|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «МИРЭА – Российский технологический университет»  РТУ МИРЭА |

Институт комплексной безопасности и специального приборостроения

(наименование института, филиала)

Кафедра КБ-3 «Управление и моделирование систем»

(наименование кафедры)

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Языки программирования»

(наименование дисциплины)

**Тема курсового проекта**  Разработка файловой системы.

**Студент группы** Гладышев Максим Алексеевич, группа БАСО-01-19 \_\_\_\_\_\_\_\_\_\_

(учебная группа, фамилия, имя, отчество студента) (подпись студента)

**Руководитель курсового проекта** Мерсов Андрей Александрович \_\_\_\_\_\_\_\_\_\_\_

(должность, звание, ученая степень) (подпись руководителя)

**Рецензент (при наличии)** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_

(должность, звание, ученая степень) (подпись рецензента)

Работа представлена к защите «25» **\_\_\_ мая\_\_\_** 2020 г.

Допущен к защите «05» **\_\_\_\_\_июня\_\_\_\_\_** 2020 г.

|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «МИРЭА – Российский технологический университет»  РТУ МИРЭА |

Институт комплексной безопасности и специального приборостроения

(наименование института, филиала)

Кафедра КБ-3 «Управление и моделирование систем»

(наименование кафедры)

Утверждаю

Заведующий кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись ФИО

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**ЗАДАНИЕ**

**На выполнение курсового проекта** по дисциплине

«Языки программирования»

Студент Гладышев Максим Алексеевич Группа БАСО-01-19

**Тема:** разработка файловой системы студентов с выполнением задания, в соответствии с вариантом, а именно нахождение всех студентов и вывод информации о них на экран, у которых за время обучения нет ни одной оценки три, три и четыре, или пять, вводимых по желанию пользователя, с указанием интервала года рождения.

**Исходные данные:** информация о группе студентов из N человек, где запись о студенте содержит определенные данные.

**Перечень вопросов, подлежащих разработке и обязательного графического материала:** изучение теоретической части объектно-ориентированного программирования, основными элементами которого являются наследование, классы, объекты и методы классов, дружественные классы, работы с файлами, чтение и запись в них, работы с динамической пямятью и элементов криптографии.

Срок представления к защите курсового проекта: до «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г.

Задание на курсовой проект выдал \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_А,А,Мерсов\_\_\_\_\_\_\_\_\_\_\_\_\_)

Подпись руководителя Ф.И.О. руководителя

Задание на курсовой проект получил \_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_М.А.Гладышев\_\_\_\_\_\_\_\_\_\_\_\_)

Подпись обучающегося Ф.И.О. исполнителя

Оглавление

[Цель курсовой работы 3](#_Toc41475897)

[Исходные данные варианта задания. 4](#_Toc41475898)

[Алгоритм решения задачи 5](#_Toc41475899)

[Описание концепции работы программы 10](#_Toc41475900)

[Блок схема программы 23](#_Toc41475901)

[Исходные данные 24](#_Toc41475902)

[Выходные данные 29](#_Toc41475903)

[Вывод 37](#_Toc41475904)

[Листинг программы 38](#_Toc41475905)

[student.h 38](#_Toc41475906)

[student.cpp 39](#_Toc41475907)

[check.h 39](#_Toc41475908)

[check.cpp 39](#_Toc41475909)

[metods.h 40](#_Toc41475910)

[metods.cpp 40](#_Toc41475911)

[Курсовая.cpp 58](#_Toc41475912)

# 

# Цель курсовой работы

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.

Задачей курсовой работы является разработка файловой системы студентов с выполнением задания, в соответствии с вариантом, а именно нахождение всех студентов и вывод информации о них на экран, у которых за время обучения нет ни одной оценки три, три и четыре, или пять, вводимых по желанию пользователя, с указанием интервала года рождения.

# Исходные данные варианта задания.

Вариант 83 Распечатать всех студентов, у которых за все время обучения нет ни одной оценки а) 3 б) 3 и 4 в) 5 Предусмотреть распечатывать лиц с указанием интервала года рождения. Варианты а)-в) выбираются по желанию пользователя.

Исходными данными для программы является информация о группе студентов из N человек, где запись о студенте содержит следующие данные:

|  |  |
| --- | --- |
| Данные | Тип данных |
| Ф.И.О. студента | fios |
| Пол | char(символьный) |
| Число, месяц, год рождения | birthdays |
| Год поступления в институт | int(целочисленный) |
| Факультет | char[] |
| Кафедра | char[] |
| Группа | char[] |
| Номер Зачетной книжки | char[] |
| Название предмета | char[] |
| Оценка | int(целочисленный) |

Максимально 9 сессий и 10 предметов в каждом семестре, которые могут быть разными.

# Алгоритм решения задачи

Для решения поставленной задачи необходимо разработать структуры – fios, birthdays , universitys, predmets, semestrs, а также абстрактный класс- file и простые классы – file\_p, file\_crypt, student, metods. Описание их назначения.

|  |  |
| --- | --- |
| Структура | Назначение |
| fios | Пользовательский тип данных для хранения ФИО студента. |
| birthdays | Пользовательский тип данных для хранения даты рождения студента. |
| universitys | Хранит в себе информацию связанную с институтом(факультет , кафедра, группа, номер зачетной книжки, год поступления) |
| predmets | Хранит в себе название предмета, оценку по нему, и указатель на следующий элемент списка предметов. |
| semestrs | Хранит в себе номер сессии и указатель на первый элемент списка предметов в сессии. |

|  |  |
| --- | --- |
| Класс | Назначение |
| student | Хранит в себе всю информацию о студенте(ФИО, пол, дату рождения, факультет и т.д.) |
| metods | Хранит в себе все методы, требуемые для работы со списком студентов. |
| file | Абстрактный класс. |
| file\_p | Производный класс от класса file, для работы с нешифрованными файлами. |
| file\_crypt | Производный класс от класса file, для считывания зашифрованных файлов, и шифрования данных с последующей записью в файл. |

Структура — это, некое объединение различных переменных (возможно разных типов данных), которому можно присвоить имя.

В таблице приведены и описаны поля структур, использованных при решении задачи.

|  |  |  |
| --- | --- | --- |
| Структура | Поля структур | |
| Тип данных | Название и характеристика |
| fios | char | name[21] – массив символов для хранения имени студента  surname[21] – массив символов для хранения фамилии студента  otchestvo[21] – массив символов для хранения отчества студента |
| birthdays | int | day – день рождения  mounth – месяц рождения  year – год рождения |
| universitys | int | year\_post – год поступления |
| char | facultet[101] – массив символов для хранения названия факультета  kafedra[71] – массив символов для хранения названия кафедры  group[21] – массив символов для хранения названия группы  number\_zachet[21] – массив символов для хранения номера зачетной книжки |
| predmets | int | id\_predmet – идентификационной номер предмета  mark – оценка по данному предмету |
| char | name\_predmet[51] – массив символов для хранения названия предмета |
| predmets\* | next\_predmet – указатель на экземпляр структуры следующего предмета |
| semestrs | int | id\_semestr – идентификационной номер сессии |
| predmets\* | predmet – указатель на экземпляр структуры первого предмета в данной сессии |
| semestrs\* | next\_semestr – указатель на экземпляр структуры следующей сессии |

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.

Класс (class) - это тип, определяемый пользователем, включающий в себя данные и функции, называемые методами или функциями-членами класса.

Таким образом, определение типа задаваемого пользователем (class) содержит спецификацию данных, требующихся для представления объекта этого типа, и набор операций (функций) для работы с подобными объектами.

В то время как объект обозначает конкретную сущность, определенную во времени и в пространстве, класс определяет лишь абстракцию существенного в объекте.

В таблицах приведены и описаны поля и методы классов, использованных при решении задачи.

|  |  |  |
| --- | --- | --- |
| Класс | Поля класса | |
| Тип данных | Название и характеристика |
| student | int | id\_student – идентификационной номер студента |
| char | pol – пол студента |
| fios | fio – экземпляр структуры для хранения ФИО студента |
| birthdays | birthday – экземпляр структуры для хранения даты рождения студента |
| universitys | university – экземпляр структуры для хранения информации студента связанной с институтом (факультет , кафедра, группа, номер зачетной книжки, год поступления) |
| Semestrs\* | semestr – указатель на экземпляр структуры первой сессии студента |
| Student\* | next\_sudent – указатель на объект следующего студента |
| metods |  | Поля отсутствуют |
| file |  | Поля отсутствуют |
| file\_p |  | Поля отсутствуют |
| file\_crypt |  | Поля отсутствуют |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Класс | Методы класса | | | |
| Название | Назначение | Аргументы (их типы) | Тип возвращаемого значения |
| student | student | Конструктор класса | - | - |
| ~ student | Деструктор класса | - | - |
| metods | metods | Конструктор класса | - | - |
| ~metods | Деструктор класса | - | - |
| create | Создает первый элемент списка студентов | - | student\* |
| create | Добавляет запись о студенте в список | pstarts(student\*) | void |
| search | Проверка на существование сессии с введенным номером | pstart(student\*), pprom(semestrs\*), id(int) | int |
| menu1\_for\_zadani | Выполнение задачи поставленной вариантом | pstart(student\*) | void |
| check\_mark | Проверяет какие оценки у студента по данному предмету(вспомогательная функция для выполнения задания) | stud(student\*), nom1(int), nom2(int) | int |
| print | Выводит список студентов на экран | pstart(student\*) | void |
| searchs | Ищет в списке по номеру зачетной книжки необходимого студента | pprom(student\*) | student\* |
| searchp | Проверка на существование в текущей сессии предмета, название которого введенного с клавиатуры | pprom(semestrs\*),  ppromp(predmets\*), p[](char) | int |
| change | Нужна для изменения данных в записях о студенте | pstart(student\*) | void |
| delet | Удаляет данные о студенте, или сессии, или предмета | pstart(student\*) | student\* |
| dobav\_semestr | Добавляет запись о сессии | pstart(student\*) | void |
| dobav\_predmet | Добавляет запись о предмете | pstart(student\*) | void |
| file | file\_load | Загружает данные из файла | - | student\* |
| file\_save | Сохраняет данные в файл | pstart(student\*) | void |
| file\_p | file\_p | Конструктор класса | - | - |
| ~ file\_p | Деструктор класса | - | - |
| file\_load | Загружает данные из файла | - | student\* |
| file\_save | Сохраняет данные в файл | pstart(student\*) | void |
| file\_crypt | file\_crypt | Конструктор класса | - | - |
| ~ file\_crypt | Деструктор класса | - | - |
| file\_load | Загружает данные из зашифрованного файла | - | student\* |
| file\_save | Сохраняет зашифрованные данные в файл | pstart(student\*) | void |

Помимо этого в программе есть отдельные функции, приведем их описание в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Назначение | Аргументы (их типы) | Тип возвращаемого значения |
| copy\_stud | Копирование информации о студенте | stud(student\*), pprom(student\*) | void |
| copy\_sem | Копирование информации о сессии | ptecp(semestrs\*),  ppromp(semestrs\*), | void |
| copy\_pre | Копирование информации о предмете | pptecpp(predmets\*), pprompp(predmets\*), | void |
| check\_input\_int | Проверка на целочисленность введенных данных с клавиатуры | - | int |
| check\_potok | Проверка на то, что в потоке ничего не осталось после предыдущего ввода | - | void |
| check\_potok | Проверка на то, что в потоке ничего не осталось после предыдущего ввода (используется перед cin.getline(), если ввод до этого происходил с помощью cin) | i(int) | void |
| menu | Функция содержащие меню программы | - | void |

Для более удобного вывода информации о студенте на экран была сделана перегрузка операции побитового сдвига (<<). ( ostream& operator << (ostream& out, student\* stud))

Следует перечислить друзей всех приведенных выше классов.

|  |  |
| --- | --- |
| Класс | Друзья |
| student | class file\_p; |
| class file\_crypt; |
| class metods; |
| ostream& operator << (ostream& out, student\* stud); |
| void copy\_stud(student\* stud, student\* pprom); |
| metods | void menu(); |
| file | void menu(); |
| file\_p | void menu(); |
| file\_crypt | void menu(); |

## Описание концепции работы программы

Все данные о студентах будут храниться в виде списка, поскольку заранее нельзя сказать, сколько памяти надо выделить, на всех студентов, которых введет пользователь. По этой же логике информация о сессиях студента и перечень предметов в каждой сессии будем тоже хранить в виде списка, а именно кольцевого односвязного списока.

Структура данных, представляющая собой конечное множество упорядоченных элементов (узлов), связанных друг с другом посредством указателей, называется связным списком. Каждый элемент связного списка содержит поле с данными, а также указатель (ссылку) на следующий и/или предыдущий элемент.

Ссылки – это адреса других узлов этого же типа, с которыми данный элемент логически связан. В языке Си для организации ссылок используются переменные указатели. При добавлении нового узла в такую структуру выделяется новый блок памяти и (с помощью ссылок) устанавливаются связи этого элемента с уже существующими. Для обозначения конечного элемента в цепи используется ссылка на первый узел, так как применяется кольцевой односвязный список.

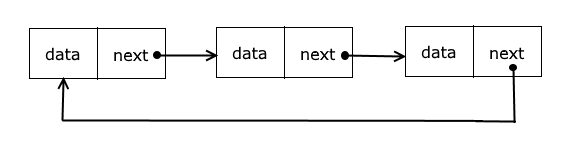


Схема кольцевого односвязного списка.

Так как неизвестно конечно число узлов списка, используется динамическое выделение памяти. Динамическое выделение памяти — это способ запроса памяти из операционной системы запущенными программами по надобности. Эта память не выделяется из ограниченной памяти стека программы, а из гораздо большего хранилища, управляемого операционной системой — кучи. Для динамического выделения памяти для одной переменной используется оператор new. Выделенный ОС адрес памяти храниться в переменной указателе. Указатель – это переменная, в которой хранится адрес другой переменной или участка памяти.

После того как список студентов был создан стоит поговорить о том, что можно с ним сделать. Начать предлагаю с изменения данных в списке.

