**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ**  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**  
**(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Расчетно-графическое задание

Дисциплина: Обьектно-ориентированное программирование

по теме «Система блогов»

Выполнил: ст. группы ВТ-22  
Воскобойников Илья Сергеевич

Проверил: Буханов Д.Г.

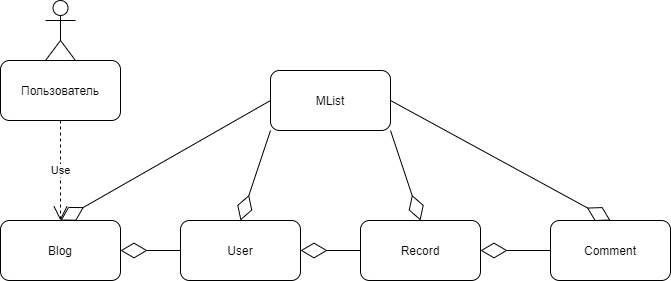
**Белгород 2020Цель работы**: приобретение практических навыков создания приложений на языке С++.

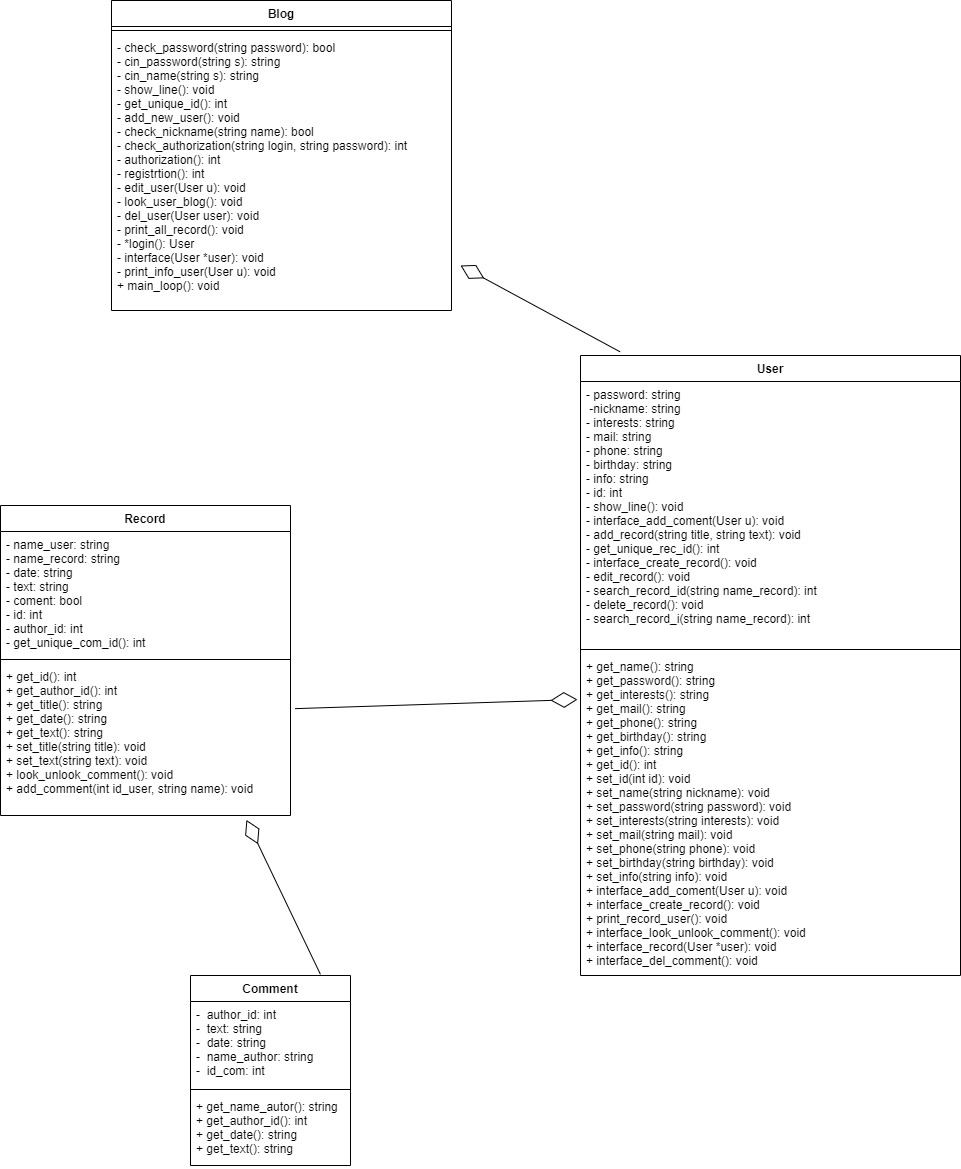
**Вариант задания 13**

Создать приложение реализующее систему блогов.

