuClass\* EDSc = new MenuClass("Изменение данных зачетки");

stud\_score stsc = EDSc->EditStudScoreMenu(stud\_id);

delete EDSc;

if (stsc.stud\_id == "-1")

return stsc;

EDSc = new MenuClass("Изменение данных оценки");

stsc = EDSc->EditScoreInfoMenu(stsc);

delete EDSc;

return stsc;

}

void preparing() {

WWC::ShowConsoleCursor(false);

cout << "Preparing |";

while (true) {

cout << "\b/";

this\_thread::sleep\_for(chrono::milliseconds(250));

cout << "\b--";

this\_thread::sleep\_for(chrono::milliseconds(250));

cout << "\b \b\b\\";

this\_thread::sleep\_for(chrono::milliseconds(250));

cout << "\b|";

this\_thread::sleep\_for(chrono::milliseconds(250));

}

}

void variant(WwDB\* wwdb) {

EditDataClass\* edc = new EditDataClass();

cout << "Введите интервал года рождения (1): "; int y1 = edc->getData(editType::onlyDigit, 0, 9999);

edc->setData("");

cout << "\nВведите интервал года рождения (2): "; int y2 = edc->getData(editType::onlyDigit, 0, 9999);

system("cls");

thread t(preparing);

t.detach();

vector<student> students = wwdb->get\_students\_by\_group(selectedItem);

ListClass\* otl\_stud = new ListClass();

ListClass\* bad\_stud = new ListClass();

bool fl;

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

fl = true;

int ysam = stoi(students[i].bday.substr(6, 10));

if (!(y1 <= ysam && ysam <= y2))

continue;

vector<stud\_score> stsc = wwdb->get\_student\_score(students[i].id);

for (int j = 0; j < stsc.size(); ++j) {

if (stsc[j].value == '-' || stsc[j].value < 4) {

bad\_stud->addItem(students[i]);

fl = false;

break;

}

}

if (fl)

otl\_stud->addItem(students[i]);

}

t.~thread();

Sleep(10);

system("cls");

student tmp;

cout << "Отличники/хорошисты:\n";

for (int i = 0; i < otl\_stud->getCount(); ++i) {

tmp = otl\_stud->getItem(i);

cout << " " << tmp.id << " " << tmp.surname << " " << tmp.name << " " << tmp.middle\_name << " " << tmp.bday << endl;

}

cout << "\nТроечники:\n";

for (int i = 0; i < bad\_stud->getCount(); ++i) {

tmp = bad\_stud->getItem(i);

cout << " " << tmp.id << " " << tmp.surname << " " << tmp.name << " " << tmp.middle\_name << " " << tmp.bday << endl;

}

delete otl\_stud;

delete bad\_stud;

}

int main()

{

setlocale(LC\_ALL, "Russian");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

WwDB\* wwdb = new WwDB();

wwdb->\_\_init\_\_();

Groupsm:

GroupMenu(wwdb->get\_groups());

if (selectedItem == "n") {

system("cls");

student stud;

EditDataClass\* edc = new EditDataClass();

while (true) {

cout << "Введите шифр студента: ";

WWC::ShowConsoleCursor(true);

stud.id = edc->getData(editType::stud\_id);

if (wwdb->check\_student(stud))

break;

}

system("cls");

while (true) {

AddStudent(stud);

if (stud.id == "-1")

goto Groupsm;

if (stud.group == "" || stud.surname == "" || stud.name == "") {

system("cls");

WWC::Cur2xy(0, 13);

WWC::ErrOut("Группа, фамилия и имя обязательны!");

WWC::Cur2xy(0, 0);

continue;

}

if (stud.bday.length() != 10) {

system("cls");

WWC::Cur2xy(0, 13);

WWC::ErrOut("Введена некорректная дата!");

WWC::Cur2xy(0, 0);

continue;

}

break;

}

wwdb->add\_student(stud);

delete edc;

system("cls");

goto Groupsm;

}

else if (selectedItem == "-1") {

system("cls");

ExitMenu();

goto Groupsm;

}

Studlistm:

student stud\_selectedItem = StudentsListMenu(wwdb->get\_students\_by\_group(selectedItem));

if (stud\_selectedItem.id == "-1")

goto Groupsm;

else if (stud\_selectedItem.id == "var") {

system("cls");

variant(wwdb);

exit(0);

}

StudEditm:

int excode = StudentEdit(stud\_selectedItem);

switch (excode) {

case -1:

goto Studlistm;

break;

case 0: {

student tmp = EditStudInfo(stud\_selectedItem);

if (tmp.id != "-1") {

wwdb->edit\_student(tmp);

stud\_selectedItem = tmp;

}

system("cls");

goto StudEditm;

}

break;

case 1: {

stud\_score stsc = EditStudScore(stud\_selectedItem.id);

if ( stsc.id == -2 || stsc.stud\_id == "-1")

goto StudEditm;

else if (stsc.stud\_id == "ddd") {

wwdb->delete\_score(stsc.id);

}

else if (stsc.id == -1) {

stsc.id = wwdb->get\_max\_score\_id() + 1;

wwdb->add\_student\_score(stsc);

}

else

wwdb->edit\_student\_score(stsc);

system("cls");

goto StudEditm;

}

case 2:

wwdb->delete\_student(stud\_selectedItem.id);

goto Studlistm;

}

delete wwdb;

