СОДЕРЖАНИЕ

[АНОТАЦИЯ 4](#_Toc135317408)

[ВВЕДЕНИЕ 5](#_Toc135317409)

[1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 7](#_Toc135317410)

[1.1 Структурное программирование 7](#_Toc135317411)

[1.2 Процедурное программирование 7](#_Toc135317412)

[1.3 Модульное программирование 8](#_Toc135317413)

[1.4 Ссылки 9](#_Toc135317414)

[1.5 Работа с памятью. Указатели 9](#_Toc135317415)

[1.6 Работа с файлами 10](#_Toc135317416)

[1.7 Динамические структуры данных 11](#_Toc135317417)

[1.8 Линейный список 11](#_Toc135317418)

[1.9 Функции 11](#_Toc135317419)

[1.10 Объектно-ориентированное программирование 12](#_Toc135317420)

[1.11 Конструкторы и деструкторы 13](#_Toc135317421)

[1.12 Классы и дружественные функции 13](#_Toc135317422)

[1.12.1 Описание классов и их методов 14](#_Toc135317423)

[1.13 Методы шифрования и расшифровки 15](#_Toc135317424)

[2 ПРАКТИЧЕСКАЯ ЧАСТЬ 19](#_Toc135317425)

[2.1 Структура программы 19](#_Toc135317426)

[2.2 Алгоритм решения задачи 24](#_Toc135317427)

[2.3 Программная реализация задания 29](#_Toc135317428)

[2.3.1 Пример выполнения задания 29](#_Toc135317429)

[2.3.2 Отображение списка студентов 30](#_Toc135317430)

[2.3.3 Изменение данных о студенте 31](#_Toc135317431)

[2.3.4 Добавление информации о новом студенте 33](#_Toc135317432)

[2.3.5 Удаление данных о студенте 35](#_Toc135317433)

[2.3.6 Шифрование данных 36](#_Toc135317434)

[2.4 Руководство пользователя 38](#_Toc135317435)

[2.5 Системные требования 39](#_Toc135317436)

[ЗАКЛЮЧЕНИЕ 40](#_Toc135317437)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 41](#_Toc135317438)

[ПРИЛОЖЕНИЕ А Исходный код программы 42](#_Toc135317439)

АНОТАЦИЯ

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

Выполнил: Кузнецов Алексей Александрович, БАСО-04-22.

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

Задача курсовой работы: распечатать всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц определенного пола.

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. Структурное программирование

Структурное программирование - это методология создания и эксплуатации программного обеспечения ЭВМ, которая воплощает принципы системного подхода. Основные положения структурного программирования довольно просты: алгоритм и программа должны составляться поэтапно; сложная задача должна разбиваться на достаточно простые части, каждая из которых имеет один вход и один выход; логика алгоритма и программы должна опираться на минимальное число достаточно простых базовых управляющих структур.

Принципы структурного программирования включают в себя следующее: программа построена из трех базовых управляющих конструкций - последовательности, ветвления и цикла; в программе базовые управляющие конструкции вложены друг в друга произвольным образом; повторяющиеся фрагменты программы были оформлены в виде подпрограмм (процедур и функций); перечисленные конструкции должны иметь один вход и один выход; разработка программы ведется пошагово, методом “сверху вниз”.

1.2 Процедурное программирование

Процедурное программирование — это тип программирования на императивном языке, при котором последовательно выполняемые операторы можно собрать в подпрограммы, то есть более крупные целостные единицы кода, с помощью механизмов самого языка. Основным оператором процедурного программирования является присваивание, предназначенное для определения и изменения содержимого памяти компьютера.

Особенности процедурного программирования включают в себя следующее:

* 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;
* 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;
* 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;
* 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;
* Передача параметров — это механизм, используемый для передачи параметров в функции, подпрограммы или процедуры.

1.3 Модульное программирование

Модуль в программировании — это фрагмент кода, который имеет определенное функциональное значение и характеризуется логической завершенностью. Модульное программирование — это способ создания программы путем объединения модулей в единую структуру. В основе модульного программирования лежат четыре основные концепции:

* каждый модуль имеет единственную точку входа и выхода;
* размер модуля по возможности должен быть минимизирован;
* вся система построена из модулей;
* каждый модуль не зависит от того, как реализованы другие модули.

Модульное программирование 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 в программировании — это объект, указывающий на определенные данные, но не хранящий их. Получение объекта по ссылке называется разыменованием.

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

int a = 0;

int& a\_ref = a;

Физически a\_ref 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 в языке Си — это переменные, в которых записаны адреса других переменных. При объявлении указателя необходимо указать тип переменных, на которых он будет указывать, а перед именем поставить знак \*.

Знак & перед именем переменной обозначает ее адрес. Знак \* перед указателем в рабочей части программы (не в объявлении) обозначает значение ячейки, на которую указывает указатель.

Нельзя записывать по указателю, который указывает непонятно куда - это вызывает сбой программы, поскольку что-то стирается в памяти. Для обозначения недействительного указателя используется константа NULL.

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.

Указатель печатаются по формату %p.

1.6 Работа с файлами

Файл в программировании — 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++ можно использовать файловые потоки. В стандартной библиотеке определён заголовочный файл fstream, который определяет базовые типы для чтения и записи файлов. В частности это:

* ifstream для чтения информации из файла (оператор >>)
* ofstream для записи информации в файл (оператор <<)
* fstream, совмещающий функции записи и чтения (операторы >> и <<)

Для открытия файла используется функция open(). Она имеет две версии open(путь) и open(путь, режим). В функцию требуется передать путь к открываемому файлу в виде строки, а также при необходимости можно указать режим его открытия.

ofstream file; file.open(path, mode);

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

file.close();

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

rename(previous, current);

Для контроля достижения конца файла есть функция eof.

while (!file.eof()) {getline(file, line)}

1.7 Динамические структуры данных

В серьезных программах часто 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.

В таких случаях применяют данные особой структуры, которые представляют собой отдельные элементы, связанные с помощью ссылок. Например, для анализа текста и определения того, какие слова и в каком количестве в нем встречаются, причем эти слова нужно расставить по алфавиту, можно использовать структуру данных “дерево”.

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.

* Класс может определять различное состояние, различные функции. Однако не всегда желательно, чтобы к некоторым компонентам класса был прямой доступ извне. Для разграничения доступа к различным компонентам класса применяются спецификаторы доступа (private, public, protected). Инкапсуляция – сокрытие поведения объекта внутри него. Это ключевой принцип ООП;
* Наследование (inheritance) представляет один из ключевых аспектов объектно-ориентированного программирования, который позволяет наследовать функциональность одного класса (базового класса) в другом - производном классе (derived class).
* Полиморфизм — это свойство программного кода изменять свое поведение в зависимости от ситуации, возникающей при выполнении программы. Как правило, полиморфизм возникает, когда существует иерархия классов, и они связаны наследованием. Полиморфизм C ++ означает, что вызов функции-члена вызовет выполнение другой функции в зависимости от типа объекта, который вызывает эту функцию.

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.

1.11 Конструкторы и деструкторы

Конструктор — это функция-член класса, которая инициализирует переменные-члены и распределяет память для их хранения. Она имеет то же имя, что и имя класса. Конструктор запускается для каждого создаваемого объекта. Он не имеет возвращаемого значения и не принимает аргументов. Конструкторы относятся к интерфейсу класса, чтобы с их помощью можно было создавать объекты данного класса из внешней части программы.

Деструктор — это функция-член класса, которая предназначена для уничтожения переменных. Она имеет имя, которое представляет собой имя класса с тильдой (~) в начале. Деструктор вызывается автоматически при уничтожении объекта.

Таким образом, конструктор и деструктор являются методами класса в C++. Конструктор облегчает инициализацию полей при создании объекта класса. Деструктор же удаляет все ресурсы, которые были выделены объекту.

1.12 Классы и дружественные функции

Класс — это составной тип данных в С++, который может использовать другие типы данных. Классы и объекты 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 — это 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 — это переменная, описание которой создает программист при создании класса. Все данные объекта хранятся в его полях. Поля могут иметь спецификаторы доступа, такие как:

* private – поля и методы, доступные только внутри самого класса, в котором они объявлены;
* protected – поля и методы, доступные внутри класса, в котором они объявлены, а также в производных от него классах;
* public – поля и методы, доступные из любого места программы;

Если для каких-то компонентов отсутствует спецификатор доступа, то по умолчанию применяется спецификатор private.

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 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 программирования С++ описывают методы и свойства, которые могут быть использованы для создания объектов. Объекты — это конкретное представление класса, которое имеет свои свойства и методы. Свойства — это данные, которые могут быть использованы для характеристики объекта класса. Методы — это функции, которые могут выполнять различные действия над данными (свойствами) класса.

Поле класса в объектно-ориентированном программировании — это 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.

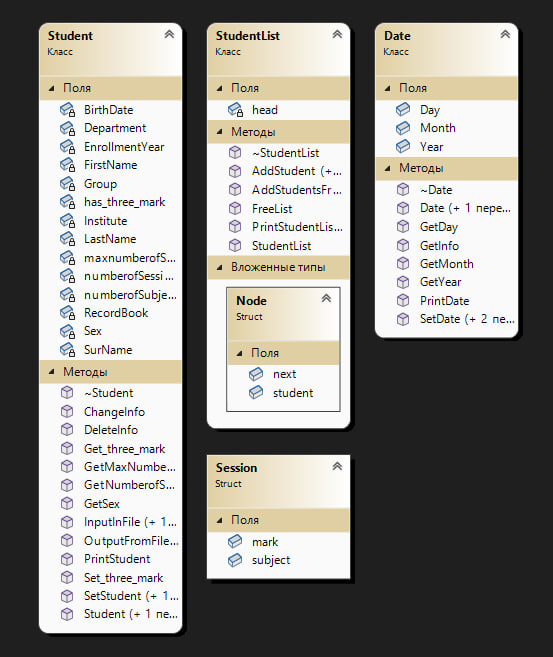


Рисунок 1 — Диаграмма классов

1.13 Методы шифрования и расшифровки

Шифрование — это процесс кодирования получаемых, отправляемых или хранимых данных с целью предотвращения несанкционированного доступа к ним.

Расшифровка — это процесс, обратный процессу шифрования, который позволяет получить исходную информацию из зашифрованного сообщения.

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

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

1) генерацию случайного пароля;

2) шифрование алгоритмом AES базы данных;

3) удаление не зашифрованной базы данных;

4) запись пароля в файл;

5) шифрование алгоритмом RSA файла с паролем с помощью публичного ключа;

6) удаление не зашифрованного файла с паролем.

Листинг 1.13.1

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | srand(time(NULL));  string pass;  for (int i = 0; i < 128; ++i) {  switch (rand() % 3) {  case 0:  pass += rand() % 10 + '0';  break;  case 1:  pass += rand() % 26 + 'A';  break;  case 2:  pass += rand() % 26 + 'a';  }  } |

Листинг 1.13.2

|  |  |
| --- | --- |
| 1  -  2  3  4 | string command = "OpenSSL\\bin\\openssl.exe enc -aes-256-cbc -salt -in DB.txt -out DB.txt.enc -pass pass:";  command += pass;  command += " -iter 1000";  system(command.c\_str()); |

Листинг 1.13.3

|  |  |
| --- | --- |
| 1  2  -  3 | if (remove("DB.txt") != 0) {  cout << "[ОШИБКА 0021091] - Не удалось удалить файл с базой." << endl;  } |

Листинг 1.13.4

|  |  |
| --- | --- |
| 1  2  3  4 | ofstream filein;  filein.open("key.bin", ios::binary);  filein << pass;  filein.close(); |

Листинг 1.13.5

|  |  |
| --- | --- |
| 1  -  2 | command = "OpenSSL\\bin\\openssl.exe rsautl -encrypt -inkey rsa.public -pubin -in key.bin -out key.bin.enc";  system(command.c\_str()); |

Листинг 1.13.6

|  |  |
| --- | --- |
| 1  2  -  3 | if (remove("key.bin") != 0) {  cout << "[ОШИБКА 0021092] - Не удалось удалить файл с ключом." << endl;  } |

Для реализации функции Decrypt поэтапно реализуем следующее:

1) Расшифровка зашифрованного файла с паролем с помощью приватного ключа и RSA;

2) Удаление зашифрованного файла с ключом;

3) Считывание ключа из файла;

4) Удаление файла с ключом;

5) Расшифровка зашифрованной базы данных с помощью ключа и алгоритма AES;

6) Удаление зашифрованной базы данных.