Для поиска необходимого студента в функции (searchs) есть массив символов zach, в котором хранится номер зачетной книжки студента, введенный с клавиатуры, которого нужно найти. Для сравнения строк, а имеенно номера зачетной книжки текущего студента и номера зачетной книжки искомого студента, используется функция strcmp (прототип int strcmp(const char \*str1, const char \*str2)), которая сравнивает строки str1 и str2. Если str1< str2, то результат отрицательный, если str1 = str2, то результат равен 0, если str1> str2, то результат положительный. Тем самым ставим условие, идти по списку, пока результат этой функции не равен 0, как только ноль, возвращаем адрес элемента списка с необходимым нам студентом. Далее уже можно менять любую информацию о студенте, выбирая нужные поля для изменения. По такому же принципу происходит поиск необходимого предмета.

Стоит отметить, что функция strcpy (прототип char\* strcpy(char\*s1, const char \*s2)), которая копирует байты из строки s1 в строку s2. Эта функция необходима, когда меняются названия предметов в сессии или номер зачетной книжки, так как в этом случае перед изменением надо проверить, нет ли уже в этой сессии такого прдмета или нет ли студента с таким номером зачетной книжки, и только после этого перезаписать данные, а для строк обычное присвоение не работает, поэтому необходимо побайтово копировать строки.

Следует перейти к вопросам ООП, которые возникли при создании проекта.

В ООП существуют парадигмы – инкапсуляция, наследование, полиморфизм.

Инкапсуляция - сведение кода и данных воедино в одном объекте, получившим название класс.

Наследование - наличие в языке ООП механизма, позволяющего объектам класса наследовать характеристики более простых и общих типов. Наследование обеспечивает как требуемый уровень общности, так и необходимую специализацию.

Полиморфизм - дословный перевод с греческого "много форм". В С++ полиморфизм реализуется с помощью виртуальных функций, которые позволяют в рамках всей иерархии классов иметь несколько версий одной и той же функции. Решение о том, какая именно версия должна выполняться в данный момент, определяется на этапе выполнения программы и носит название позднего связывания.

Класс – структура, в которой вместе с данными определяются функции. Это и есть инкапсуляция в терминах ООП.

Для каждого объекта класса устанавливается область видимости либо явно – указанием уровня доступа одним из ключевых слов public, private, protected с двоеточием, либо неявно – по умолчанию. Указание области видимости относится ко всем последующим объектам класса, пока не встретится указание другой области видимости. Область видимости public разрешает доступ к объектам класса из любой части программы, в которой известен этот объект (общедоступный). Область видимости private разрешает доступ к объектам класса только из методов этого класса. Объекты с такой областью видимости называют частными. Область видимости protected определяется для защищенных объектов, она имеет смысл только в иерархической системе классов и разрешает доступ к объектам этой области из методов производных классов. В теле класса ключевое слово области видимости может использоваться неоднократно. Область видимости для объектов типа «класс» по умолчанию private.

Если это применять конкретно к данному решению. То все поля и методы описанные в классах находятся в области private, для того чтобы только друзья класса могли иметь к ним доступ. Только конструкторы и диструкторы находятся в области public, чтобы все участки программы имели возможность создавать и удалять объекты классов.

Конструктор – это метод класса, имя которого совпадает с именем класса. Конструктор вызывается автоматически после выделения памяти для переменной и обеспечивает инициализацию компонент – данных. Конструктор не имеет никакого типа (даже типа void) и не возвращает никакого значения в результате своей работы. Конструктор нельзя вызывать как обычную компонентную функцию в программе. Для класса может быть объявлено несколько конструкторов, различающихся числом и типами параметров. При этом даже если для объектного типа не определено ни одного конструктора, компилятор создает для него конструктор по умолчанию, не использующий параметров, а также конструктор ко-пирования, необходимый в том случае, если переменная объектного типа передается в конструктор как аргумент. В этом случае создаваемый объект будет точной копией аргумента конструктора.

Деструктор вызывается перед освобождением памяти, занимаемой объектной переменной, и предназначен для выполнения дополнительных действий, связанных с уничтожением объектной переменной, например, для освобождения динамической памяти, закрытия, уничтожения файлов и т.п. Деструктор всегда имеет то же имя, что и имя класса, но перед именем записывается знак ~ (тильда). Деструктор не имеет параметров и подобно конструктору не возвращает никакого значения. Таким образом, деструктор не может быть перегружен и должен существовать в классе в единственном экземпляре. Деструктор вызывается автоматически при уничтожении объекта. Таким образом, для статически определенных объектов деструктор вызывается, когда заканчивается блок программы, в котором определен объект. Для объектов, память для которых выделена динамически, деструктор вызывается при уничтожении объекта операцией delete.

В программе деструктор не имеет пустого тела только в классе student. В его теле происходит обнуление данных из полей, для того чтобы никто сторонний обращявшийся по адресу, который имел ранее удаляемый объект, не смог считать информацию хранящуюся по этому адресу ранее.

Ранее упоминалось об использовании друзей класса, для доступа к приватным элементам класса. Друзьмя класса могут быть как функции, так и классы. Использование и тех, и других наглядно показывает класс student, друзьями которого являются как классы metods, file\_p, file\_crypt, так и функцию copy\_stud, также его другом является перегруженная операция побитового сдвига.

Чтобы предоставить функции возможность выполнения действий над различными классами можно определить обычную функцию языка С++ и предоставить ей право доступа к элементам класса типа private, protected. Для этого нужно в описании класса поместить заголовок функции, перед которым поставить ключевое слово friend. Дружественная функция не является методом класса, не зависит от позиции в классе и спецификаторов прав доступа. Дружественные функции получают доступ к членам класса через указатель, передаваемый им явно. Можно сделать все функции класса Y друзьями класса X в одном объявлении.

Теперь поговорим о двух остальных парадигмах ООП вместе, так как в проекти они тестно связанны из-за использования абстрактного класса.

Наследование - наличие в языке ООП механизма, позволяющего объектам класса наследовать характеристики более простых и общих типов. То есть, к существующему классу можно что-либо добавить, или изменять его каким-либо образом для создания нового (производного) класса.

В программе наследование классов следующее.

file

file\_crypt

file\_p

То есть классы file\_p и file\_crypt являются производными от класса file.

Наследование позволяет создавать иерархию связанных типов, совместно использующих код и интерфейс. Модификатор прав доступа используется для изменения доступа к наследуемым объектам в соответствии с правилами, указанными в таблице

|  |  |  |
| --- | --- | --- |
| Доступ в классах при наследовании Доступ в базовом классе | Модификатор прав доступа | Доступ в производном классе |
| private | private | не доступны |
| private | public | не доступны |
| protected | private | private |
| protected | public | protected |
| public | private | private |
| public | public | public |

При определении производного класса не наследуются из базового:

1. конструкторы;

2. деструкторы;

3. операторы new, определенные пользователем;

4. операторы присвоения, определенные пользователем;

5. отношения дружественности.

Обычную функцию-метод можно переопределить в наследуемых классах. Однако без атрибута virtual такая функция-метод будет связана с сообщением на этапе компиляции. Атрибут virtual гарантирует позднее связывание в пределах полиморфического кластера.

Часто возникает необходимость передачи сообщений объектам, принадлежащим разным классам в иерархии. В этом случае требуется реализация механизма позднего связывания. Чтобы добиться позднего связывания для объекта, его нужно объявить как указатель или ссылку на объект соответствующего класса. Для открытых производных классов указатели и ссылки на объекты этих классов совместимыми с указателями и ссылками на объекты базового класса (т.е. к объекту производного класса можно обращаться, как будто это объект базового класса). Выбранная функция-метод зависит от класса, на объект которого указывается, но не от типа указателя.

С++ поддерживает virtual функции-методы, которые объявлены в основном классе и переопределены в порожденном классе. Иерархия классов, определенная общим наследованием, создает связанный набор типов пользователя, на которые можно ссылаться с помощью указателя базового класса. При обращении к виртуальной функции через этот указатель в С++ выбирается соответствующее функциональное определение во время выполнения. Объект, на который указывается, должен содержать в себе информацию о типе, поскольку различия между ними может быть сделано динамически. Это особенность типична для ООП кода. Каждый объект “знает” как на него должны воздействовать. Эта форма полиморфизма называется чистым полиморфизмом.

В С++ функции-методы класса с различным числом и типом параметров есть действительно различные функции, даже если они имеют одно и тоже имя. Виртуальные функции позволяют переопределять в управляемом классе функции, введенные в базовом классе, даже если число и тип аргументов то же самое. Для виртуальных функций нельзя переопределять тип функции. Если две функции с одинаковым именем будут иметь различные аргументы, С++ будет считать их различными и проигнорирует механизм виртуальных функций. Виртуальная функция обязательно метод класса.

Абстрактные классы

Если базовый класс используется только для порождения производных классов, то виртуальные функции в базовом классе могут быть "пустыми", поскольку никогда не будут вызваны для объекта базового класса. Базовый класс в котором есть хотя бы одна такая функция, называется *абстрактным*.

В проекте абстрактным классом является класс file, с чисто виртуальными функциями file\_save и file\_load, которые переопределяются в производных классах file\_p и file\_crypt.

Помимо всего выше сказанного всю информацию о студентах можно загружать и сохранять в файл. С этой целью были написаны классы file\_p и file\_crypt.

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.

Файлы бывают текстовыми и двоичными

Текстовый файл – файл, в котором каждый символ из используемого набора хранится в виде одного байта (код, соответствующий символу). Текстовые файлы разбиваются на несколько строк с помощью специального символа "конец строки". Текстовый файл заканчивается специальным символом "конец файла".

Двоичный файл – файл, данные которого представлены в бинарном виде. При записи в двоичный файл символы и числа записываются в виде последовательности байт (в своем внутреннем двоичном представлении в памяти компьютера).

Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику.

Функции библиотеки ввода-вывода языка С++, поддерживающие обмен данными с файлами на уровне потока, позволяют обрабатывать данные различных размеров и форматов, обеспечивая при этом буферизованный ввод и вывод. Таким образом, поток представляет собой этот файл вместе с предоставленными средствами буферизации.

Чтение данных из потока называется извлечением, вывод в поток – помещением (включением).

ifstream - файл ввода, производный от istream

ofstream - файл вывода, производный от ostream

fstream - файл ввода-вывода, производный от iostream

Для записи данных в файл используется метод класса ofstream write, который записывает в файл определенное количество байтов, а также метод put для записи одиночного символа. А для чтения из файла используется метод классса ifstream read, который считывает из файла определенное количество байт, а также методы peek для считывания одного символа, но не выборке его в поток, и seekg для изменения положения указателя при считывании файла.

В проекте присутствуют элементы криптографии. Инициализация криптопровайдера выполняется при помощи функции CryptAcquireContext. В качестве параметров эта функция принимает имя контейнера ключей, имя криптопровайдера, тип провайдера и флаги, определяющие тип и действия с контейнером ключей и режим работы криптопровайдера:

Криптопровайдер– это независимый модуль, содержащий библиотеку криптографических функций со стандартизованным интерфейсом. Криптопровайдер отвечает за реализацию функций интерфейса, а также играет роль хранилища для ключей всех типов. Подобная архитектура позволяет переходить от одного провайдера к другому с минимальными изменениями исходного кода, так как интерфейс (т. е. сами функции) не меняется.

Делается это с помощью функции CryptAcquireContext:

BOOL CryptAcquireContext (

HCRYPTPROV\* hCryptProvider, //дескриптор провайдера

LPCTSTR pszContainer, // имя контейнера ключей

LPCTSTR pszProvider, // имя провайдера

DWORD dwProvType, // тип провайдера

DWORD dwFlags // флаги )

где: hCryptProvider- указатель на дескриптор CSP, является выходным параметром. По завершении процесса работы с криптопровайдером(CSP), необходимо вызвать функцию CryptReleaseContext для освобождения(очистки) дескриптора и контейнера ключей.

pszContainer- имя ключевого контейнера. Это строка с нулевым завершением, которая идентифицирует контейнер ключей для CSP. Это имя не зависит от метода, используемого для хранения ключей (может быть массивом символов, строкой). Некоторые CSP хранят свои контейнеры ключей внутри (в аппаратном обеспечении), некоторые используют системный реестр, а другие-файловую систему. В большинстве случаев, когда dwFlags имеет значение CRYPT\_VERIFYCONTEXT, pszContainer должен иметь значение NULL

pszProvider- имя провайдера. Это строка с нулевым завершением, которая идентифицирует имя используемого провайдера.

dwProvType – тип провайдера:

* PROV\_RSA\_FULL
* PROV\_RSA\_AES
* PROV\_RSA\_SIG
* PROV\_RSA\_SCHANNEL
* PROV\_DSS
* PROV\_DSS\_DH
* PROV\_DH\_SCHANNEL

Функция CryptAcquireContext работает в два этапа: сначала она ищет криптопровайдер по имени и типу, указанному в аргументах, а затем контейнер ключей с заданным именем.

Базовая функция шифрования данных имеет следующее объявление:

BOOL CryptEncrypt(HCRYPTKEY hKey,

HCRYPTHAS hHash,

BOOL Final,

DWORD dwFlags,

BYTE\* pbData,

DWORD\* pdwDataLen,

DWORD dwBufLen);

Первым параметром данной функции передается хендл сессионного ключа, применяемого для шифрования. Второй параметр достаточно редко используется и предназначен для получения хеша данных одновременно с их шифрованием.

Эта функция может обрабатывать данные блоками. То есть, нет необходимости сразу загружать в память целиком весь массив данных, а лишь потом передавать ссылку на него криптографической функции. Достаточно передавать массив данных поблочно, специальным образом отметив лишь последний блок данных (это обычно нужно, чтобы криптопровайдер провел некоторые действия после использования сессионного ключа). Для указания того, что это последний блок данных, в функции CryptEncryptиспользуется третий параметр Final. Четвертый параметр служит указателем на массив входных/выходных данных. Здесь нужно сразу отметить некоторую общую схему работы с данными в Crypto API. Если возвращаемые данные могут быть любого размера (а это возможно, ведь, скажем, в алгоритме может происходить простая замена, когда одна буква кодируется четырьмя цифрами), то работа с функцией состоит из двух этапов. На первом этапе в функцию передается общий размер входных данных и NULL в качестве ссылки на сам массив выходных данных. Функция возвращает длину выходного массива данных, пользователь инициализирует память необходимого размера и лишь затем заново передает функции ссылку на этот массив. Такая же схема используется и в работе с функцией CryptEncrypt. Параметр pdwDataLenслужит для возврата размера данных, возвращаемых функцией. Параметр dwBufLenслужит для указания длины входного буфера данных. Параметр dwFlagsобычно не используется и устанавливается в 0.

