Создайте чат со следующим функционалом:

1. консольная программа
2. регистрация пользователей - логин, пароль, имя
3. вход в чат по логину/паролю
4. отправка сообщений конкретному пользователю
5. обмен сообщениями между всеми пользователями чата одновременно

Обязательным условием является использование классов.Создайте чат со следующим функционалом:

1. консольная программа
2. регистрация пользователей - логин, пароль, имя
3. вход в чат по логину/паролю
4. отправка сообщений конкретному пользователю
5. обмен сообщениями между всеми пользователями чата одновременно

Обязательным условием является использование классов.  
  
#include <iostream>

#include <string>

#include <vector>

class User {

private:

std::string username;

std::string password;

std::string name;

public:

User(const std::string& username, const std::string& password, const std::string& name) :

username(username), password(password), name(name) {}

const std::string& getUsername() const {

return username;

}

const std::string& getPassword() const {

return password;

}

const std::string& getName() const {

return name;

}

};

class Chat {

private:

std::vector<User> users;

public:

void registerUser(const std::string& username, const std::string& password, const std::string& name) {

users.emplace\_back(username, password, name);

std::cout << "Пользователь " << username << " успешно зарегистрирован." << std::endl;

}

User\* login(const std::string& username, const std::string& password) {

for (User& user : users) {

if (user.getUsername() == username && user.getPassword() == password) {

std::cout << "Пользователь " << username << " успешно вошел в чат." << std::endl;

return &user;

}

}

std::cout << "Ошибка входа. Проверьте введенные данные." << std::endl;

return nullptr;

}

void sendMessage(const User\* sender, const User\* recipient, const std::string& message) {

std::cout << sender->getName() << " отправил сообщение пользователю " << recipient->getName() << ": " << message << std::endl;

}

void broadcastMessage(const User\* sender, const std::string& message) {

for (const User& user : users) {

if (&user != sender) {

std::cout << sender->getName() << " отправил сообщение всем пользователям чата: " << message << std::endl;

}

}

}

};

int main() {

Chat chat;

std::string choice;

while (true) {

std::cout << std::endl;

std::cout << "1. Зарегистрироваться" << std::endl;

std::cout << "2. Войти в чат" << std::endl;

std::cout << "3. Отправить сообщение пользователю" << std::endl;

std::cout << "4. Отправить сообщение всем пользователям" << std::endl;

std::cout << "5. Выйти из чата" << std::endl;

std::cout << std::endl;

std::cout << "Введите номер действия: ";

std::cin >> choice;

if (choice == "1") {

std::string username, password, name;

std::cout << "Введите логин: ";

std::cin >> username;

std::cout << "Введите пароль: ";

std::cin >> password;

std::cout << "Введите имя: ";

std::cin >> name;

chat.registerUser(username, password, name);

}

else if (choice == "2") {

std::string username, password;

std::cout << "Введите логин: ";

std::cin >> username;

std::cout << "Введите пароль: ";

std::cin >> password;

User\* currentUser = chat.login(username, password);

// Дополнительная логика при успешном входе в чат

}

else if (choice == "3") {

// Логика отправки сообщения конкретному пользователю

}

else if (choice == "4") {

// Логика отправки сообщения всем пользователям

}

else if (choice == "5") {

// Логика выхода из чата

break;

}

else {

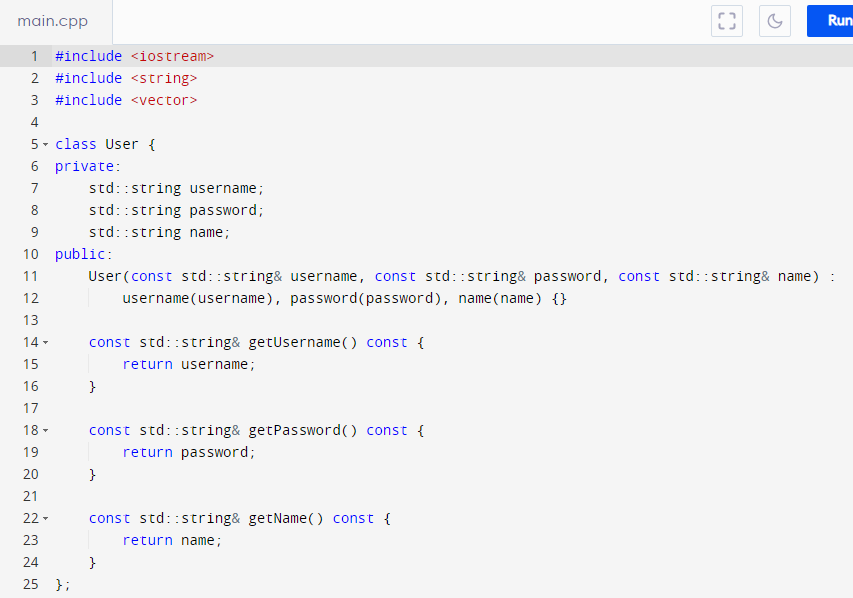
std::cout << "Неверный выбор. Повторите попытку." << std::endl;

