Messenger

Step 5

Database STL

Каркас программы уже написан. Выберите вариант `one` или вариант `two` (усложненный). Допишите код вместо `//...` реализующий требуемую функцию. Ниже приведено описание архитектуры для варианта `one`.

Класс `Msg` включает основной функционал мессенджера. База данных реализована в следующих контейнерах из библиотеки STL:

```
std::map<std::string, int> d_login_id_; // словарь логин – id пользователя
```

```
std::map<int, User> d_id_user_; // словарь id пользователя – User
```

```
std::map<int, std::vector<int>> d_id_contacts_; // словарь id пользователя – список друзей пользователя в виде вектора, содержащего id его друзей
```

```
std::vector<Message> d_message_; // все сообщения всех пользователей в виде вектора сообщений
```

В классе присутствуют следующие функции:

`void singup(std::string login, std::string pass)`

Если пользователь с указанным `login` существует выбрасывает исключение, иначе производит регистрацию пользователя. Для этого сперва увеличивается счётчик `USER_LAST_ID`, это значение будет являться `id` для регистрируемого пользователя. Далее добавляет `id`, `login` и `pass` регистрируемого пользователя в словари **`d_login_id_`** и **`d_id_user_`**

void singin(std::string login, std::string pass)

Производит аутентификацию пользователя по login и pass. Из словаря **d_id_user_** находится пароль для пользователя. Если он равен pass, переменной AUTH_ID присваивается id пользователя, иначе выбрасывается исключение

std::vector<User> search();

Возвращает вектор всех пользователей, сформированный из словаря **d_id_user_**

void add(int friend_id)

Добавляет пользователя с идентификатором friend_id в список контактов: в словарь **d_id_contacts_** по ключу AUTH_ID добавляется friend_id. Обратите внимание, что в качестве значения в словаре **d_id_contacts_** выступает вектор целого типа (id друзей) std::vector<int>:

```
d_id_contacts_[AUTH_ID].push_back(friend_id)
```

std::vector<User> contacts()

Возвращает список контактов пользователя AUTH_ID в виде вектора пользователей. Вектор пользователей формируется из словаря **d_id_contacts_** и **d_id_user_**

void remove(int friend_id)

удаляет из вектора словаря контактов **d_id_contacts_** по ключу AUTH_ID друга

void send(int friend_id, std::string mess)

Сохраняет в вектор **d_message_** сообщение mess от пользователя AUTH_ID пользователю friend_id

vector<Message> read(int friend_id)

Возвращает переписку между AUTH_ID и login: из вектора **d_message_** отфильтровывает все сообщения от AUTH_ID к friend_id и от friend_id к AUTH_ID