*Расшифровывание*

Базовая функция расшифровывания имеет следующее описание:

BOOL CryptDecrypt(HCRYPTKEY hKey,

HCRYPTHASH hHash,

BOOL Final,

DWORD dwFlags,

BYTE\* pbData,

DWORD\* pdwDataLen);

Первым параметром данной функции передается инициализированный контекст сессионного ключа, применяемого для расшифровывания данных. Второй параметр, как и в предыдущем примере, связан, по большей части, с функцией получения и проверки цифровой подписи. Обычно он не используется и устанавливается в 0. Параметр dwFlags чаще всего не используется и также устанавливается в 0. Параметры pbDataи pdwDataLenиспользуются точно так же, как и у CryptEncrypt и представляют собой ссылку на входной/выходной массив данных и длину этого массива данных.

CryptoAPI существуют ключи двух типов:

* сессионные ключи (session keys)
* пары открытый/закрытый ключ (public/private key pairs).

Сессионные ключи- это симметричные ключи, так как один и тот же ключ применяется и для шифрования, и для расшифровки. Сессионные ключи меняются. Алгоритмы, использующие сессионные ключи (так называемые симметричные алгоритмы), – RC2, RC4, DES. Microsoft RSA Base Provider работает с 40-разрядными сессионными ключами.

Пары открытый/закрытый ключ используются в так называемых асимметричных алгоритмах шифрования. Если шифрование выполнялось одним ключом из пары, то дешифрование производится другим. Открытые (public) ключи могут передаваться другим лицам для проверки цифровых подписей и шифрования пересылаемых данных. Закрытые (private) ключи не могут быть экспортированы; они используются для создания цифровых подписей и дешифровки данных. Закрытый ключ должен быть известен только его владельцу. При шифровании с открытым ключом жизненно важна абсолютно достоверная ассоциация открытого ключа и передавшей его стороны, поскольку в обратном случае возможна подмена открытого ключа и осуществление несанкционированного доступа к передаваемым зашифрованным данным.

Для генерации ключей в CryptoAPI предусмотрены две функции –CryptGenKey и CryptDeriveKey. Первая из них генерирует ключи случайным образом, а вторая – на основе пользовательских данных. При этом гарантируется, что для одних и тех же входных данных CryptDeriveKey всегда выдает один и тот же результат. Это способ генерации ключей может быть полезен для создания симметричного ключа шифрования на базе пароля.

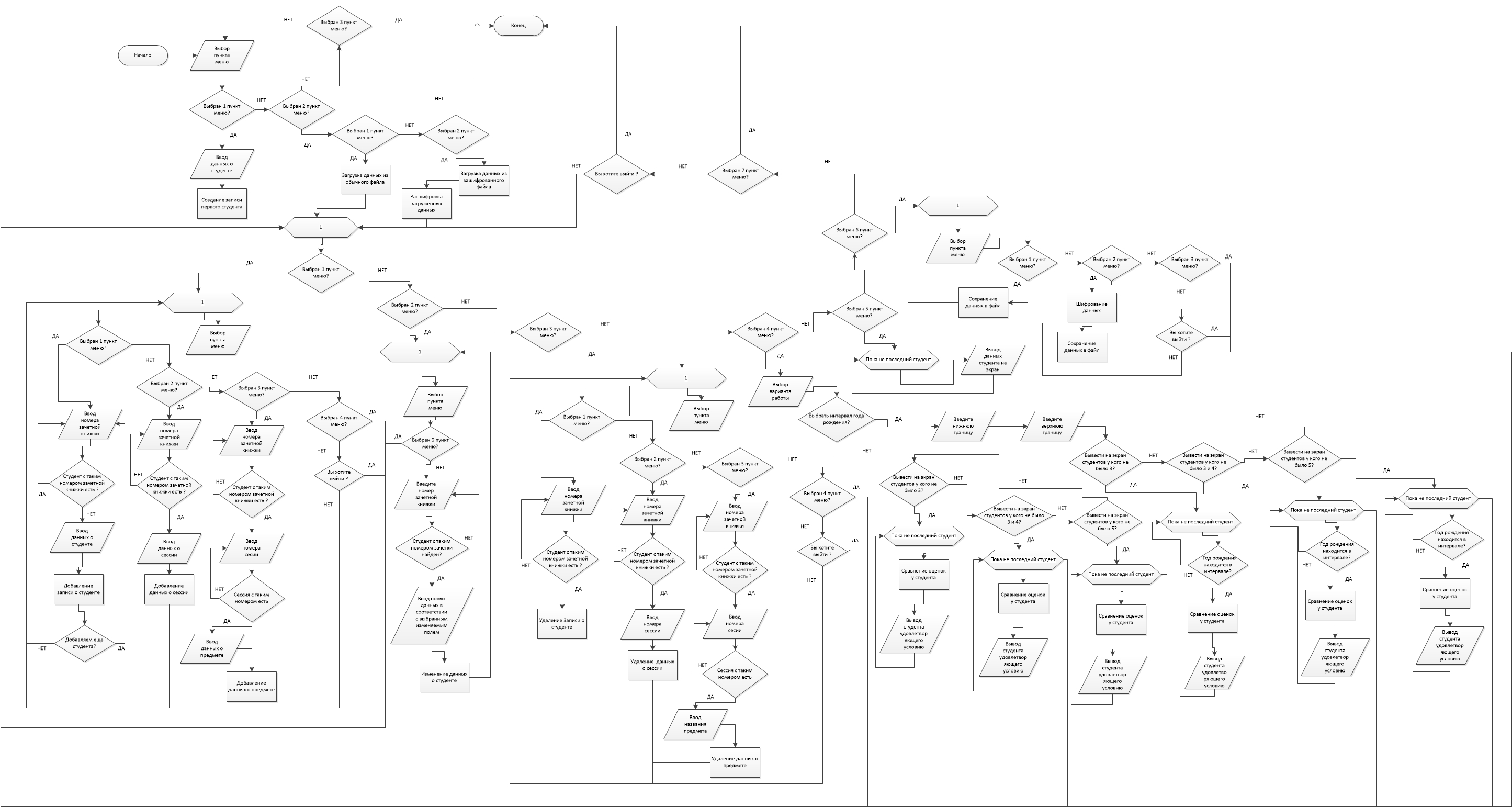
BOOL WINAPI CryptGenKey (HCRYPTPROV hProv, ALG\_ID Algid, DWORD dwFlags, HCRYPTKEY\* phKey);

Первый и четвертый параметры говорят сами за себя. Вторым параметром передается идентификатор алгоритма шифрования, для которого генерируется ключ. При генерации ключевых пар RSA для шифрования и подписи используются специальные значения AT\_KEYEXCHANGE и AT\_SIGNATURE. Третий параметр задает различные опции ключа, которые зависят от алгоритма и провайдера.

Обмен ключами в CryptoAPI реализуется с помощью функций CryptExportKey и CryptImportKey .

В проекте для работы с криптографией надо подключить криптопровайдер с помощью CryptAcquireContext, далее сгенирировать сессионный ключ с помощью CryptGenKey, причем третий параметр должен быть CRYPT\_EXPORTABLE, чтобы потом можно было его экспортировать. Далее с помощью CryptEncrypt шифруем, после шифрования данных необходимо экспортировать сесионный ключ, для того чтобы его можно было экспортировать без шифрования в функции CryptExportKey третий параметр должен быть PLAINTEXTKEYBLOB, а не SIMPLEBLOB, так как последний обязательно требует шифрования сессионного ключа. Причем CryptExportKey работает в два этапа, сначала делаем пятый параметр в NULL, чтобы узнать размер ключа и сохраняем его в переменную указанную в шестом параметре, далее уже зная длину, экспортируем ключ и записываем в файл ключа. При дешифровке также подключаем криптопровайдер, считываем ключ из файла и импортируем его в контейнер ключей. После производим дешифровку данных считанных из файла с помощью данного ключа.

# Блок схема программы



# Исходные данные

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| id | ФИО | Дата рождения | Пол | Номер зачетной книжки | Группа | Год поступления | кафедра | факультет |
| 1 | Макарова Виктория Александровна | 25.06.2001 | Ж | 19К0450 | КМБО-01-19 | 2019 | ВМ | Институт Кибернетики |
| 2 | Новиков Дмитрий Олегович | 16.02.2001 | М | 18К0080 | КББО-01-18 | 2018 | БСТ | Институт Кибернетики |
| 3 | Карпова Мария Ивановна | 27.09.2002 | Ж | 19Б0597 | БАСО-01-19 | 2019 | КБ-1 | Институт комплексной безопасности и специального приборостроения |
| 4 | Кынёв Сергей Витальевич | 05.05.1999 | М | 17Б0872 | БББО-04-17 | 2017 | КБ-8 | Институт комплексной безопасности и специального приборостроения |
| 5 | Невская Мирослава Олеговна | 15.07.2000 | Ж | 19Б1025 | БИСО-01-19 | 2019 | КБ-2 | Институт комплексной безопасности и специального приборостроения |
| 6 | Герасимов Никита Антонович | 30.09.2001 | М | 19Б0752 | БСБО-01-19 | 2019 | КБ-3 | Институт комплексной безопасности и специального приборостроения |
| 7 | Пенькова Елена Павловна | 09.12.1999 | Ж | 16Э0053 | ЭЛБО-06-16 | 2016 | ОЭПС | Физико-технологический институ |
| 8 | Фелосеев Леонид Егорович | 18.08.2002 | М | 19Э0476 | ЭЭБО-05-19 | 2019 | ИЭ | Физико-технологический институ |
| 9 | Василевский Владимир Дмитриевич | 22.06.2000 | М | 18Р0286 | РССО-01-18 | 2018 | РСПИ | Институт радиотехнических и телекоммуникационных систем |
| 10 | Захарчук Ольга Ивановна | 26.06.2001 | Ж | 19Р0985 | РРБО-02-19 | 2019 | ТРР | Институт радиотехнических и телекоммуникационных систем |

Зачетная книжка 1 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 4 |
| Мат. анализ | 5 |
| Дискр. математика | 4 |
| Основы алгоритмики | 3 |
| 2 | НИ работы | 5 |
| ОО программирования | 4 |
| Методы мат. анализа | 5 |
| Ин. язык | 5 |
| Правоведение | 5 |

Зачетная книжка 2 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 5 |
| Мат. анализ | 5 |
| Дискр. математика | 4 |
| Основы алгоритмики | 4 |
| 2 | НИ работы | 4 |
| ОО программирования | 5 |
| Методы мат. анализа | 4 |
| Ин. язык | 5 |
| Правоведение | 4 |
| 3 | Комплексный анализ | 5 |
| Арх. компьютеров | 5 |
| Методы КА | 4 |
| Диф. уравнения | 4 |
| 4 | Философия | 5 |
| Базы данных | 4 |
| Автоматы и алгоритмы | 5 |
| Алгебра и геометрия | 5 |

Зачетная книжка 3 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Линейная алгебра | 5 |
| Мат. анализ | 5 |
| Физика | 5 |
| Основы ИБ | 5 |
| 2 | Практика | 5 |
| Языки программирования | 5 |
| Мат. анализа | 5 |
| Ин. язык | 5 |
| Правоведение | 5 |

Зачетная книжка 4 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 5 |
| Мат. анализ | 3 |
| Физика | 4 |
| Основы ИБ | 4 |
| 2 | Стандарты ИБ | 4 |
| Языки программирования | 5 |
| Мат. анализа | 4 |
| Ин. язык | 5 |
| Правоведение | 4 |
| 3 | Анализ сложности алгоритмов | 3 |
| Арх. компьютеров | 4 |
| Физика | 4 |
| Диф. уравнения | 4 |
| 4 | Философия | 5 |
| Базы данных | 4 |
| Организация ЭВМ | 5 |
| Алгебра и геометрия | 4 |
| 5 | Криптографические методы ЗИ | 3 |
| Безопасность систем БД | 4 |
| БЖД | 5 |
| Угрозы ИБ АС | 4 |
| Аппаратные средства вычислительной техники | 4 |
| 6 | Теория информации | 5 |
| Нац. система ИБ | 4 |
| Практика | 3 |
| Основы Управленческой деятельности | 3 |

Зачетная книжка 5 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Линейная алгебра и геометрия | 5 |
| Мат. анализ | 4 |
| Физика | 3 |
| Основы ИБ | 4 |
| 2 | Стандарты ИБ | 4 |
| Языки программирования | 4 |
| Мат. анализа | 4 |
| Ин. язык | 5 |
| Правоведение | 4 |

Зачетная книжка 6 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Линейная алгебра и геометрия | 5 |
| Мат. анализ | 4 |
| Физика | 4 |
| Технология программирования | 4 |
| 2 | Архитектура ИС | 4 |
| Дискретная математика | 4 |
| Мат. анализа | 4 |
| Ин. язык | 5 |
| Правоведение | 4 |

Зачетная книжка 7 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 5 |
| Мат. анализ | 5 |
| Физика | 5 |
| Программирование | 5 |
| 2 | Физика | 5 |
| Программирование | 5 |
| Мат. анализа | 5 |
| Ин. язык | 5 |
| Правоведение | 5 |
| 3 | История | 5 |
| Программирование | 5 |
| Физика | 5 |
| Диф. уравнения | 5 |
| 4 | Философия | 5 |
| Пром. Применение лазеров | 5 |
| Теория вероятности и МС | 5 |
| Алгебра и геометрия | 5 |
| 5 | ФО лазерной тех. | 5 |
| Электроника | 5 |
| Ин. язык | 5 |
| Мат. анализа | 5 |
| ГиФ оптика | 5 |
| 6 | Материалы квант. и опт. эл-ки | 5 |
| Оптико-электронные системы | 5 |
| Практика | 5 |
| Пром. Дизайн ОЭП | 5 |
| 7 | Оптика лазеров | 5 |
| Прикладная оптика | 5 |
| Гражд. оборона | 5 |
| Источники и приемники ОИ | 5 |
| 8 | Электронные системы СН | 5 |
| Приборы А/Т диагностики | 5 |
| Методы и средства ОДСН | 5 |
| Системы АП и ВП В О/Т | 5 |

