ОПИСАНИЕ ИТОГОВОГО ЗАДАНИЯ ПО МОДУЛЮ 16 «КОНТЕЙНЕРЫ STL»

Представленная программа реализует итоговое задание модуля 16 «STL» - реализацию модели чата с использованием контейнеров STL. Программа состоит из четырех классов, разделенных на заголовочные файлы и файлы реализации, одного заголовочного файла для реализации некоторых вспомогательных функций, и управляющего блока main. Упомянутые классы и файлы представлены в таблице:

|  |  |  |
| --- | --- | --- |
| КЛАСС | ФАЙЛЫ КЛАССА | НАЗНАЧЕНИЕ |
| HashTable | HashTable.h  HashTable.cpp | Запись паролей и логинов в хеш-таблицу, представленную контейнером unordered\_map, сохранение таблицы в файл hashTable.txt и ее загрузка из файла при запуске программы для проверки пароля |
| User | User.h  User.cpp | Регистрация и создание списка пользователей чата, поиск и удаление пользователей. Хранение списка пользователей в файле listOfUsers.txt |
| Message | Message.h  Message.cpp | Создание индивидуальных сообщений конкретным пользователям, их сохранение в текстовом файле Messages.txt, загрузка и чтение из файла |
| AllMessages | AllMessages.h  AllMessages.cpp | Создание общих сообщений для всех пользователей, их сохранение в текстовом файле AllMessages.txt, загрузка и чтение из файла |

Класс HashTable

В этом классе определен контейнер std::unordered\_map <std::string, int> list\_of\_users, предназначенный для записи и сохранения пар «пароль – логин».

В качестве логинов используются личные номера пользователей (хотя, в принципе, можно использовать и номера телефонов), автоматически создаваемые при регистрации функцией-счетчиком int countNumber(), определенной в заголовке specialFunc.h. Работа функции основана на извлечении значения номера из файла «Numbers.txt» и его инкрементировании, после чего новое значение присваивается переменной номера пользователя и перезаписывается в тот же файл. Таким образом, каждый пользователь при регистрации получает уникальный идентификационный номер.

Пароль при регистрации пользователя также генерируется автоматически функцией std::string random\_string(std::size\_t length), определенной в том же заголовке. Функция создает случайную комбинацию символов, длина которой определяется параметром length. В нашей программе задан восьмисимвольный пароль, но его длина может быть задана любой, для чего также придется изменить аргументы функции substr(), извлекающей подстроки пароля и личного номера в функциях getpass() и getID().

Запись и сохранение пароля и номера осуществляется функцией fillMap(std::string password, int num). Аргументы функции задаются в блоке main переменными std::string password и int ID\_Number{}, значения которым присваиваются автоматически, после чего сохранются в файл hashTable.txt.

Извлечение паролей и номеров из файла и загрузка их в контейнер list\_of\_users при запуске программы осуществляется функцией insert({ getpass(i), getID(i) }), находящейся внутри цикла в функции extractMap(). Число итераций цикла определяется переменной n, значение которой задается функцией countPasses(), подсчитывающей число строк в файле hashTable.txt. При этом указанное число уменьшается на единицу, чтобы избежать загрузки последней пустой строки.

Аргументами функции insert() являются вспомогательные функции getpass() и getID(). Эти функции читают строки из файла hashTable.txt с помощью находящейся в них функции поиска подстроки substr(). В первом случае функция substr() имеет аргументы (0, 8), которыми задается извлечение восьми символов пароля. Во втором она имеет аргумент 9 (извлечение всех символов после девятого, поскольку длина номера может быть любой).