Листинг 1.13.7

|  |  |
| --- | --- |
| 1  -  2 | string command = "OpenSSL\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.bin.enc -out key.bin";  system(command.c\_str()); |

Листинг 1.13.8

|  |  |
| --- | --- |
| 1  2  -  3 | if (remove("key.bin.enc") != 0) {  cout << "[ОШИБКА 0021093] - Не удалось удалить файл с зашифрованным ключом." << endl;  } |

Листинг 1.13.9

|  |  |
| --- | --- |
| 1  2  3  4  5 | string pass;  ifstream fileout;  fileout.open("key.bin", ios::binary);  fileout >> pass;  fileout.close(); |

Листинг 1.13.10

|  |  |
| --- | --- |
| 1  2  -  3 | if (remove("key.bin") != 0) {  cout << "[ОШИБКА 0021094] - Не удалось удалить файл с ключом." << endl;  } |

Листинг 1.13.11

|  |  |
| --- | --- |
| 1  -  2  3  4 | command = "OpenSSL\\bin\\openssl.exe enc -aes-256-cbc -d -in DB.txt.enc -out DB.txt -pass pass:";  command += pass;  command += " -iter 1000";  system(command.c\_str()); |

Листинг 1.13.12

|  |  |
| --- | --- |
| 1  2  -  3 | if (remove("DB.txt.enc") != 0) {  cout << "[ОШИБКА 0021095] - Не удалось удалить файл с зашифрованной базой." << endl;  } |

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Структура программы

Исходными данными для программы является информация о группе студентов из 14 человек, где запись о студенте содержит следующие данные (таблица 2.1.1).

Таблица 2.1.1 – Структура и типы данных

|  |  |
| --- | --- |
| Основная структура данных | Типы данных |
| Фамилия | string (строковый) |
| Имя | string (строковый) |
| Отчество | string (строковый) |
| Пол | string (строковый) |
| Число, месяц, год рождения | Date (дата) |
| Год поступления в институт | int (целочисленный) |
| Институт | string (строковый) |
| Кафедра | string (строковый) |
| Группа | string (строковый) |
| Номер зачетной книжки | string (строковый) |
| Кол-во сданных сессий | int (целочисленный) |
| Кол-во предметов в сессии | int (целочисленный) |
| Макс. кол-во предметов среди всех сессий | int (целочисленный) |
| Отметка троечника | bool (логический) |

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 2.1.2).

Таблица 2.1.2 – Классы

|  |  |
| --- | --- |
| Класс | Назначение |
| Date | Пользовательский тип данных, хранит день, месяц и год. |
| Session | Содержит в себе оценку и название предмета. |
| Student | Содержит в себе информацию о студенте: пол, номер зачетной книжки, сведения о фамилии, имени, отчестве, дате рождения, годе поступления, группе, институте, кафедре, кол-ве сессий/предметов, отметке троечника. |
| StudentList | Содержит в себе список студентов, представленный в виде объектов структуры Node. |
| Класс | Назначение |
| Node | Вспомогательная структура для StudentList, объекты которой представляют из себя элементы списка. |

Для разработки данного проекта понадобятся следующие классы. В таблице ниже представлены и описаны поля и методы класса, используемые в проекте (таблица 2.1.3).

Таблица 2.1.3 – Поля классов

|  |  |  |
| --- | --- | --- |
| Классы/Структуры | Поле класса | |
| Тип данных | Название и характеристика |
| Date | int | Day – день,  Month – месяц  Year – год |
| Student | string | LastName – фамилия  FirstName – имя  SurName – отчество Institute — институт Department — кафедра Group — группа  RecordBook — номер зачетной книжки Sex — пол |
| Date | BirthDate – дата рождения |
| int | EnrollmentYear — год поступления numberofSessions — кол-во сессий numberofSubjects — кол-во предметов в сессии maxnumberofSubjects — макс. кол-во предметов среди всех сессий |
| bool | has\_three\_mark — отметка, является ли студент троечником |
| Session | string | subject – наименование предмета |
| mark – оценка по предмету |
| StudentList | Node\* | head – указатель на первый элемент списка |
| Node | Student\* | student – хранимая информация о студенте в элементе списка |
| Node\* | next – указатель на следующий элемент списка |

Далее методы классов, представленные в таблице 2.1.4.

Таблица 2.1.4 Методы классов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Date | Date | Конструктор по умолчанию | - | - |
| Date | Конструктор с указанием значений полей класса | \_Day, \_Month, \_Year (int) | - |
| SetDate | Указание значений даты | \_Day, \_Month, \_Year (int) | - |
| SetDate | Указание значений даты | Date (Date) | - |
| SetDate | Указание значений даты с клавиатуры | - | - |
| GetDay | Получить значение Day | - | int |
| GetMonth | Получить значение Month | - | int |
| GetYear | Получить значение Year | - | int |
| GetInfo | Получить значение Day/Month/Year | \_info (string) | int |
| PrintDate | Вывести на экран значения Day, Month, Year | - | - |
| ~Date | Обозначение деструктора для лучшей читаемости кода | - | - |
| Student | Student | Конструктор по умолчанию | - | - |
| Student | Конструктор с указанием значений полей класса | \_LastName, \_FirstName, \_SurName, \_BirthDate, \_EnrollmentYear, \_Institute, \_Department, \_Group, \_RecordBook, \_Sex (string, Date, int) | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Student | SetStudent | Указание значений полей класса | \_LastName, \_FirstName, \_SurName, \_BirthDate, \_EnrollmentYear, \_Institute, \_Department, \_Group, \_RecordBook, \_Sex (string, Date, int) | - |
| SetStudent | Указание значений полей класса c клавиатуры | - | - |
| Set\_three\_mark | Отметить студента-троечника | is\_mark (bool) | - |
| InputInFile | Добавить значения полей класса в файл | - | - |
| InputInFile | Добавить значения полей класса в файл | \_filename[] (const char) | - |
| OutputFromFile | Считать информацию из файла о конкретном студенте | \_filename[] (const char) | - |
| OutputFromFile | Считать информацию из файла о конкретном студенте | \_filename[], o (const char, int) | - |
| DeleteInfo | Удалить информацию о студенте из файла | \_filename[] (const char) | - |
| ChangeInfo | Изменить информацию о студенте в файле | \_filename[] (const char) | - |
| PrintStudent | Вывод в консоль информации о студенте | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| Student | GetSex | Возвращает значение пола | - | string |
| Get\_three\_mark | Возвращает значение отметки троечника | - | bool |
| GetNumberofSessions | Возвращает кол-во сессий | - | int |
| GetMaxNumberofSubjects | Возвращает максимальное кол-во предметов среди всех сессий | - | int |
| ~Student | Обозначение деструктора для лучшей читаемости кода | - | - |
| StudentList | StudentList | Конструктор по умолчанию. Указатель на первый элемент = nullptr | - | - |
| ~StudentList | Деструктор полностью очищает список | - | - |
| AddStudent | Добавляет студента как элемент списка | student (Student&)  (ссылка на объект класса) | - |
| AddStudent | Добавляет студента как элемент списка | student (Student\*)  (указатель на объект класса) | - |
| AddStudentsFromFile | Заполняет список объектами класса Student | \_filename[] (const char) | - |
| PrintStudentList | Выводит в консоль все элементы списка (информацию студентов) | - | - |

Продолжение таблицы 2.1.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы  (их тип) | Тип возвращаемого значения |
| StudentList | PrintStudentList | Выводит в консоль информацию студентов без троек и с разделением по полу (вариант 34) | \_sex (string) | - |
| FreeList | Полностью очищает список | - | - |
| - | Check | Проверяет, есть ли цифры в строке | \_word (string) | bool |
| - | Clear | Очищает буфер | - | - |
| - | HowMuchLinesInFile | Возвращает кол-во строк в файле | \_filename[] (const char) | int |
| - | menu | Вызывает главное меню программы | - | int |
| - | Crypt | Шифрует информацию файла базы данных | - | - |
| - | Decrypt | Расшифровывает информацию в файле | - | - |

2.2 Алгоритм решения задачи

Задача: распечатать всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц определенного пола. Вариант 34.

Описание алгоритма выполнения задания:

1. Создается список студентов, заносимых из файла. Сессии каждого студента не заносятся в список — вместо этого идет проверка на наличие оценки “3” хотя бы по одному предмету среди всех сессий. Если “тройка” есть, то логическое поле студента “has\_three\_mark” принимает значение ИСТИНА.

2. Пользователь выбирает, студентов какого пола необходимо вывести в консоль.

3. Каждый элемент списка проходит проверку:

1. Пол студента должен совпадать с введенным.

2. Логическое поле студента “has\_three\_mark” должно иметь значение ЛОЖЬ (по умолчанию).

Если оба требования выполняются, то в консоль выводится информация о данном студенте.

4. Вышеперечисленные пункты повторяются для каждого элемента в списке.

