

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

КУРСОВОЙ ПРОЕКТ (РАБОТА)

по дисциплине: «Языки программирования»

(наименование дисциплины)

Тема курсового проекта (работы)\_\_\_\_Файловая система\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Студент группы БББО-05-19 Сердюков Матвей Романович

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

учебная группа, фамилия, имя, отчество студента

подпись студента

Руководитель курсового проекта (работы) Мерсов А. А.

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

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 «3» июня 2020 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт комплексной безопасности и специального приборостроения

Кафедра КБ-3 «Управление и моделирование систем»

Утверждаю

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

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

фамилия, имя, отчество подпись

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_202\_ г.

ЗАДАНИЕ

на выполнение курсового проекта (работы) по дисциплине

«Языки программирования»

Студент: Сердюков Матвей Романович Группа: БББО-05-19

Тема:\_\_Файловые системы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

Исходные данные: информация о группе студентов: ФИО, дата рождения, пол, год поступления, факультет, кафедра, номер зачётной книжки, оценки

Перечень вопросов, подлежащих разработке, и обязательного графического материала: ввод данных с клавиатуры, выполнение программы согласно варианту, функции чтения и записи в/из файл(-а), использование динамической памяти, функции добавления или удаления записей, функция изменения записей, использование классов и объектов, конструкторы и деструкторы, друзья классов, наследование, перегрузка операторов, виртуальные функции, шифрование и дешифрование на основе системы CRYPTO++.

Срок представления к защите курсового проекта (работы) до «25» мая 2020 г.

Задание на выполнение курсовой проект (работу) выдал Мерсов А.А.(\_\_\_\_\_\_\_\_\_\_\_\_\_)

ф.и.о. руководителя

подпись руководителя

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2020

г.

Задание на курсовой проект (работу) получил Сердюков М.Р.

(\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

ф.и.о. исполнителя

подпись обучающегося

Содержание

[Введение 1](#__RefHeading___Toc6879_3528458755)

[Исходные данные 2](#__RefHeading___Toc6899_3528458755)

[Оценка использованных средств ООП 3](#__RefHeading___Toc6901_3528458755)

[Разработанные классы и структуры 4](#__RefHeading___Toc6919_3528458755)

[Содержание классов и структур 5](#__RefHeading___Toc6939_3528458755)

[Схема классов 11](#__RefHeading___Toc6953_3528458755)

[Алгоритм решения задачи 12](#__RefHeading___Toc6969_3528458755)

[Руководство пользователя 16](#__RefHeading___Toc6971_3528458755)

[Контрольные данные 17](#__RefHeading___Toc7001_3528458755)

[Анализ результатов 18](#__RefHeading___Toc7007_3528458755)

[Листинг программы 21](#__RefHeading___Toc6026_80162864)

# Введение

Цель разработки: программа, написанная на языке программирования C++, позволяющая работать с группой студентов, и выполняющая задание согласно варианту.

Задачи:

1. Реализовать хранение данных об одном студенте
2. Реализовать хранение списка студентов
3. Реализовать заполнение списка студентов с клавиатуры
4. Реализовать редактирование, добавление или удаление студентов в списке
5. Реализовать сортировку студентов в списке по критериям, указанным в варианте
6. Реализовать чтение и запись данных в/из файл(-а)
7. Реализовать шифрование и дешифрование данных с помощью криптографической библиотеки

# Исходные данные

Информация о группе студентов из N человек, где запись о студенте содержит

следующие данные:

1. Ф.И.О. студента.
2. Число, месяц, год рождения.
3. Год поступления в институт.
4. Факультет, кафедра.
5. Группа.
6. Номер зачетной книжки.
7. Номер зачетной книжки.
8. Названия предметов и оценки по каждому предмету в каждой сессии.

(максимально 9 сессий и 10 предметов в каждом семестре, которые

м.б. разные).

Вариант 72:

Разбить группу на 3 части, с указанием интервала года рождения:

1. отличников;
2. хорошистов;
3. троечников

по сессии, вводимой по желанию пользователя за всё время обучения.

# Оценка использованных средств ООП

В ходе разработки программы были использованы следующие элементы парадигмы ООП:

* Класс — это тип, определяемый пользователем, включающий в себя данные и функции, называемые методами или функциями-членами класса;
* Объект — переменная типа класс;
* Наследование - наличие в языке ООП механизма, позволяющего объектам класса наследовать характеристики более простых и общих типов.
* 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;
* Деструктор — метод класса, который вызывается перед освобождением памяти, занимаемой объектной переменной, и предназначен для выполнения дополнительных действий, связанных с уничтожением объектной переменной;
* Дружественная функция — функция, не являющаяся методом класса, которой предоставлен доступ к частным и защищённым элементам класса;
* Виртуальная функция — метод класса, который может быть переопределён в производном классе при наследовании.

# Разработанные классы и структуры

В программе были разработаны следующие классы и структуры:

1. basic\_string - класс для работы со строками;
2. faculty - факультет;
3. department - кафедра;
4. group - учебная группа;
5. id - номер зачётной книжки;
6. fullname - ФИО;
7. subject - структура для работы с предметами
8. session - сессия;
9. student - вся информация о студенте;
10. my\_list - структура для работы со списками.

# Содержание классов и структур

1. Классы basic\_string, faculty, department, group, id

Поля:

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| data | string | хранение строки |

Методы:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Аргументы | Тип возвращаемого значения | Назначение |
| get | нет | string | получение данных |
| set | string input | void | установка значения поля data |

2. Класс fullname

Поля:

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| name | string[3] | хранение массива из 3 строк |

Методы:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Аргументы | Тип возвращаемого значения | Назначение |
| get | нет | string | получение данных |
| get\_i\_data | int i | string | получение одного из 3 значений поля name |
| set | string last\_name, string first\_name, string middle\_name | void | установка значений поля name |

3. Структура subject

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| name | string | название предмета |
| mark | unsigned short | оценка |

4. Класс session

Поля:

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| subjects | subject[10] | массив структур subject, в которых содержится информация о предмете |
| n | unsigned short | количество предметов в сессии |

Методы:

|  |  |  |  |
| --- | --- | --- | --- |
| Поле | Аргументы | Тип возвращаемого значения | Назначение |
| set\_n | unsigned short x | void | установка количества предметов в сессии |
| set | subject input[] | void | установка предметов в сессии |
| get\_n | нет | int | получение кол-ва предметов в сессии |
| get | нет | subject\* | получение указателя на массив предметов |
| get\_avrg | нет | double | получение среднего балла в сессии |
| print | нет | void | вывод информации о сессии |
| add\_subject | subject input | void | добавление предмета в сессию |
| edit\_subject | string input, int new\_mark | void | редактирование предмета |
| delete\_subject | string input | void | удаление предмета из сессии |

5. Класс student

Поля:

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| fio | fullname | ФИО |
| birthdate | unsigned short[3] | дата рождения |
| sex | char | пол |
| enroll\_year | unsigned short | год поступления |
| fac | faculty | факультет |
| kaf | department | кафедра |
| gr | group | группа |
| zach | id | номер зачётной книжки |
| marks | session[9] | оценки |