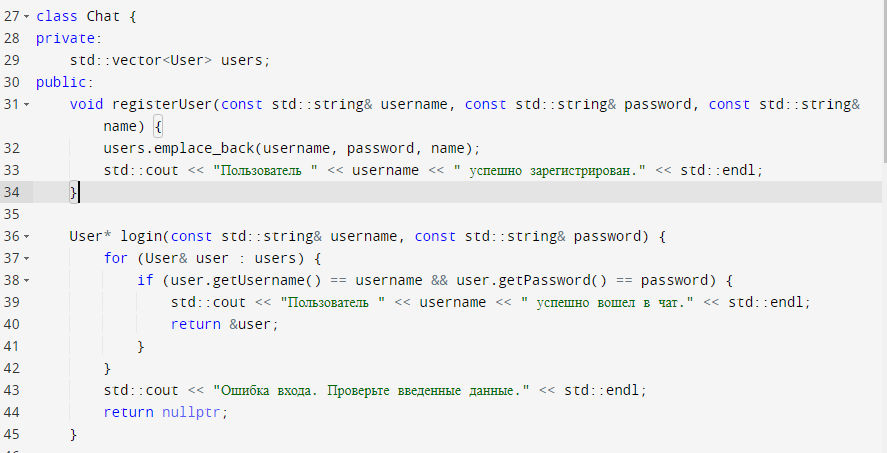
}

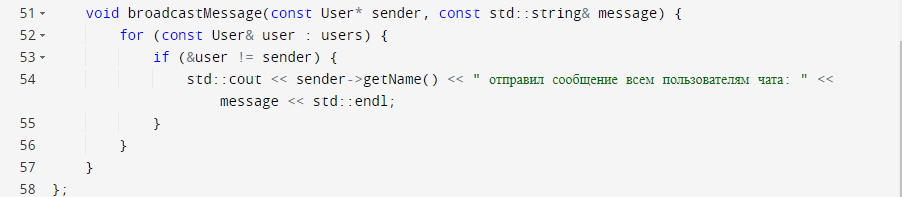
}

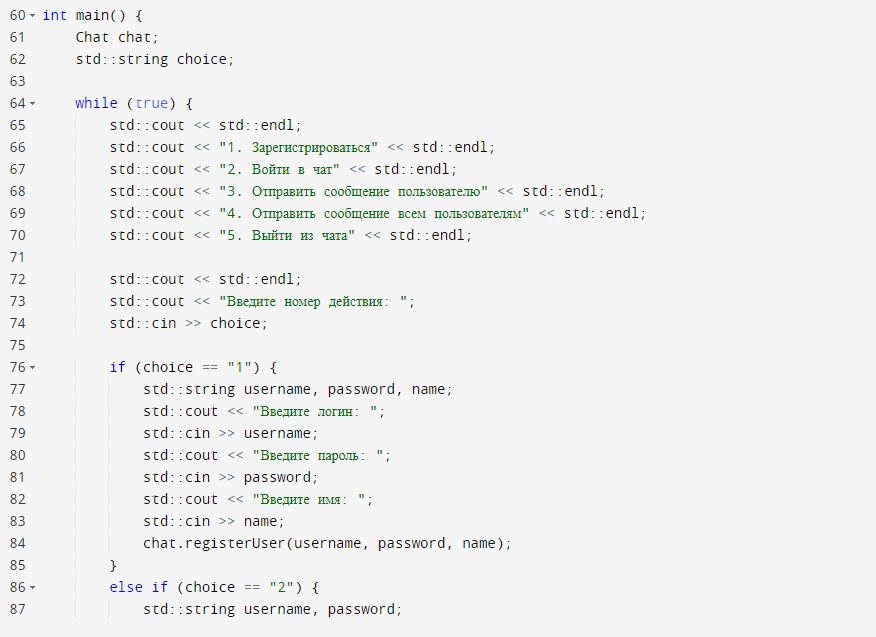
return 0;