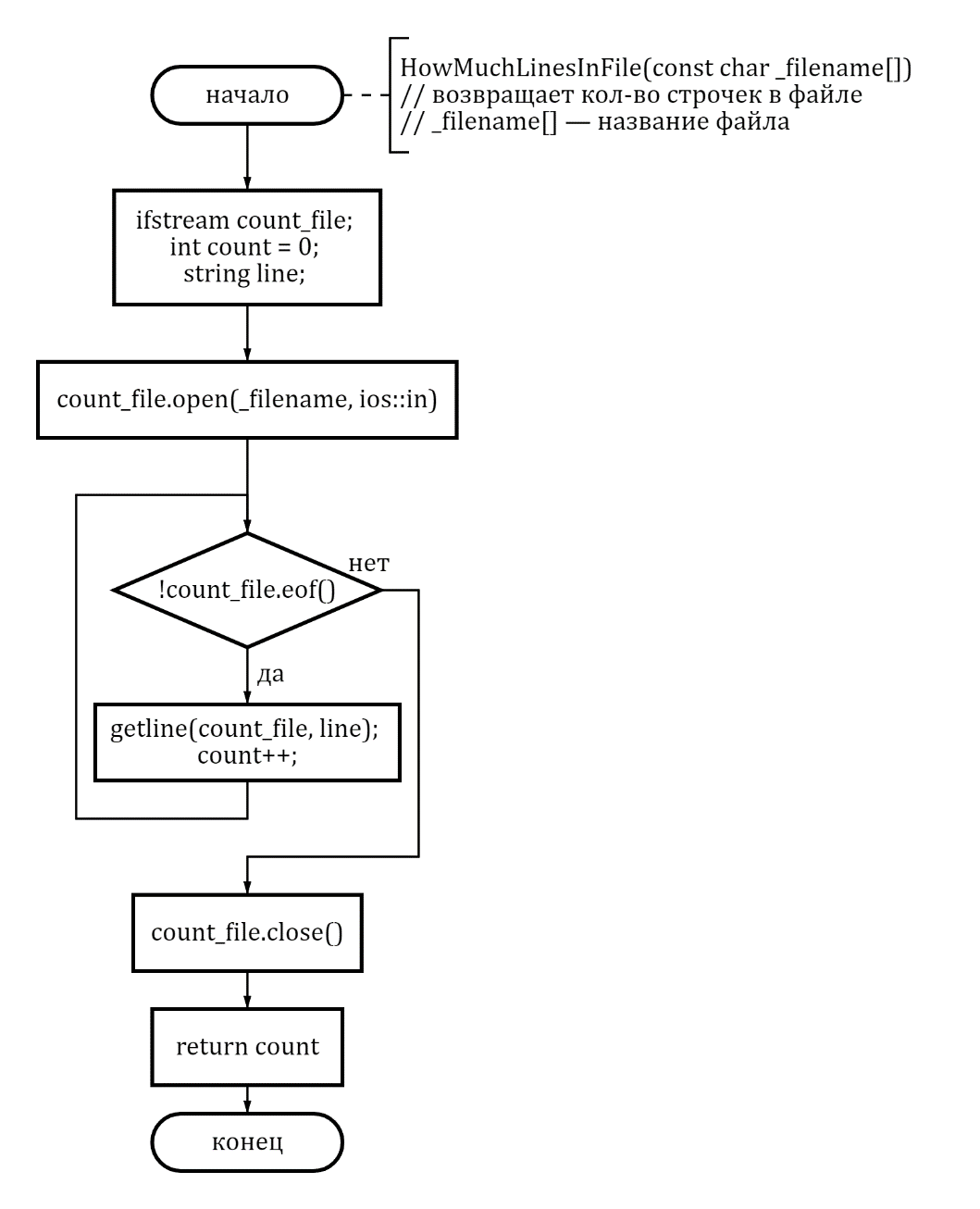


Рисунок 2.1 — Схема алгоритма подсчета количества строк в файле

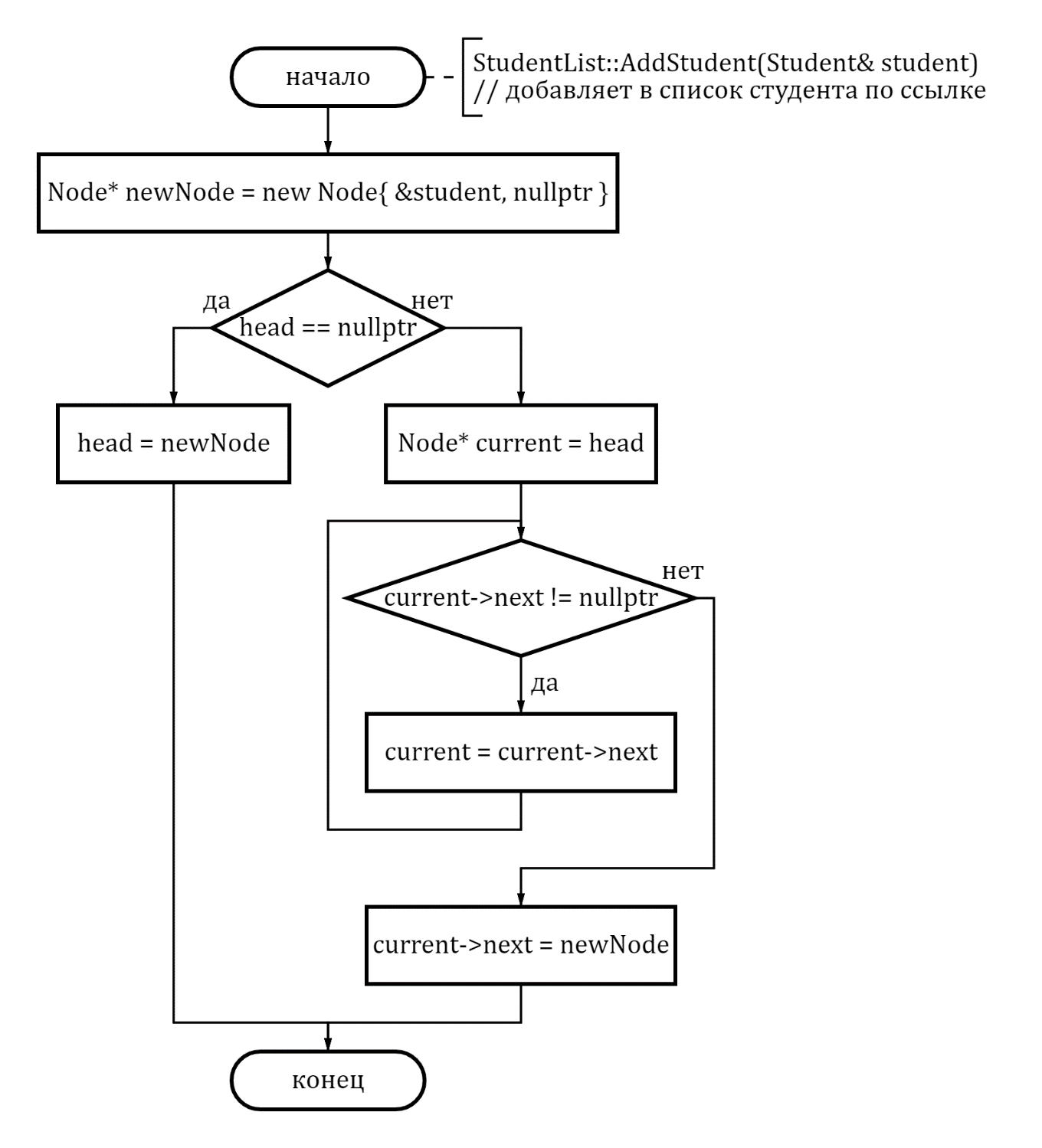


Рисунок 2.2 — Схема алгоритма добавления студента в конец списка

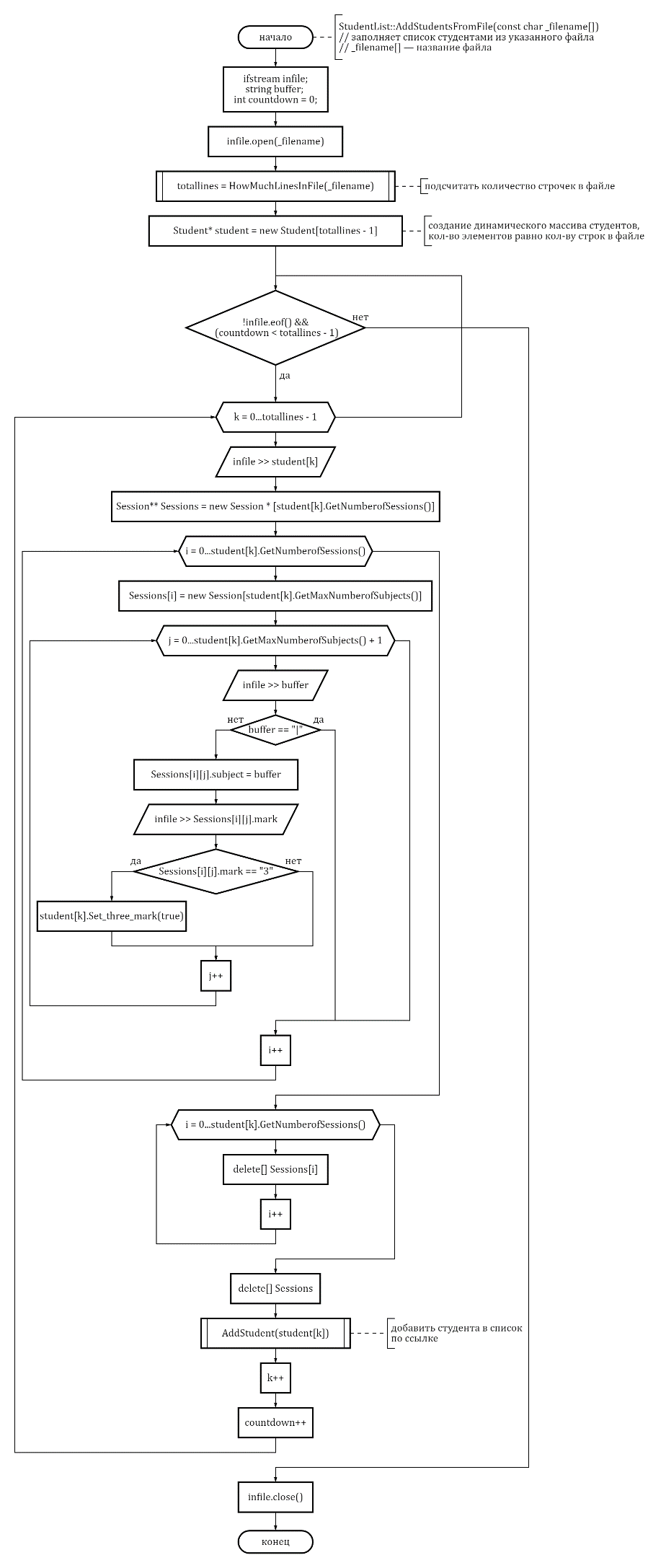


Рисунок 2.3 — Схема алгоритма заполнения списка студентов из файла

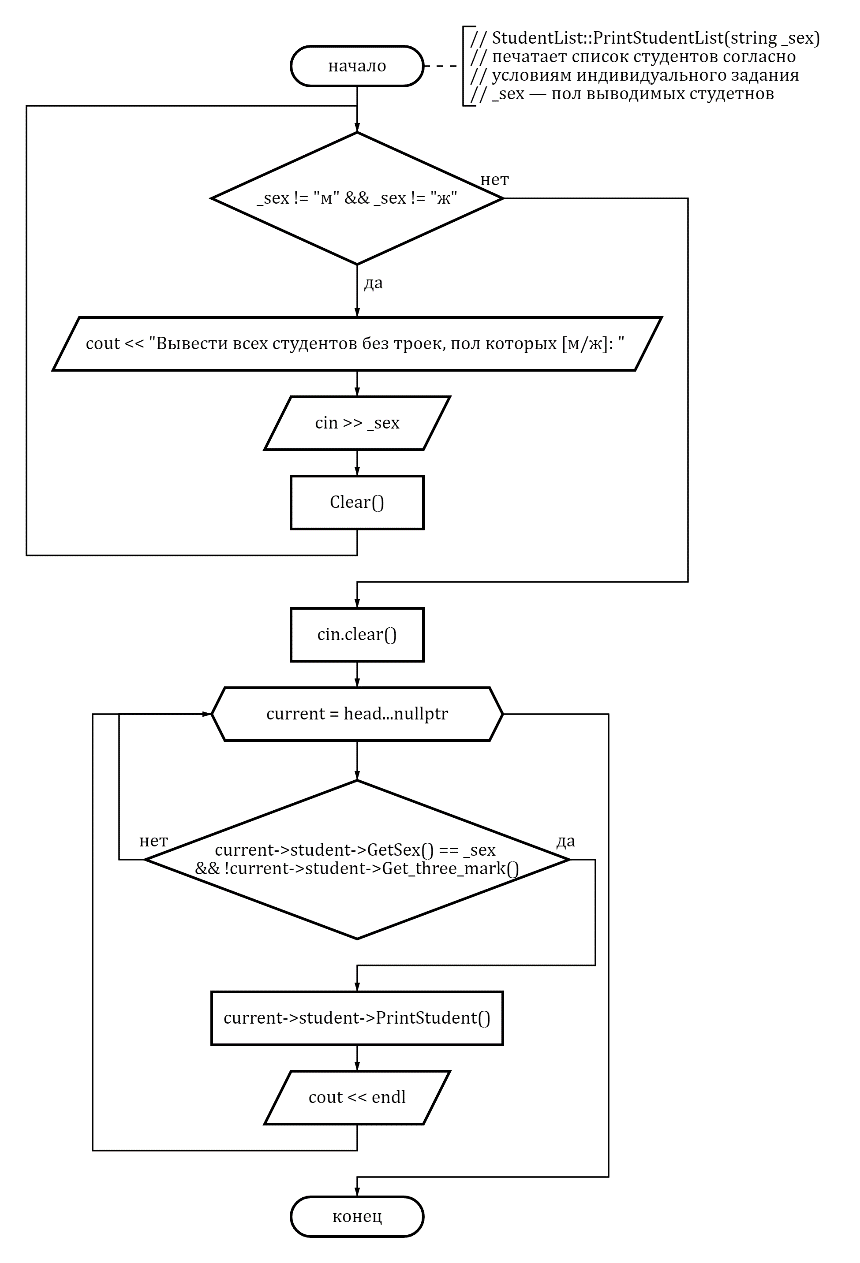


Рисунок 2.4 — Схема алгоритма печати списка студентов согласно условиям индивидуального задания

