SKILLFACTORY

Курс: разработчик на С++

Поток: CPLUS-1005

Итоговый проект по Блоку «Основные конструкции C++»

Исполнитель: Литвинович Вячеслав Александрович.

-= Орел 2023 =-

Содержание:

1. Оглавление Список участников команды.3
2. Описание выбранной идеи решения4
3. Описание пользовательских типов и функций в проекте 5

**Для работы с программой обязательно наличие файлов Users.txt, Message.txt и MessagePC.txt в папке с программой!!!**

1. Список участников команды.

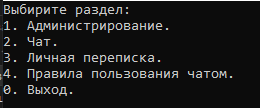
В виду отсутствия свободных сокурсников работа выполнена единолично Литвиновичем Вячеславом Александровичем.

1. Описание выбранной идеи решения.

Не смотря на простоту реализации, проект является самодостаточным рабочим продуктом.

В данном проекте реализована переписка: на ряду с общей перепиской, реализована личная переписка с выбранным пользователем, а также раздел администрирования.

При запуске программы пользователю дается выбор в каком разделе он хочет продолжить:



Раздел 1: Администрирование, дает возможность:

1. Участвовать в общей переписке.
2. Вести личную переписку с любым пользователем.
3. Закрывать доступ любому пользователю к программе.
4. Удалять сообщения, не соответствующие цензуре.

Раздел 2: Чат, дает возможность:

1. Войти в чат под своими логином и паролем.
2. Зарегистрировать нового пользователя.
3. Просматривать общий чат.
4. Добавлять сообщения в общий чат.

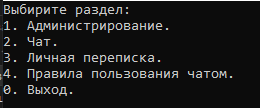
Раздел 3: Личная переписка, дает возможность:

1. Войти под своими логином и паролем.
2. Выбрать собеседника.
3. Просматривать сообщения с выбранным собеседником.
4. Добавлять сообщения в личную переписку с выбранным собеседником.

Раздел 4: Правила пользования чатом, описывает правила и ограничения.

1. Описание пользовательских типов и функций в проекте

Вход в программу (функция int main()) находится в файле main.cpp, в функции main прописан бесконечный цикл, дающий возможность повторного выбора, после того как пользователь закончит работу с выбранной веткой дерева развития событий. Первоочередно пользователю дается выбор:



В зависимости от выбора запускается одно из четырех функций, объявленных в файле Menu.h и определенных в файле Menu.cpp.

void administration();

void chat();

void perCor();

void regulation();

Функции самодостаточны, ни каких параметров не принимают и значения не возвращают.

Функция administration –предназначена для работы администратора.

void administration()

{

Admin Admin;

User user(string, const string&, const string&, const string&);

Admin.loginMenu();

if (Admin.getLogin()) Admin.adminMenu();

}

Сначала объявляется экземпляр класса Admin, с именем Admin. Далее объявляется класс User с именем user. Далее запускается метод loginMenu экземпляра класса Admin (выбор пользователя - будет рассмотрен далее). Если при выполнении метода loginMenu экземпляра класса Admin пароль введен корректно запускается метод adminMenu экземпляра класса Admin.

Функция chat – предназначена для реализации общего чата.

void chat()

{

Chat Сhat;

User user (const string&, const string&, const string&);

Сhat.loginMenu();

if (Сhat.getCurrentUser())Сhat.showChat();

}

Сначала объявляется экземпляр класса Сhat, с именем Сhat. Далее объявляется класс User с именем user. Далее запускается метод loginMenu экземпляра класса Сhat (выбор пользователя - будет рассмотрен далее). Если при выполнении метода loginMenu экземпляра класса Сhat пользователь был выбран корректно запускается метод showChat экземпляра класса Сhat (общий чат - будет рассмотрен далее).

Функция perCor – предназначена для реализации личной переписки между пользователями.

void perCor()

{

PerCor perCor;

User user (const string&, const string&, const string&);

perCor.loginMenuPC();

if (perCor.getCurrentUser())perCor.userMenuPC();

if (perCor.getParty() != "") perCor.showPC();

}

Сначала объявляется экземпляр класса PerCor, с именем perCor. Далее объявляется класс User с именем user. Далее запускается метод loginMenuPC экземпляра класса perCor (выбор пользователя - будет рассмотрен далее). Если при выполнении метода loginMenuPC экземпляра класса perCor пользователь был выбран корректно запускается метод userMenuPC экземпляра класса perCor (выбор собеседника - будет рассмотрен далее). Далее, если собеседник выбран, запускается метод showPC экземпляра класса perCor (личная переписка - будет рассмотрен далее).

Функция regulation предназначена для ознакомления пользователя с правилами пользования мессенджером. После ознакомления пользователь возвращается в главное меню.

void regulation()

{

system("cls");

cout << "\t\t--- Правила пользования программой ---\n1. Закрытые логины и имена при регистрации:\n\t(admin)\n\t(0)\n2. В сообщениях запрещено:\n";

cout << "\tОскорблять\n\tПризывать к насилию\n3. Возможные наказания за нарушения:\n\tБан по логину\n\tУдаление сообщения\n\n\n";

system("pause");

}

Файл Times.cpp содержит определение функции time, объявленной в заголовочном файле Times.h. Функция time служит для определения системного времени и возврата его значения в виде строки.