Методы:

|  |  |  |  |
| --- | --- | --- | --- |
| Метод | Аргументы | Тип возвращаемого значения | Назначение |
| get\_full\_name | нет | string | получение ФИО |
| get\_name | int i | string | получение одного из значений ФИО |
| set\_full\_name | string last\_name, string first\_name, string middle\_name | void | установка ФИО |
| get\_birthdate | нет | string | получение даты рождения |
| get\_birth\_year | нет | unsigned short | получение года рождения |
| set\_birthdate | unsigned short day, unsigned short month, unsigned short year | void | установка даты рождения |
| get\_sex | нет | char | получение пола |
| set\_sex | char input | void | установка пола |
| get\_enroll\_year | нет | unsigned short | получение года поступления |
| set\_enroll\_year | unsigned short input | void | установка года поступления |
| get\_fac | нет | string | получение факультета |
| set\_fac | string input | void | установка факультета |
| get\_kaf | нет | string | получение кафедры |
| set\_kaf | string input | void | установка кафедры |
| get\_group | нет | string | получение группы |
| set\_group | string input | void | установка группы |
| get\_id | нет | string | получение номера зачётной книжки |
| set\_id | string input | void | установка номера зачётной книжки |
| get\_n | нет | unsigned short | получение кол-ва сессий |
| set\_n | unsigned short x | void | установка кол-ва сессий |
| get\_marks | нет | session\* | получение указателя на массив сессий |
| set\_marks | session input[] | void | установка оценок |
| get\_nth\_session | нет | session | получение n-ой сессии |
| print | нет | void | вывод всей информации о студенте |

6. Структура my\_list

Поля структуры:

|  |  |  |
| --- | --- | --- |
| Поле | Тип | Назначение |
| prv | my\_list\* | указатель на предыдущий элемент списка |
| val | student | информация о студенте |
| nxt | my\_list\* | указатель на следующий элемент списка |

Функции, разработанные для работы со списками:

|  |  |  |  |
| --- | --- | --- | --- |
| Имя функции | Аргументы | Тип возвращаемого значения | Назначение |
| init\_list | student x | my\_list\* | инициализация списка |
| add\_element | my\_list\* lst, student x | my\_list\* | добавление нового элемента в список |
| delete\_element | my\_list\* lst | my\_list\* | удаление элемента из списка |
| delete\_list | my\_list\* lst | my\_list\* | полное удаление списка |
| find\_element\_by\_id | my\_list\* lst, string input | my\_list\* | поиск элемента списка по номеру зачётной книжки |
| get\_students\_by\_year | my\_list\* lst, unsigned short start, unsigned short end | my\_list\* | получения списка студентов, входящих в определённый интервал года рождения |
| get\_students\_by\_mark | my\_list\* lst, unsigned short ses, unsigned short mark | my\_list\* | получение списка студентов, у которых в введённой сессии получена введённая оценка (средний балл) |
| print\_list | my\_list\* lst | void | вывод списка на экран |

# Схема классов

1. Класс basic\_string наследуется в следующих классах (спецификатор public):
   1. faculty
   2. department
   3. group
   4. id
   5. fullname
2. Класс student имеет дружественную функцию student edit(student input)

# Алгоритм решения задачи

Первая проблема, которую нужно решить при разработке программы — хранение данных о студенте. Для этого был объявлен класс student, позволяющий хранить всю необходимую информацию внутри одного объекта, а также методы для работы с ней. Класс должен содержать следующие поля:

1. ФИО (3 строки)
2. Дата рождения (3 числа)
3. Пол (символ)
4. Год поступления (число)
5. Факультет (строка)
6. Кафедра (строка)
7. Номер зачетной книжки (строка)
8. Оценки (до 9 сессий с 1-10 предметов(строка и число) в каждой)

Как можно заметить, примерно половина из этих полей будет представлять собой одну строку, за исключением ФИО, где будет необходимо хранить уже 3 строки. В связи с этим было решено сделать каждое из этих строковых полей классами, которые наследуют класс basic\_string, который состоит из одного поля типа string, а также двух методов для получения и изменения значения этого поля. Методы были объявлены виртуальным, благодаря чему их можно было переопределить для работы с массивом строк (для работы с классом, содержащим ФИО).

Также стоит обратить внимание на оценки. Для комфортной работы с ними сначала была создана структура subject, содержащая строку (имя предмета) и целое число (оценка). Затем был объявлен класс session, в котором хранится массив этих структур. Также для этого класса были разработаны методы по добавлению, удалению и редактированию предметов в сессии, а также получения среднего балла в сессии. А в самом классе student уже хранится массив из объектов типа session.

Для всех полей основного класса были разработаны методы по получению и установки значений, а для классов session, student и структуры subject были перегружены операторы присваивания, для более комфортной работы с ними в ходе дальнейшей разработки программы. Принцип работы каждой из перегрузок схож: всем значениям полей текущего объекта присваиваются соответствующие им значения из объекта-аргумента. ДЛя обращения к полям текущего объекта используется неявный указатель this. Также для класса student была объявлена дружественная функция edit, позволяющая редактировать поля класса и функция для вывода всей инфоормации о студенте.

Хоть и проблема хранения данных о студенте была решена, надо было так же решить проблем хранения уже группы студентов, количество которых может постоянно меняться в ходе работы программы. Для этого было решено использовать такую структуру данных, как двусвязный кольцевой список. Каждый её элемент содержит в себе какое-то значение, а также ссылки на следующий и предыдущий элемент. Это свойство списков позволит в дальнейшем быстро и эффективно работать с группой студентов: добавлять, удалять или редактировать их данные. Каждый элемент разработанного списка представляет собой структуру my\_list, состоящей из объекта типа student, а также двух указателей на эту же структуру. Для работы со списком были разработаны функции, для инициализиации списка, добавления или удаления элементов, очистики списка с использованием динамической памяти, в частности операторов new и delete, а также функция вывода всего списка студентов. Для дальнейшей сортировки списка были реализованы функция, возвращающая список студентов, которые входят в введённый интервал года рождения, и функция, возвращающая список студентов, которые получили тот или иной средний балл в одной из сессий.

После того, как все данные, с которыми работает программа, были объявлены, был реализован ввод данных с клавиатуры. Перед реализацией данного функционала были также реализованы функции проверки введённых данных на корректность. Ввод данных происходит посредством двух функций, одна из которых создаёт объект и заполняет все его поля, а вторая бесконечно вызывает первую, добавляя новые элемент в список. После добавления каждого элемента функция спрашивает у пользователя, нужно ли добавить ещё одного студента, и лишь при получении ответа «нет» прекращает свою работу. После того как функция ввода была реализована, были также разработаны функции добавления, редактирования и удаления студентов из введенного списка. Функции редактирования и удаления сначала запрашивают номер зачётной книжки, и начинают свою работу лишь в случае успешного поиска по этому номеру.