}

Crypto.h

#pragma once

#include <iostream>

#include <stdio.h>

#include <windows.h>

class Crypto {

public:

static string Encrypt(string to\_enc) {

if (to\_enc == "")

return "";

string pass = "Zargarov";

string command = "echo " + to\_enc + " | openssl\\bin\\openssl.exe aes-128-ecb -pbkdf2 -a -A -pass pass:" + pass + " 1>crptmp 2>NUL";

system(command.c\_str());

fstream crptmpf("crptmp", fstream::in);

getline(crptmpf, to\_enc);

crptmpf.close();

remove("crptmp");

return to\_enc;

}

static string Decrypt(string to\_dec) {

if (to\_dec == "")

return "";

string pass = "Zargarov";

string command = "echo " + to\_dec + " | openssl\\bin\\openssl.exe aes-128-ecb -pbkdf2 -a -d -pass pass:" + pass + " 1>crptmp 2>NUL";

system(command.c\_str());

fstream crptmpf("crptmp", fstream::in);

getline(crptmpf, to\_dec);

crptmpf.close();

remove("crptmp");

return to\_dec;

}

};

EditDataClass.h

#pragma once

#include "windows.h"

#include <iostream>

#include <string.h>

#include <conio.h>

#include <stdio.h>

#include "WorkWithConsole.h"

using namespace std;

enum class editType : char { date, group, stud\_id, onlyDigit, onlyAlpha, all };

class EditDataClass {

private:

string label;

string data;

public:

EditDataClass() {

label = "";

data = "";

}

void setData(string \_data = "") {

if (\_data.length() > 1)

this->data = \_data;

else

data = "";

}

bool isDigit(char ch) {

if (ch >= 48 and ch <= 57)

return true;

else

return false;

}

bool isStringDigit(string \_str) {

for (int i = 0; i < \_str.length(); i++)

if (not isDigit(\_str[i]))

return false;

if (\_str.length() == 0)

return false;

return true;

}

bool isAlpha(int ch) {

if (ch >= 'a' and ch <= 'z')

return true;

if (ch >= 'A' and ch <= 'Z')

return true;

if (ch >= -200 and ch <= -1)

return true;

if (ch == ' ')

return true;

return false;

}

bool isSpace(char ch) {

if (ch == 32)

return true;

else

return false;

}

bool isSpec(char ch) {

if (ch >= 33 and ch <= 47)

return true;

else

return false;

}

void clear(string \_data = "") {

system("cls");

data = \_data;

}

string getData(enum class editType et) {

char ch = 0;

WWC::ShowConsoleCursor(true);

cout << data;

while (ch != 13) {

ch = \_getch();

if (ch == 8) {

if (data.length() > 0) {

data.pop\_back();

//system("cls");

cout << "\b \b";

}

continue;

}

if (et == editType::stud\_id) {

if (data.size() < 2)

if (isDigit(ch)) {

cout << ch;

data.push\_back(ch);

}

if (data.size() == 2)

if (isAlpha(ch)) {

cout << ch;

data.push\_back(ch);

}

if (data.size() > 2 && data.size() < 7)

if (isDigit(ch)) {

cout << ch;

data.push\_back(ch);

}

}

if (et == editType::date) {

if (0 <= data.size() && data.size() < 10 && data.size() != 2 && data.size() != 5)

if (isDigit(ch)) {

cout << ch;

data.push\_back(ch);

}

if (data.size() == 2 || data.size() == 5) {

cout << ".";

data.push\_back('.');

}

}

if (et == editType::group) {

if (0 <= data.size() && data.size() <= 4)

if (isAlpha(ch)) {

cout << ch;

data.push\_back(ch);

}

if (data.size() == 5 || data.size() == 6 || data.size() == 8 || data.size() == 9)

if (isDigit(ch)) {

cout << ch;

data.push\_back(ch);

}

if (data.size() == 4 || data.size() == 7) {

cout << "-";

data.push\_back('-');

}

}

if (et == editType::onlyDigit)

if (isDigit(ch)) {

cout << ch;

data.push\_back(ch);

}

if (et == editType::onlyAlpha)

if (isAlpha(ch)) {

cout << ch;

data.push\_back(ch);

}

if (et == editType::all) {

if (ch != '\"' && ch != 13 && ch != '\n') {

cout << ch;

data.push\_back(ch);

}

}

}

WWC::ShowConsoleCursor(false);

return data;

}

int getData(enum class editType et, int min, int max) {

DiapError:

if (et == editType::onlyDigit) {

getData(et);

int num = max + 1;

if (isStringDigit(data))

num = atoi(data.c\_str());

if (data.length() < 1) {

WWC::ConsColor(FOREGROUND\_RED);

cout << endl << "Egor: Введите число\n";

WWC::ConsColor(15);

goto DiapError;

}

if (!(num >= min and num <= max)) {

WWC::ConsColor(FOREGROUND\_RED);

cout << endl << "Egor: Введенное число (" << num << ") Не входит в отрезок [" << min << "; " << max << "]\n";

WWC::ConsColor(15);

goto DiapError;

}

return num;

}

}

string getData(enum class editType et, int len) {

LenError:

if (et == editType::onlyAlpha) {

getData(et);

if (data.length() < 1) {

WWC::ConsColor(FOREGROUND\_RED);

cout << endl << "Egor: Введите строку\n";

WWC::ConsColor(15);

goto LenError;

}

if (data.length() > len) {

WWC::ConsColor(FOREGROUND\_RED);

cout << endl << "Egor: Длина строки больше допустимой (" << data.length() << ") Разрешено: " << len << endl;