Файл User.h содержит определение класса User.

class User

{

string ban\_;

const string login\_;

string password\_;

const string name\_;

int num\_;

public:

User() {}

User(string ban, const string& login, const string& password, const string& name) :ban\_(ban), login\_(login), password\_(password), name\_(name){}

const string getBan() const { return ban\_; }

const string& getLogin() const { return login\_; }

const string& getPassword() const { return password\_; }

const string& getName() const { return name\_; }

void setPassword(string& password) { password\_ = password; }

void setBan(int ban) { ban? ban\_ = "false": ban\_ = "true"; }

void setNum(int num) { num\_ = num; }

};

Класс User предназначен для хранения и структурирования информации о пользователях.

Класс содержит переменные:

ban\_ - индикатор наличия бана на пользователе.

login\_ - логин пользователя

password\_ - пароль пользователя

name\_ - имя пользователя

num\_ - порядковый номер пользователя, необходим в некоторых методах других классов.

Также класс User содержит конструкторы, гетеры и сетеры необходимые для работы с классом.

Файл message.h содержит определение класса Message.

class Message

{

string del\_;

const string time\_;

const string from\_;

const string text\_;

public:

Message() { }

Message(string del, const string& times, const string& from, const string& text) :del\_(del), time\_(times), from\_(from), text\_(text) {}

string& getDelete() { return del\_; }

const string& getTime() const { return time\_; }

const string& getFrom() const { return from\_; }

const string& getText()const { return text\_; }

void setDelete(string& del) { del\_ = del; }

};

Класс Message предназначен для хранения и структурирования информации о сообщения для общего чата.

Класс содержит переменные:

del\_ - индикатор, удалено сообщение или нет;

time\_ - время написания сообщения;

from\_ - от кого поступило сообщение;

text\_ - содержание сообщения.

Также класс Message содержит конструкторы, гетеры и сетеры необходимые для работы с классом.

Файл messagePC.h содержит определение класса MessagePC.

class MessagePC

{

string del\_;

const string time\_;

const string from\_;

const string to\_;

const string text\_;

public:

MessagePC() {}

MessagePC(string del, const string& times, const string from, const string to, const string text) : del\_(del), time\_(times), from\_(from), to\_(to), text\_(text) {}

string& getDelete() { return del\_; }

const string& getTime() const { return time\_; }

const string& getFrom() const { return from\_; }

const string& getTo() const { return to\_; }

const string& getText()const { return text\_; }

void setDelete(string& deletes) { del\_ = deletes; }

};

Класс MessagePC предназначен для хранения и структурирования информации о сообщения для личной переписки.

Класс содержит переменные:

del\_ - индикатор, удалено сообщение или нет;

time\_ - время написания сообщения;

from\_ - от кого поступило сообщение;

to\_ - кому послано сообщение;

text\_ - содержание сообщения.

Также класс MessagePC содержит конструкторы, гетеры и сетеры необходимые для работы с классом.

Файл Chat.h содержит определение класса Chat.

class Chat {

string me\_;

vector<User> users\_;

vector<Message> message\_;

shared\_ptr<User>currentUser\_ = nullptr;

public:

shared\_ptr<User>getCurrentUser() const { return currentUser\_; }

void loginMenu();

void showChat();

private:

void getUsersFile();

void login();

void registration();

void setUsersFile(string, string, string);

void getMessage\_File();

void chat();

void addMessage(string txt);

void expCin();

shared\_ptr<User> getUserByLogin(const string& login)const;

shared\_ptr<User> getUserByName(const string& name)const;

};

Класс Chat - основной класс обеспечивающий общую переписку.

Класс содержит переменные:

me\_ - переменная определяющая пользователя;

users\_ - контейнер пользователей;

message\_ - контейнер с перепиской;

currentUser\_ - указатель на пользователя (меня) из общего контейнера пользователей.

Основные методы:

LoginMenu - вход и регистрацию пользователя;

showChat – переписка.

Второстепенные методы - метода LoginMenu:

getUsersFile - загрузка пользователей в контейнер из файла;

login - вход пользователя;

registration - регистрация нового пользователя;

setUsersFile (дополнительный метод метода registration) -внесение зарегистрированного пользователя в файл;

Второстепенные методы - метода showChat:

getMessage\_File - загрузка переписки из файла;

chat - вывод переписки;

addMessage - добавление сообщения в чат и сохранение его в контейнер переписки.

Вспомогательные методы:

getUserByLogin - указатель, проверяющий логин на предмет повторов;

getUserByName - указатель, проверяющий имя на предмет повторов;

expCin - обработка ошибок при использовании оператора cin.

Определение методов прописано в файле Chat.cpp.