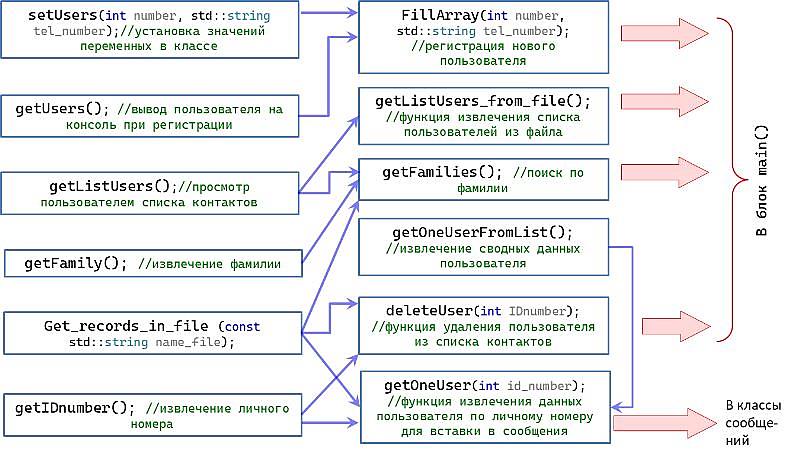
После извлечения пароль и номер с помощью функции push\_back() загружаются во вспомогательные контейнеры std::vector<std::string> arr\_pass и std::vector<int> arr\_id, и уже из этих контейнеров функция insert() загружает пары значений (соответственно их индексам в массивах) в контейнер list\_of\_users. Так приходится делать, во-первых, потому, что сохранение контейнеров в базах данных еще не рассматривалось. Во-вторых, прямо записать строки в контейнер list\_of\_users нельзя (во всяком случае, я такого способа не нашел), их нужно разделить на пары значений для пропуска паролей через внутреннюю хеш-функцию контейнера. К тому же строки текстового файла индексов, как таковых, не имеют, и любой цикл выведет только первую строку. И только загрузка в массивы позволяет эти данные проиндексировать.

Далее, функция deleteElement(std::string passwrd) с помощью внутренней функции erase(passwrd) участвует в удалении элемента по введенному паролю из списка list\_of\_users и перезаписи списка в текстовый файл уже без данных удаленного пользователя.

И наконец, функция on\_guard(std::string password, int ID\_Number) с помощью функции find(password) проводит поиск введенного пароля в контейнере list\_of\_users для допуска или недопуска в чат. Причем проверяется не только сам пароль, но и соответствие пароля номеру пользователя (чтобы недобросовестный пользователь, введя не свой номер, не смог бы послать сообщение от чужого имени).

Класс User

В этом классе создается список пользователей чата. В нем определен массив объектов класса для хранения списка – контейнер std::vector<User> arr. При этом в функции-установщике setUsers(int number, std::string tel\_number) номер телефона задается при регистрации в блоке main() функцией-маской autonumber(), определенной в заголовке specialFunc.h, а личный номер задается функцией-счетчиком countNumber(), определенной там же. При этом функция FillArray(), вызываемая при регистрации пользователя, сохраняет эти данные в текстовый файл listOfUsers.txt, а функция getListUsers\_from\_file() при загрузке программы считывает список пользователей из файла и выводит на экран. Взаимодействие функций класса User представлено на нижеследующей схеме:



Класс Message

В этом классе осуществляется создание и сохранение сообщений конкретным пользователям. Созданные сообщения функцией createMessages(int id\_number\_of, int id\_number\_in) загружаются в массив объектов класса – контейнер std::vector<Message> array и из него сохраняются в файл Messages.txt. Находящаяся внутри функции createMessages() устновочная функция setData(int id\_number\_of, int id\_number\_in) устанавливает следующие значения переменных:

1. Переменная времени получает значение от функции Time(), генерирующей текущую дату и время отправки сообщения;
2. Переменная номера отправителя получает значение номера, введенного пользователем при входе в программу;
3. Значение переменной номера получателя вводится пользователем при создании сообщения;
4. Данные отправителя и получателя загружаются автоматически функцией User::getOneUser() соответственно введенным номерам

Функция countRecords() подсчитывает число строк в файле сообщений для чтения сообщений из файла, а функция readPersonalMessages(int ID\_Number) получает номер, введенный пользователем при входе в программу и выводит на экран только сообщения, отправленные пользователем либо адресованные ему, что исключает возможность чтения чужих сообщений.

Класс AllMessades, создающий и читающий сообщения для всех пользователей, полностью подобен классу Message, за исключением того, что для переменной имени получателя установлено значение «всем».