WWC::ConsColor(15);

goto LenError;

}

return data;

}

}

};

List.h

#pragma once

#include <iostream>

#include <string>

#include <algorithm>

#include "Work\_with\_DB.h"

using namespace std;

struct list {

student data;

struct list\* next;

};

class ListClass {

private:

list\* myHead;

int countItem = 0;

public:

ListClass() {

myHead;

countItem = 0;

}

~ListClass() {

struct list\* old = NULL;

struct list\* current = myHead;

while (current != NULL) {

old = current;

current = current->next;

delete old;

}

}

void addItem(student \_data)

{

struct list\* newItem = new list();

newItem->data = \_data;

if (countItem == 0)

newItem->next = NULL;

else

newItem->next = myHead;

myHead = newItem;

countItem++;

}

void insertItem(int index, student \_data) {

if (not (index >= 0 and index <= countItem and countItem >= 0))

return;

if (index == 0)

addItem(\_data);

else {

struct list\* current = myHead;

for (int i = 0; i < index - 1; i++) {

current = current->next;

}

struct list\* newItem = new list();

newItem->data = \_data;

newItem->next = current->next;

current->next = newItem;

countItem++;

}

}

void push\_back(student \_data) {

insertItem(countItem, \_data);

}

int getCount() {

return countItem;

}

student getItem(int index) {

student \_data = { "Error::list\_error" };

if (index >= 0 && index < countItem && countItem > 0) {

list\* current = myHead;

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

current = current->next;

}

\_data = current->data;

}

return \_data;

}

};

Menu.h

#pragma once

#include <iostream>

#include <string.h>

#include "windows.h"

#include <conio.h>

#include <stdio.h>

#include <vector>

#include <map>

#include <iomanip>

#include "EditDataClass.h"

#include "WorkWithConsole.h"

#include "Work\_with\_DB.h"

using namespace std;

enum class SelectType : char { string, student, EditField };

struct text\_color {

int forc;

int bgc;

int hoverc;

};

class MenuClass {

private:

int count\_items = 0;

vector<string> string\_items;

vector<pair<string, string>> string2\_items;

vector<student> stud\_items;

string upTitle; //Заголовок меню (сверху)

int selectedItem = 0;

SelectType st;

public:

MenuClass(const string \_upTitle) {

upTitle = move(\_upTitle);

}

void addMenuItem(string \_item) {

string\_items.push\_back(\_item);

count\_items++;

}

void addMenuItem(student \_item) {

stud\_items.push\_back(\_item);

count\_items++;

}

void addMenuItem(pair<string, string> \_item) {

string2\_items.push\_back(make\_pair(\_item.first, \_item.second));

count\_items++;

}

string get\_stud\_string(int i) {

return stud\_items[i].id + " " + stud\_items[i].surname + " " + stud\_items[i].name + " " + stud\_items[i].middle\_name;

}

void item\_print(int i, bool fl = false) {

switch (st) {

case SelectType::string:

cout << "•" << string\_items[i] << endl;

break;

case SelectType::student:

cout << "•" << get\_stud\_string(i) << endl;

break;

case SelectType::EditField:

WWC::ConsColor(fl ? FOREGROUND\_GREEN : 15); cout << string2\_items[i].first;

WWC::ConsColor(FOREGROUND\_INTENSITY); cout << string2\_items[i].second << endl;

break;

}

}

void draw(map<int, text\_color> colors) {

cout << upTitle << endl;

for (int i = 0; i < count\_items; ++i) {

if ((colors[i].forc | colors[i].bgc | colors[i].hoverc) != 0)

WWC::ConsColor(colors[i].forc, colors[i].bgc);

if (i == selectedItem) {

WWC::ConsColor(FOREGROUND\_GREEN);

item\_print(i, true);

WWC::ConsColor(FOREGROUND\_INTENSITY);

}

else

item\_print(i);

}

}

int ItemSelect(map<int, text\_color> colors = {}, short fl = 0) {

char ch = 0;

draw(colors);

WWC::ShowConsoleCursor(false);

while (ch != 13) {

ch = \_getch();

if (fl == 1 && (ch == 'n' || ch == 'N' || ch == 'т' || ch == 'Т')) {

selectedItem = 'n';

break;

}

else if (fl == 2 && ch == 83) {

selectedItem = 'd';

break;

}

else if (fl == 3 && (ch == 'v' || ch == 'V' || ch == 'м' || ch == 'М')) {

selectedItem = 'v';

break;

}

else if (ch == 75 || ch == 97 || ch == 8) // Left

{

selectedItem = -1;

break;

}

if (count\_items == 0)

continue;

if (ch == 72 || ch == 119) // Up

{

WWC::Cur2xy(0, selectedItem + 1);

if ((colors[selectedItem].forc | colors[selectedItem].bgc) == 0)

WWC::ConsColor(FOREGROUND\_INTENSITY);

else

WWC::ConsColor(colors[selectedItem].forc, colors[selectedItem].bgc);

item\_print(selectedItem);

selectedItem -= selectedItem > 0 ? 1 : 0;

WWC::Cur2xy(0, selectedItem + 1);

if (colors[selectedItem].hoverc == 0)

WWC::ConsColor(FOREGROUND\_GREEN);

else

WWC::ConsColor(colors[selectedItem].hoverc);

item\_print(selectedItem, true);