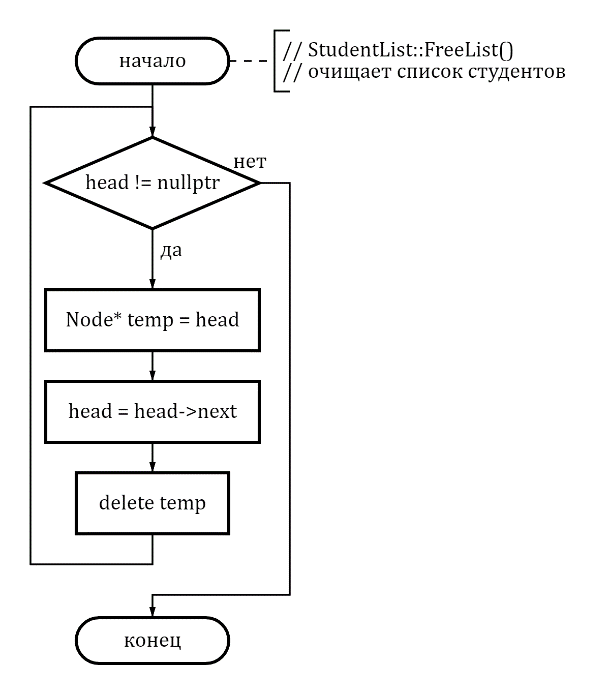


Рисунок 2.5 — Схема алгоритма очистки списка студентов

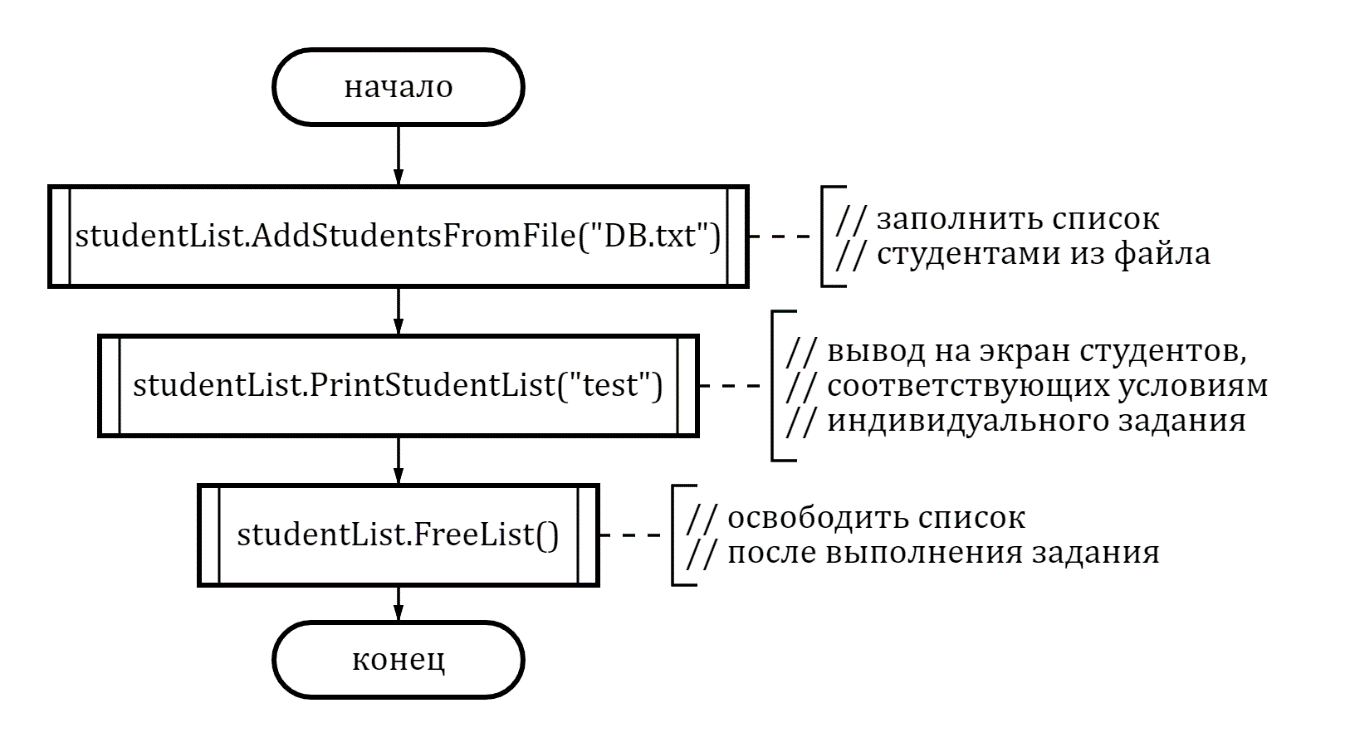


Рисунок 2.6 — Схема алгоритма выполнения индивидуального задания

2.3 Программная реализация задания

2.3.1 Пример выполнения задания

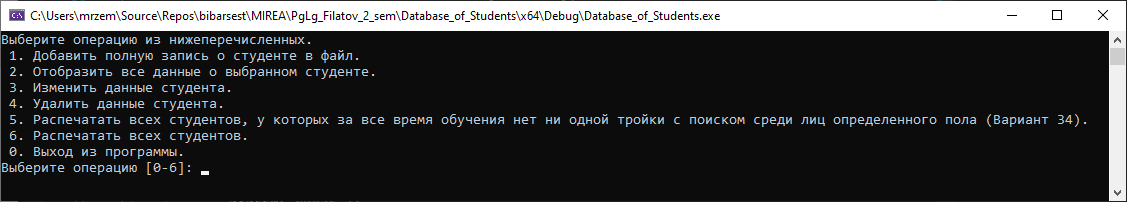


Рисунок 3 — Главное меню

Для того, чтобы выполнить задание по варианту, необходимо в главном меню ввести значение “5”. После этого программа предложит выбрать, студентов какого пола необходимо вывести на экран.

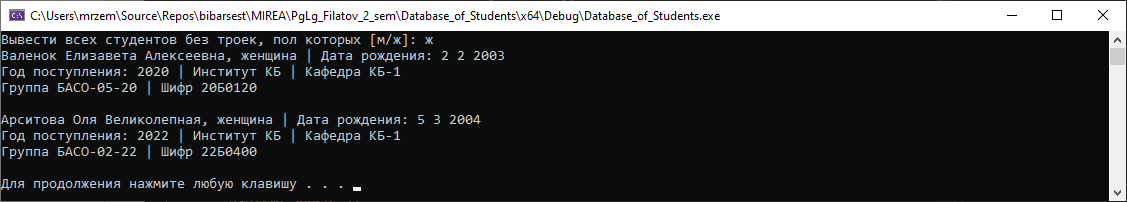


Рисунок 4 — Список студентов, подходящих по требованиям

Программа отработала и вывела студентов, подходящих по требованиям. Проверим, действительно ли у них нет троек.

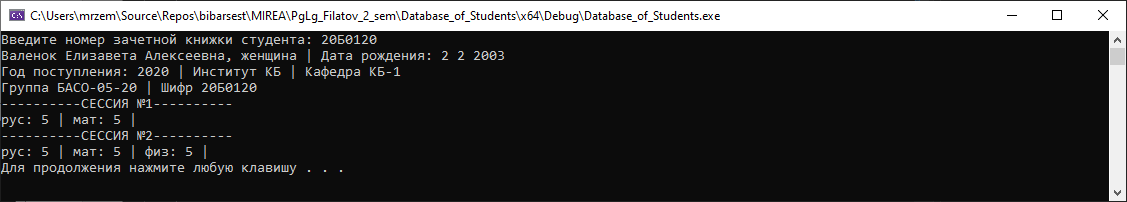


Рисунок 5 — Полная информация о студенте

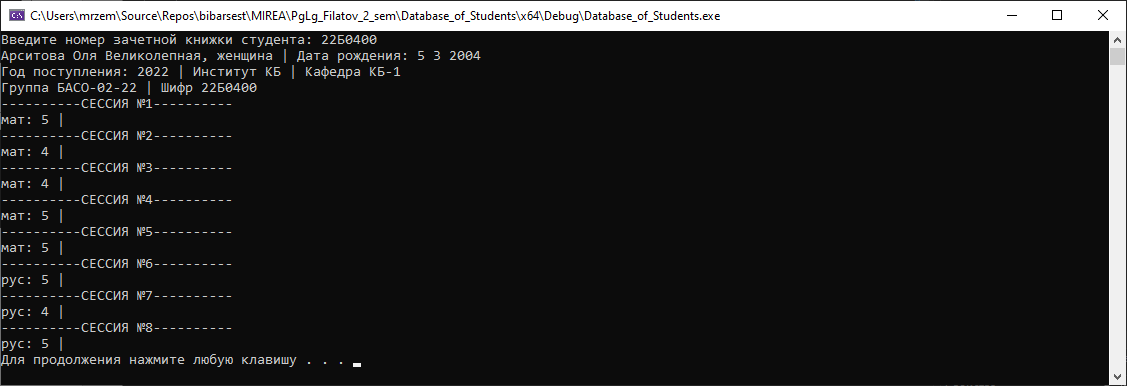


Рисунок 6 — Полная информация о втором студенте

Действительно, у выведенных студентов нет троек за весь период обучения.

2.3.2 Отображение списка студентов

Для отображения списка студентов необходимо ввести “6” в главном меню. Выводится информация полей класса Student. Для печати сессий необходимо обращаться к конкретному студенту по номеру зачетной книжки.

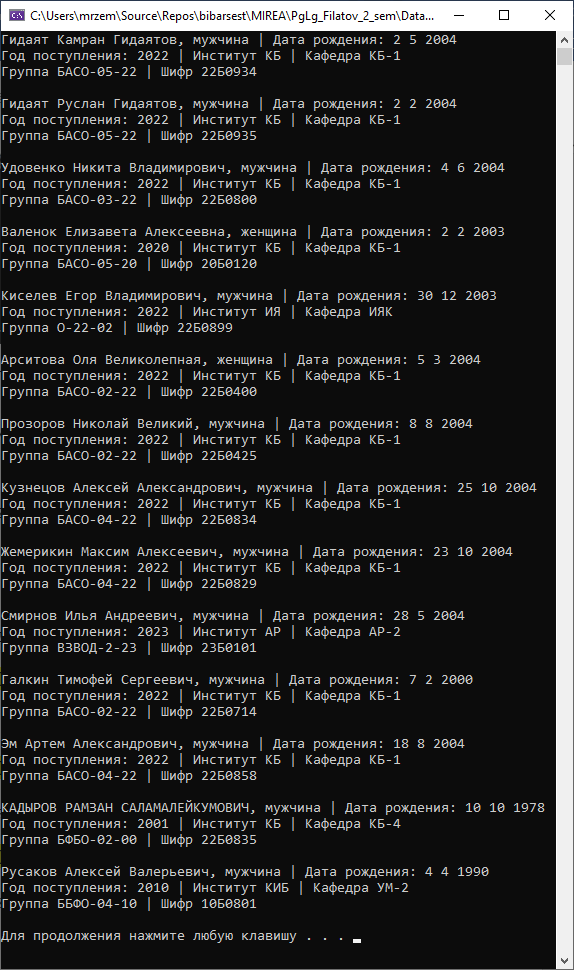


Рисунок 7 — Список студентов

2.3.3 Изменение данных о студенте

Для изменения данных студента необходимо ввести “3”, находясь в главном меню программы. Далее нужно ввести номер зачетной книжки студента, информацию которого надо изменить. Если в базе нашелся студент с введенным номер зачетной книжки, то выводится его полная информация вместе с сессиями. Открывается меню изменения данных студента.