Зачетная книжка 8 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 4 |
| Мат. анализ | 5 |
| Физика | 5 |
| Программирование | 4 |
| 2 | Физика | 5 |
| Программирование | 4 |
| Мат. анализ | 4 |
| Ин. язык | 5 |
| Правоведение | 5 |

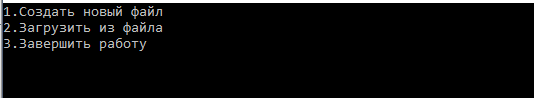
Зачетная книжка 9 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 5 |
| Мат. анализ | 5 |
| Физика | 4 |
| Языки программирования | 4 |
| 2 | Практика | 4 |
| Языки программирования | 5 |
| Мат. анализ | 4 |
| Ин. язык | 5 |
| Правоведение | 4 |
| 3 | Теория вероятн. и МС | 5 |
| Радиоматериалы и Р/К | 5 |
| Основы теории цепей | 4 |
| Экономика | 4 |
| 4 | Радиоинжиниринг | 5 |
| Основы ТН | 4 |
| Метрология СиС | 5 |
| Экономика | 5 |

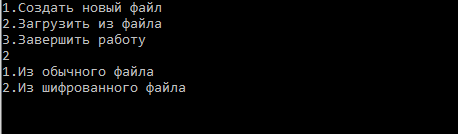
Зачетная книжка 10 студента.

|  |  |  |
| --- | --- | --- |
| Семестр | Предмет | Оценка |
| 1 | Алгебра и геометрия | 4 |
| Мат. анализ | 3 |
| Физика | 4 |
| Языки программирования | 4 |
| 2 | Практика | 4 |
| Языки программирования | 4 |
| Мат. анализа | 4 |
| Ин. язык | 4 |
| Правоведение | 4 |

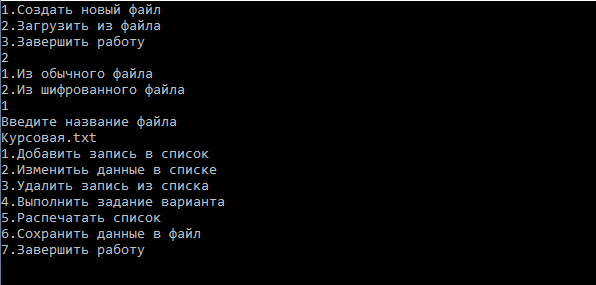
# Выходные данные

При запуске программы появляется первое меню.

При выборе первого пункта меню запускается создание первой записи о студенте.

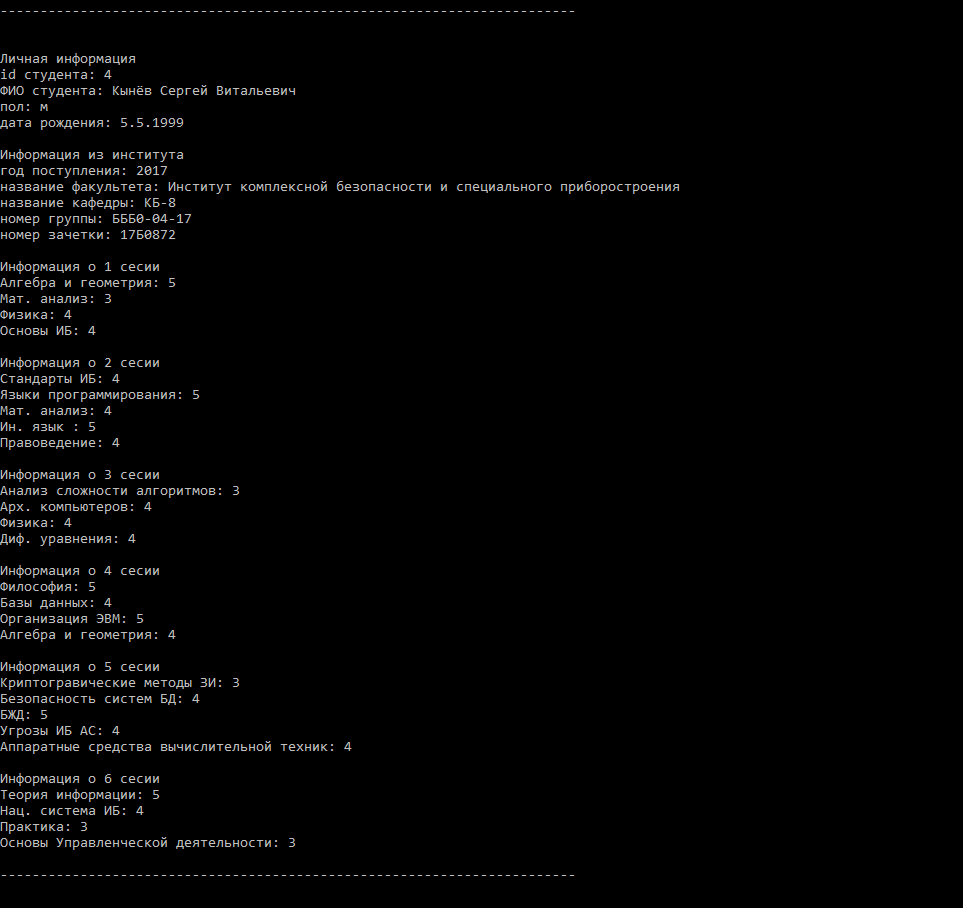
Выбирая второй пункт меню, открывается выбор.

Выберем второй пункт меню и загрузим данные, приведенные в таблицах, из файла.



Вводим название файла, и после загрузки файла появляется следующее меню для работы со списком студентов.

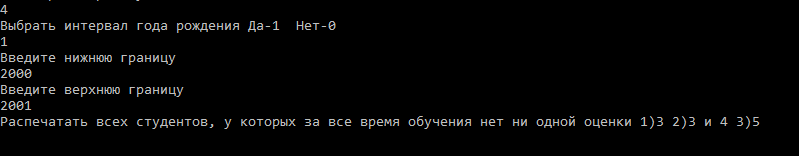
Если выбрать 5 пункт меню, то выведется весь список студентов со всей информацией о них. Вывод одной записи о студенте выглядит следующим образом.



Чтобы выполнить задание данное вариантом нужно в меню выбрать 4 пункт.

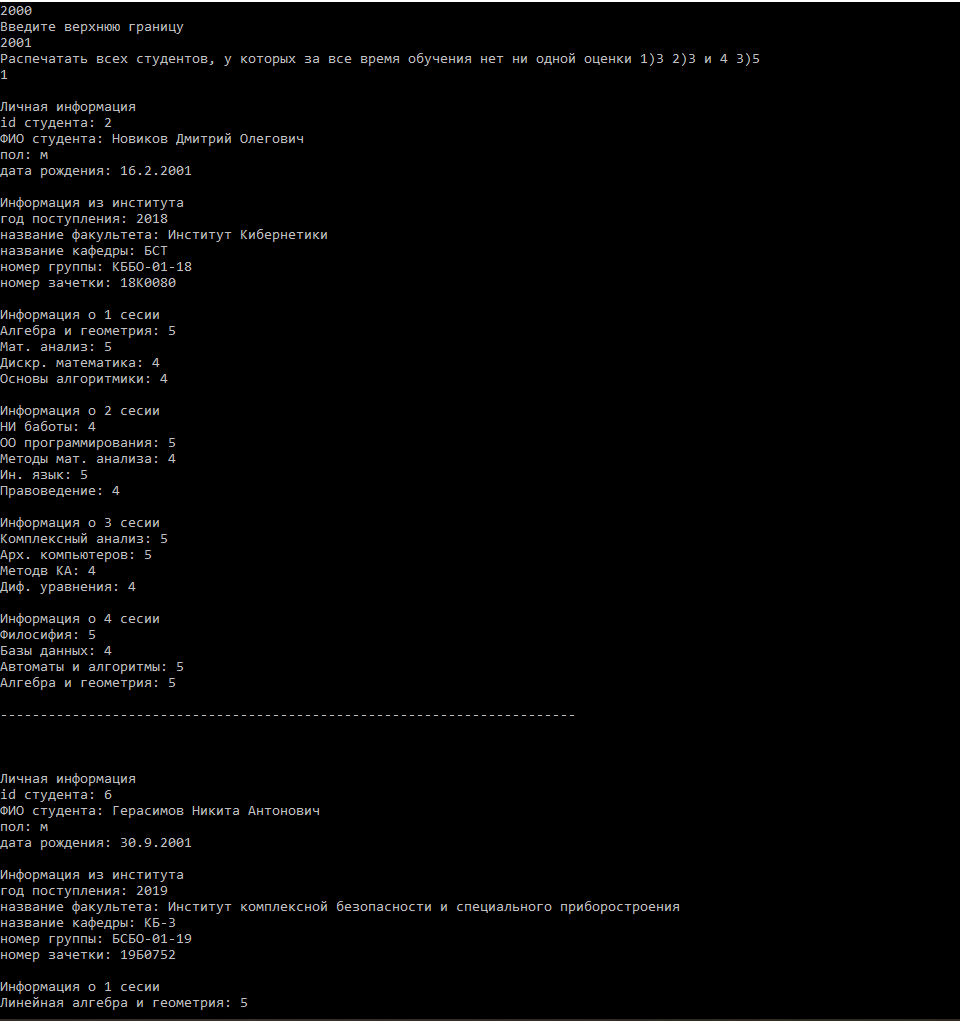
При выборе этого пункта будет задан вопрос:задание

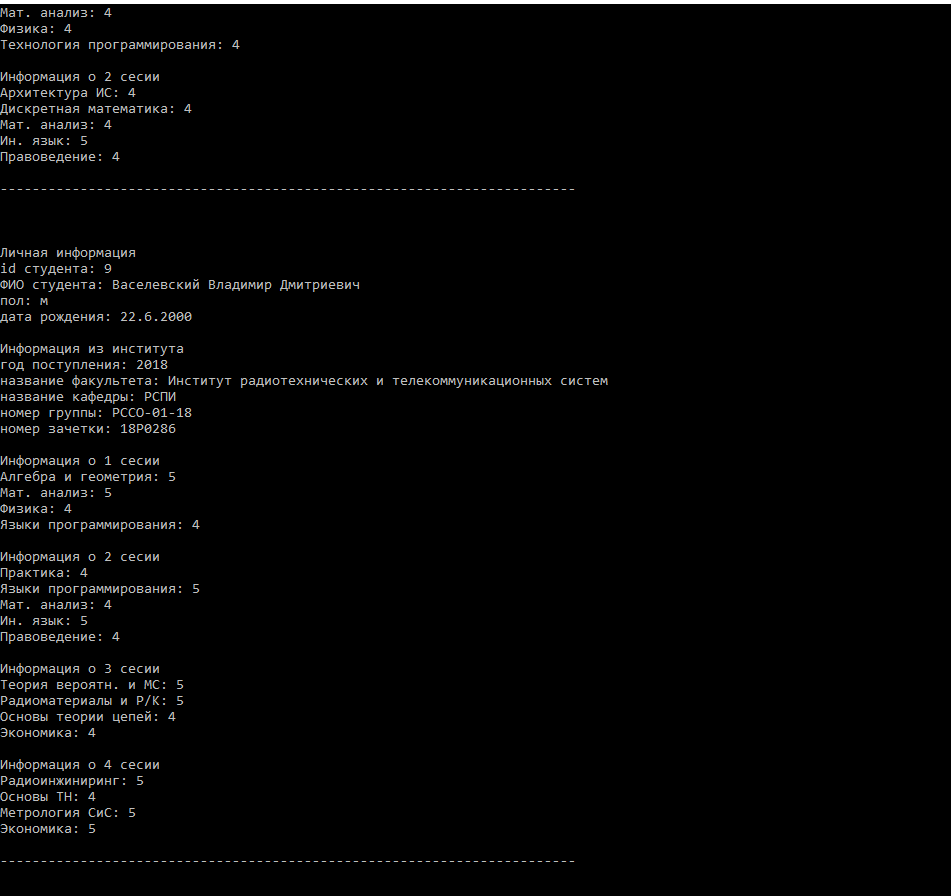
Выберем интервал между 2000 и 2001 годами.



Теперь предоставляется выбор, какой из 3 подпунктов задания выполнить.

Выберем 1 подпункт. И прогрмамма вывела.





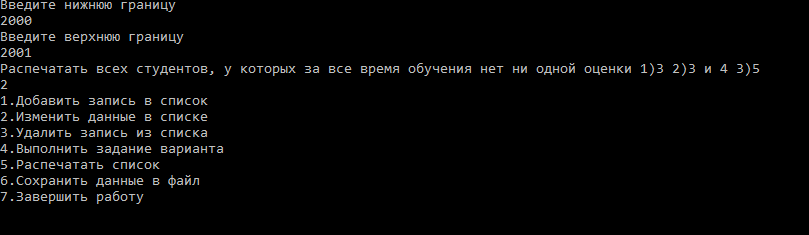
Результатом программы является вывод информации о студентах год рождения, которых находится в интервале между 2000 и 2001 годом, и у кого нет оценки 3 за все время обучения, таких студентов было найдено трое.

Для проверки правильности работы вернемся к таблице с данными и сами, вручную, найдем студентов год рождения, которых находится в интервале между 2000 и 2001 годом, и у кого нет оценки 3 за все время обучения.

|  |  |
| --- | --- |
| id | ФИО |
| 2 | Новиков Дмитрий Олегович |
| 6 | Герасимов Никита Антонович |
| 9 | Василевский Владимир Дмитриевич |

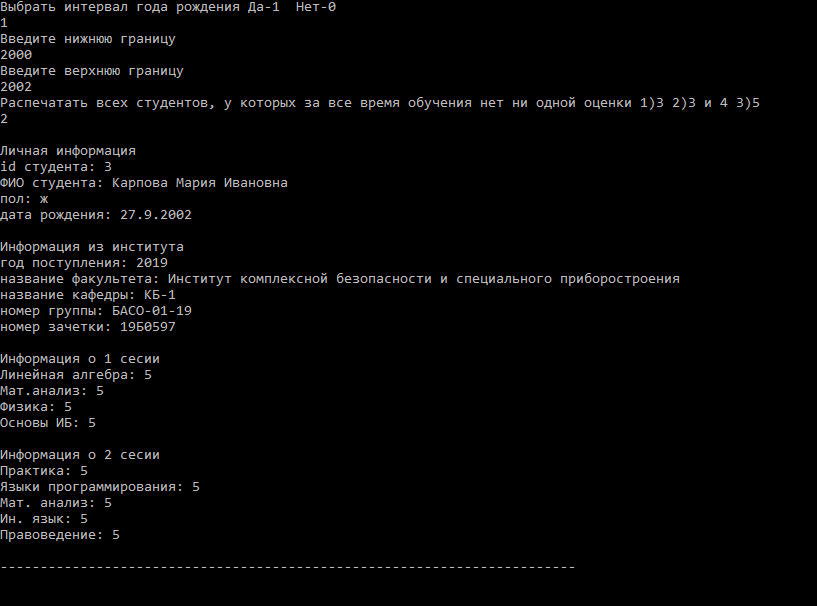
Оценки можно посмотреть в таблицах выше.