**Постановка задачи:**

Каждый блог принадлежит одному пользователю и состоит из нескольких записей. В момент создания блога в нем записей нет. Пользователь может создавать записи лишь в собственном блоге. Читать записи любого пользователя может каждый пользователь. В каждой записи есть заголовок, дата, текст. На каждой странице блога находятся ссылка на профиль. На начальной странице блога отображаются 10 последних записей (или менее, если в блоге их недостаточно). Если записей в блоге больше 10, то с начальной страницы можно перейти на вторую, где отображается второй десяток записей и т. д. Записи в блоге упорядочены по убыванию даты. Любая запись может быть отредактирована, но дата записи не может быть изменена. Запись может быть удалена автором. Перейдя на страницу отдельной записи, пользователь может ее комментировать, независимо от того, чужая это запись или его собственная. Комментарий состоит из текста и даты, указывающей, когда он написан. Комментарий может быть дан к записи блога или к ранее написанному комментарию. Пользователь может запретить комментирование любой записи в своем блоге. Комментарии, полученные раньше запрета, сохраняются. Запрет на комментирование может быть снят. Все комментарии к записи отображаются под ее текстом в виде дерева. Каждый комментарий помечен именем его автора. Имя может быть использовано для перехода к начальной странице блога с этим именем. Комментарии не могут быть отредактированы, но могут быть удалены автором записи. При удалении комментария также стираются все ответы на него, если таковые были. Любой желающий вести блог должен зарегистрироваться. При регистрации пользователь заполняет форму, указывая желаемое имя, пароль, список своих интересов, дату рождения, контакты, краткие сведения о себе. Если имя не занято, и форма заполнена полностью, система заводит блог нового пользователя. Сведения из регистрационной формы (кроме пароля) отображаются на странице профиля пользователя. Пользователь может отредактировать собственный профиль, изменив в нем все, кроме имени. Пользователь может изменить свой пароль. Пароль не может быть короче 8 символов. Зарегистрированный пользователь, осуществляет вход в собственный блог, указывая имя и пароль. Все описанные выше действия с записями, комментариями и собственным профилем, действия с друзьями доступны пользователю только после входа. Пользователь может полностью удалить свой блог со всеми записями, комментариями и профилем. После удаления блога регистрационные данные стираются из системы.





User.h

#pragma once  
  
#ifndef **RGZ\_USER\_H**#define **RGZ\_USER\_H**#include **<string>**#include **"Record.h"**#include **"listM.hpp"**#include **<iostream>**#include **"Record.h"  
using namespace** std;  
  
**class** User  
{  
 **int** id;  
 string nickname;  
 string password;  
 string interests;  
 string mail;  
 string phone;  
 string birthday;  
 string info;  
 **void** show\_line();  
 **void** interface\_add\_coment(User u);  
 **void** add\_record(string title, string text);  
 **int** get\_unique\_rec\_id();  
 **friend** std::ostream& **operator**<< (std::ostream &out, **const** User &user);  
 **void** interface\_create\_record();  
 **void** edit\_record();  
 **int** search\_record\_id(string name\_record);  
 **void** delete\_record();  
 **int** search\_record\_i(string name\_record);  
**public**:  
 User(**int** user\_id, string nickname, string password, string interests, string mail, string phone, string birthday, string info){  
 **this**->id = user\_id;  
 **this**->nickname = nickname;  
 **this**->password = password;  
 **this**->interests = interests;  
 **this**->mail = mail;  
 **this**->phone = phone;  
 **this**->birthday = birthday;  
 **this**->info = info;  
 };  
 MList<Record> records;  
 MList<Record> getRecords();  
 string get\_name();  
 string get\_password();  
 string get\_interests();  
 string get\_mail();  
 string get\_phone();  
 string get\_birthday();  
 string get\_info();  
 **int** get\_id();  
  
 **void** set\_id(**int** id);  
 **void** set\_name(string nickname);  
 **void** set\_password(string password);  
 **void** set\_interests(string interests);  
 **void** set\_mail(string mail);  
 **void** set\_phone(string phone);  
 **void** set\_birthday(string birthday);  
 **void** set\_info(string info);  
 **void** print\_record\_user();  
 **void** interface\_record(User \*user);  
 **void** interface\_look\_unlook\_comment();  
 **void** interface\_del\_comment();  
};  
  