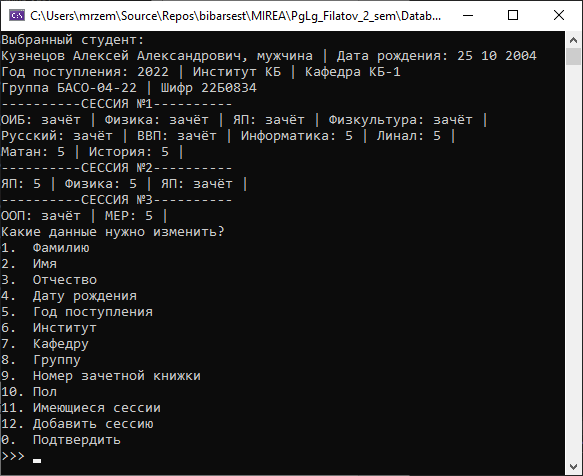


Рисунок 8 — Меню изменения данных студента

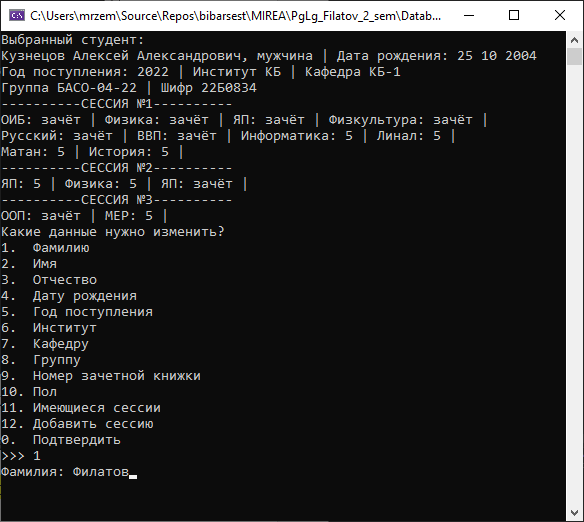


Рисунок 9 — Изменение фамилии студента

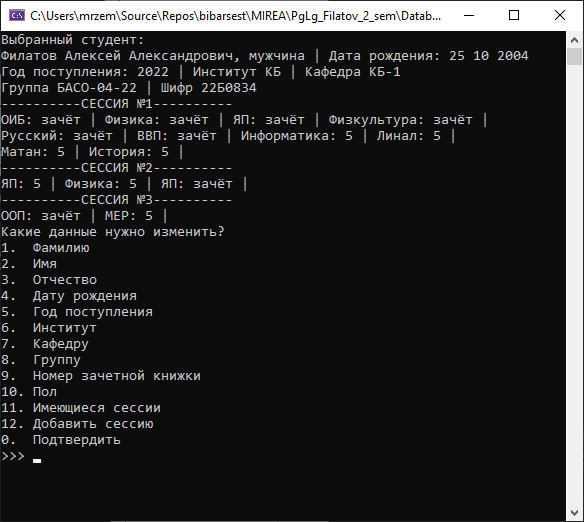


Рисунок 10 — Отображение изменений в меню

После изменения тех или иных данных студента, новая информация отображается во все том же меню. Для завершения редактирования необходимо ввести “0” — сохранение изменений и переход в главное меню.

2.3.4 Добавление информации о новом студенте

Для добавления нового студента необходимо ввести “1”, находясь в главном меню программы. Поэтапно программа предложит ввести фамилию, имя, отчество, дату рождения, год поступления, институт, кафедру, группу, номер зачетной книжки, пол, сессии. При этом действуют ограничения. Например, при вводе номера зачетной книжки, если он уже имеется в базе, то программа потребует ввести другой номер.

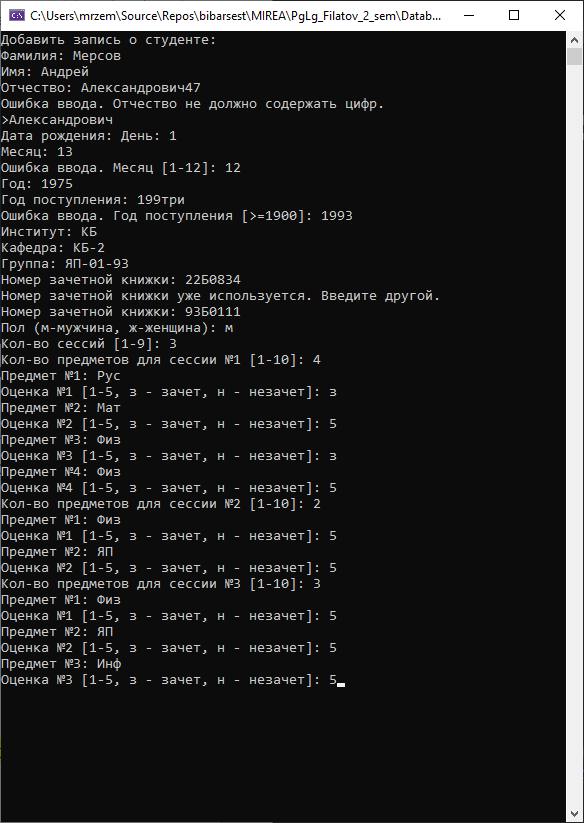


Рисунок 11 — Добавление информации о новом студенте

После ввода всех необходимых значений программа выводит все данные нового студента и запрашивает подтверждение на их запись в файл.

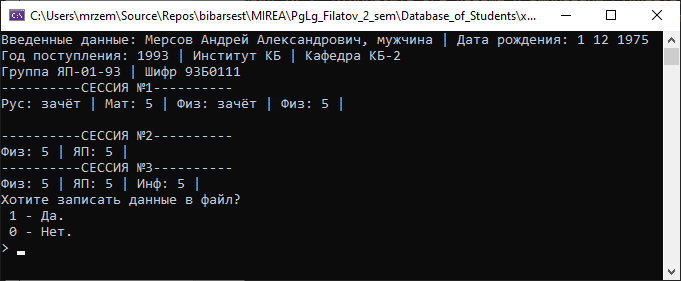


Рисунок 12 — Подтверждение записи информации нового студента

При помощи пункта меню, описанного в 2.3.2, проверим, действительно ли студент появился в базе.

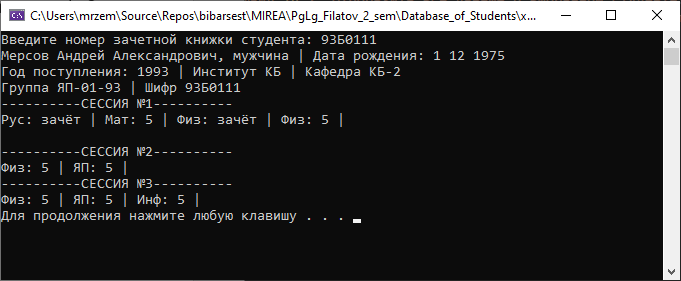


Рисунок 13 — Студент добавлен в базу

2.3.5 Удаление данных о студенте

Для удаления студента из базы необходимо ввести “4”, находясь в главном меню программы. Программа потребует ввести номер зачетной книжки студента, которого нужно удалить.

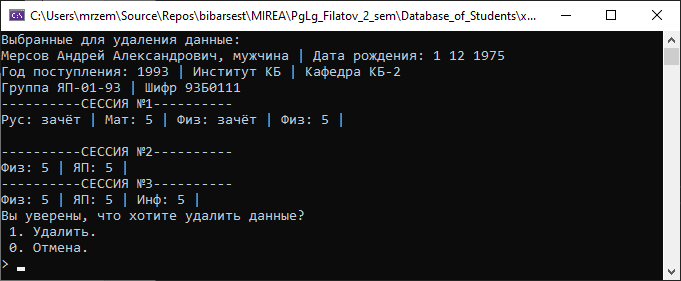


Рисунок 14 — Предпросмотр удаляемых данных

Программа показывает данные студента, которые будут удалены после подтверждения путем ввода “1” в консоль.

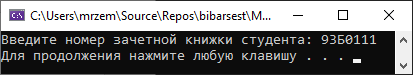


Рисунок 15 — Данные удалены

При помощи пункта меню, описанного в 2.3.2, мы удостоверились, что, действительно, студента нет в базе.

2.3.6 Шифрование данных

Шифрование и расшифровка происходят, соответственно, в момент запуска и закрытия программы. Когда пользователь запускает программу, перед тем, как вывести главное меню, вызывается функция Decrypt(), расшифровывающая файл с информацией о студентах. После работы программы, когда пользователь вводит “0” для выхода, файл шифруется при помощи функции Crypt(). Процесс шифрования/расшифровки описан в теоретической части, п. 1.13.

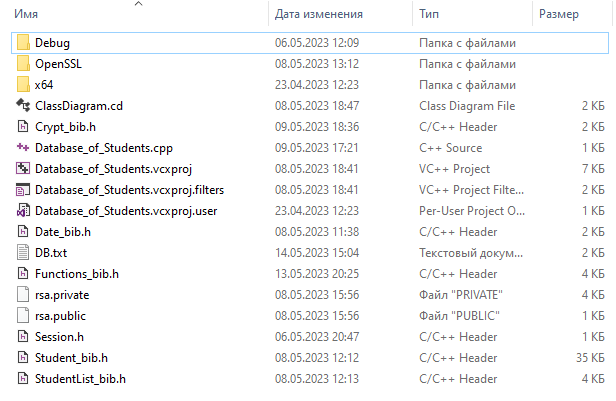


Рисунок 16 — Файловая база в момент работы программы (данные не зашифрованы)

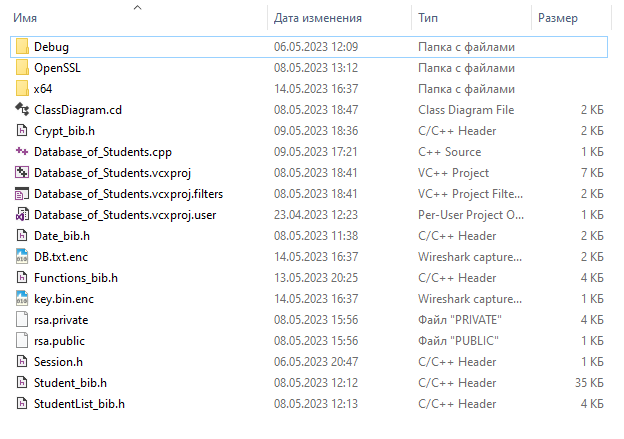


Рисунок 17 — Файловая база после отработки программы (данные зашифрованы)

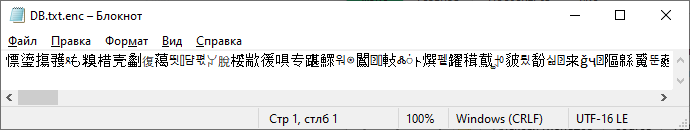


Рисунок 18 — База данных студентов в зашифрованном виде

2.4 Руководство пользователя

При запуске программы на экран выводится консоль с меню, где пользователь может выбрать один из нескольких пунктов, таких как: добавление полной записи о студенте в базу, отображение всех данных конкретного студента по номеру зачетной книжки, изменение данных конкретного студента, удаление информации о студенте из базы, выборка из базы студентов без троек с указанием пола, печать представленных в базе студентов, выход. В зависимости от выбранного пользователем варианта выполняется определенная часть программы.

При вводе “0” осуществляется выход из программы и шифрование файловой базы данных.

При вводе “1” происходит добавление в файл нового студента и его данных с клавиатуры пользователем.

При вводе “2” программа запрашивает номер зачетной книжки, после чего, если студент с таким номером представлен в файле, в консоль выводятся сведения о данном студенте.

При вводе “3” программа запрашивает номер зачетной книжки, после чего, если студент с таким номером представлен в файле, пользователь в праве корректировать данные выбранного студента и после сохранить их в файл.

При вводе “4” программа запрашивает номер зачетной книжки, после чего, если студент с таким номером представлен в файле, программа выводит в консоль данные, которые будут удалены из файла после подтверждения пользователя.

При вводе “5” программа запрашивает пол, чтобы вывести на экран всех студентов данного пола, подходящих требованиям задания — не имеют ни одной тройки за весь период обучения.

При вводе “6” программа выводит на экран всех студентов, представленных в файле.

2.5 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, распечатывающая всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц определенного пола.