Выберем 2 подпункт, с этим же интервалом года рождения.



Программа нам никого не выдала, так как все отличники не входят в этот интерван годов рождения. Единственные отличники 2002 и 1999 годов рождения.

Теперь изменим интервал на между 2000 и 2002 и выберем 2 подпункт.

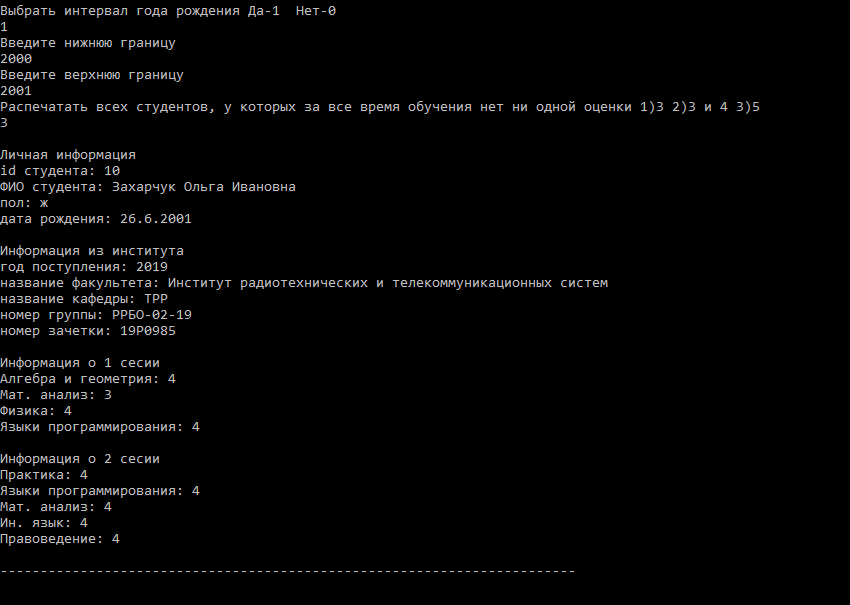


Результатом программы является вывод информации о студентах год рождения, которых находится в интервале между 2000 и 2002 годом, и у кого нет оценки 3 и 4 за все время обучения, такой всего один студент.

Для проверки правильности работы вернемся к таблице с данными и сами, вручную, найдем студентов год рождения, которых находится в интервале между 2000 и 2002 годом, и у кого нет оценки 3 и 4 за все время обучения.

|  |  |
| --- | --- |
| id | ФИО |
| 3 | Карпова Мария Ивановна |

Теперь интервал года рождения межуд 2000 и 2001 и выберем 3 подпункт.

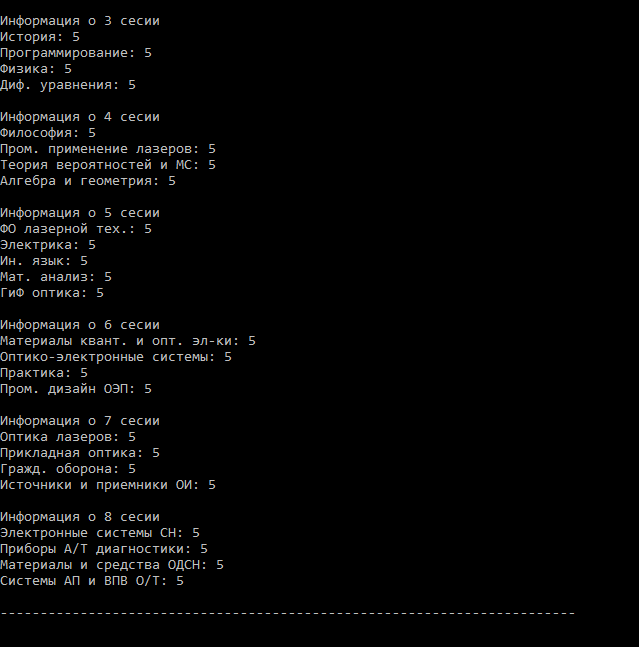
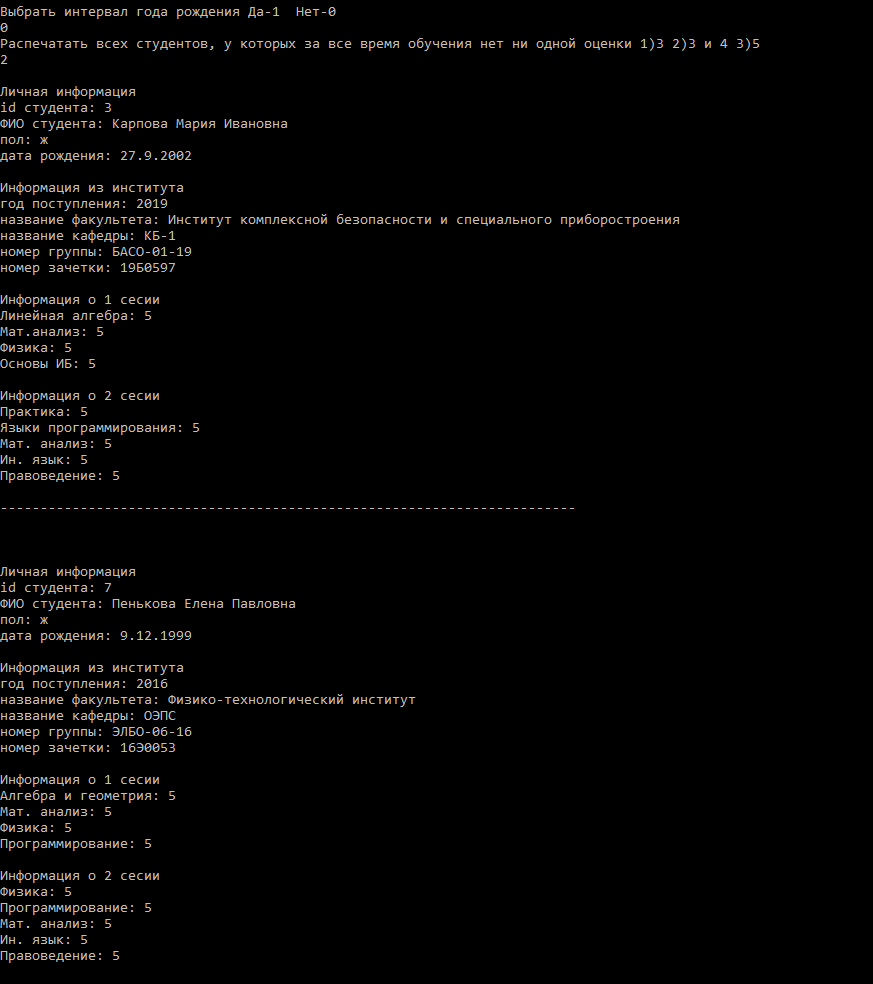


Результатом программы является вывод информации о студентах год рождения, которых находится в интервале между 2000 и 2001 годом, и у кого нет оценки 5 за все время обучения, такой всего один студент.

Для проверки правильности работы вернемся к таблице с данными и сами, вручную, найдем студентов год рождения, которых находится в интервале между 2000 и 2001 годом, и у кого нет оценки 5 за все время обучения.

|  |  |
| --- | --- |
| id | ФИО |
| 10 | Захарчук Ольга Ивановна |

Теперь не будем вводить интерала года рождения и выберем 2 подпункт.



Результатом программы является, вывод информации о студентах без задания интервала года рождения и у кого нет оценки 3 и 4 за все время обучения, такой всего один студент.

Для проверки правильности работы вернемся к таблице с данными и сами, вручную, найдем студентов, у которых нет оценки 3 и 4 за все время обучения.

|  |  |
| --- | --- |
| id | ФИО |
| 3 | Карпова Мария Ивановна |
| 7 | Пенькова Елена Павловна |

# Вывод

В ходе работы над курсовой работой была создана программа, которая находит всех студентов и выводить информацию о них на экран, у которых за время обучения нет ни одной оценки три, три и четыре, или пять, вводимых по желанию пользователя, с указанием интервала года рождения, а также реализованы возможности по добовлению и удалению записей, изменению информации, сохранению и загрузки информации в/из файла, и некторые элементы криптографии, с применением ООП.

# Листинг программы

## student.h

#pragma once

#include <iostream>

#include <cstring>

using namespace std;

struct fios {

char surname[21];

char name[21];

char otchestvo[21];

};

struct birthdays {

int day;

int mounth;

int year;

};

struct univesitys {

int year\_post;

char facultet[101];

char kafedra[71];

char group[21];

char number\_zachet[21];

};

struct predmets {

int id\_predmet;

char name\_predmet[51];

int mark;

predmets\* next\_predmet;

};

struct semestrs {

int id\_semestr;

predmets\* predmet;

semestrs\* next\_semestr;

};

class student {

private:

int id\_student;

char pol;

fios fio;

birthdays birthday;

univesitys univesity;

semestrs\* semestr;

student\* next\_student;

public:

student();

~student();

friend class file\_p;

friend class file\_crypt;

friend class metods;

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

friend void copy\_stud(student\* stud, student\* pprom);

};

## student.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include"student.h"

student::student() {

id\_student = 0;

pol = ' ';

strcpy(fio.name, " ");

strcpy(fio.surname, " ");

strcpy(fio.otchestvo, " ");

birthday.day = 0;

birthday.mounth = 0;

birthday.year = 0;

strcpy(univesity.facultet, " ");

strcpy(univesity.group, " ");

strcpy(univesity.kafedra, " ");

strcpy(univesity.number\_zachet, " ");

univesity.year\_post = 0;

next\_student = nullptr;

semestr = nullptr;

};

student::~student(){

id\_student = 0;

pol = ' ';

strcpy(fio.name, " ");

strcpy(fio.surname, " ");

strcpy(fio.otchestvo, " ");

birthday.day = 0;

birthday.mounth = 0;

birthday.year = 0;

strcpy(univesity.facultet, " ");

strcpy(univesity.group, " ");

strcpy(univesity.kafedra, " ");

strcpy(univesity.number\_zachet, " ");

univesity.year\_post = 0;

next\_student = nullptr;

semestr = nullptr;

cout << "Запись о студенте удалена" << endl;

}

## check.h

#pragma once

#include<iostream>

using namespace std;

int check\_input\_int();

void check\_potok();

void check\_potok(int);

## check.cpp

#include "check.h"

int check\_input\_int() {

int check;

do {

cin >> check;

if (!cin) {

cout << "Вы ввели не целое число " << endl;

cin.clear();

cin.ignore(1000, '\n');

}

else break;

} while (1);

return check;

}

void check\_potok() {

if (!cin) {

cin.clear();

cin.ignore(1000, '\n');

}

}

void check\_potok(int i) {

if (!cin) {

cin.clear();

cin.ignore(1000, '\n');

}

else cin.ignore();

}

## metods.h

#pragma once

#include"student.h"

#include"check.h"

class metods {

student\* create();

void create(student\* pstarts);

int search(student\* pstart, semestrs\* pprom, int id);

void menu1\_for\_zadani(student\* pstart);

int check\_mark(student\* stud, int nom1, int nom2);

void print(student\* pstart);

student\* searchs(student\* pprom);

int searchp(semestrs\* pprom, predmets\* ppromp, char p[]);

void change(student\* pstart);

student\* delet(student\* pstart);

void dobav\_semestr(student\* pstart);

void dobav\_predmet(student\* pstart);

public:

metods();

~metods();

friend void menu();

};

## metods.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include"metods.h"

student\* metods::create() {

student\* pstart = new student;

pstart->id\_student = 1;

check\_potok(1);

cout << "Введите фамилию " << endl;

cin.getline(pstart->fio.surname, 20);

check\_potok();

cout << "Введите имя " << endl;

cin.getline(pstart->fio.name, 20);

check\_potok();

cout << "Введите отчество " << endl;

cin.getline(pstart->fio.otchestvo, 20);

check\_potok();

cout << "Введите пол 'м' или 'ж'" << endl;

do {

cin >> pstart->pol;

if (pstart->pol == 'м' || pstart->pol == 'ж') {

check\_potok();

break;

}

cout << "Вы ввели некоректное значение для пола" << endl;

check\_potok();

} while (1);

cout << "Введите день рождения " << endl;

do {

cin >> pstart->birthday.day;

if (pstart->birthday.day >= 1 && pstart->birthday.day <= 31) {

check\_potok();

break;

}

cout << "Вы ввели некоректный день" << endl;

check\_potok();

} while (1);

cout << "Введите месяц рождения " << endl;

do {

cin >> pstart->birthday.mounth;

if (pstart->birthday.mounth >= 1 && pstart->birthday.mounth <= 12) {

check\_potok();

break;

}

cout << "Вы ввели некоректный месяц" << endl;

check\_potok();

} while (1);

cout << "Введите год рождения " << endl;

do {

cin >> pstart->birthday.year;

if (pstart->birthday.year >= 1910 && pstart->birthday.year <= 2004) {

check\_potok();

break;

}

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

cout << "Введите год поступления" << endl;

do {

cin >> pstart->univesity.year\_post;

if (pstart->birthday.year + 16 <= pstart->univesity.year\_post) {

check\_potok(1);

break;