}

else if (ch == 80 || ch == 115) // Down

{

WWC::Cur2xy(0, selectedItem + 1);

if ((colors[selectedItem].forc | colors[selectedItem].bgc) == 0)

WWC::ConsColor(FOREGROUND\_INTENSITY);

else

WWC::ConsColor(colors[selectedItem].forc, colors[selectedItem].bgc);

item\_print(selectedItem);

selectedItem += selectedItem < count\_items - 1 ? 1 : 0;

WWC::Cur2xy(0, selectedItem + 1);

if (colors[selectedItem].hoverc == 0)

WWC::ConsColor(FOREGROUND\_GREEN);

else

WWC::ConsColor(colors[selectedItem].hoverc);

item\_print(selectedItem, true);

}

else if (ch == 77 || ch == 100) // Right

break;

Sleep(10);

}

//WWC::ShowConsoleCursor(true); ZAD

WWC::Cur2xy(0, 0);

return selectedItem;

}

string group\_select() {

st = SelectType::string;

ItemSelect({}, 1);

system("cls");

WWC::ConsColor(15);

if (selectedItem == 'n')

return "n";

else if (selectedItem == -1)

return "-1";

return string\_items[selectedItem];

}

student stud\_select() {

st = SelectType::student;

ItemSelect({}, 3);

system("cls");

WWC::ConsColor(15);

if (selectedItem == -1)

{

student stud = { "", "-1" };

return stud;

}

if (selectedItem == 'v')

return { "", "var" };

return stud\_items[selectedItem];

}

int stud\_edit(string stud\_id) {

st = SelectType::string;

cout << "Preparing...";

addMenuItem("Изменить данные о студенте");

addMenuItem("Изменить данные зачетной книжки студента");

addMenuItem("Удалить студента");

WWC::Cur2xy(0, 7);

WWC::ConsColor(15);

WwDB\* wwdb = new WwDB();

vector<stud\_score> scr = wwdb->get\_student\_score(stud\_id);

cout.width(85); cout.fill('\_'); cout << "\_" << endl;

cout.width(4); cout << left << "id";

cout.width(10); cout << left << "|\_Семестр";

cout.width(52); cout << left << "|\_Предмет";

cout.width(9); cout << left << "|\_Вид";

cout.width(10); cout << left << "|\_Оценка"; cout << "|" << endl;

cout.fill(' ');

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

if (i >= 1 && scr[i].sem != scr[i - 1].sem)

{

cout.fill('-');

cout.width(4); cout << "-";

cout.width(10); cout << "|-";

cout.width(52); cout << "|-";

cout.width(9); cout << "|-";

cout.width(10); cout << "|-"; cout << "|" << endl;

cout.fill(' ');

}

WWC::ConsColor(FOREGROUND\_INTENSITY);

cout.width(3); cout << left << scr[i].id;

WWC::ConsColor(15);

cout << " | ";

cout.width(8);

cout << left << scr[i].sem << "| ";

cout.width(50);

cout << left << scr[i].subj << "| ";

cout.width(7);

cout << ((scr[i].extype == ExamType::exam) ? "exam" : "zach") << "| ";

cout.width(8);

if (scr[i].value == '+') {

WWC::ConsColor(FOREGROUND\_GREEN);

cout << "Зачёт";

WWC::ConsColor(15);

cout << "|" << endl;

continue;

}

else if (scr[i].value == '-') {

WWC::ConsColor(12);

cout << "Незачёт";

WWC::ConsColor(15);

cout << "|" << endl;

continue;

}

else if (scr[i].value < 3)

WWC::ConsColor(12);

else if (scr[i].value >= 3 && scr[i].value < 4)

WWC::ConsColor(14);

else if (scr[i].value >= 4)

WWC::ConsColor(FOREGROUND\_GREEN);

cout << scr[i].value;

WWC::ConsColor(15);

cout << "|" << endl;

}

WWC::ConsColor(15);

WWC::Cur2xy(0, 0);

ItemSelect({ {2, {12, 0, FOREGROUND\_RED}} });

system("cls");

WWC::ConsColor(15);

if (selectedItem == -1)

return -1;

delete wwdb;

return selectedItem;

}

student EditStudInfoMenu(student stud) {

st = SelectType::EditField;

addMenuItem(make\_pair("Группа: ", stud.group));

addMenuItem(make\_pair("Фамилия: ", stud.surname));

addMenuItem(make\_pair("Имя: ", stud.name));

addMenuItem(make\_pair("Отчество: ", stud.middle\_name));

addMenuItem(make\_pair("Дата рождения: ", stud.bday));

addMenuItem(make\_pair("Год поступления: ", stud.admyear));

addMenuItem(make\_pair("Институт: ", stud.inst));

addMenuItem(make\_pair("Кафедра: ", stud.kaf));

string sex = ((stud.sex == Sex::man) ? "Мужской" : (stud.sex == Sex::woman ? "Женский" : "Боевой вертолет КА-52"));

addMenuItem(make\_pair("Пол: ", sex));

addMenuItem(make\_pair("\nСохранить", ""));

EditDataClass\* edc = new EditDataClass();

while (true) {

ItemSelect();

if (selectedItem == 9)

break;

if (selectedItem == -1) {

system("cls");

return {"", "-1"};

}

WWC::Cur2xy(0, selectedItem + 1);

cout << string2\_items[selectedItem].first; WWC::ConsColor(15); // cout << string2\_items[selectedItem].second;