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. 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. 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)
3. Страуструп Б. Программирование. Принципы и практика использования C++. – Litres, 2022.
4. Лафоре Р. Объектно-ориентированное программирование в C++. 4-е изд., перераб. и доп //Санкт-Петербург.: Питер. – 2022.
5. Стенли Липпман Язык программирования С++: полное руководство / Липпман Стенли, ЛажойеЖози. – Саратов: Профобразование, 2023. 1104 c.
6. Страуструп Б. Дизайн и эволюция языка С++. – Litres, 2022.
7. Уильямс Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ. – Litres, 2022.
8. Аммерааль Л. STL для программистов на C++. – Litres, 2022.
9. Дейл Н., Уимз Ч., Хедингтон М. Программирование на C++. – Litres, 2022.
10. Нефедов Д. Г., Русяк И. Г., Вавилова Д. Д. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. – 2020.
11. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – " Издательский дом"" Питер""", 2018.
12. Гудлиф П. Ремесло программиста. – 2009.

ПРИЛОЖЕНИЕ А Исходный код программы

Исходный код программы расположен в репозитории GitHub: <https://github.com/bibarsest/MIREA/tree/master/PgLg_Filatov_2_sem/Database_of_Students/Database_of_Students>

**Database\_of\_Students.cpp**

#include <iostream>

using namespace std;

#include <windows.h>

#include <fstream>

#include <sstream>

#include <string>

#include "Session.h"

#include "Functions\_bib.h"

#include "Date\_bib.h"

#include "Student\_bib.h"

#include "StudentList\_bib.h"

#include "Crypt\_bib.h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

}

**Session.h**

#pragma once

#include <iostream>

using namespace std;

struct Session

{

string subject = "#";

string mark = "#";

};

**Functions\_bib.h**

#pragma once

#include <iostream>

using namespace std;

bool Check(string \_word)

{

string int\_alphabet = "0123456789";

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

{

for (int j = 0; j < int\_alphabet.length(); j++)

if (\_word[i] == int\_alphabet[j]) { return false; }

}

return true;

}

void Clear()

{

cin.clear(); cin.ignore(INT\_MAX, '\n');

}

int HowMuchLinesInFile(const char \_filename[])

{

ifstream count\_file; int count = 0; string line;

count\_file.open(\_filename, ios::in);

while (!count\_file.eof()) { getline(count\_file, line); count++; }

count\_file.close();

return count;

}

#include "Session.h"

#include "Date\_bib.h"

#include "Student\_bib.h"

#include "StudentList\_bib.h"

#include "Crypt\_bib.h"

int menu()

{

Decrypt();

int choice = 10;

Student OperatingStudent;

StudentList studentList;

bool firstcall = true;

while (true)

{

while (cin.fail() || choice < 0 || choice > 6)

{

system("cls");

if (!firstcall) Clear();

cout << "Выберите операцию из нижеперечисленных. \n 1. Добавить полную запись о студенте в файл.\n 2. Отобразить все данные о выбранном студенте.\n 3. Изменить данные студента.\n";

cout << " 4. Удалить данные студента.\n 5. Распечатать всех студентов, у которых за все время обучения нет ни одной тройки с поиском среди лиц определенного пола (Вариант 34).";

cout << "\n 6. Распечатать всех студентов.\n 0. Выход из программы." << endl;

cout << "Выберите операцию [0-6]: "; cin >> choice;

firstcall = false;

}

Clear();

switch (choice)

{

case 1:

system("cls");

cout << "Добавить запись о студенте: " << endl;

OperatingStudent.SetStudent(); OperatingStudent.InputInFile("DB.txt");

system("pause");

firstcall = true; choice = 10;

break;

case 2:

system("cls");

OperatingStudent.OutputFromFile("DB.txt");

system("pause");

firstcall = true; choice = 10;

break;

case 3:

system("cls");

OperatingStudent.ChangeInfo("DB.txt");

system("pause");

firstcall = true; choice = 10;

break;

case 4:

system("cls");

OperatingStudent.DeleteInfo("DB.txt");

system("pause");

firstcall = true; choice = 10;

break;

case 5:

system("cls");

studentList.AddStudentsFromFile("DB.txt");

studentList.PrintStudentList("test");

studentList.FreeList();

system("pause");

firstcall = true; choice = 10;

break;

case 6:

system("cls");

studentList.AddStudentsFromFile("DB.txt");

studentList.PrintStudentList();

studentList.FreeList();

system("pause");

firstcall = true; choice = 10;

break;

case 0:

Crypt();

return 0;

}

}

}

**Date\_bib.h**

#pragma once

#include <iostream>

using namespace std;

#include "Functions\_bib.h"

class Date

{

public:

int Day, Month, Year;

Date()

{

Day = 0; Month = 0; Year = 0;

}

Date(int \_Day, int \_Month, int \_Year)

{

Day = \_Day; Month = \_Month; Year = \_Year;

}

void SetDate(int \_Day, int \_Month, int \_Year)

{

Day = \_Day; Month = \_Month; Year = \_Year;

}

void SetDate(Date Date)

{

SetDate(Date.Day, Date.Month, Date.Year);

}

void SetDate()

{

cout << "День: "; cin >> Day;

while (cin.fail() || Day > 31 || Day < 1)

{

cout << "Ошибка ввода. День [1-31]: "; Clear(); cin >> Day;

}

Clear();

cout << "Месяц: "; cin >> Month;

while (cin.fail() || Month > 12 || Month < 1)

{

cout << "Ошибка ввода. Месяц [1-12]: "; Clear(); cin >> Month;

}

Clear();

cout << "Год: "; cin >> Year;

while (cin.fail() || Year < 1900 || Year > 2023)

{

cout << "Ошибка ввода. Год [1900-2023]: "; Clear(); cin >> Year;

}

cin.clear();

}

friend istream& operator>>(istream& is, Date& dt)

{

is >> dt.Day >> dt.Month >> dt.Year;

return is;

}

int GetDay() { return Day; }

int GetMonth() { return Month; }

int GetYear() { return Year; }

int GetInfo(string \_info)

{

if (\_info == "Day") return Day;

if (\_info == "Month") return Month;

if (\_info == "Year") return Year;

}

void PrintDate()

{

cout << Day << " " << Month << " " << Year;

}

~Date() { ; }

};

**Student\_bib.h**

#pragma once

#include <iostream>

using namespace std;

#include "Functions\_bib.h"

class Student

{

string LastName, FirstName, SurName;

Date BirthDate;

int EnrollmentYear;

string Institute, Department, Group, RecordBook;

string Sex;

int numberofSessions, numberofSubjects, maxnumberofSubjects = -10;

bool has\_three\_mark = false;

public:

Student()

{

LastName = "undefined"; FirstName = "undefined"; SurName = "undefined";

BirthDate.SetDate(0, 0, 0); EnrollmentYear = 0;

Institute = "undefined"; Department = "undefined"; Group = "undefined"; RecordBook = "undefined";

Sex = 'u';

}

Student(string \_LastName, string \_FirstName, string \_SurName, Date \_BirthDate, int \_EnrollmentYear,

string \_Institute, string \_Department, string \_Group, string \_RecordBook, string \_Sex)

{

LastName = \_LastName; FirstName = \_FirstName; SurName = \_SurName;

BirthDate.SetDate(\_BirthDate); EnrollmentYear = \_EnrollmentYear;

Institute = \_Institute; Department = \_Department; Group = \_Group; RecordBook = \_RecordBook;

Sex = \_Sex;

}

friend ostream& operator<<(ostream& os, Student& st)

{

os << st.LastName << " " << st.FirstName << " " << st.SurName;

if (st.Sex == "м" || st.Sex == "ж")

{

if (st.Sex == "м") os << ", мужчина";

if (st.Sex == "ж") os << ", женщина";

}

else os << ", пол не определен";

os << " | ";

os << "Дата рождения: "; st.BirthDate.PrintDate(); os << endl;

os << "Год поступления: " << st.EnrollmentYear << " | Институт " << st.Institute << " | Кафедра " << st.Department << endl;

os << "Группа " << st.Group << " | Шифр " << st.RecordBook << " " << endl;

return os;

}

friend istream& operator>>(istream& is, Student& st)

{

is >> st.LastName >> st.FirstName >> st.SurName >> st.BirthDate >> st.EnrollmentYear >> st.Institute >> st.Department >> st.Group >> st.RecordBook >> st.Sex >> st.numberofSessions >> st.maxnumberofSubjects;

return is;

}

void Set\_three\_mark(bool is\_mark) { has\_three\_mark = is\_mark; }

void SetStudent(string \_LastName, string \_FirstName, string \_SurName, Date \_BirthDate, int \_EnrollmentYear,

string \_Institute, string \_Department, string \_Group, string \_RecordBook, string \_Sex)

{

LastName = \_LastName; FirstName = \_FirstName; SurName = \_SurName;

BirthDate.SetDate(\_BirthDate); EnrollmentYear = \_EnrollmentYear;

Institute = \_Institute; Department = \_Department; Group = \_Group; RecordBook = \_RecordBook;

Sex = \_Sex;

}

void SetStudent()

{

ifstream file;

string line, searchline;

bool RecordBook\_search = true, count = true;

cout << "Фамилия: "; cin >> LastName;

while (cin.fail() || !(Check(LastName))) { Clear(); cout << "Ошибка ввода. Фамилия не должна содержать цифр. \n>"; cin >> LastName; }

Clear();

cout << "Имя: "; cin >> FirstName;

while (cin.fail() || !(Check(FirstName))) { Clear(); cout << "Ошибка ввода. Имя не должно содержать цифр. \n>"; cin >> FirstName; }

Clear();

cout << "Отчество: "; cin >> SurName;

while (cin.fail() || !(Check(SurName))) { Clear(); cout << "Ошибка ввода. Отчество не должно содержать цифр. \n>"; cin >> SurName; }

Clear();

cout << "Дата рождения: "; BirthDate.SetDate();

cout << "Год поступления: "; cin >> EnrollmentYear;

while (cin.fail() || EnrollmentYear < 1900) { cout << "Ошибка ввода. Год поступления [>=1900]: "; Clear(); cin >> EnrollmentYear; }

Clear();

cout << "Институт: "; cin >> Institute; Clear();

cout << "Кафедра: "; cin >> Department; Clear();

cout << "Группа: "; cin >> Group; Clear();

cout << "Номер зачетной книжки: "; cin >> RecordBook; Clear();

while (RecordBook\_search)

{

file.open("DB.txt", ios::out);

count = false;

while (getline(file, line))

{

while (line.find(RecordBook) != string::npos)

{

cout << "Номер зачетной книжки уже используется. Введите другой." << endl;

cout << "Номер зачетной книжки: "; cin >> RecordBook; Clear();

count = true;

}

}

file.close();

if (count == false) RecordBook\_search = false;

}

cout << "Пол (м-мужчина, ж-женщина): "; cin >> Sex;

while (cin.fail() || ((Sex != "м") && (Sex != "ж"))) { Clear(); cout << "Ошибка ввода. Пол [м/ж]: "; cin >> Sex; }

Clear();

}

void InputInFile()

{

ofstream file;

file.open("DB.txt", ios::app);

file << LastName << " " << FirstName << " " << SurName << " " << BirthDate.GetInfo("Day") << " "

<< BirthDate.GetInfo("Month") << " " << BirthDate.GetInfo("Year") << " " << EnrollmentYear << " "

<< Institute << " " << Department << " " << Group << " " << RecordBook << " " << Sex << endl;

file.close();

}

void InputInFile(const char \_filename[])

{

cout << "Кол-во сессий [1-9]: "; cin >> numberofSessions;

while (cin.fail() || numberofSessions < 1 || numberofSessions > 9) { cout << "Ошибка ввода. Кол-во сессий [1-9]: "; Clear(); cin >> numberofSessions; }

Clear();

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

cout << "Кол-во предметов для сессии №" << i + 1 << " [1-10]: "; cin >> numberofSubjects;

while (cin.fail() || numberofSubjects < 1 || numberofSubjects > 10) { cout << "Ошибка ввода. Кол-во предметов [1-10]: "; Clear(); cin >> numberofSubjects; }