}

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

cout << "Введите название факультета " << endl;

cin.getline(pstart->univesity.facultet, 100);

check\_potok();

cout << "Введите казвание кафедры " << endl;

cin.getline(pstart->univesity.kafedra, 70);

check\_potok();

cout << "Введите название группы " << endl;

cin.getline(pstart->univesity.group, 20);

check\_potok();

cout << "Введите номер зачетной книжки" << endl;

cin.getline(pstart->univesity.number\_zachet, 20);

check\_potok();

int flug, ocen, id;

semestrs\* ptec, \* pprom;

predmets\* ptecp, \* ppromp, \* pstartp;

char p[51];

pstart->semestr = new semestrs;

pprom = pstart->semestr;

pprom->next\_semestr = pstart->semestr;

for (int i = 1; i < 10; i++) {

do {

do {

cout << "Введите номер сессии" << endl;

id = check\_input\_int();

if (id > 0 && id < 10) break;

else cout << "Вы ввели не корректный номер сессии" << endl;

} while (1);

} while (search(pstart, pstart->semestr, id));

pprom->id\_semestr = id;

pprom->predmet = new predmets;

ppromp = pprom->predmet;

pstartp = ppromp;

ppromp->next\_predmet = ppromp;

check\_potok(1);

for (int j = 1; j < 11; j++) {

do {

cout << "Введите название предмета " << endl;

cin.getline(p, 50);

check\_potok();

} while (searchp(pprom, pprom->predmet, p));

strcpy(ppromp->name\_predmet, p);

cout << "Оценка по нему " << endl;

do {

ocen = check\_input\_int();

if (ocen > 1 && ocen < 6) {

ppromp->mark = ocen;

check\_potok();

break;

}

else {

cout << "Вы ввели число не соответствующие оценкам по пяти бальной шкале " << endl;

check\_potok();

}

} while (1);

cout << "Есть еще предметы в этой сессии? Да-1 Нет-0 " << endl;

flug = check\_input\_int();

if (flug < 1) {

ppromp->next\_predmet = pstartp;

break;

}

if (j < 10) {

ptecp = new predmets;

ppromp->next\_predmet = ptecp;

ppromp = ptecp;

ppromp->next\_predmet = pprom->predmet;

check\_potok(1);

}

else {

cout << "Введено максимальное количество предметов для одной ссесии" << endl;

ppromp->next\_predmet = pprom->predmet;

check\_potok();

}

}

cout << "Есть еще сессии у данного студента ? Да-1 Нет-0 " << endl;

flug = check\_input\_int();

if (flug < 1) {

pprom->next\_semestr = pstart->semestr;

break;

}

if (i < 9) {

ptec = new semestrs;

pprom->next\_semestr = ptec;

pprom = ptec;

pprom->next\_semestr = pstart->semestr;

check\_potok();

}

else {

cout << "Введено максимальное количество сессий" << endl;

pprom->next\_semestr = pstart->semestr;

}

}

pstart->next\_student = pstart;

return pstart;

}

void metods::create(student\* pstarts) {

char zach[21];

int flug;

student\* pprompst = pstarts;

while (1) {

check\_potok(1);

do {

cout << "Введите номер зачетки студента" << endl;

cin.getline(zach, 20);

do {

if (strcmp(pprompst->univesity.number\_zachet, zach) == 0) {

cout << "Студент с таким номером заченой книжки уже есть" << endl;

break;

}

pprompst = pprompst->next\_student;

} while (pstarts != pprompst);

check\_potok();

if (pprompst == pstarts) break;

pprompst = pstarts;

} while (1);

student\* pproms = pstarts;

do {

pproms = pproms->next\_student;

} while (pproms->next\_student != pstarts);

student\* ptecs = new student;

strcpy(ptecs->univesity.number\_zachet, zach);

ptecs->id\_student = pproms->id\_student + 1;

cout << "Введите фамилию " << endl;

cin.getline(ptecs->fio.surname, 20);

check\_potok();

cout << "Введите имя " << endl;

cin.getline(ptecs->fio.name, 20);

check\_potok();

cout << "Введите отчество " << endl;

cin.getline(ptecs->fio.otchestvo, 20);

check\_potok();

cout << "Введите пол 'м' или 'ж'" << endl;

do {

cin >> ptecs->pol;

if (ptecs->pol == 'м' || ptecs->pol == 'ж') {

check\_potok();

break;

}

cout << "Вы ввели некоректное значение для пола" << endl;

check\_potok();

} while (1);

cout << "Введите день рождения " << endl;

do {

cin >> ptecs->birthday.day;

if (ptecs->birthday.day >= 1 && ptecs->birthday.day <= 31) {

check\_potok();

break;

}

cout << "Вы ввели некоректный день" << endl;

check\_potok();

} while (1);

cout << "Введите месяц рождения " << endl;

do {

cin >> ptecs->birthday.mounth;

if (ptecs->birthday.mounth >= 1 && ptecs->birthday.mounth <= 12) {

check\_potok();

break;

}

cout << "Вы ввели некоректный месяц" << endl;

check\_potok();

} while (1);

cout << "Введите год рождения " << endl;

do {

cin >> ptecs->birthday.year;

if (ptecs->birthday.year >= 1910 && ptecs->birthday.year <= 2004) {

check\_potok();

break;

}

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

cout << "Введите год поступления" << endl;

do {

cin >> ptecs->univesity.year\_post;

if (ptecs->birthday.year + 16 <= ptecs->univesity.year\_post) {

check\_potok(1);

break;

}

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

cout << "Введите название факультета " << endl;

cin.getline(ptecs->univesity.facultet, 100);

check\_potok();

cout << "Введите казвание кафедры " << endl;

cin.getline(ptecs->univesity.kafedra, 70);

check\_potok();

cout << "Введите название группы " << endl;

cin.getline(ptecs->univesity.group, 20);

check\_potok();

int flug, ocen, id;

semestrs\* ptec, \* pprom;

predmets\* ptecp, \* ppromp, \* pstartp;

char p[51];

ptecs->semestr = new semestrs;

pprom = ptecs->semestr;

pprom->next\_semestr = ptecs->semestr;

for (int i = 1; i < 10; i++) {

do {

do {

cout << "Введите номер сессии" << endl;

id = check\_input\_int();

if (id > 0 && id < 10) break;

else {

cout << "Вы ввели не корректный номер сессии" << endl;

check\_potok();

}

} while (1);

} while (search(ptecs, ptecs->semestr, id));

pprom->id\_semestr = id;

pprom->predmet = new predmets;

ppromp = pprom->predmet;

pstartp = ppromp;

ppromp->next\_predmet = ppromp;

check\_potok(1);

for (int j = 1; j < 11; j++) {

do {

cout << "Введите название предмета " << endl;

cin.getline(p, 50);

check\_potok();

} while (searchp(pprom, pprom->predmet, p));

strcpy(ppromp->name\_predmet, p);

cout << "Оценка по нему " << endl;

do {

ocen = check\_input\_int();

if (ocen > 1 && ocen < 6) {

ppromp->mark = ocen;

break;

}

else {

cout << "Вы ввели число не соответствующие оценкам по пяти бальной шкале " << endl;

check\_potok();

}

} while (1);

cout << "Есть еще предметы в этой сессии? Да-1 Нет-0 " << endl;

flug = check\_input\_int();

if (flug < 1) {

ppromp->next\_predmet = pstartp;

break;

}

if (j < 10) {

ptecp = new predmets;

ppromp->next\_predmet = ptecp;

ppromp = ptecp;

ppromp->next\_predmet = pprom->predmet;

check\_potok(1);

}

else {

cout << "Введено максимальное количество предметов для одной ссесии" << endl;

ppromp->next\_predmet = pprom->predmet;

check\_potok();

}

}

cout << "Есть еще сессии у данного студента ? Да-1 Нет-0 " << endl;

flug = check\_input\_int();

if (flug < 1) {

pprom->next\_semestr = ptecs->semestr;

break;

}

if (i < 9) {

ptec = new semestrs;

pprom->next\_semestr = ptec;

pprom = ptec;

pprom->next\_semestr = ptecs->semestr;

check\_potok();

}

else {

cout << "Введено максимальное количество сессий" << endl;

pprom->next\_semestr = ptecs->semestr;

check\_potok();

}

}

pproms->next\_student = ptecs;

cout << "Добавлять еще студента ? Да-1 Нет-0" << endl;

flug = check\_input\_int();

ptecs->next\_student = pstarts;

if (flug < 1) break;

pproms = ptecs;

}

}

int metods::search(student\* pstart, semestrs\* pprom, int id) {

do {

if (pprom->id\_semestr == id) {

cout << "Сессия с таким номер уже есть" << endl;

return 1;

}

pprom = pprom->next\_semestr;

} while (pstart->semestr != pprom);

return 0;

}

int metods::searchp(semestrs\* pprom, predmets\* ppromp, char p[]) {

do {

if (strcmp(ppromp->name\_predmet, p) == 0) {

cout << "Такой предмет уже есть" << endl;

return 1;

}

ppromp = ppromp->next\_predmet;

} while (pprom->predmet != ppromp);

return 0;

}

int metods::searchp(semestrs\* pprom, predmets\* ppromp, char p[]) {

do {

if (strcmp(ppromp->name\_predmet, p) == 0) {

cout << "Такой предмет уже есть" << endl;

return 1;

}

ppromp = ppromp->next\_predmet;

} while (pprom->predmet != ppromp);

return 0;

}

student\* metods::searchs(student\* pprom) {

char zach[21];

check\_potok(1);

do {

cout << "Введите номер зачетки студента" << endl;

cin.getline(zach, 20);

student\* pstart = pprom;

do {

if (strcmp(pprom->univesity.number\_zachet, zach) == 0) {

return pprom;

}

pprom = pprom->next\_student;

} while (pstart != pprom);

cout << "Студента с таким номером заченой книжки не был найден, проверьте правильность ввода данных" << endl;

check\_potok();

} while (1);

}

int metods::check\_mark(student\* stud, int nom1, int nom2) {

semestrs\* pstart = stud->semestr;

semestrs\* pprom = stud->semestr;

do {

predmets\* pstartp = stud->semestr->predmet;

predmets\* ppromp = stud->semestr->predmet;

do {

if (ppromp->mark == nom1 || ppromp->mark == nom2) return 0;

ppromp = ppromp->next\_predmet;

} while (pstartp != ppromp);

pprom = pprom->next\_semestr;

} while (pstart != pprom);

return 1;

}

void metods::menu1\_for\_zadani(student\* stud) {

int low, high, flug;

cout << "Выбрать интервал года рождения Да-1 Нет-0" << endl;

check\_potok();

if (check\_input\_int()) {

do {

cout << "Введите нижнюю границу" << endl;

check\_potok();

low = check\_input\_int();

if (low >= 1910 && low <= 2004) break;

else cout << "Вы ввели некоректные данные" << endl;

} while (1);

do {

cout << "Введите верхнюю границу" << endl;

check\_potok();

high = check\_input\_int();

if (high <= 2004 && high >= low) break;

else cout << "Вы ввели некоректные данные" << endl;

} while (1);

do {

cout << "Распечатать всех студентов, у которых за все время обучения нет ни одной оценки 1)3 2)3 и 4 3)5" << endl;

check\_potok();

flug = check\_input\_int();

if (flug == 1) {

student\* pprom = stud;

do {

if (pprom->birthday.year >= low && pprom->birthday.year <= high) {

if (check\_mark(pprom, 3, 0)) cout << pprom << endl;

}

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug == 2) {

student\* pprom = stud;

do {

if (pprom->birthday.year >= low && pprom->birthday.year <= high) {

if (check\_mark(pprom, 3, 4)) cout << pprom << endl;

}

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug == 3) {

student\* pprom = stud;

do {

if (pprom->birthday.year >= low && pprom->birthday.year <= high) {

if (check\_mark(pprom, 5, 0)) cout << pprom << endl;

}

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug < 1 || flug>3) cout << "Вы не выбрали ни одного пункта, произведите повторный выбор" << endl;

} while (1);

}

else {

do {

cout << "Распечатать всех студентов, у которых за все время обучения нет ни одной оценки 1)3 2)3 и 4 3)5" << endl;

check\_potok();

flug = check\_input\_int();

if (flug == 1) {

student\* pprom = stud;

do {

if (check\_mark(pprom, 3, 0)) cout << pprom << endl;

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug == 2) {

student\* pprom = stud;

do {

if (check\_mark(pprom, 3, 4)) cout << pprom << endl;

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug == 3) {

student\* pprom = stud;

do {

if (check\_mark(pprom, 5, 0)) cout << pprom << endl;

pprom = pprom->next\_student;

} while (pprom != stud);

break;

}

if (flug < 1 || flug>3) cout << "Вы не выбрали ни одного пункта, произведите повторный выбор" << endl;

} while (1);

}

}

void metods::print(student\* pstart) {

student\* pprom = pstart;

do {

cout << pprom;

pprom = pprom->next\_student;

} while (pstart != pprom);

}

void metods::change(student\* pstart) {

int menu, flugs = 0, flugp = 0;

student\* pprom = searchs(pstart);

cout << pprom << endl << endl;

char menu1[] = { "1.Изменить ФИО" };

char menu2[] = { "2.Изменить пол" };

char menu3[] = { "3.Изменить данные о дате рождения" };

char menu4[] = { "4.Изменить данные об институте" };

char menu5[] = { "5.Изменить данные о сесиях" };

char menu6[] = { "6.Выйти из этого меню" };

while (1) {

cout << menu1 << endl << menu2 << endl << menu3 << endl << menu4 << endl << menu5 << endl << menu6 << endl;

check\_potok();

menu = check\_input\_int();

if (menu == 1) {

while (1) {

cout << "1.Изменить Фамилию" << endl << "2.Изменить Имя" << endl << "3.Изменить Отчество" << endl << "4.Выйти из этого меню" << endl;

check\_potok();

menu = check\_input\_int();

check\_potok(1);

if (menu == 1) {

cout << "Введите новую фамилию" << endl;

cin.getline(pprom->fio.surname, 20);

}

if (menu == 2) {

cout << "Введите новое имя" << endl;

cin.getline(pprom->fio.name, 20);

}

if (menu == 3) {

cout << "Введите новое отчество" << endl;

cin.getline(pprom->fio.otchestvo, 20);

}

if (menu == 4) {

break;

}

if (menu < 1 || menu > 4) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

menu = 1;