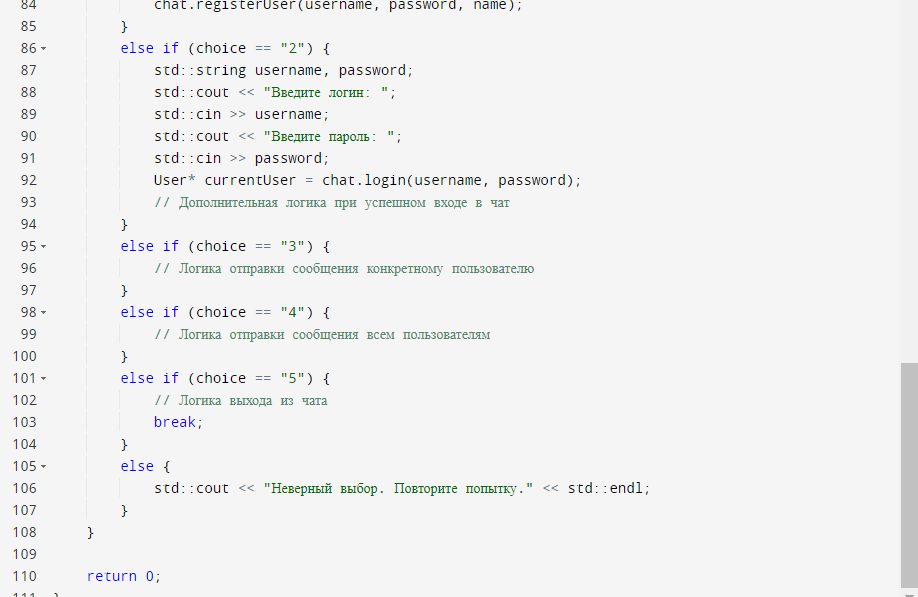
}

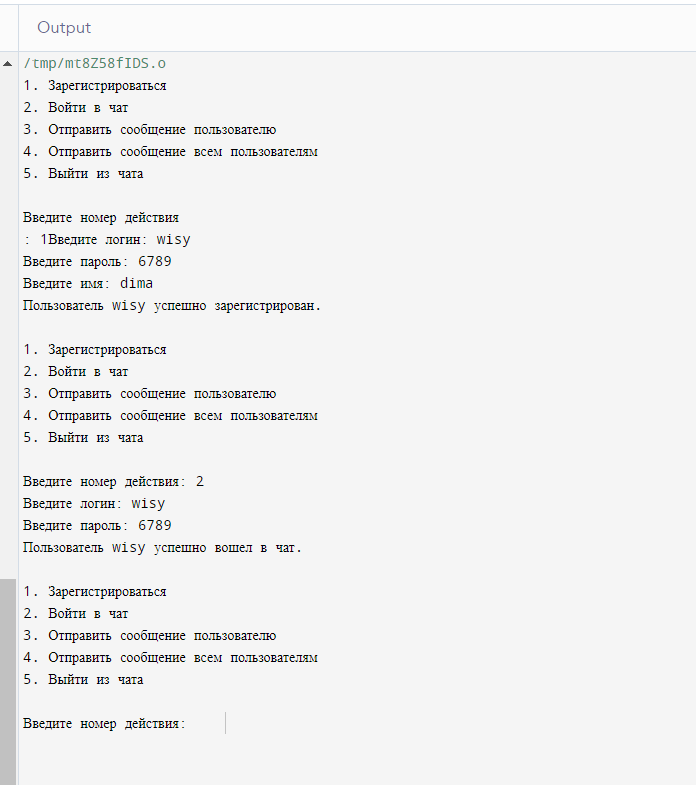








**Результат:**

****

Пошагово:

1. Сформулировать и оформить идею и техническое задание
2. Составить архитектуру – какие классы будут, как они будут взаимодействовать между собой. Оформить архитектуру в схему.
3. Не стремитесь сделать большое и сложное приложение. Заложите архитектуру самой базовой функциональности — отправка и чтение сообщений, авторизация и регистрация пользователей.

**Идея:**

Разработка простого чата с функционалом отправки и чтения сообщений, а также авторизацией и регистрацией пользователей.

Техническое задание:

1. Регистрация пользователей:

- Пользователь должен иметь уникальное имя пользователя (username), пароль и имя;

- Данные о пользователях должны храниться в базе данных;

- Необходимо реализовать функцию регистрации, которая будет проверять уникальность имени пользователя и добавлять нового пользователя в базу данных.