WWC::ShowConsoleCursor(true);

editType et;

switch (selectedItem)

{

case 0:

et = editType::group;

break;

case 4:

et = editType::date;

break;

case 5:

et = editType::onlyDigit;

break;

case 8:

{

MenuClass\* sex\_sel = new MenuClass("");

string sex = sex\_sel->sexselect(5, selectedItem + 1);

delete sex\_sel;

string2\_items[selectedItem].second = sex;

system("cls");

continue;

}

case 7:

et = editType::all;

break;

default:

et = editType::onlyAlpha;

break;

}

edc->setData(string2\_items[selectedItem].second);

string data = edc->getData(et);

string2\_items[selectedItem].second = data;

system("cls");

}

delete edc;

stud.group = string2\_items[0].second;

stud.surname = string2\_items[1].second;

stud.name = string2\_items[2].second;

stud.middle\_name = string2\_items[3].second;

stud.bday = string2\_items[4].second;

stud.admyear = string2\_items[5].second;

stud.inst = string2\_items[6].second;

stud.kaf = string2\_items[7].second;

stud.sex = (string2\_items[8].second == "Мужской") ? Sex::man : (string2\_items[8].second == "Женский" ? Sex::woman : Sex::CombatHelicopter);

return stud;

}

stud\_score EditStudScoreMenu(string stud\_id) {

st = SelectType::string;

cout << "Preparing...";

WWC::ConsColor(15);

WwDB\* wwdb = new WwDB();

vector<stud\_score> scr = wwdb->get\_student\_score(stud\_id);

for (int i = 0; i < scr.size(); ++i)

addMenuItem("Семестр " + to\_string(scr[i].sem) + ": " + scr[i].subj);

addMenuItem("\b \b\nДобавить");

system("cls");

ItemSelect();

system("cls");

WWC::ConsColor(15);

if (selectedItem == -1)

return { -2, "-1" };

if (selectedItem == scr.size()) {

return { -1, stud\_id };

}

return scr[selectedItem];

}

stud\_score EditScoreInfoMenu(stud\_score stsc) {

st = SelectType::EditField;

addMenuItem(make\_pair("Семестр: ", to\_string(stsc.sem)));

addMenuItem(make\_pair("Предмет: ", stsc.subj));

addMenuItem(make\_pair("Тип оценивания: ", ((stsc.extype == ExamType::exam) ? "Экзамен" : "Зачёт")));

string oc;

if (stsc.extype == ExamType::exam)

oc = to\_string(stsc.value);

else if (stsc.value == '+')

oc = "Зачёт";

else

oc = "Незачёт";

addMenuItem(make\_pair("Оценка: ", oc));

addMenuItem(make\_pair("\nСохранить", ""));

EditDataClass\* edc = new EditDataClass();

while (true) {

ItemSelect({}, 2);

if (selectedItem == 'd') {

system("cls");

return { stsc.id, "ddd"};

}

if (selectedItem == 4)

break;

if (selectedItem == -1) {

system("cls");

return { -2 };

}

WWC::Cur2xy(0, selectedItem + 1);

cout << string2\_items[selectedItem].first; WWC::ConsColor(15);

WWC::ShowConsoleCursor(true);

editType et;

switch (selectedItem)

{

case 0:

et = editType::onlyDigit;

break;

case 1:

et = editType::all;

break;

case 2:

{

MenuClass\* ExTSel = new MenuClass("");

ExTSel->addMenuItem("Экзамен");

ExTSel->addMenuItem("Зачёт");

string2\_items[selectedItem].second = ExTSel->string\_items[ExTSel->GorItemSelect(16, selectedItem + 1)];

delete ExTSel;

if (string2\_items[2].second == "Зачёт")

string2\_items[3].second = "Незачёт";

else

string2\_items[3].second = "0";

system("cls");

continue;

}

case 3:

if (string2\_items[2].second == "Зачёт") {

MenuClass\* ZNZSel = new MenuClass("");

ZNZSel->addMenuItem("Зачёт");

ZNZSel->addMenuItem("Незачёт");

string2\_items[selectedItem].second = ZNZSel->string\_items[ZNZSel->GorItemSelect(8, selectedItem + 1)];

delete ZNZSel;

system("cls");

continue;

}

et = editType::onlyDigit;

break;

default:

et = editType::onlyAlpha;

break;

}

edc->setData(string2\_items[selectedItem].second);

string data = edc->getData(et);

string2\_items[selectedItem].second = data;

system("cls");

}

delete edc;

stsc.sem = stoi(string2\_items[0].second);

stsc.subj = string2\_items[1].second;

stsc.extype = (string2\_items[2].second == "Экзамен" ? ExamType::exam : ExamType::zach);

if (stsc.extype == ExamType::exam)

stsc.value = stoi(string2\_items[3].second);

else

stsc.value = string2\_items[3].second == "Зачёт" ? '+' : '-';

return stsc;

}

void GorDraw() {

for (int i = 0; i < count\_items; ++i) {

if (i == selectedItem) {

WWC::ConsColor(FOREGROUND\_GREEN);

cout << string\_items[i] << " ";

WWC::ConsColor(FOREGROUND\_INTENSITY);

}

else

cout << string\_items[i] << " ";

}

}

int GorItemSelect(int x = 0, int y = 0) {

WWC::Cur2xy(x, y);

GorDraw();

char ch = 0;