}

if (menu == 2) {

check\_potok();

cout << "Введите новый пол" << endl;

do {

cin >> pprom->pol;

if (pprom->pol == 'м' || pprom->pol == 'ж') {

check\_potok();

break;

}

cout << "Вы ввели некоректное значение для пола" << endl;

check\_potok();

} while (1);

}

if (menu == 3) {

while (1) {

cout << "1.Изменить день рождения" << endl << "2.Изменить месяц рождения" << endl << "3.Изменить год рождения" << endl << "4.Выйти из этого меню" << endl;

check\_potok();

menu = check\_input\_int();

check\_potok();

if (menu == 1) {

cout << "Введите новый день рождения " << endl;

do {

cin >> pprom->birthday.day;

if (pprom->birthday.day >= 1 && pprom->birthday.day <= 31) break;

cout << "Вы ввели некоректный день" << endl;

check\_potok();

} while (1);

}

if (menu == 2) {

cout << "Введите новый месяц рождения " << endl;

do {

cin >> pprom->birthday.mounth;

if (pprom->birthday.mounth >= 1 && pprom->birthday.mounth <= 12) break;

cout << "Вы ввели некоректный месяц" << endl;

check\_potok();

} while (1);

}

if (menu == 3) {

cout << "Введите новый год рождения " << endl;

do {

cin >> pprom->birthday.year;

if (pprom->birthday.year >= 1910 && pprom->birthday.year <= 2004)break;

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

}

if (menu == 4) {

break;

}

if (menu < 1 || menu > 4) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

menu = 3;

}

if (menu == 4) {

while (1) {

cout << "1.Изменить год поступления" << endl << "2.Изменить название факультета" << endl << "3.Изменить название кафедры" << endl << "4.Изменить название группы" << endl << "5.Изменить номер зачетной книжки" << endl << "6.Выйти из этого меню" << endl;

check\_potok();

menu = check\_input\_int();

if (menu == 1) {

check\_potok();

cout << "Введите новый год поступления" << endl;

do {

cin >> pprom->univesity.year\_post;

if (pprom->birthday.year + 16 <= pprom->univesity.year\_post)break;

cout << "Вы ввели некоректный год" << endl;

check\_potok();

} while (1);

}

if (menu == 2) {

check\_potok(1);

cout << "Введите название факультета " << endl;

cin.getline(pprom->univesity.facultet, 100);

}

if (menu == 3) {

check\_potok(1);

cout << "Введите казвание кафедры " << endl;

cin.getline(pprom->univesity.kafedra, 70);

}

if (menu == 4) {

check\_potok(1);

cout << "Введите название группы " << endl;

cin.getline(pprom->univesity.group, 20);

}

if (menu == 5) {

char zach[21];

student\* pprompst = pstart;

check\_potok(1);

do {

cout << "Введите новый номер зачетки студента" << endl;

cin.getline(zach, 20);

do {

if (strcmp(pprompst->univesity.number\_zachet, zach) == 0) {

cout << "Студент с таким номером заченой книжки уже есть" << endl;

break;

}

pprompst = pprompst->next\_student;

} while (pstart != pprompst);

if (pprompst == pstart) break;

check\_potok();

pprompst = pstart;

} while (1);

strcpy(pprom->univesity.number\_zachet, zach);

}

if (menu == 6) {

break;

}

if (menu < 1 || menu > 6) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

menu = 4;

}

if (menu == 5) {

char pre[51];

semestrs\* ppromp, \* ptecp = nullptr;

predmets\* pprompp, \* ptecpp = nullptr;

while (1) {

cout << "1.Изменить номер сессии " << endl << "2.Изменить оценку по предмету" << endl << "3.Изменить название предмета" << endl << "4.Выйти из этого меню" << endl;

check\_potok();

menu = check\_input\_int();

if (menu == 1) {

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели не корректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == menu) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

while (1) {

while (1) {

cout << "Введите новый номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели не корректное значение для номера сессии" << endl;

}

if (!search(pprom, pprom->semestr, menu)) {

ptecp->id\_semestr = menu;

break;

}

}

menu = 1;

}

if (menu == 2) {

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели не корректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == menu) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

check\_potok(1);

while (1) {

pprompp = ptecp->predmet;

cout << "Введите название предмета " << endl;

cin.getline(pre, 50);

do {

if (strcmp(pprompp->name\_predmet, pre) == 0) {

ptecpp = pprompp;

flugp = 1;

break;

}

pprompp = pprompp->next\_predmet;

} while (ptecp->predmet != pprompp);

if (flugp == 1)break;

else {

cout << "Такой предмет не был найден у данного студента" << endl;

check\_potok();

}

}

cout << "Введите новую оценку по предмету" << endl;

do {

check\_potok();

menu = check\_input\_int();

if (menu > 1 && menu < 6) {

ptecpp->mark = menu;

break;

}

else cout << "Вы ввели число не соответствующие оценкам по пяти бальной шкале " << endl;

} while (1);

menu = 2;

}

if (menu == 3) {

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели не корректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == menu) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

check\_potok(1);

while (1) {

pprompp = ptecp->predmet;

cout << "Введите название предмета " << endl;

cin.getline(pre, 50);

do {

if (strcmp(pprompp->name\_predmet, pre) == 0) {

ptecpp = pprompp;

flugp = 1;

break;

}

pprompp = pprompp->next\_predmet;

} while (ptecp->predmet != pprompp);

if (flugp == 1)break;

else {

cout << "Такой предмет не был найден у данного студента" << endl;

check\_potok();

}

}

while (1) {

cout << "Введите новое название предмета" << endl;

check\_potok();

cin.getline(pre, 50);

if (!searchp(ptecp, ptecp->predmet, pre)) {

strcpy(ptecpp->name\_predmet, pre);

break;

}

}

menu = 3;

}

if (menu == 4) break;

if (menu < 1 || menu > 4) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

}

if (menu == 6)break;

if (menu < 1 || menu > 6) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

}

student\* metods::delet(student\* pstart) {

int menu, flugs = 0, flugp = 0;

while (1) {

cout << "1.Удалить запись о студенте" << endl << "2.Удалить запись сессии у студента" << endl << "3.Удалить запись о предмете" << endl << "4.Выход из меню" << endl;

check\_potok();

menu = check\_input\_int();

if (menu > 0 && menu < 4) {

student\* pprom = searchs(pstart);

if (menu == 1) {

student\* ptecp = pstart;

if (pprom == pprom->next\_student) {

cout << "В файле была всего одна запись о студенте, послее её удаления файл будет пустым" << endl;

delete pprom;

return nullptr;

}

while (ptecp->next\_student != pprom) {

ptecp = ptecp->next\_student;

}

if (pprom == pstart) {

pstart = pprom->next\_student;

}

ptecp->next\_student = pprom->next\_student;

delete pprom;

}

if (menu == 2) {

semestrs\* ppromp, \* ptecp = nullptr, \* pprompe;

//predmets\* pprompp, \* ptecpp;

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели не корректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == menu) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

ppromp = pprom->semestr;

pprompe = pprom->semestr;

if (ptecp == ptecp->next\_semestr) {

cout << "У студента была всего одна запись о сесии" << endl;

break;

}

do {

pprompe = pprompe->next\_semestr;

} while (pprompe->next\_semestr != pprom->semestr);

while (ppromp->next\_semestr != ptecp) {

ppromp = ppromp->next\_semestr;

}

if (pprompe->next\_semestr == ptecp) {

pprom->semestr = ptecp->next\_semestr;

}

ppromp->next\_semestr = ptecp->next\_semestr;

delete ptecp;

menu = 2;

}

if (menu == 3) {

char pre[51];

semestrs\* ppromp, \* ptecp = nullptr;

predmets\* pprompp, \* ppromppe, \* ptecpp = nullptr;

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

menu = check\_input\_int();

if (menu >= 1 && menu <= 9) break;

else cout << "Вы ввели некорректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == menu) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

check\_potok(1);

while (1) {

pprompp = ptecp->predmet;

cout << "Введите название предмета " << endl;

cin.getline(pre, 50);

do {

if (strcmp(pprompp->name\_predmet, pre) == 0) {

ptecpp = pprompp;

flugp = 1;

break;

}

pprompp = pprompp->next\_predmet;

} while (ptecp->predmet != pprompp);

if (flugp == 1)break;

else {

cout << "Такой предмет не был найден у данного студента" << endl;

check\_potok();

}

}

pprompp = ptecp->predmet;

ppromppe = ptecp->predmet;

if (ptecpp == ptecpp->next\_predmet) {

cout << "У студента была всего один предмет в этой сесии" << endl;

break;

}

do {

ppromppe = ppromppe->next\_predmet;

} while (ppromppe->next\_predmet != ptecp->predmet);

while (pprompp->next\_predmet != ptecpp) {

pprompp = pprompp->next\_predmet;

}

if (ppromppe->next\_predmet == ptecpp) {

ppromp->predmet = ptecpp->next\_predmet;

}

pprompp->next\_predmet = ptecpp->next\_predmet;

delete ptecpp;

menu = 3;

}

}

if (menu == 4) break;

if (menu < 1 || menu > 4) {

cout << "Вы не выбрали ни одного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

if (check\_input\_int() > 0) break;

}

}

return pstart;

}

void metods::dobav\_semestr(student\* pstart) {

int flug, ocen, id, count = 0;

char p[51];

student\* pprom = searchs(pstart);

semestrs\* ppromp, \* ptecp = nullptr;

predmets\* pprompp, \* ptecpp, \* pstartpp;

ppromp = pprom->semestr;

do {

ppromp = ppromp->next\_semestr;

count++;

} while (ppromp->next\_semestr != pprom->semestr);

if (count != 9) {

ppromp->next\_semestr = new semestrs;

ptecp = ppromp->next\_semestr;

do {

do {

cout << "Введите номер сессии" << endl;

check\_potok();

id = check\_input\_int();

if (id > 0 && id < 10) break;

else cout << "Вы ввели не корректный номер сессии" << endl;

} while (1);

} while (search(pprom, pprom->semestr, id));

ptecp->id\_semestr = id;

ptecp->next\_semestr = pstart->semestr;

ptecp->predmet = new predmets;

pprompp = ptecp->predmet;

pstartpp = pprompp;

pprompp->next\_predmet = pprompp;

check\_potok(1);

for (int j = 1; j < 11; j++) {

do {

cout << "Введите название предмета " << endl;

cin.getline(p, 50);

check\_potok();

} while (searchp(ptecp, ptecp->predmet, p));

strcpy(pprompp->name\_predmet, p);

cout << "Оценка по нему " << endl;

do {

ocen = check\_input\_int();

if (ocen > 1 && ocen < 6) {

pprompp->mark = ocen;

break;

}

else cout << "Вы ввели число не соответствующие оценкам по пяти бальной шкале " << endl;

} while (1);

cout << "Есть еще предметы в этой сессии? Да-1 Нет-0 " << endl;

flug = check\_input\_int();

if (flug < 1) {

pprompp->next\_predmet = pstartpp;

break;

}

if (j < 10) {

ptecpp = new predmets;

pprompp->next\_predmet = ptecpp;

pprompp = ptecpp;

pprompp->next\_predmet = ptecp->predmet;

check\_potok(1);

}

else {

cout << "Введено максимальное количество предметов для одной ссесии" << endl;

pprompp->next\_predmet = ptecp->predmet;

}

}

}

else {

cout << "У данного студента максимальное количество сессий" << endl;

}

}

void metods::dobav\_predmet(student\* pstart) {

int flug, flugs, ocen, count = 0;

char pre[51];

student\* pprom = searchs(pstart);

semestrs\* ppromp, \* ptecp = nullptr;

predmets\* pprompp, \* ptecpp, \* pstartpp;

while (1) {

while (1) {

cout << "Введите номер сессии" << endl;

check\_potok();

flug = check\_input\_int();

if (flug >= 1 && flug <= 9) break;

else cout << "Вы ввели некорректное значение для номера сессии" << endl;

}

ppromp = pprom->semestr;

do {

if (ppromp->id\_semestr == flug) {

ptecp = ppromp;

flugs = 1;

break;

}

ppromp = ppromp->next\_semestr;

} while (pprom->semestr != ppromp);

if (flugs == 1)break;

else cout << "Сессия с таким номером не была найденая у данного студента" << endl;

}

pprompp = ptecp->predmet;

do {

pprompp = pprompp->next\_predmet;

count++;

} while (pprompp->next\_predmet != ptecp->predmet);

check\_potok(1);

if (count != 10) {

pprompp->next\_predmet = new predmets;

ptecpp = pprompp->next\_predmet;

ptecpp->next\_predmet = ptecp->predmet;

do {

cout << "Введите название предмета " << endl;

cin.getline(pre, 50);

check\_potok();

} while (searchp(ptecp, ptecp->predmet, pre));

strcpy(pprompp->name\_predmet, pre);

cout << "Оценка по нему " << endl;

do {

ocen = check\_input\_int();

if (ocen > 1 && ocen < 6) {

pprompp->mark = ocen;

break;

}

else cout << "Вы ввели число не соответствующие оценкам по пяти бальной шкале " << endl;

} while (1);

}

else {

cout << "У данного студента в этой сессии максимальное количество предметов" << endl;

}

}

metods::~metods() {};

metods::metods() {};

## Курсовая.cpp

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <windows.h>

#include <fstream>

#include <wincrypt.h>

#include "metods.h"

using namespace std;

void copy\_stud(student\* stud, student\* pprom);

void copy\_sem(semestrs\* ptecp, semestrs\* ppromp);

void copy\_pre(predmets\* ptecpp, predmets\* pprompp);

class file {

protected:

virtual student\* file\_load() = 0;

virtual void file\_save(student\* pstart) = 0;

public:

friend void menu();

};

class file\_p : public file {

void file\_save(student\* pstart) {

char name[51];

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

check\_potok(1);

cin.getline(name, 50);

ofstream file\_save(name, ios::binary);

if (!file\_save) {

cout << "Файл не создан" << endl;

return;

}

if (pstart != nullptr) {

student\* pprom = pstart;

do {

file\_save.write((char\*)pprom, sizeof(student));

semestrs\* pstartp, \* ppromp;

predmets\* pstartpp, \* pprompp;

pstartp = pprom->semestr;

ppromp = pprom->semestr;

do {

pstartpp = ppromp->predmet;

file\_save.write((char\*)ppromp, sizeof(semestrs));

pprompp = ppromp->predmet;

do {

file\_save.write((char\*)pprompp, sizeof(predmets));

pprompp = pprompp->next\_predmet;