Файл PerCor.h содержит определение класса PerCor.

class PerCor {

string me\_;

string party\_;

vector<User> users\_;

vector<string> perCorParty\_;

vector<MessagePC> messagePC\_;

shared\_ptr<User>currentUser\_ = nullptr;

shared\_ptr<User>\_party = nullptr;

public:

shared\_ptr<User>getCurrentUser() const { return currentUser\_; }

void loginMenuPC();

void userMenuPC();

void showPC();

void loginAdminPC();

auto getParty() { return party\_; }

private:

void getUsersFile();

void login();

void registration();

void setUsersFile(string, string, string);

void getMessagePC\_File();

void perCorParty();

void setParty(string party) { party\_ = party; }

void addParty();

bool expParty(string name);

void chat();

void addMessage(string txt);

void expCin();

shared\_ptr<User> getUserByLogin(const string& login) const;

shared\_ptr<User> getUserByName(const string& name)const;

};

Класс PerCor - основной класс обеспечивающий личную переписку.

*Класс PerCor содержит переменные:*

me\_ - переменная определяющая пользователя;

party\_ - переменная определяющая собеседника;

users\_ - контейнер пользователей;

perCorParty\_ - контейнер собеседников;

messagePC\_ - контейнер с личной перепиской;

currentUser\_ - указатель на пользователя (меня) из общего контейнера пользователей;

\_party – указатель на собеседника, для проверки наличия бана.

*Основные методы:*

loginMenuPC – обеспечивает вход и регистрацию пользователя;

userMenuPC – обеспечивает выбор собеседника;

showPC – обеспечивает переписку;

loginAdminPC – обеспечивает вход администратора, запускается из класса Admin.

Вспомогательные методы метода loginMenuPC:

getUsersFile – загрузка пользователей в контейнер из файла User.txt;

login - вход пользователя

registration - регистрация пользователя

setUsersFile (дополнительный метод метода registration) - внесение зарегистрированного пользователя в файл User.txt.

Вспомогательные методы метода userMenuPC:

getMessagePC\_File - загрузка личной переписки из файла MessagePC.txt, в контейнер личной переписки;

perCorParty - заполнение контейнера пользователей с которыми открыта личная переписка, на основе контейнера личной переписки;

setParty - выбор текущего собеседника;

addParty - добавление и выбор нового собеседника;

expParty (дополнительный метод метода addParty) - проверка выбранного пользователя на ошибки;

Вспомогательные методы метода showPC:

chat - вывод личной переписки с выбранным собеседником;

addMessage - добавление сообщения в чат и сохранения его в контейнер переписки.

Общий вспомогательный метод:

expCin - обработка ошибок при использовании оператора cin.

Определение методов прописано в файле PerCor.cpp.

Файл Admin.h содержит определение класса Admin.

class Admin

{

bool login\_{ false };

int numAdmin\_;

vector<User> users\_;

vector<Message> message\_;

shared\_ptr<User>currentAdmin\_ = nullptr;

public:

void loginMenu();

void adminMenu();

bool getLogin() { return login\_; }

private:

void inPassword();

void passChange();

bool setUsersFile();

void ban();

void showChat();

void chat();

void getMessage\_File();

void delMess();

void upDelMess();

bool setMessage\_File();

void addMessage(string txt);

void expCin();

void getUsersFile();

void setNumAdmin(int numAdmin) { numAdmin\_ = numAdmin; }

shared\_ptr<User> getAdminByLogin(const string& login);// const;

shared\_ptr<User> getAdminByName(const string& name)const;

};

Класс Admin - основной класс обеспечивающий администрирование чата.

Класс Admin содержит переменные:

login\_ - индикатор входа администратора;

numAdmin\_ - номер администратора в контейнере пользователей;

users\_ - контейнер пользователей;

message\_ - контейнер с перепиской;

currentAdmin\_ - указатель на администратора из общего контейнера пользователей.

Основные методы:

loginMenu – ввод пароля администратора и возможность его смены;

adminMenu – меню работы администратора;

Вспомоательные методы метода loginMenu:

inPassword – ввод и проверка пароля;

passChange – смена пароля;

setUsersFile – сохранение обновленного контейнера пользователей;

Вспомогательные методы метода adminMenu:

ban – наложение, снятие бана на пользователя;

showChat – работа с общим чатом;

Вспомогательные методы метода showChat:

chat – вывод переписки на экран;

getMessage\_File – загрузка переписки из файла Message.txt;

delMess – пометка выбранного сообщения удаленным;

upDelMess – снятие пометки удалено, с выбранного сообщения;

setMessage\_File – обновление файла сообщений Message.txt, в соответствии с контейнером сообщений;

addMessage – добавление нового сообщения.

Методы используемые всем классом:

expCin - обработка ошибок при использовании оператора cin;

getUsersFile – загрузка пользователей в контейнер из файла User.txt;

setNumAdmin – сохранение значения порядкового номера логина администратора в контейнере пользователей.