Clear();

if (numberofSubjects > maxnumberofSubjects) maxnumberofSubjects = numberofSubjects;

Sessions[i] = new Session[10]();

for (int j = 0; j < numberofSubjects; j++)

{

cout << "Предмет №" << j + 1 << ": "; cin >> Sessions[i][j].subject; Clear();

cout << "Оценка №" << j + 1 << " [1-5, з - зачет, н - незачет]: "; cin >> Sessions[i][j].mark;

while (cin.fail() || (Sessions[i][j].mark != "з" && Sessions[i][j].mark != "н" &&

Sessions[i][j].mark != "5" && Sessions[i][j].mark != "4" && Sessions[i][j].mark != "3" && Sessions[i][j].mark != "2" && Sessions[i][j].mark != "1"))

{

cout << "Ошибка ввода. Оценка: [1-5, з - зачет, н - незачет]: "; Clear(); cin >> Sessions[i][j].mark;

}

Clear();

}

}

string choice = "z";

while (cin.fail() || (choice != "1" && choice != "0"))

{

system("cls");

cout << "Введенные данные: "; PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт | ";

if (Sessions[i][j].mark == "н") cout << "незачёт | ";

}

else if (stoi(Sessions[i][j].mark) > 0 && stoi(Sessions[i][j].mark) < 6) cout << Sessions[i][j].mark << " | ";

if (j == 3 || j == 7) cout << endl;

}

cout << endl;

}

cout << "Хотите записать данные в файл?\n 1 - Да.\n 0 - Нет.\n> ";

cin >> choice;

if (choice == "1")

{

ofstream file;

file.open(\_filename, ios::app);

file << LastName << " " << FirstName << " " << SurName << " " << BirthDate.GetInfo("Day") << " "

<< BirthDate.GetInfo("Month") << " " << BirthDate.GetInfo("Year") << " " << EnrollmentYear << " "

<< Institute << " " << Department << " " << Group << " " << RecordBook << " " << Sex << " " <<

numberofSessions << " " << maxnumberofSubjects << " ";

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

{

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

file << Sessions[i][j].subject << " " << Sessions[i][j].mark << " ";

}

file << "| ";

}

file << endl;

file.close();

}

if (choice == "0") { cout << "Данные не записаны. " << endl; }

if (choice != "1" && choice != "0") { Clear(); system("cls"); cout << "Ошибка ввода. " << endl; }

}

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

{

delete[] Sessions[i];

}

delete[] Sessions;

Clear();

}

void OutputFromFile(const char \_filename[])

{

ifstream file;

string searchline, buffer;

cout << "Введите номер зачетной книжки студента: "; cin >> searchline;

file.open(\_filename, ios::in);

while (file >> LastName >> FirstName >> SurName >> BirthDate.Day >> BirthDate.Month >> BirthDate.Year >>

EnrollmentYear >> Institute >> Department >> Group >> RecordBook >> Sex >> numberofSessions >> maxnumberofSubjects)

{

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

Sessions[i] = new Session[maxnumberofSubjects];

}

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

{

for (int j = 0; j < maxnumberofSubjects + 1; j++)

{

file >> buffer;

if (buffer == "|") break;

Sessions[i][j].subject = buffer;

file >> Sessions[i][j].mark;

}

}

if (RecordBook.find(searchline) != string::npos)

{

PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт | ";

if (Sessions[i][j].mark == "н") cout << "незачёт | ";

}

else cout << Sessions[i][j].mark << " | ";

if (j == 3 || j == 7) cout << endl;

}

cout << endl;

}

}

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

{

delete[] Sessions[i];

}

delete[] Sessions;

}

file.close();

}

void OutputFromFile(const char \_filename[], int o)

{

ifstream file;

string line, searchline, buffer;

cout << "Введите номер зачетной книжки студента: "; cin >> searchline;

file.open(\_filename, ios::in);

while (getline(file, line))

{

stringstream ss(line);

ss >> LastName >> FirstName >> SurName >> BirthDate.Day >> BirthDate.Month >> BirthDate.Year >>

EnrollmentYear >> Institute >> Department >> Group >> RecordBook >> Sex >> numberofSessions >> maxnumberofSubjects;

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

Sessions[i] = new Session[maxnumberofSubjects];

}

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

{

for (int j = 0; j < maxnumberofSubjects + 1; j++)

{

ss >> buffer;

if (buffer == "|") break;

Sessions[i][j].subject = buffer;

ss >> Sessions[i][j].mark;

}

}

if (RecordBook.find(searchline) != string::npos)

{

PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт ";

if (Sessions[i][j].mark == "н") cout << "незачёт ";

}

else cout << Sessions[i][j].mark << " ";

}

cout << endl;

}

}

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

{

delete[] Sessions[i];

}

delete[] Sessions;

}

file.close();

}

void DeleteInfo(const char \_filename[])

{

ifstream file;

string bufer, line, searchline;

int deleting\_choice = 0;

remove("DBtemp.txt");

cout << "Введите номер зачетной книжки студента: "; cin >> searchline;

file.open(\_filename, ios::in);

while (file >> LastName >> FirstName >> SurName >> BirthDate.Day >> BirthDate.Month >> BirthDate.Year >>

EnrollmentYear >> Institute >> Department >> Group >> RecordBook >> Sex >> numberofSessions >> maxnumberofSubjects)

{

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

Sessions[i] = new Session[maxnumberofSubjects];

}

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

{

for (int j = 0; j < maxnumberofSubjects + 1; j++)

{

file >> bufer;

if (bufer == "|") break;

Sessions[i][j].subject = bufer;

file >> Sessions[i][j].mark;

}

}

if (RecordBook.find(searchline) != string::npos)

{

cout << "Выбранные для удаления данные: " << endl; PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт | ";

if (Sessions[i][j].mark == "н") cout << "незачёт | ";

}

else cout << Sessions[i][j].mark << " | ";

if (j == 3 || j == 7) cout << endl;

}

cout << endl;

}

cout << "Вы уверены, что хотите удалить данные?\n 1. Удалить.\n 0. Отмена.\n> ";

cin >> deleting\_choice;

while (cin.fail() || (deleting\_choice != 0 && deleting\_choice != 1))

{

Clear(); system("cls");

cout << "Ошибка ввода.\nВы уверены, что хотите удалить данные?\n 1. Удалить.\n 0. Отмена.\n> "; cin >> deleting\_choice;

}

Clear();

}

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

{

delete[] Sessions[i];

}

delete[] Sessions;

}

switch (deleting\_choice)

{

case 0:

file.close();

cout << "Данные не удалены." << endl;

break;

case 1:

file.close();

file.open(\_filename, ios::in);

while (file >> LastName >> FirstName >> SurName >> BirthDate.Day >> BirthDate.Month >> BirthDate.Year >>

EnrollmentYear >> Institute >> Department >> Group >> RecordBook >> Sex >> numberofSessions >> maxnumberofSubjects)

{

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

Sessions[i] = new Session[maxnumberofSubjects];

}

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

{

for (int j = 0; j < maxnumberofSubjects + 1; j++)

{

file >> bufer;

if (bufer == "|") break;

Sessions[i][j].subject = bufer;

file >> Sessions[i][j].mark;

}

}

if (!(RecordBook.find(searchline) != string::npos))

{

ofstream file\_out;

file\_out.open("DBtemp.txt", ios::app);

file\_out << LastName << " " << FirstName << " " << SurName << " " << BirthDate.GetInfo("Day") << " "

<< BirthDate.GetInfo("Month") << " " << BirthDate.GetInfo("Year") << " " << EnrollmentYear << " "

<< Institute << " " << Department << " " << Group << " " << RecordBook << " " << Sex << " " << numberofSessions << " "

<< maxnumberofSubjects << " ";

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

{

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

file\_out << Sessions[i][j].subject << " " << Sessions[i][j].mark << " ";

}

file\_out << "| ";

}

file\_out << endl;

file\_out.close();

}

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

{

delete[] Sessions[i];

}

delete[] Sessions;

}

file.close();

remove(\_filename);

rename("DBtemp.txt", \_filename);

cout << "Данные удалены." << endl;

break;

}

}

void ChangeInfo(const char \_filename[])

{

ifstream file;

string searchline, buffer, line;

int choice = 20, session\_num = 0, subject\_num = 0;

bool menu\_is\_true = true, RecordBook\_search = true, is\_RecordBook\_changed = false, RecordBook\_changed = false, first\_change\_sessions = true;

remove("DBtemp.txt");

cout << "Введите номер зачетной книжки студента: "; cin >> searchline;

file.open(\_filename, ios::in);

while (file >> LastName >> FirstName >> SurName >> BirthDate.Day >> BirthDate.Month >> BirthDate.Year >>

EnrollmentYear >> Institute >> Department >> Group >> RecordBook >> Sex >> numberofSessions >> maxnumberofSubjects)

{

Session\*\* Sessions = new Session \* [numberofSessions];

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

{

Sessions[i] = new Session[maxnumberofSubjects]();

}

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

{

for (int j = 0; j < maxnumberofSubjects + 1; j++)

{

file >> buffer;

if (buffer == "|") break;

Sessions[i][j].subject = buffer;

file >> Sessions[i][j].mark;

}

}

if (RecordBook.find(searchline) != string::npos)

{

while (menu\_is\_true)

{

while (cin.fail() || choice < 0 || choice > 12)

{

Clear(); system("cls");

cout << "Выбранный студент: " << endl; PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт | ";

if (Sessions[i][j].mark == "н") cout << "незачёт | ";

}

else cout << Sessions[i][j].mark << " | ";

if (j == 3 || j == 7) cout << endl;

}

cout << endl;

}

cout << "Какие данные нужно изменить?\n1. Фамилию\n2. Имя\n3. Отчество\n4. Дату рождения\n5. Год поступления\n6. Институт\n7. Кафедру\n8. Группу\n9. ";

cout << "Номер зачетной книжки\n10. Пол\n11. Имеющиеся сессии\n12. Добавить сессию\n0. Подтвердить\n>>> ";

cin >> choice;

}

Clear();

switch (choice)

{

system("cls");

case 0: menu\_is\_true = false; break;

case 1:

cout << "Фамилия: "; cin >> LastName;

while (cin.fail() || !(Check(LastName))) { Clear(); cout << "Ошибка ввода. Фамилия не должна содержать цифр. \n>"; cin >> LastName; }

choice = 20; break;

case 2:

cout << "Имя: "; cin >> FirstName;

while (cin.fail() || !(Check(FirstName))) { Clear(); cout << "Ошибка ввода. Имя не должно содержать цифр. \n>"; cin >> FirstName; }

choice = 20; break;

case 3:

cout << "Отчество: "; cin >> SurName;

while (cin.fail() || !(Check(SurName))) { Clear(); cout << "Ошибка ввода. Отчество не должно содержать цифр. \n>"; cin >> SurName; }

choice = 20; break;

case 4:

cout << "Дата рождения: "; BirthDate.SetDate();

choice = 20; break;

case 5:

cout << "Год поступления: "; cin >> EnrollmentYear;

while (cin.fail() || EnrollmentYear < 1900) { cout << "Ошибка ввода. Год поступления [>=1900]: "; Clear(); cin >> EnrollmentYear; }

choice = 20; break;

case 6:

cout << "Институт: "; cin >> Institute; cin.clear();

choice = 20; break;

case 7:

cout << "Кафедра: "; cin >> Department; cin.clear();

choice = 20; break;

case 8:

cout << "Группа: "; cin >> Group; cin.clear();

choice = 20; break;

case 9:

cout << "Номер зачетной книжки: "; cin >> RecordBook; Clear();

while (RecordBook\_search)

{

ifstream filecheck;

filecheck.open("DB.txt", ios::out);

is\_RecordBook\_changed = false;

while (getline(filecheck, line))

{

while (line.find(RecordBook) != string::npos)