  
#endif *//RGZ\_USER\_H*

User.cpp  
  
#include **"User.h"**string User:: get\_name() {**return** nickname;}  
string User::get\_password() {**return** password;}  
string User::get\_interests() {**return** interests;}  
string User::get\_mail() {**return** mail;}  
string User::get\_phone() {**return** phone;}  
string User::get\_birthday() {**return** birthday;}  
string User::get\_info() {**return** info;}  
**int** User::get\_id() {**return** id;}  
  
**void** User::set\_id(**int** id){ **this**->id=id;}  
**void** User::set\_name(string nickname){**this**->nickname = nickname;}  
**void** User::set\_password(string password){**this**->password = password;}  
**void** User::set\_mail(string mail){**this**->mail = mail;}  
**void** User::set\_interests(string interests){**this**->interests = interests;}  
**void** User::set\_phone(string phone){**this**->phone=phone;}  
**void** User::set\_info(string info) {**this**->info = info;}  
**void** User::set\_birthday(string birthday) {**this**->birthday = birthday;}  
  
**int** User::get\_unique\_rec\_id(){  
 **return** rand() ;  
};  
  
**void** User::add\_record(string title, string text) {  
 Record record=Record(get\_unique\_rec\_id(), get\_id(), title, text,get\_name());  
 records.push\_front(record);  
};  
  
**void** User::interface\_create\_record() {  
 string title,text;  
 cout <<**"Введите название записи"** <<endl;  
 cin >> title;  
 cout << **"Введите текст записи"**<<endl;  
 cin >>text;  
 add\_record(title,text);  
}  
  
**int** User::search\_record\_id(string name\_record){  
 **for** (**auto** record:records) {  
 **if**(record.get\_title()==name\_record){  
 **return** record.get\_id();  
 }  
 }  
 **return** -1;  
}  
  
**int** User::search\_record\_i(string name\_record) {  
 **int** i=0;  
 **for** (**auto** record:records) {  
 **if**(record.get\_title()!=name\_record){  
 i++;  
 }  
 }  
 **if**(records.size()<i){ **return** -1;}  
 **return** i;  
}  
  
**void** User::show\_line() {  
 cout<<**"====================================================================================================="**<<endl;  
}  
  
**void** User::edit\_record(){  
 string title,text;  
 cout <<**"Введите название записи которую хотите изменить"** <<endl;  
 cin >> title;  
 **int** i\_record=search\_record\_i(title);  
 **if**(i\_record==-1){  
 cout<<**"Запись не найдена"**<<endl;  
 **return**;  
 }  
 cout<<**"Введите новое имя записи"**<<endl;  
 cin>>title;  
 records[i\_record].set\_title(title);  
 cout << **"Введите текст записи"**<<endl;  
 cin >>text;  
 records[i\_record].set\_text(text);  
}  
  
**void** User:: delete\_record(){  
 string title;  
 cout<<**"Введеите название записи которую зотите удалить"**<<endl;  
 cin>>title;  
 **int** id\_record=search\_record\_id(title);  
 **if**(id\_record==-1){  
 cout<<**"Запись не найдена"**<<endl;  
 **return**;  
 }  
 **int** index=0;  
 **for** (**auto** record:records) {  
 **if**(record.get\_id()!=id\_record) index++;  
 **else break**;  
 }  
 **auto** it = records.begin();  
 advance(it,index);  
 records.erase(it);  
}  
  
std::ostream& **operator**<< (std::ostream &out, **const** User &user) {  
 out << **"Имя: "** << user.nickname << endl;  
 out << **"Интресы: "** << user.interests << endl;  
 out << **"Почта: "** << user.mail << endl;  
 out << **"Интeресы: "** << user.interests << endl;  
 out << **"День рождения: "** << user.birthday<< endl;  
 out << **"Информация о пользователе: "** << user.info << endl;  
 **return** out;  
}  
**void** User::interface\_look\_unlook\_comment(){  
 cout<< **"Введите имя записи в которой хотите изменить парамеры комментирования"**<<endl;  
 string title;  
 cin>>title;  
 **int** i\_record=search\_record\_i(title);  
 **if**(i\_record==-1){  
 cout<<**"Запись не найдена"**<<endl;  
 **return**;  
 }  
 records[i\_record].look\_unlook\_comment();  
}  
  