Теперь, когда в программе есть возможность ввести данные, можно реализовать выполнение задания по варианту. Для этого была создана функция, которая сначала запрашивает у пользователя интервал года рождения и вызывает функцию поиска по этому интервалу, объявленную ранее. И только после этого, передавая уже список лишь тех студентов, родившихся в указанному интервале, три раза вызываются функции поиска по оценкам(3, 4 и 5 соответственно), и полученные списки уже выводятся на экран.

И последнее, что было разработано — ввод/вывод в/из файл(-а) и шифрование данных. Эти два элемента тесно связаны друг с другом, в связи с чем сперва было реализовано шифрование. Для его реализации была использована библиотека Crypto++, которая представляет широкий функционал в области криптографии. Особенностью данной библиотеки является использование парадигмы под названием pipelining (рус. конвейерная обработка). Её суть заключается в передаче данных от "источника" (Source) к "стоку" (Sink). А данные на своем пути могут проходить через различные фильтры, так или иначе обрабатывающие эти данные. В результате, как правило, для реализации тех или иных преобразований данных, в динамической памяти последовательно вызываются конструкторы требуемых классов, от самого вложенного (как правило это "сток", к внешнему - "источнику". В документации библиотеки сказано, что её разработчиками предусмотрена опасность утечки памяти, и оператор delete для созданных объектов вызывается автоматически, что упрощает работу пользователю этой библиотеки. В данном случае для реализации шифрования и дешифрования были использованы классы DefaultEncryptorWithMAC и DefaultDecryptorWithMAC, позволяющие реализовать шифрование и дешифрование с использованием пароля. В своей работе они используют такие алгоритмы как AES и SHA256, что обеспечивает высокую защищенность данных. После реализации данных функций, были разработаны функции чтения/записи из/в файл(-а). Они сначала запрашивают у пользователя имя файла, после чего вводится пароль, с помощью которого будет (де-)шифроваться информация. В дальнейшем эти функции полностью противоположны: функция записи проходится по списку студентов, и записывает каждое поле в файл, предварительно зашифровав его, а функция чтения создаёт новый список, считывает каждую строчку файла, и после её дешифровки записывает информацию в поле объекта. После считывания каждого студента в список добавляется соответствующий элемент.

После реализаций всех функций для удобной работы с программой так же было разработано меню, которое позволяет проводить все необходимые операции над списком студентов:

1. Ввод списка с клавиатуры
2. Чтение списка из файла
3. Добавление записей в список
4. Удаление записей из списка
5. Редактирование записей в списке
6. Вывод всего списка
7. Вывод результатов работы программы по варианту
8. Сохранение списка в файл
9. Выход из программы

В результате программа состоит из 15 исходных файлов. 14 из них представляют собой пары из заголовочного файла с объявлением переменных и функций и файла .cpp с их реализацией. Структура программы выглядит так:

1. students — переменные, структуры и классы, необходимые для хранения и обработки информации о студенте
2. my\_list — реализация списка
3. read — ввод с клавиатуры
4. validation — проверка входных данных
5. files — чтение и запись из файлов
6. crypto — шифрование и дешифрование
7. menu — меню программы
8. main.cpp — основной файл программы

# Руководство пользователя

Внимание: для компиляции и работы программы необходима библиотека CRYPTO++. Также могут возникнуть проблемы с совместимостью на ОС Windows.

Для сборки программы на ОС Linux, находясь в папке src, воспользуйтесь следующей командой:

g++ \*.cpp -I include/ -lcryptopp

Внимание: программа работает только на английском языке.

1. После запуска программы, введите данные студентов с клавиатуры(опция 1), либо считайте данные с файла (опция 2). Остальные функции будут недоступны, пока не будет осуществлен ввод каким-либо способом.

Внимание: при повторном вызове функции ввода в дальнейшей работе программы, список студентов будет введён заново и информация, не сохранённая в файл, удалится.

Внимание: считать можно только файл, предварительно созданный в программе.

1. После успешного считывания данных, вам доступны следующие функции, которые можно вызывать в любом порядке неограниченное кол-во раз:
   1. Добавить студента (опция 3)
   2. Редактировать студента (опция 4)
   3. Удалить студента (опция 5)
   4. Вывести список студентов (опция 6)
   5. Вывести результат сортировки согласно варианту (опция 7)
   6. Сохранить список студентов в файл (опция 8)

Внимание: функции редактирования и удаления студентов требуют ввода номера зачётной книжки необходимого студента. Чтобы узнать эту и иную информацию, воспользуйтесь опцией 6.

1. После завершения работы с данными, введите 0, чтобы выйти из программы. Внимание: файл со списком студентом будет сохранён только в том случае, если вы сохранили его, используя опцию 8.

# Контрольные данные

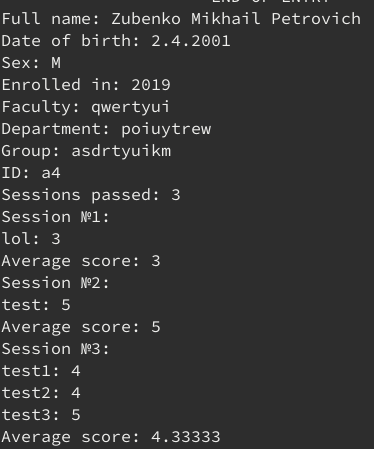
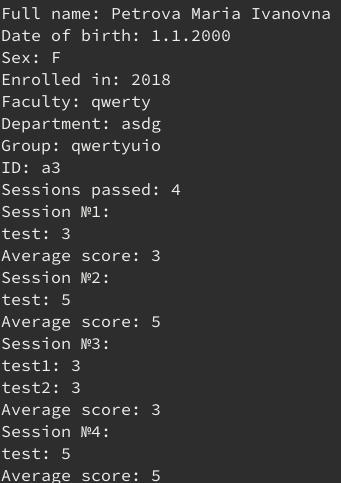
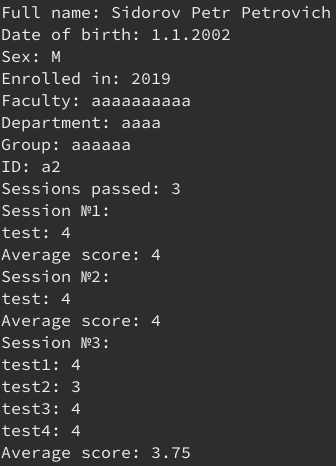
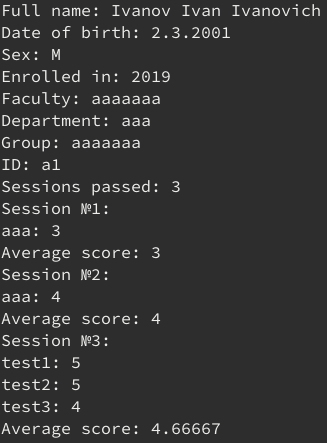
В программу были введены данные о шести студентах. Для сортировки будет использован интервал 2000-2001 года рождения, по третьей сессии. Для удобства в таблице приведены лишь те данные, от которых зависит результат сортировки:

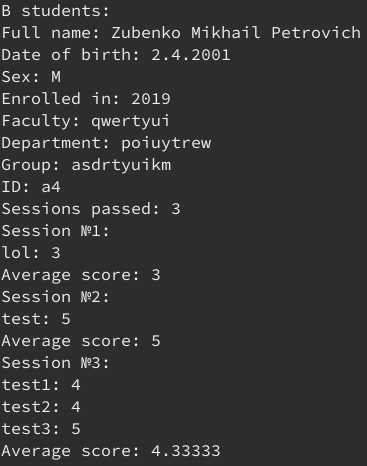
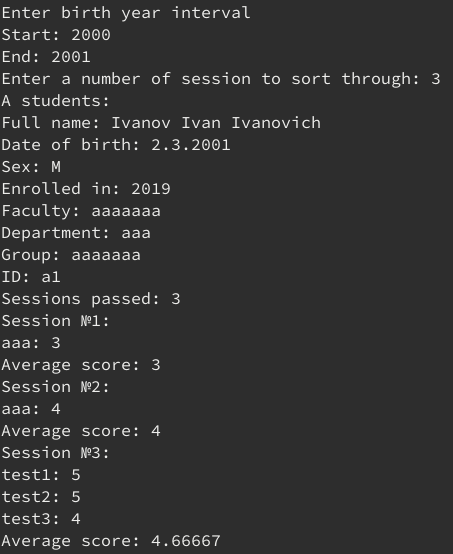
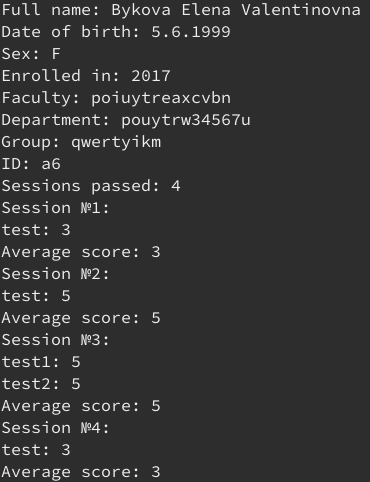
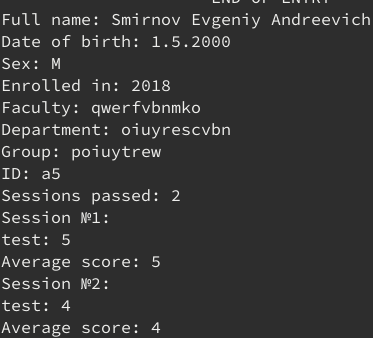
|  |  |  |
| --- | --- | --- |
| Номер зачётной книжки (ID) | Год рождения | Средний балл (округленный) |
| a1 | 2001 | 5 |
| a2 | 2002 | 4 |
| a3 | 2000 | 3 |
| a4 | 2001 | 4 |
| a5 | 2000 | отсутствует |
| a6 | 1999 | 5 |

В теории, программа должна вывести лишь студентов с ID a1 (отличник), a3 (хорошист) и a3(троечник). Остальные студенты выведены не будут, так как их год рождения не удовлетворяет указанному интервалу, либо же у них отсутствует введённая сессия.

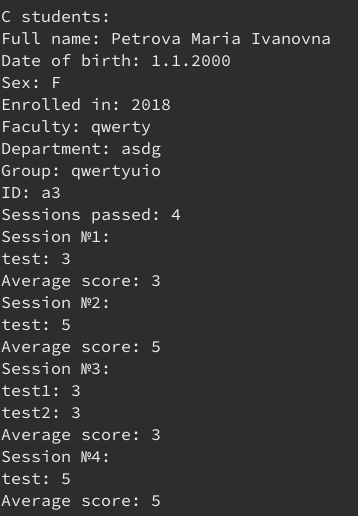
# Анализ результатов

Введенные данные:





Результаты сортировки:



Как можно заметить, результат работы программы совпал с теоретическим, что говорит о корректной работе программы

# Листинг программы

## Исходный файл students.h

#pragma once

#include <string>

#include <iostream>

using namespace std;

class data\_string

{

private:

string data;

public:

virtual void set\_data(string input)

{

data = input;

}

virtual string get\_data() const

{

return data;

}

};

class faculty: public data\_string

{

string data;

};

class department: public data\_string

{

string data;

};

class group: public data\_string

{

string data;

};

class id: public data\_string

{

string data;

};

class fullname: public data\_string

{

string name[3];

public:

void set\_data(string last\_name, string first\_name, string middle\_name)

{

name[0] = last\_name;

name[1] = first\_name;

name[2] = middle\_name;

}

string get\_data() const

{

string s = name[0] + ' ' + name[1] + ' ' + name[2];

return s;

}

string get\_i\_data(int i) const

{

return name[i];

}

};

struct subject

{

string name;

unsigned short mark;

subject& operator =(const subject& right)

{

if (this == &right)

{

return \*this;

}

this->name = right.name;

this->mark = right.mark;

return \*this;

}

};

typedef struct subject subject;

class session

{

subject subjects[10];

unsigned short n;

public:

void set\_n(unsigned short x)

{

n = x;

}

void set(subject input[])

{

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

{

subjects[i] = input[i];

}

}

int get\_n() const

{

return n;

}

subject\* get()

{

return subjects;

}

double get\_avrg()

{

double x = 0;

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

{

x += subjects[i].mark;

}

x /= n;

return x;

}

void print()

{

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

{

cout << subjects[i].name << ": " << subjects[i].mark << endl;

}

}

void add\_subject(subject input)

{

subjects[n] = input;

n++;

}

void delete\_subject(string input)

{

int i;

for (i = 0; i < n; i++)

{

if (subjects[i].name == input)

{

break;

}

}

for (; i < n-1; i++)

{

subjects[i] = subjects[i+1];

}

n--;

}

void edit\_subject(string input, int new\_mark)

{

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

{

if (input == subjects[i].name)

{

subjects[i].mark = new\_mark;

}

}

}

session& operator=(const session& right)

{

if (this == &right)

{

return \*this;

}

this->n = right.n;

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

{

this->subjects[i] = right.subjects[i];

}

return \*this;

}

};

class student

{

private:

fullname fio;

unsigned short birthdate[3];

char sex;

unsigned short enroll\_year;

faculty fac;

department kaf;

group gr;

id zach;

unsigned short n\_sessions;

session marks[9];

friend student edit(student input);

public:

student(){;}

student(unsigned short x)

{

n\_sessions = x;

}

string get\_full\_name() const;

string get\_name(int i)

{

return fio.get\_i\_data(i);

}

void set\_full\_name(string last\_name, string first\_name, string middle\_name);

string get\_birthdate();

unsigned short get\_birth\_year();

void set\_birthdate(unsigned short day, unsigned short month, unsigned short year);

char get\_sex() const;

void set\_sex(char input);

unsigned short get\_enroll\_year() const;

void set\_enroll\_year(unsigned short input);

string get\_fac() const;

void set\_fac(string input);

string get\_kaf() const;

void set\_kaf(string input);

string get\_group() const;

void set\_group(string input);

string get\_id() const;

void set\_id(string input);

unsigned short get\_n() const;

void set\_n(unsigned short x)