WWC::ShowConsoleCursor(false);

while (ch != 13) {

ch = \_getch();

if (ch == 8) {

selectedItem = -1;

break;

}

if (ch == 75 || ch == 97) // Left

{

selectedItem -= selectedItem > 0 ? 1 : 0;

WWC::Cur2xy(x, y);

GorDraw();

}

if (ch == 77 || ch == 100) { // Right

selectedItem += selectedItem < count\_items - 1 ? 1 : 0;

WWC::Cur2xy(x, y);

GorDraw();

}

Sleep(10);

}

return selectedItem;

}

string sexselect(int x = 0, int y = 0) {

addMenuItem("Мужской");

addMenuItem("Женский");

addMenuItem("Боевой вертолет КА-52");

GorItemSelect(x, y);

return string\_items[selectedItem];

}

};

Work\_with\_DB.h

#pragma once

#include "windows.h"

#include <iostream>

#include <fstream>

#include <string.h>

#include <set>

#include <conio.h>

#include <algorithm>

#include <stdio.h>

#include <io.h>

#include "Crypto.h"

using namespace std;

enum class Sex : char { man, woman, CombatHelicopter }; // А давай

enum class ExamType : char { exam, zach };

struct student {

string group;

string id;

string surname;

string name;

string middle\_name;

string bday = "01.01.1999";

string admyear = "1999"; // Год поступления

string inst;

string kaf;

Sex sex = Sex::CombatHelicopter;

};

struct stud\_score {

int id;

string stud\_id;

string subj = "";

ExamType extype = ExamType::exam;

short value = 0;

int sem = 1;

};

class WwDB {

private:

int stud\_count = 0;

int max\_scid = 0;

set<string> groups;

public:

void \_\_init\_\_() {

WWC::ShowConsoleCursor(false);

cout << "Loading and preparing...";

if (\_access("\_Score.bin", 0) != -1)

rename("\_Score.bin", "Score.bin");

if (\_access("\_Students.bin", 0) != -1)

rename("\_Students.bin", "Students.bin");

fstream students\_file("Students.bin", fstream::out | fstream::app | fstream::binary);

students\_file.close();

fstream stud\_score\_file("Score.bin", fstream::out | fstream::app | fstream::binary);

stud\_score\_file.close();

stud\_count = students\_count();

students\_file.open("Students.bin", fstream::in | fstream::binary);

string tmp;

while (!students\_file.eof()) {

getline(students\_file, tmp);

groups.insert(stud\_parse(tmp, 2).group);

}

groups.erase(groups.begin());

students\_file.close();

init\_max\_score\_id();

system("cls");

}

vector<string> get\_groups() {

vector<string> v(groups.begin(), groups.end());

return v;

}

void init\_max\_score\_id() {

fstream stsc\_file("Score.bin", fstream::in | fstream::binary);

string tmp;

int \_maxid = 0;

while (!stsc\_file.eof()) {

getline(stsc\_file, tmp);

\_maxid = stud\_score\_parse(tmp, 1).id;

if (\_maxid > max\_scid)

max\_scid = \_maxid;

}

stsc\_file.close();

}

int get\_max\_score\_id() {

return max\_scid;

}

int students\_count() {

fstream students\_file("Students.bin", fstream::in | fstream::binary);

string tmp;

int n = 0;

while (!students\_file.eof()) {

getline(students\_file, tmp);

if (tmp != "\n")

n++;

}

students\_file.close();

return n-1;

}

student stud\_parse(string st, int jsr = 0) {

st = Crypto::Decrypt(st);

string tmp;

bool fl = false;

student stout;

for (int i = 0, j = 0; i < st.length(); ++i) {

if (st[i] == '"') {

j++;

fl = j % 2;

if (fl)

tmp = "";

else {

switch (j)

{

case 2:

stout.group = tmp;

break;

case 4:

stout.id = tmp;

break;

case 6:

stout.surname = tmp;

break;

case 8:

stout.name = tmp;

break;

case 10:

stout.middle\_name = tmp;

break;

case 12:

stout.bday = tmp;

break;

case 14:

stout.admyear = tmp;

break;

case 16:

stout.inst = tmp;

break;

case 18:

stout.kaf = tmp;

break;

case 20:

stout.sex = (tmp == "man") ? Sex::man : (tmp == "woman" ? Sex::woman : Sex::CombatHelicopter);

break;

default:

break;

}

if (j > 0 && j == jsr)

return stout;

}

continue;

}

if (fl) {

tmp += st[i];

}

}

return stout;

}

stud\_score stud\_score\_parse(string st, int jsr = 0) {

st = Crypto::Decrypt(st);

string tmp;

bool fl = false;

stud\_score stout;

for (int i = 0, j = 0; i < st.length(); ++i) {

if (st[i] == '"') {

j++;

fl = j % 2;

if (fl) {

if (tmp != "" && j == 1)

stout.id = stoi(tmp);

tmp = "";

}

else {

switch (j)

{

case 2:

stout.stud\_id = tmp;

break;

case 4:

stout.subj = tmp;

break;

case 6:

stout.extype = (tmp == "exam") ? ExamType::exam : ExamType::zach;

break;

case 8:

stout.value = stoi(tmp);

break;

case 10:

stout.sem = stoi(tmp);

break;

default:

break;

}

}

continue;

}

if (j > 0 && j == jsr)

return stout;

if (fl) {

tmp += st[i];

}

else {

if (isdigit(st[i])) {

tmp += st[i];