} while (pstartpp != pprompp);

file\_save.put('?');

ppromp = ppromp->next\_semestr;

} while (pstartp != ppromp);

file\_save.put('!');

pprom = pprom->next\_student;

} while (pstart != pprom);

}

file\_save.close();

}

student\* file\_load() {

char name[51];

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

check\_potok(1);

cin.getline(name, 50);

ifstream file\_load(name, ios::binary);

if (!file\_load) {

cout << "Файл не найден" << endl;

return nullptr;

}

student\* pstart = nullptr, \* pprom, \* ptec;

semestrs\* pstartp, \* ppromp, \* ptecp;

predmets\* pstartpp, \* pprompp, \* ptecpp;

if (file\_load.peek() != EOF) {

pstart = new(student);

pprom = pstart;

while (1) {

file\_load.read((char\*)pprom, sizeof(student));

pprom->semestr = new semestrs;

ppromp = pprom->semestr;

pstartp = pprom->semestr;

do {

file\_load.read((char\*)ppromp, sizeof(semestrs));

ppromp->predmet = new predmets;

pstartpp = ppromp->predmet;

pprompp = ppromp->predmet;

do {

file\_load.read((char\*)pprompp, sizeof(predmets));

if (file\_load.peek() != '?') {

ptecpp = new predmets;

pprompp->next\_predmet = ptecpp;

pprompp = ptecpp;

}

else {

pprompp->next\_predmet = pstartpp;

file\_load.seekg(1, ios::cur);

break;

}

} while (1);

if (file\_load.peek() != '!') {

ptecp = new semestrs;

ppromp->next\_semestr = ptecp;

ppromp = ptecp;

}

else {

ppromp->next\_semestr = pstartp;

file\_load.seekg(1, ios::cur);

break;

}

} while (1);

if (file\_load.peek() != EOF) {

ptec = new student;

pprom->next\_student = ptec;

pprom = ptec;

}

else {

pprom->next\_student = pstart;

break;

}

}

}

else { cout << "Файл пустой из него нечего загружать" << endl; }

file\_load.close();

return pstart;

}

public:

file\_p() {};

~file\_p() {};

friend void menu();

};

class file\_crypt : public file {

void file\_save(student\* pstart) {

HCRYPTPROV discrip;

HCRYPTKEY key;

DWORD counts = sizeof(student);

DWORD countse = sizeof(semestrs);

DWORD countp = sizeof(predmets);

DWORD sizebuf = 0;

student\* pprom, \* ptec;

semestrs\* pstartp, \* ppromp, \* ptecp;

predmets\* pstartpp, \* pprompp, \* ptecpp;

pprom = new student;

ppromp = new semestrs;

pprompp = new predmets;

ptec = pstart;

char name[51];

char name\_key[51];

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

check\_potok(1);

cin.getline(name, 50);

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

check\_potok();

cin.getline(name\_key, 50);

ofstream file\_save(name, ios::binary);

ofstream file\_key(name\_key, ios::binary);

if (!file\_save) {

cout << "Файл не создан" << endl;

return;

}

if (!file\_key) {

cout << "Файл ключа не создан" << endl;

return;

}

if (pstart != nullptr) {

if (CryptAcquireContext(&discrip, NULL, NULL, PROV\_RSA\_FULL, CRYPT\_VERIFYCONTEXT)) {

if (CryptGenKey(discrip, CALG\_RC4, CRYPT\_EXPORTABLE, &key)) {

do {

copy\_stud(ptec, pprom);

if (CryptEncrypt(key, 0, true, 0, (BYTE\*)pprom, &counts, (DWORD)sizeof(student))) {

file\_save.write((char\*)pprom, sizeof(student));

}

pstartp = ptec->semestr;

ptecp = ptec->semestr;

do {

pstartpp = ptecp->predmet;

copy\_sem(ptecp, ppromp);

if (CryptEncrypt(key, 0, true, 0, (BYTE\*)ppromp, &countse, (DWORD)sizeof(semestrs))) {

file\_save.write((char\*)ppromp, sizeof(semestrs));

}

ptecpp = ptecp->predmet;

do {

copy\_pre(ptecpp, pprompp);

if (CryptEncrypt(key, 0, true, 0, (BYTE\*)pprompp, &countp, (DWORD)sizeof(predmets))) {

file\_save.write((char\*)pprompp, sizeof(predmets));

}

ptecpp = ptecpp->next\_predmet;

} while (pstartpp != ptecpp);

file\_save.put('?');

ptecp = ptecp->next\_semestr;

} while (pstartp != ptecp);

file\_save.put('!');

ptec = ptec->next\_student;

} while (pstart != ptec);

}

}

if (CryptExportKey(key, 0, PLAINTEXTKEYBLOB, 0, NULL, &sizebuf)) {

BYTE\* keys = static\_cast<BYTE\*>(malloc(sizebuf));

if (CryptExportKey(key, 0, PLAINTEXTKEYBLOB, 0, keys, &sizebuf)) {

file\_key.write((char\*)keys, sizebuf);

}

}

}

file\_key.close();

file\_save.close();

}

student\* file\_load() {

HCRYPTPROV discrip;

HCRYPTKEY key;

DWORD counts = sizeof(student);

DWORD countse = sizeof(semestrs);

DWORD countp = sizeof(predmets);

student\* pstart = nullptr, \* pprom, \* ptec, \* ptecc;

semestrs\* pstartp, \* ppromp, \* ptecp, \* ptecpc;

predmets\* pstartpp, \* pprompp, \* ptecpp, \* ptecppc;

char name[51];

char name\_key[51];

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

check\_potok(1);

cin.getline(name, 50);

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

check\_potok();

cin.getline(name\_key, 50);

ifstream file\_load(name, ios::binary);

ifstream file\_key(name\_key, ios::binary);

if (!file\_load) {

cout << "Файл не найден" << endl;

return nullptr;

}

if (!file\_key) {

cout << "Файл ключа не найден" << endl;

return nullptr;

}

if (file\_load.peek() != EOF) {

if (CryptAcquireContext(&discrip, NULL, NULL, PROV\_RSA\_FULL, CRYPT\_VERIFYCONTEXT)) {

BYTE\* keys = static\_cast<BYTE\*>(malloc(28));

file\_key.read((char\*)keys, 28);

if (CryptImportKey(discrip, keys, 28, 0, 0, &key)) {

pstart = new(student);

pprom = pstart;

ptecc = new student;

ptecpc = new semestrs;

ptecppc = new predmets;

while (1) {

file\_load.read((char\*)ptecc, sizeof(student));

if (CryptDecrypt(key, 0, true, 0, (BYTE\*)ptecc, &counts)) {

copy\_stud(ptecc, pprom);

}

pprom->semestr = new semestrs;

ppromp = pprom->semestr;

pstartp = pprom->semestr;

do {

file\_load.read((char\*)ptecpc, sizeof(semestrs));

if (CryptDecrypt(key, 0, true, 0, (BYTE\*)ptecpc, &countse)) {

copy\_sem(ptecpc, ppromp);

}

ppromp->predmet = new predmets;

pstartpp = ppromp->predmet;

pprompp = ppromp->predmet;

do {

file\_load.read((char\*)ptecppc, sizeof(predmets));

if (CryptDecrypt(key, 0, true, 0, (BYTE\*)ptecppc, &countp)) {

copy\_pre(ptecppc, pprompp);

}

if (file\_load.peek() != '?') {

ptecpp = new predmets;

pprompp->next\_predmet = ptecpp;

pprompp = ptecpp;

}

else {

pprompp->next\_predmet = pstartpp;

file\_load.seekg(1, ios::cur);

break;

}

} while (1);

if (file\_load.peek() != '!') {

ptecp = new semestrs;

ppromp->next\_semestr = ptecp;

ppromp = ptecp;

}

else {

ppromp->next\_semestr = pstartp;

file\_load.seekg(1, ios::cur);

break;

}

} while (1);

if (file\_load.peek() != EOF) {

ptec = new student;

pprom->next\_student = ptec;

pprom = ptec;

}

else {

pprom->next\_student = pstart;

break;

}

}

}

}

}

else { cout << "Файл пустой из него нечего загружать" << endl; }

file\_key.close();

file\_load.close();

return pstart;

}

public:

file\_crypt() {};

~file\_crypt() {};

friend void menu();

};

ostream& operator << (ostream& out, student\* stud) {

semestrs\* pstart, \* pprom;

predmets\* pstartp, \* ppromp;

out << endl;

out << "Личная информация" << endl;

out << "id студента: " << stud->id\_student << endl;

out << "ФИО студента: " << stud->fio.surname << " " << stud->fio.name << " " << stud->fio.otchestvo << endl;

out << "пол: " << stud->pol << endl;

out << "дата рождения: " << stud->birthday.day << "." << stud->birthday.mounth << "." << stud->birthday.year << endl << endl;

out << "Информация из института" << endl;

out << "год поступления: " << stud->univesity.year\_post << endl;

out << "название факультета: " << stud->univesity.facultet << endl;

out << "название кафедры: " << stud->univesity.kafedra << endl;

out << "номер группы: " << stud->univesity.group << endl;

out << "номер зачетки: " << stud->univesity.number\_zachet << endl << endl;

pstart = stud->semestr;

pprom = stud->semestr;

do {

pstartp = pprom->predmet;

out << "Информация о " << pprom->id\_semestr << " сесии" << endl;

ppromp = pprom->predmet;

do {

out << ppromp->name\_predmet << ": " << ppromp->mark << endl;

ppromp = ppromp->next\_predmet;

} while (pstartp != ppromp);

out << endl;

pprom = pprom->next\_semestr;

} while (pstart != pprom);

out << "------------------------------------------------------------------------" << endl << endl;

return out;

}

void copy\_stud(student\* stud, student \*pprom) {

pprom->id\_student = stud->id\_student;

strcpy(pprom->fio.name, stud->fio.name);

strcpy(pprom->fio.surname, stud->fio.surname);

strcpy(pprom->fio.otchestvo, stud->fio.otchestvo);

pprom->pol = stud->pol;

pprom->birthday.day = stud->birthday.day;

pprom->birthday.mounth = stud->birthday.mounth;

pprom->birthday.year = stud->birthday.year;

strcpy(pprom->univesity.facultet, stud->univesity.facultet);

strcpy(pprom->univesity.group, stud->univesity.group);

strcpy(pprom->univesity.kafedra, stud->univesity.kafedra);

strcpy(pprom->univesity.number\_zachet, stud->univesity.number\_zachet);

pprom->univesity.year\_post = stud->univesity.year\_post;

pprom->next\_student = stud->next\_student;

pprom->semestr = stud->semestr;

}

void copy\_sem(semestrs\* ptecp,semestrs\* ppromp) {

ppromp->id\_semestr = ptecp->id\_semestr;

ppromp->predmet = ptecp->predmet;

ppromp->next\_semestr = ppromp->next\_semestr;

}

void copy\_pre(predmets\* ptecpp, predmets\* pprompp) {

strcpy(pprompp->name\_predmet , ptecpp->name\_predmet);

pprompp->mark = ptecpp->mark;

pprompp->id\_predmet = ptecpp->id\_predmet;

pprompp->next\_predmet = ptecpp->next\_predmet;

}

void menu() {

student\* pstart = nullptr;

int menus=0;

metods metod;

file\* files;

file\_p file\_prost;

file\_crypt file\_crypto;

while (pstart==nullptr) {

cout << "1.Создать новый файл" << endl << "2.Загрузить из файла" << endl << "3.Завершить работу" << endl;

check\_potok();

menus = check\_input\_int();

if (menus == 1) {

pstart = metod.create();

}

if (menus == 2) {

cout << "1.Из обычного файла" << endl << "2.Из шифрованного файла" << endl;

check\_potok();

menus = check\_input\_int();

if (menus == 1) {

files = &file\_prost;

pstart=files->file\_load();

}

if (menus == 2) {

files = &file\_crypto;

pstart=files->file\_load();

}

menus = 2;

}

if (menus == 3) return;

if (menus < 1 || menus > 3) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

check\_potok();

if (check\_input\_int() > 0) return;

}

}

while (1) {

cout << "1.Добавить запись в список" << endl << "2.Изменить данные в списке" << endl << "3.Удалить запись из списка" << endl << "4.Выполнить задание варианта" << endl << "5.Распечатать список" << endl << "6.Сохранить данные в файл" << endl << "7.Завершить работу" << endl;

check\_potok();

menus = check\_input\_int();

if (menus == 1) {

while (1) {

cout << "1.Запись о студенте" << endl << "2.Запись о семестре" << endl << "3.Запись о предмете" << endl << "4.Выход из этого меню" << endl;

check\_potok();

menus = check\_input\_int();

if (menus == 1) metod.create(pstart);

if (menus == 2) metod.dobav\_semestr(pstart);

if (menus == 3) metod.dobav\_predmet(pstart);

if (menus == 4) break;

if (menus < 1 || menus > 4) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

check\_potok();

if (check\_input\_int() > 0) break;

}

}

menus = 1;

}

if (menus == 2) metod.change(pstart);

if (menus == 3) metod.delet(pstart);

if (menus == 4) metod.menu1\_for\_zadani(pstart);

if (menus == 5) metod.print(pstart);

if (menus == 6) {

while (1) {

cout << "1.Сохранить в обычном виде" << endl << "2.Сохранить в зашифрованном виде" << endl << "3.Выход из меню" << endl;

check\_potok();

menus = check\_input\_int();

if (menus == 1) {

files = &file\_prost;

files->file\_save(pstart);

break;

}

if (menus == 2) {

files = &file\_crypto;

files->file\_save(pstart);

break;

}

if (menus == 3) break;

if (menus < 1 || menus > 3) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

check\_potok();

if (check\_input\_int() > 0) break;

}

}

menus = 6;

}

if (menus == 7) return;

if (menus < 1 || menus > 7) {

cout << "Вы не выбрали ниодного пункта меню. Хотите выйти? да-1 , нет-0" << endl;

check\_potok();

if (check\_input\_int() > 0) return;

}

}

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

menu();

}