{

n\_sessions = x;

}

session\* get\_marks();

void set\_marks(session input[]);

session get\_nth\_session(unsigned short x);

void print();

student& operator=(const student& right)

{

if (this == &right)

{

return \*this;

}

this->set\_full\_name(right.fio.get\_i\_data(0), right.fio.get\_i\_data(1), right.fio.get\_i\_data(2));

this->set\_birthdate(right.birthdate[0], right.birthdate[1], right.birthdate[2]);

sex = right.get\_sex();

enroll\_year = right.enroll\_year;

this->set\_fac(right.get\_fac());

this->set\_kaf(right.get\_kaf());

this->set\_group(right.get\_group());

this->set\_id(right.get\_id());

n\_sessions = right.n\_sessions;

for(int i = 0; i < n\_sessions; i++)

{

marks[i].set\_n(right.marks[i].get\_n());

marks[i] = right.marks[i];

}

return \*this;

}

};

## Исходный файл students.cpp

#include <iostream>

#include "students.h"

#include "validation.h"

string student::get\_full\_name() const

{

return fio.get\_data();

}

void student::set\_full\_name(string last\_name, string first\_name, string middle\_name)

{

fio.set\_data(last\_name, first\_name, middle\_name);

}

string student::get\_birthdate()

{

string output = to\_string(birthdate[0]) + '.' + to\_string(birthdate[1]) + '.' + to\_string(birthdate[2]);

return output;

}

unsigned short student::get\_birth\_year()

{

return birthdate[2];

}

void student::set\_birthdate(unsigned short day, unsigned short month, unsigned short year)

{

birthdate[0] = day;

birthdate[1] = month;

birthdate[2] = year;

}

char student::get\_sex() const

{

return sex;

}

void student::set\_sex(char input)

{

sex = input;

}

unsigned short student::get\_enroll\_year() const

{

return enroll\_year;

}

void student::set\_enroll\_year(unsigned short input)

{

enroll\_year = input;

}

string student::get\_fac() const

{

return fac.get\_data();

}

void student::set\_fac(string input)

{

fac.set\_data(input);

}

string student::get\_kaf() const

{

return kaf.get\_data();

}

void student::set\_kaf(string input)

{

kaf.set\_data(input);

}

string student::get\_group() const

{

return gr.get\_data();

}

void student::set\_group(string input)

{

gr.set\_data(input);

}

string student::get\_id() const

{

return zach.get\_data();

}

void student::set\_id(string input)

{

zach.set\_data(input);

}

unsigned short student::get\_n() const

{

return n\_sessions;

}

session\* student::get\_marks()

{

return marks;

}

void student::set\_marks(session input[])

{

for(int i = 0; i < n\_sessions; i++)

{

marks[i].set\_n(input[i].get\_n());

marks[i] = input[i];

}

}

session student::get\_nth\_session(unsigned short x)

{

return marks[x-1];

}

void student::print()

{

cout << "Full name: " << get\_full\_name() << endl;

cout << "Date of birth: " << get\_birthdate() << endl;

cout << "Sex: " << get\_sex() << endl;

cout << "Enrolled in: " << get\_enroll\_year() << endl;

cout << "Faculty: " << get\_fac() << endl;

cout << "Department: " << get\_kaf() << endl;

cout << "Group: " << get\_group() << endl;

cout << "ID: " << get\_id() << endl;

cout << "Sessions passed: " << get\_n() << endl;

for (int i = 0; i < n\_sessions; i++)

{

cout << "Session №" << i+1 << ":\n";

marks[i].print();

cout << "Average score: " << marks[i].get\_avrg() << endl;

}

}

student edit(student input)

{

int n;

cout << "What would you like to edit?" << endl;

while(true)

{

cout << "\t1. Name" << endl;

cout << "\t2. Date of birth" << endl;

cout << "\t3. Sex" << endl;

cout << "\t4. Year of enroll" << endl;

cout << "\t5. Faculty" << endl;

cout << "\t6. Department" << endl;

cout << "\t7. Group" << endl;

cout << "\t8. Marks" << endl;

cout << "\t0. Exit edit mode" << endl;

cin >> n;

if (n == 1)

{

string new\_name[3];

cout << "Enter last name: ";

cin.ignore();

getline(cin, new\_name[0]);

cout << "Enter first name: ";

cin.ignore();

getline(cin, new\_name[1]);

cout << "Enter middle name: ";

cin.ignore();

getline(cin, new\_name[2]);

input.set\_full\_name(new\_name[0], new\_name[1], new\_name[2]);

cout << "Name edited!" << endl;

}

if (n == 2)

{

unsigned short new\_date[3];

cout << "Enter date of birth\n";

do

{

cout << "Day: ";

while(!(cin >> new\_date[0]))

{

clearbuf();

input\_error();

}

cout << "Month: ";

while(!(cin >> new\_date[1]))

{

clearbuf();

input\_error();

}

cout << "Year: ";

while(!(cin >> new\_date[2]))

{

clearbuf();

input\_error();

}

}while(check\_birthdate(new\_date[0], new\_date[1], new\_date[2]) != true);

for(int i = 0; i < 3; i++)

{

input.birthdate[i] = new\_date[i];

}

}

if (n == 3)

{

char new\_s;

do

{

cout << "Enter sex: ";

cin >> new\_s;

clearbuf();

}while(check\_sex(new\_s) != true);

input.sex = new\_s;

}

if (n == 4)

{

unsigned short new\_enr;

do

{

cout << "Enter enroll year: ";

while(!(cin >> new\_enr))

{

clearbuf();

input\_error();

}

}while(check\_enroll\_year(new\_enr) != true);

input.enroll\_year = new\_enr;

}

if (n == 5)

{

string new\_fac;

cin.ignore();

getline(cin, new\_fac);

input.set\_fac(new\_fac);

}

if (n == 6)

{

string new\_kaf;

cin.ignore();

getline(cin, new\_kaf);

input.set\_kaf(new\_kaf);

}

if (n == 7)

{

string new\_group;

cin.ignore();

getline(cin, new\_group);

input.set\_group(new\_group);

}

if (n == 8)

{

int choice;

cout << "What would you like to do?" << endl;

cout << "1. Add new session" << endl;

cout << "2. Edit existing session" << endl;

while(!(cin >> choice))

{

clearbuf();

input\_error();

}

if (choice == 1)

{

if (input.get\_n() < 9)

{

int newsubj;

cout << "Enter number of subjects in new session: ";

while(!(cin >> newsubj))

{

clearbuf();

input\_error();

}

if (newsubj >= 1 && newsubj <= 10)

{

input.n\_sessions++;