}

}

}

return stout;

}

bool check\_student(student st) {

fstream stfile("Students.bin", fstream::in | fstream::binary);

string tmp;

while (!stfile.eof()) {

getline(stfile, tmp);

if (stud\_parse(tmp, 4).id == st.id) {

WWC::ErrOut("Egor: Студент с таким шифром уже существует!");

return false;

}

}

stfile.close();

return true;

}

void add\_student(student st) {

fstream stfile("Students.bin", fstream::out | fstream::app | fstream::binary);

string sex = (st.sex == Sex::man) ? "man" : (st.sex == Sex::woman ? "woman" : "CombatHelicopter");

//stfile << "{\"" << encryptDecrypt(st.group) << "\"\"" << encryptDecrypt(st.id) << "\"\"" << encryptDecrypt(st.surname) << "\"\"" << encryptDecrypt(st.name)

// << "\"\"" << encryptDecrypt(st.middle\_name) << "\"\"" << encryptDecrypt(st.bday) << "\"\"" << encryptDecrypt(st.admyear) << "\"\"" << encryptDecrypt(st.inst)

// << "\"\"" << encryptDecrypt(st.kaf) << "\"\"" << encryptDecrypt(sex) << "\"}\n";

stfile << Crypto::Encrypt("{\"" + st.group + "\"\"" + st.id + "\"\"" + st.surname + "\"\"" + st.name

+ "\"\"" + st.middle\_name + "\"\"" + st.bday + "\"\"" + st.admyear + "\"\"" + st.inst

+ "\"\"" + st.kaf + "\"\"" + sex + "\"}") << "\n";

stfile.close();

groups.insert(st.group);

stud\_count += 1;

}

void edit\_student(student st) {

fstream stfile("Students.bin", fstream::in | fstream::binary);

fstream tmp\_file("tmp.bin", fstream::out | fstream::binary);

int n = 0;

string tmp;

while (!stfile.eof()) {

getline(stfile, tmp);

student tmp\_st = stud\_parse(tmp);

if (tmp\_st.id == st.id) {

// замена данных

tmp\_st = st;

string sex = (st.sex == Sex::man) ? "man" : (st.sex == Sex::woman ? "woman" : "CombatHelicopter");

//tmp\_file << "{\"" << encryptDecrypt(st.group) << "\"\"" << encryptDecrypt(st.id) << "\"\"" << encryptDecrypt(st.surname) << "\"\"" << encryptDecrypt(st.name)

// << "\"\"" << encryptDecrypt(st.middle\_name) << "\"\"" << encryptDecrypt(st.bday) << "\"\"" << encryptDecrypt(st.admyear) << "\"\"" << encryptDecrypt(st.inst)

// << "\"\"" << encryptDecrypt(st.kaf) << "\"\"" << encryptDecrypt(sex) << "\"}\n";

tmp\_file << Crypto::Encrypt("{\"" + st.group + "\"\"" + st.id + "\"\"" + st.surname + "\"\"" + st.name

+ "\"\"" + st.middle\_name + "\"\"" + st.bday + "\"\"" + st.admyear + "\"\"" + st.inst

+ "\"\"" + st.kaf + "\"\"" + sex + "\"}") << "\n";

groups.insert(st.group);

while (!stfile.eof()) {

getline(stfile, tmp);

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stfile.close();

try {

rename("Students.bin", "\_Students.bin");

rename("tmp.bin", "Students.bin");

remove("\_Students.bin");

}

catch (exception e) {

WWC::ErrOut("Egor: Ошибка записи базы!");

}

return;

}

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stfile.close();

remove("tmp.bin");

return;

}

vector<student> get\_students\_by\_group(string \_group) {

vector<student> students;

fstream students\_file("Students.bin", fstream::in | fstream::binary);

string tmp;

while (!students\_file.eof()) {

getline(students\_file, tmp);

//cout << "ch = " << ch << endl;

student st = stud\_parse(tmp);

if (st.group == \_group)

students.push\_back(st);

}

students\_file.close();

if (students.size() == 0) {

groups.erase(\_group);

return students;

}

sort(students.begin(), students.end(), [](student& left, student right) {

return left.id < right.id;

});

return students;

}

vector<stud\_score> get\_student\_score(string \_stud\_id) {

fstream stsc\_file("Score.bin", fstream::in | fstream::binary);

string tmp;

vector<stud\_score> scores;

while (!stsc\_file.eof()) {

getline(stsc\_file, tmp);

stud\_score tmp\_ssc = stud\_score\_parse(tmp);

//cout << tmp\_ssc.stud\_id << " " << \_stud\_id << "\n";

if (tmp\_ssc.stud\_id == \_stud\_id) {

scores.push\_back(tmp\_ssc);

}

}

sort(scores.begin(), scores.end(), [](stud\_score& left, stud\_score& right) {

return left.sem < right.sem;

}); // Сортировка по семестру

stsc\_file.close();

return scores;

}

int add\_student\_score(stud\_score stud\_sc) {

fstream stud\_score\_file("Score.bin", fstream::out | fstream::app | fstream::binary);

string ex\_type = (stud\_sc.extype == ExamType::exam) ? "exam" : "zach";

//stud\_score\_file << stud\_sc.id << " " << stud\_sc.stud\_id << " " << stud\_sc.subj << " " << ex\_type << " " << stud\_sc.value << " endl ";

stud\_score\_file << Crypto::Encrypt("{" + to\_string(stud\_sc.id) + "\"" + stud\_sc.stud\_id + "\"\"" + stud\_sc.subj