{

cout << "Номер зачетной книжки уже используется. Введите другой." << endl;

cout << "Номер зачетной книжки: "; cin >> RecordBook; Clear();

is\_RecordBook\_changed = true;

}

}

filecheck.close();

if (is\_RecordBook\_changed == false) RecordBook\_search = false;

}

choice = 20; RecordBook\_changed = true; menu\_is\_true = false; break;

case 10:

cout << "Пол (м-мужчина, ж-женщина): "; cin >> Sex;

while (cin.fail() || ((Sex != "м") && (Sex != "ж"))) { Clear(); cout << "Ошибка ввода. Пол [м/ж]: "; cin >> Sex; }

choice = 20; break;

case 11:

session\_num = 0; subject\_num = 0;

while (cin.fail() || session\_num < 1 || session\_num > numberofSessions)

{

if (!first\_change\_sessions) Clear();

cout << "Выберите, какую сессию изменить [1-" << numberofSessions << "]: "; cin >> session\_num; first\_change\_sessions = false;

}

session\_num--;

while (cin.fail() || subject\_num < 1 || subject\_num > maxnumberofSubjects)

{

Clear();

cout << "Выберите позицию предмета, который надо изменить [1-" << maxnumberofSubjects << "]: "; cin >> subject\_num;

}

subject\_num--;

cout << "Выбрано:\n" << Sessions[session\_num][subject\_num].subject << ": " << Sessions[session\_num][subject\_num].mark << endl;

if (Sessions[session\_num][subject\_num].mark == "#" && Sessions[session\_num][subject\_num - 1].mark == "#") { cout << "Сессия/предмет вне диапазона" << endl; system("pause"); }

else

{

cout << "Введите предмет: "; cin >> Sessions[session\_num][subject\_num].subject; Clear();

cout << "Введите оценку [1-5, з - зачет, н - незачет]: "; cin >> Sessions[session\_num][subject\_num].mark;

while (Sessions[session\_num][subject\_num].mark != "5" && Sessions[session\_num][subject\_num].mark != "4" &&

Sessions[session\_num][subject\_num].mark != "3" && Sessions[session\_num][subject\_num].mark != "2" &&

Sessions[session\_num][subject\_num].mark != "1" && Sessions[session\_num][subject\_num].mark != "з" &&

Sessions[session\_num][subject\_num].mark != "н")

{

Clear(); cout << "Введите оценку [1-5, з - зачет, н - незачет]: "; cin >> Sessions[session\_num][subject\_num].mark;

}

}

choice = 20; break;

case 12: numberofSessions++;

cout << "Кол-во предметов для сессии №" << numberofSessions << " [1-10]: "; cin >> numberofSubjects;

while (cin.fail() || numberofSubjects < 1 || numberofSubjects > 10) { cout << "Ошибка ввода. Кол-во предметов [1-10]: "; Clear(); cin >> numberofSubjects; }

Clear();

if (numberofSubjects > maxnumberofSubjects) maxnumberofSubjects = numberofSubjects;

Sessions[numberofSessions - 1] = new Session[numberofSubjects + 1]();

for (int j = 0; j < numberofSubjects; j++)

{

cout << "Предмет №" << j + 1 << ": "; cin >> Sessions[numberofSessions - 1][j].subject; Clear();

cout << "Оценка №" << j + 1 << " [1-5, з - зачет, н - незачет]: "; cin >> Sessions[numberofSessions - 1][j].mark;

while (cin.fail() || (Sessions[numberofSessions - 1][j].mark != "з" && Sessions[numberofSessions - 1][j].mark != "н" &&

Sessions[numberofSessions - 1][j].mark != "5" && Sessions[numberofSessions - 1][j].mark != "4" &&

Sessions[numberofSessions - 1][j].mark != "3" && Sessions[numberofSessions - 1][j].mark != "2" && Sessions[numberofSessions - 1][j].mark != "1"))

{

cout << "Ошибка ввода. Оценка: [1-5, з - зачет, н - незачет]: "; Clear(); cin >> Sessions[numberofSessions - 1][j].mark;

}

Clear();

}

choice = 0;

break;

}

}

system("cls");

cout << "Измененные данные:\n";

PrintStudent();

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

{

cout << "----------СЕССИЯ №" << i + 1 << "----------" << endl;

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

cout << Sessions[i][j].subject << ": ";

if (Sessions[i][j].mark == "з" || Sessions[i][j].mark == "н")

{

if (Sessions[i][j].mark == "з") cout << "зачёт | ";

if (Sessions[i][j].mark == "н") cout << "незачёт | ";

}

else cout << Sessions[i][j].mark << " | ";

if (j == 3 || j == 7) cout << endl;

}

cout << endl;

}

if (!RecordBook\_changed)

{

ofstream file\_out;

file\_out.open("DBtemp.txt", ios::app);

file\_out << LastName << " " << FirstName << " " << SurName << " " << BirthDate.GetInfo("Day") << " "

<< BirthDate.GetInfo("Month") << " " << BirthDate.GetInfo("Year") << " " << EnrollmentYear << " "

<< Institute << " " << Department << " " << Group << " " << RecordBook << " " << Sex << " " <<

numberofSessions << " " << maxnumberofSubjects << " ";

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

{

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

file\_out << Sessions[i][j].subject << " " << Sessions[i][j].mark << " ";

}

file\_out << "| ";

}

file\_out << endl;

file\_out.close();

}

}

if (!(RecordBook.find(searchline) != string::npos))

{

ofstream file\_out\_other\_students;

file\_out\_other\_students.open("DBtemp.txt", ios::app);

file\_out\_other\_students << LastName << " " << FirstName << " " << SurName << " " << BirthDate.GetInfo("Day") << " "

<< BirthDate.GetInfo("Month") << " " << BirthDate.GetInfo("Year") << " " << EnrollmentYear << " "

<< Institute << " " << Department << " " << Group << " " << RecordBook << " " << Sex << " " << numberofSessions << " "

<< maxnumberofSubjects << " ";

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

{

for (int j = 0; j < maxnumberofSubjects; j++)

{

if (Sessions[i][j].mark == "#") break;

file\_out\_other\_students << Sessions[i][j].subject << " " << Sessions[i][j].mark << " ";

}

file\_out\_other\_students << "| ";

}

file\_out\_other\_students << endl;

file\_out\_other\_students.close();

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

{

delete[] Sessions[i];

}

delete[] Sessions;

}

}

file.close();

remove(\_filename);

rename("DBtemp.txt", \_filename);

}

void PrintStudent()

{

cout << LastName << " " << FirstName << " " << SurName;

if (Sex == "м" || Sex == "ж")

{

if (Sex == "м") cout << ", мужчина";

if (Sex == "ж") cout << ", женщина";

}

else cout << ", пол не определен";

cout << " | ";

cout << "Дата рождения: "; BirthDate.PrintDate(); cout << endl;

cout << "Год поступления: " << EnrollmentYear << " | Институт " << Institute << " | Кафедра " << Department << endl;

cout << "Группа " << Group << " | Шифр " << RecordBook << " " << endl;

}

string GetSex() { return Sex; }

bool Get\_three\_mark() { return has\_three\_mark; }

int GetNumberofSessions() { return numberofSessions; }

int GetMaxNumberofSubjects() { return maxnumberofSubjects; }

~Student() { ; }

};

**StudentList\_bib.h**

#pragma once

#include <iostream>

using namespace std;

#include "Functions\_bib.h"

class StudentList

{

struct Node

{

Student\* student;

Node\* next;

};

Node\* head;

public:

StudentList() { head = nullptr; }

~StudentList()

{

while (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

}

}

void AddStudent(Student& student)

{

Node\* newNode = new Node{ &student, nullptr };

if (head == nullptr) { head = newNode; }

else

{

Node\* current = head;

while (current->next != nullptr) { current = current->next; }

current->next = newNode;

}

}

void AddStudent(Student\* student)

{

Node\* newNode = new Node{ student, nullptr };

if (head == nullptr) { head = newNode; }

else

{

Node\* current = head;

while (current->next != nullptr) { current = current->next; }

current->next = newNode;

}

}

void AddStudentsFromFile(const char \_filename[])

{

ifstream infile;

infile.open(\_filename);

string buffer;

int totallines = HowMuchLinesInFile(\_filename), countdown = 0;

Student\* student = new Student[totallines - 1];

while (!infile.eof() && (countdown < totallines - 1))

{

for (int k = 0; k < totallines - 1; k++, countdown++)

{

infile >> student[k];

Session\*\* Sessions = new Session \* [student[k].GetNumberofSessions()];

for (int i = 0; i < student[k].GetNumberofSessions(); i++)

{

Sessions[i] = new Session[student[k].GetMaxNumberofSubjects()];

for (int j = 0; j < student[k].GetMaxNumberofSubjects() + 1; j++)

{

infile >> buffer;

if (buffer == "|") break;

Sessions[i][j].subject = buffer;

infile >> Sessions[i][j].mark;

if (Sessions[i][j].mark == "3") student[k].Set\_three\_mark(true);

}

}

for (int i = 0; i < student[k].GetNumberofSessions(); i++)

{

delete[] Sessions[i];

}

delete[] Sessions;

AddStudent(student[k]);

}

}

infile.close();

}

void PrintStudentList()

{

for (Node\* current = head; current != nullptr; current = current->next)

{

current->student->PrintStudent(); cout << endl;

}

}

void PrintStudentList(string \_sex)

{

while (\_sex != "м" && \_sex != "ж")

{

cout << "Вывести всех студентов без троек, пол которых [м/ж]: "; cin >> \_sex; Clear();

}

cin.clear();

for (Node\* current = head; current != nullptr; current = current->next)

{

if (current->student->GetSex() == \_sex && !current->student->Get\_three\_mark()) { current->student->PrintStudent(); cout << endl; }

}

}

void FreeList()

{

while (head != nullptr)

{

Node\* temp = head;

head = head->next;

delete temp;

}

}

};

**Crypt\_bib.h**

#pragma once

#include <iostream>

using namespace std;

#include <fstream>

#include <time.h>

#include <windows.h>

#include <string>

void Crypt()

{

srand(time(NULL));

string pass;

for (int i = 0; i < 128; ++i) {

switch (rand() % 3) {

case 0:

pass += rand() % 10 + '0';

break;

case 1:

pass += rand() % 26 + 'A';

break;

case 2:

pass += rand() % 26 + 'a';

}

}

string command = "OpenSSL\\bin\\openssl.exe enc -aes-256-cbc -salt -in DB.txt -out DB.txt.enc -pass pass:";

command += pass;

command += " -iter 1000";

system(command.c\_str());

if (remove("DB.txt") != 0)

{

cout << "[ОШИБКА 0021091] - Не удалось удалить файл с базой." << endl;

}

ofstream filein;

filein.open("key.bin", ios::binary);

filein << pass;

filein.close();

command = "OpenSSL\\bin\\openssl.exe rsautl -encrypt -inkey rsa.public -pubin -in key.bin -out key.bin.enc";

system(command.c\_str());

if (remove("key.bin") != 0)

{

cout << "[ОШИБКА 0021092] - Не удалось удалить файл с ключом." << endl;

}

}

void Decrypt()

{

string command = "OpenSSL\\bin\\openssl.exe rsautl -decrypt -inkey rsa.private -in key.bin.enc -out key.bin";

system(command.c\_str());

if (remove("key.bin.enc") != 0)

{

cout << "[ОШИБКА 0021093] - Не удалось удалить файл с зашифрованным ключом." << endl;

}

string pass;

ifstream fileout;

fileout.open("key.bin", ios::binary);

fileout >> pass;

fileout.close();

if (remove("key.bin") != 0)

{

cout << "[ОШИБКА 0021094] - Не удалось удалить файл с ключом." << endl;

}

command = "OpenSSL\\bin\\openssl.exe enc -aes-256-cbc -d -in DB.txt.enc -out DB.txt -pass pass:";

command += pass;

command += " -iter 1000";

system(command.c\_str());

if (remove("DB.txt.enc") != 0)

{

cout << "[ОШИБКА 0021095] - Не удалось удалить файл с зашифрованной базой." << endl;

}

}