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

{

subject tmp;

string s;

unsigned short x;

cout << "Enter name of the subject: ";

cin.ignore();

getline(cin, s);

do

{

cout << "Enter mark: ";

while(!(cin >> x))

{

clearbuf();

input\_error();

}

}while(check\_mark(x) != true);

tmp.name = s;

tmp.mark = x;

input.marks[input.n\_sessions-1].add\_subject(tmp);

}

}

else

{

cout << "Session can't have more than 10 subjects!" << endl;

}

}

else

{

cout << "Can't add more than 9 sessions!" << endl;

}

}

if (choice == 2)

{

unsigned short x;

cout << "Enter number of a session you want to edit(1-" << input.n\_sessions << "): ";

session\* cur;

while (!(cin >> x))

{

clearbuf();

input\_error();

}

if (x <= input.n\_sessions)

{

cur = &input.marks[x-1];

cout << "Session №" << x << ":\n";

cur->print();

while(true)

{

unsigned short seschoice;

cout << "1. Add subject" << endl;

cout << "2. Delete subject" << endl;

cout << "3. Edit mark for existing subject" << endl;

cout << "0. Quit session" << endl;

while(!(cin >> seschoice))

{

clearbuf();

input\_error();

}

if (seschoice == 1)

{

subject tmp;

string s;

unsigned short m;

cout << "Enter name of the subject: ";

cin.ignore();

getline(cin, s);

do

{

cout << "Enter mark: ";

while(!(cin >> x))

{

clearbuf();

input\_error();

}

}while(check\_mark(x) != true);

tmp.name = s;

tmp.mark = x;

cur->add\_subject(tmp);

}

if (seschoice == 2)

{

string del\_s;

cout << "Enter name of the subject: ";

cin.ignore();

getline(cin, del\_s);

cur->delete\_subject(del\_s);

}

if (seschoice == 3)

{

string edit\_sub;

cout << "Enter name of the subject: ";

cin.ignore();

getline(cin, edit\_sub);

unsigned short new\_mark;

cout << "Enter mark: ";

do

{

while(!(cin >> new\_mark))

{

clearbuf();

input\_error();

}

}while(check\_mark(new\_mark) != true);

cur->edit\_subject(edit\_sub, new\_mark);

}

if (seschoice == 0)

{

break;

}

}

}

else

{

cout << "Session not found!" << endl;

}

}

}

if (n == 0)

{

break;

}

}

student output = input;

return output;

}

## Исходный файл **my\_list**.h

#pragma once

#include "students.h"

typedef struct my\_list my\_list;

struct my\_list

{

struct my\_list\* nxt;

student val;

struct my\_list\* prv;

};

my\_list\* init\_list(student x);

my\_list\* add\_element(my\_list\* lst, student x);

my\_list\* delete\_element(my\_list\* lst);

my\_list\* delete\_list(my\_list\* lst);

my\_list\* find\_element\_by\_id(my\_list\* lst, string input);

my\_list\* get\_students\_by\_year(my\_list\* lst, unsigned short start, unsigned short end);

my\_list\* get\_students\_by\_mark(my\_list\* lst, unsigned short ses, unsigned short mark);

void print\_list(my\_list\* lst);

## Исходный файл my\_list.cpp

#include <iostream>

#include "my\_list.h"

my\_list\* init\_list(student x)

{

my\_list\* lst;

lst = new my\_list;

lst->val = x;

lst->prv = lst;

lst->nxt = lst;

return lst;

}

my\_list\* add\_element(my\_list\* lst, student x)

{

my\_list\* tmp;

my\_list\* p;

tmp = new my\_list;

p = lst->nxt;

lst->nxt = tmp;

tmp->val = x;

tmp->nxt = p;

tmp->prv = lst;

p->prv = tmp;

return tmp;

}

my\_list\* delete\_element(my\_list\* lst)

{

my\_list \*prev, \*next;

prev = lst->prv;

next = lst->nxt;

prev->nxt = next;

next->prv = prev;

delete lst;

return prev;

}

my\_list\* delete\_list(my\_list\* lst)

{

my\_list\* cur = lst;

my\_list\* fin = lst->prv;

do

{

my\_list\* tmp = cur->nxt;

delete cur;

cur = tmp;

}while(cur != fin);

return NULL;

}

my\_list\* find\_element\_by\_id(my\_list\* lst, string input)

{

my\_list\* cur;

cur = lst;

do

{

if (cur->val.get\_id() == input)

{

return cur;

}

cur = cur->nxt;

}while(cur != lst);

return NULL;

}

my\_list\* get\_students\_by\_year(my\_list\* lst, unsigned short start, unsigned short end)

{

my\_list\* cur = lst;

my\_list\* res = init\_list(student(1));

my\_list\* root = res;

int x;

bool flag = false;

do

{

x = cur->val.get\_birth\_year();

if(x >= start && x <= end)

{

res = add\_element(res, cur->val);

flag = true;

}

cur = cur->nxt;

}while(cur != lst);

if(flag)

{

res = delete\_element(root);

return res;

}

else

{

return NULL;

}

}

my\_list\* get\_students\_by\_mark(my\_list\* lst, unsigned short ses, unsigned short mark)

{

my\_list\* cur = lst;

my\_list\* res = init\_list(student(1));

my\_list\* root = res;

bool flag = false;

do

{

if(cur->val.get\_n() >= ses)

{

session tmp;

tmp = cur->val.get\_nth\_session(ses);

if (tmp.get\_avrg() >= (mark-0.5) && tmp.get\_avrg() < (mark+0.5))

{

res = add\_element(res, cur->val);

flag = true;

}

}

cur = cur->nxt;

}while(cur != lst);

if (flag)

{

res = delete\_element(root);

return res;

}

else

{

return NULL;

}

}

void print\_list(my\_list\* lst)

{

my\_list\* cur;

cur = lst;

do

{

cur->val.print();

cout << "=====================END OF ENTRY=====================" << endl;

cur = cur->nxt;

}while(cur != lst);

}

## Исходный файл validation.h

#pragma once

#include <string>

bool check\_birthdate(unsigned short day, unsigned short month, unsigned short year);

bool check\_sex(char input);

bool check\_enroll\_year(unsigned short input);

bool check\_mark(unsigned short input);

void clearbuf();

void input\_error();

## Исходный файл **validation**.cpp

#include <iostream>

#include <limits>

#include "validation.h"

using namespace std;

bool check\_birthdate(unsigned short day, unsigned short month, unsigned short year)

{

if (day == 0 || month ==0)

{

return false;

}

if (year < 1900 || year > 2020)

{

return false;

}

switch (month)

{

case 1:

if (day > 31)

{

return false;

}

break;

case 2:

if ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0)

{

if (day > 29)

{

return false;

}

}

else

{

if (day > 28)

{

return false;

}

}

break;

case 3:

if (day > 31)

{

return false;

}

break;

case 4:

if (day > 30)

{

return false;

}

break;

case 5:

if (day > 31)

{

return false;

}

break;

case 6:

if (day > 30)

{

return false;

}

break;

case 7:

if (day > 31)