+ "\"\"" + ex\_type + "\"\"" + to\_string(stud\_sc.value) + "\"\"" + to\_string(stud\_sc.sem) + "\"}") << "\n";

stud\_score\_file.close();

max\_scid++;

return stud\_sc.id;

}

void edit\_student\_score(stud\_score stud\_sc) {

fstream stud\_score\_file("Score.bin", fstream::in | fstream::binary);

fstream tmp\_file("tmp.bin", fstream::out | fstream::binary);

int n = 0;

string tmp;

//int id;

while (!stud\_score\_file.eof()) {

getline(stud\_score\_file, tmp);

stud\_score tmp\_ssc = stud\_score\_parse(tmp);

if (tmp\_ssc.id == stud\_sc.id) {

// замена данных

tmp\_ssc.extype = stud\_sc.extype;

tmp\_ssc.subj = stud\_sc.subj;

tmp\_ssc.value = stud\_sc.value;

string ex\_type = (tmp\_ssc.extype == ExamType::exam) ? "exam" : "zach";

cout << ex\_type;

tmp\_file << Crypto::Encrypt("{" + to\_string(stud\_sc.id) + "\"" + stud\_sc.stud\_id + "\"\"" + stud\_sc.subj

+ "\"\"" + ex\_type + "\"\"" + to\_string(stud\_sc.value) + "\"\"" + to\_string(stud\_sc.sem) + "\"}") << "\n";

while (!stud\_score\_file.eof()) {

getline(stud\_score\_file, tmp);

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_score\_file.close();

try {

rename("Score.bin", "\_Score.bin");

rename("tmp.bin", "Score.bin");

remove("\_Score.bin");

}

catch (exception e) {

WWC::ErrOut("Egor: Ошибка записи базы!");

}

return;

}

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_score\_file.close();

remove("tmp.bin");

return;

}

void delete\_score(int \_id) {

fstream stud\_score\_file("Score.bin", fstream::in | fstream::binary);

fstream tmp\_file("tmp.bin", fstream::out | fstream::binary);

int n = 0;

string tmp;

while (!stud\_score\_file.eof()) {

getline(stud\_score\_file, tmp);

stud\_score tmp\_ssc = stud\_score\_parse(tmp);

if (tmp\_ssc.id == \_id) {

while (!stud\_score\_file.eof()) {

getline(stud\_score\_file, tmp);

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_score\_file.close();

try {

rename("Score.bin", "\_Score.bin");

rename("tmp.bin", "Score.bin");

remove("\_Score.bin");

}

catch (exception e) {

WWC::ErrOut("Egor: Ошибка записи базы!");

}

return;

}

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_score\_file.close();

remove("tmp.bin");

return;

}

void delete\_students\_score(string stud\_id) {

fstream stud\_score\_file("Score.bin", fstream::in | fstream::binary);

fstream tmp\_file("tmp.bin", fstream::out | fstream::binary);

int n = 0;

string tmp;

while (!stud\_score\_file.eof()) {

getline(stud\_score\_file, tmp);

stud\_score tmp\_ssc = stud\_score\_parse(tmp, 2);

if (tmp\_ssc.stud\_id == stud\_id)

continue;

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_score\_file.close();

try {

rename("Score.bin", "\_Score.bin");

rename("tmp.bin", "Score.bin");

remove("\_Score.bin");

}

catch (exception e) {

WWC::ErrOut("Egor: Ошибка записи базы!");

}

remove("tmp.bin");

return;

}

bool delete\_student(string stud\_id) {

cout << "Deleting student, please wait...";

delete\_students\_score(stud\_id);

fstream stud\_file("Students.bin", fstream::in | fstream::binary);

fstream tmp\_file("tmp.bin", fstream::out | fstream::binary);

int n = 0;

string tmp;

while (!stud\_file.eof()) {

getline(stud\_file, tmp);

student tmp\_st = stud\_parse(tmp, 4);

if (tmp\_st.id == stud\_id)

continue;

tmp\_file << tmp << "\n";

}

tmp\_file.close();

stud\_file.close();

try {

rename("Students.bin", "\_Students.bin");

rename("tmp.bin", "Students.bin");

remove("\_Students.bin");

}

catch (exception e) {

WWC::ErrOut("Egor: Ошибка записи базы!");

return false;

}

remove("tmp.bin");

system("cls");

return true;

}

};

WorkWithConsole.h

#pragma once

#include <iostream>

#include <string.h>

#include "windows.h"

#include <conio.h>

#include <stdio.h>

using namespace std;

class WWC {

public:

static void Cur2xy(int xpos, int ypos)

{

COORD scrn;

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

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

SetConsoleCursorPosition(hOuput, scrn);

}

static void ConsColor(int ForC = 0, int BackC = 0) {

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), ((BackC & 0x0F) << 4) + (ForC & 0x0F));

}

static void ShowConsoleCursor(bool showFlag)

{

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

CONSOLE\_CURSOR\_INFO cursorInfo;

GetConsoleCursorInfo(out, &cursorInfo);

cursorInfo.bVisible = showFlag; // set the cursor visibility

SetConsoleCursorInfo(out, &cursorInfo);

}

static void ErrOut(string err) {

WWC::ConsColor(FOREGROUND\_RED);

std::cout << std::endl << err << "\n";

WWC::ConsColor(15);

}

};