2. Авторизация пользователей:

- Пользователь должен вводить свой логин и пароль для входа в чат;

- Необходимо реализовать функцию авторизации, которая будет проверять соответствие введенных пользователем данных данным в базе данных.

3. Отправка сообщений:

- Зарегистрированный и авторизованный пользователь должен иметь возможность отправлять сообщения другим пользователям в чате;

- Отправленные сообщения должны сохраняться в базе данных;

- Необходимо реализовать функцию отправки сообщений, которая будет сохранять сообщение в базе данных.

4. Чтение сообщений:

- Зарегистрированный и авторизованный пользователь должен иметь возможность прочитать все полученные им сообщения;

- Необходимо реализовать функцию чтения сообщений, которая будет возвращать список сообщений, полученных данным пользователем.

**Архитектура:**

1. Класс User:

- Свойства: username (строка), password (строка), name (строка).

- Методы: конструктор, геттеры.

2. Класс Chat:

- Свойства: users (вектор объектов User).

- Методы: registerUser(username, password, name), login(username, password), sendMessage(sender, recipient, message), readMessages(user).

3. Класс Database:

- Свойства: users (таблица пользователей), messages (таблица сообщений).

- Методы: addUser(user), findUser(username), addMessage(message), getMessages(user).



Readme:

Зыков, Артюх, Федорович;

имя тимлида (Дмитрий Артюх);

Дмитрий-Код:

Роман-Архитектура;

Игорь-Заполнение документа;

Выбранная идея решения - разработка консольного чата на языке C++. Цель проекта - создать программу, которая позволяет пользователям зарегистрироваться, войти в систему и обмениваться сообщениями друг с другом.

Основные функции, которые будет реализовывать программа:

1. Регистрация пользователя: Пользователь может создать учетную запись, указав логин, пароль и имя.

2. Вход в систему: Зарегистрированные пользователи могут войти в систему, вводя свои данные - логин и пароль.

3. Отправка сообщений: Зарегистрированные пользователи могут отправлять сообщения другим пользователям, указывая имя получателя и текст сообщения.

4. Просмотр сообщений: Пользователи могут просматривать все отправленные и полученные ими сообщения.

Для управления всеми пользователями и сообщениями будет использована структура данных стандартной библиотеки C++ - мапа для хранения информации о зарегистрированных пользователях и вектор для хранения сообщений.

Программа будет иметь простой интерфейс командной строки, в котором пользователь будет выбирать нужные ему опции, вводя соответствующие команды.Приведенный выше пример кода демонстрирует основные функции, которые можно реализовать в рамках данной задачи. Однако, в реальном проекте возможно добавление дополнительных функций, как например, дополнительные команды для удаления пользователя, работа с файлами и т.д.;



1. Пользовательский тип "User":

- Описывает атрибуты и методы, связанные с пользователем.

- Атрибуты:

- username (строка): уникальное имя пользователя.

- password (строка): пароль пользователя.

- name (строка): имя пользователя.

- Методы:

- register(): метод для регистрации нового пользователя.

- login(): метод для авторизации пользователя.

- send\_message(receiver, message): метод для отправки сообщения другому пользователю.

- receive\_message(sender, message): метод для получения сообщения от другого пользователя.

2. Функция "register\_user":

- Отвечает за регистрацию нового пользователя.

- Принимает имя пользователя, пароль и имя в качестве параметров.

- Создает экземпляр класса "User" соответствующего нового пользователя.

- Сохраняет данные пользователя в базе данных.

3. Функция "login\_user":

- Отвечает за авторизацию пользователя.

- Принимает имя пользователя и пароль в качестве параметров.

- Проверяет соответствие введенных данных с данными пользователей, хранящихся в базе данных.

- Возвращает экземпляр класса "User", если пользователь успешно авторизован, или None, если данные неверны.

4. Функция "send\_message":

- Отправляет сообщение от одного пользователя другому.

- Принимает отправителя, получателя и текст сообщения в качестве параметров.

- Создает и сохраняет сообщение в базе данных.

- В зависимости от требований проекта, может оповещать получателя о новом сообщении.

5. Функция "get\_messages":

- Получает сообщения для указанного пользователя.

- Принимает пользователя в качестве параметра.

- Извлекает из базы данных все сообщения, адресованные данному пользователю, и возвращает их.