{

return false;

}

break;

case 8:

if (day > 31)

{

return false;

}

break;

case 9:

if (day > 30)

{

return false;

}

break;

case 10:

if (day > 31)

{

return false;

}

break;

case 11:

if (day > 30)

{

return false;

}

break;

case 12:

if (day > 31)

{

return false;

}

break;

}

return true;

}

bool check\_sex(char input)

{

return (input == 'M' || input == 'F');

}

bool check\_enroll\_year(unsigned short input)

{

return (input >= 1947);

}

bool check\_mark(unsigned short input)

{

return (input >= 2 && input <= 5);

}

void clearbuf()

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

}

void input\_error()

{

cout << "Invalid data! Try again:\n";

}

## Исходный файл **read**.h

#pragma once

#include "students.h"

#include "my\_list.h"

#include "validation.h"

using namespace std;

student read\_student(my\_list\* data);

my\_list\* read\_from\_keyboard();

## Исходный файл **read**.cpp

#include <iostream>

#include "read.h"

#include "my\_list.h"

#include "validation.h"

using namespace std;

student read\_student(my\_list\* data)

{

string lastname;

cout << "Enter last name: ";

cin.clear();

getline(cin, lastname);

string firstname;

cout << "Enter first name: ";

cin.clear();

getline(cin, firstname);

string middlename;

cout << "Enter middle name: ";

cin.clear();

getline(cin, middlename);

unsigned short date[3];

cout << "Enter date of birth\n";

do

{

cout << "Day: ";

while(!(cin >> date[0]))

{

clearbuf();

input\_error();

}

cout << "Month: ";

while(!(cin >> date[1]))

{

clearbuf();

input\_error();

}

cout << "Year: ";

while(!(cin >> date[2]))

{

clearbuf();

input\_error();

}

}while(check\_birthdate(date[0], date[1], date[2]) != true);

char s;

do

{

cout << "Enter sex: ";

cin >> s;

clearbuf();

}while(check\_sex(s) != true);

unsigned short enroll;

do

{

cout << "Enter enroll year: ";

while(!(cin >> enroll))

{

clearbuf();

input\_error();

}

}while(check\_enroll\_year(enroll) != true);

clearbuf();

string facul;

cout << "Enter faculty: ";

cin.clear();

getline(cin, facul);

string dep;

cout << "Enter department: ";

cin.clear();

getline(cin, dep);

string gro;

cout << "Enter group: ";

cin >> gro;

clearbuf();

string zachotka;

cout << "Enter ID: ";

while(true)

{

cin >> zachotka;

clearbuf();

if (data != NULL)

{

if (find\_element\_by\_id(data, zachotka) != NULL)

{

cout << "Error: this ID already exists. Try again: ";

continue;

}

}

break;

}

int sesnum;

do{

cout << "Enter number of sessions: ";

while(!(cin >> sesnum))

{

clearbuf();

input\_error();

}

}while(!(sesnum >= 1 && sesnum <= 9));

session ses[sesnum];

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

{

int subjnum = 0;

do

{

cout << "Enter number of subjects in session №" << i+1 << ": ";

while(!(cin >> subjnum))

{

clearbuf();

input\_error();

}

}while(!(subjnum >= 1 && subjnum <= 10));

ses[i].set\_n(subjnum);

subject input[subjnum];

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

{

string subname;

unsigned short submark = 0;

cout << "Enter name of the subject: ";

cin.ignore();

getline(cin, subname);

do

{

cout << "Enter mark: ";

while(!(cin >> submark))

{

clearbuf();

input\_error();

}

}while(check\_mark(submark) != true);

input[j].name = subname;

input[j].mark = submark;

}

ses[i].set(input);

}

student x = student(sesnum);

x.set\_full\_name(lastname, firstname, middlename);

x.set\_birthdate(date[0], date[1], date[2]);

x.set\_sex(s);

x.set\_enroll\_year(enroll);

x.set\_fac(facul);

x.set\_kaf(dep);

x.set\_group(gro);

x.set\_id(zachotka);

x.set\_marks(ses);

return x;

}

my\_list\* read\_from\_keyboard()

{

clearbuf();

my\_list\* studlist = NULL;

student x = read\_student(studlist);

studlist = init\_list(x);

char flag;

while(true)

{

cout << "Add more(y/n)? ";

do

{

cin >> flag;

clearbuf();

}while(!(flag == 'y' || flag == 'n'));

if (flag == 'n')

{

break;

}

studlist = add\_element(studlist, read\_student(studlist));

}

return studlist;

}

## Исходный файл crypto.h

#pragma once

#include <iostream>

#include "cryptopp/default.h"

#include "cryptopp/hex.h"

std::string encrypt(std::string input, std::string password);

std::string decrypt(std::string input, std::string password);

## Исходный файл **crypto**.cpp

#include "crypto.h"

std::string encrypt(std::string input, std::string password)

{

std::string output;

using namespace CryptoPP;

StringSource ss1(input, true,

new DefaultEncryptorWithMAC(

(byte\*)password.data(), password.size(),

new HexEncoder(

new StringSink(output)

)

)

);

return output;

}

std::string decrypt(std::string input, std::string password)

{

std::string output;

using namespace CryptoPP;

StringSource ss1(input, true,

new HexDecoder(

new DefaultDecryptorWithMAC(

(byte\*)password.data(), password.size(),

new StringSink(output)

)

)

);

return output;

}

## Исходный файл files.h

#pragma once

#include "my\_list.h"

void save\_data(my\_list\* lst);

my\_list\* read\_from\_file(my\_list\* lst);

## Исходный файл **files**.cpp

#include <iostream>

#include <fstream>

#include "files.h"

#include "crypto.h"

using namespace std;

void save\_data(my\_list\* lst)

{

my\_list\* cur = lst;

ofstream fout;

string filename;

string password;

cout << "Enter name of the file: ";

cin >> filename;

cout << "Enter password: ";

cin >> password;

fout.open(filename);

do

{

student tmp = cur->val;

fout << encrypt(tmp.get\_name(0), password) << endl;

fout << encrypt(tmp.get\_name(1), password) << endl;

fout << encrypt(tmp.get\_name(2), password) << endl;

fout << encrypt(tmp.get\_birthdate(), password) << endl;

fout << encrypt(string(1, tmp.get\_sex()), password) << endl;

fout << encrypt(to\_string(tmp.get\_enroll\_year()), password) << endl;

fout << encrypt(tmp.get\_fac(), password) << endl;

fout << encrypt(tmp.get\_kaf(), password) << endl;

fout << encrypt(tmp.get\_group(), password) << endl;

fout << encrypt(tmp.get\_id(), password) << endl;

fout << encrypt(to\_string(tmp.get\_n()), password) << endl;

for(int i = 0; i < tmp.get\_n(); i++)

{

session tmp\_ses = tmp.get\_nth\_session(i+1);

subject\* tmp\_subj = tmp\_ses.get();

for(int j = 0; j < tmp\_ses.get\_n(); j++)

{

fout << encrypt(((tmp\_subj + j)->name + ": " + to\_string((tmp\_subj + j)->mark)), password) << endl;

}

fout << encrypt("END OF SESSION", password) << endl;

}

fout << encrypt("END OF ENTRY", password) << endl;

cur = cur->nxt;

}while(cur != lst);

fout << encrypt("END OF FILE", password) << endl;

fout.close();

}

my\_list\* read\_from\_file(my\_list\* lst)

{

lst = init\_list(student(1));

my\_list\* root = lst;

ifstream fin;

string filename;

string password;

string s = "";

cout << "Enter name if the file: ";

cin >> filename;

cout << "Enter password: ";

cin >> password;

fin.open(filename);

while(true)

{

student x;

getline(fin, s);

s = decrypt(s, password);

if (s == "END OF FILE")

{

break;

}

string name[3];

name[0] = s;

getline(fin, s);

s = decrypt(s, password);

name[1] = s;

getline(fin, s);

s = decrypt(s, password);

name[2] = s;

x.set\_full\_name(name[0], name[1], name[2]);

getline(fin, s);

s = decrypt(s, password);

unsigned short date[3];

int cur, prev = 0;

cur = s.find('.');

date[0] = stoi(s.substr(prev, cur - prev));

prev = cur + 1;

cur = s.find('.', prev);

date[1] = stoi(s.substr(prev, cur - prev));

prev = cur + 1;

cur = s.find('.', prev);

date[2] = stoi(s.substr(prev, cur - prev));

x.set\_birthdate(date[0], date[1], date[2]);

getline(fin, s);

s = decrypt(s, password);

if(s == "M")

{

x.set\_sex('M');

}

else

{

x.set\_sex('F');

}

getline(fin, s);

s = decrypt(s, password);

x.set\_enroll\_year(stoi(s));

getline(fin, s);

s = decrypt(s, password);

x.set\_fac(s);

getline(fin, s);

s = decrypt(s, password);

x.set\_kaf(s);

getline(fin, s);

s = decrypt(s, password);

x.set\_group(s);

getline(fin, s);

s = decrypt(s, password);

x.set\_id(s);

getline(fin, s);

s = decrypt(s, password);

unsigned short n = stoi(s);

x.set\_n(n);

session mrks[n];

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

{

getline(fin, s);

s = decrypt(s, password);

mrks[i].set\_n(0);

while(s != "END OF SESSION")

{

subject tmp;

tmp.name = s.substr(0, s.find(':'));

tmp.mark = stoi(s.substr(s.find(':') + 1, string::npos));

mrks[i].add\_subject(tmp);

getline(fin, s);

s = decrypt(s, password);

}

}

x.set\_marks(mrks);

getline(fin, s);

s = decrypt(s, password);

lst = add\_element(lst, x);

}

fin.close();

lst = delete\_element(root);

return lst;

}

## Исходный файл menu.h

#pragma once

#include "my\_list.h"

my\_list\* menu(int x, my\_list\* lst);

my\_list\* edit\_entry(my\_list\* input);

my\_list\* delete\_entry(my\_list\* input);

void print\_result(my\_list\* input);

## Исходный файл **menu**.cpp

#include <iostream>

#include "menu.h"

#include "read.h"

#include "files.h"

using namespace std;

my\_list\* menu(int x, my\_list\* lst)

{

switch(x)

{

case 1:

if (lst != NULL)

lst = delete\_list(lst);

lst = read\_from\_keyboard();

break;

case 2:

if (lst != NULL)

lst = delete\_list(lst);

lst = read\_from\_file(lst);

break;

case 3:

if (lst != NULL)

{

cin.ignore();

lst = add\_element(lst, read\_student(lst));

}

break;

case 4:

if (lst != NULL)

lst = edit\_entry(lst);

break;

case 5:

if (lst != NULL)

lst = delete\_entry(lst);

break;

case 6:

if (lst != NULL)

print\_list(lst);

break;

case 7:

if (lst != NULL)

print\_result(lst);

break;

case 8:

if (lst != NULL)

save\_data(lst);

break;

case 0:

if (lst != NULL)

lst = delete\_list(lst);

exit(0);

break;

default:

cout << "Error: no matching options\n";

break;

}

return lst;

}

my\_list\* edit\_entry(my\_list\* input)

{

string ed\_id;

cout << "Enter ID of student you want to edit: ";

cin >> ed\_id;

my\_list\* x = find\_element\_by\_id(input, ed\_id);

if (x == NULL)

{

cout << "Entry not found!" << endl;

}

else

{

x->val = edit(x->val);

}

return input;

}

my\_list\* delete\_entry(my\_list\* input)

{

string del\_id;

cout << "Enter ID of student you want to delete: ";

cin >> del\_id;

my\_list\* del = find\_element\_by\_id(input, del\_id);

if (del == NULL)

{

cout << "Entry not found!" << endl;

}

else

{

if (del == del->nxt)

{

input = delete\_list(input);

}

else

{

input = delete\_element(del);

}

}

return input;

}

void print\_result(my\_list\* input)

{

unsigned short beg, end;

cout << "Enter birth year interval\n";

cout << "Start: ";

while(!(cin >> beg))

{

clearbuf();

input\_error();

}

cout << "End: ";

while(!(cin >> end))

{

clearbuf();

input\_error();

}

my\_list\* output = get\_students\_by\_year(input, beg, end);

if(output == NULL)

{

cout << "No students found in interval " << beg << "-" << end << endl;

}

else

{

unsigned short sesnum;

cout << "Enter a number of session to sort through: ";

while(!(cin >> sesnum))

{

clearbuf();

input\_error();

}

my\_list\* a = get\_students\_by\_mark(output, sesnum, 5);

if (a == NULL)

{

cout << "No A students found!" << endl;

}

else

{

cout << "A students:" << endl;

print\_list(a);

}

my\_list\* b = get\_students\_by\_mark(output, sesnum, 4);

if (b == NULL)

{

cout << "No B students found!" << endl;

}

else

{

cout << "B students:" << endl;

print\_list(b);

}

my\_list\* c = get\_students\_by\_mark(output, sesnum, 3);

if (c == NULL)

{

cout << "No C students found!" << endl;

}

else

{

cout << "C students:" << endl;

print\_list(c);

}

}

}

## Исходный файл **main**.cpp

#include <iostream>

#include <limits>

#include "menu.h"

using namespace std;

int main()

{

int x;

my\_list\* foo;

cout << "Welcome! Choose an option:\n";

while(true)

{

cout << "1. Enter data from keyboard\n";

cout << "2. Read data from file\n";

cout << "3. Add an entry\n";

cout << "4. Edit an entry\n";

cout << "5. Delete an entry\n";

cout << "6. Print all entries\n";

cout << "7. Print sort result\n";

cout << "8. Save data to a file\n";

cout << "0. Exit\n";

cout << "Enter number(0-9): ";

while (!(cin >> x))

{

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cout << "Invalid data! Try again:\n";

}

foo = menu(x, foo);

}

return 0;

}