  
**void** User::print\_record\_user() {  
 **for** (**auto** record:records) {  
 cout<<record<<endl;  
 }  
}  
  
**void** User::interface\_add\_coment(User u){  
 string title;  
 cout << **"Введите название записи к которой хотите оставить комментарий"**<<endl;  
 cin >> title;  
 **int** i\_record=search\_record\_i(title);  
 **if**(i\_record==-1){  
 cout<<**"Запись не найдена"**<<endl;  
 **return**;  
 }  
 **if**(records[i\_record].get\_coment()){  
 records[i\_record].add\_comment(get\_id(),u.get\_name());  
 } **else**{  
 cout <<**"К этой записи нельзя оставить комментарий"**<<endl;  
 }  
}  
  
**void** User::interface\_del\_comment() {  
 string title, name, text;  
 cout << **"Введите имя записи в которой хотите удалить комментарий "** << endl;  
 cin >> title;  
 **int** q=0;  
 **int** i\_r = search\_record\_i(title);  
 **if**(i\_r==-1){  
 cout<<**"Запись не найдена"**<<endl;  
 **return**;  
 }  
 MList<Commentaire> \*commentss = records[i\_r].getCommends();  
 **for** (**auto** com:\*commentss) {  
 q++;  
 cout<<**"Комментарий:"**<< q<<endl<<com<<endl;  
 }  
 cout << **"Введите номер комментраий которы хотите удалить "** << endl;  
 cin >> q;  
 **auto** it = commentss->begin();  
 advance(it, q-1);  
 commentss->erase(it);  
}  
  
**void** User:: interface\_record(User \*user){  
 **int** v;  
 cout << endl<<**"Создать новую запись - 1"** << endl;  
 cout << **"Отредактировать запись - 2"** << endl;  
 cout << **"Удалить запись - 3"** << endl;  
 cout << **"Создать комментарий к записи - 4"** << endl;  
 cout<<**"Удалить комментарий к записи - 5"** << endl;  
 cout << **"Возможность оставлять комментарии записи-6 "** << endl;  
 cout<<**"Вернуться назад - 7"**<<endl<<endl;  
 cin>>v;  
 **switch** (v){  
 **case** 1:  
 show\_line();  
 interface\_create\_record();  
 **break**;  
 **case** 2:  
 show\_line();  
 edit\_record();  
 **break**;  
 **case** 3:  
 show\_line();  
 delete\_record();  
 **break**;  
 **case** 4:  
 show\_line();  
 interface\_add\_coment(\*user);  
 **break**;  
 **case** 5:  
 show\_line();  
 interface\_del\_comment();  
 **break**;  
 **case** 6:  
 show\_line();  
 interface\_look\_unlook\_comment();  
 **break**;  
 **case** 7:  
 show\_line();  
 **break**;  
 **default**:  
 show\_line();  
 interface\_record(user);  
 **break**;  
 }  
}  
MList<Record> User ::getRecords() {  
 **return** records;  
}

Record.h

#include **<time.h>**#include **<iostream>**#include **<string>**#include **"Commentaire.h"**#include **"listM.hpp"**#include **<time.h>  
  
  
using namespace** std;  
  
**class** Record{  
 **int** id;  
 **int** author\_id;  
 string title;  
 string text;  
 string date;  
 string name\_autor;  
 **bool** coment = **true**;  
 MList<Commentaire> comments;  
 **int** get\_unique\_com\_id();  
  
**public**:  
 Record(**int** rec\_id, **int** author\_id, string title, string text , string name\_autor){  
 **this**->id = rec\_id;  
 **this**->author\_id = author\_id;  
 **this**->title = title;  
 **this**->text = text;  
 time\_t seconds = time(**NULL**);  
 tm\* timeinfo = localtime(&seconds);  
 **this**->date = asctime(timeinfo);  
 **this**->name\_autor=name\_autor;  
 }  
 **int** get\_id();  
 **int** get\_author\_id();  
 string get\_name\_autor();  
 string get\_title();  
 string get\_date();  
 string get\_text();  
 **bool** get\_coment();  
 MList<Commentaire>\* getCommends();  
 **void** set\_title(string title);  
 **void** set\_text(string text);  
 **void** look\_unlook\_comment();  
 **void** add\_comment(**int** id\_user, string name);  
 **friend** std::ostream& **operator**<< (std::ostream &out, **const** Record &rec);  
};  
  
  
  
#endif *//RGZ\_RECORD\_H*

Record.cpp  
  
#include **"Record.h"**string Record::get\_title() {**return** title;}  
string Record::get\_date() {**return** date;}  
string Record::get\_text() {**return** text;}  
  
**void** Record::set\_title(string title) {**this**->title=title;}  
**void** Record::set\_text(string text) {**this**->text;}  
  
  
**void** Record::look\_unlook\_comment() {  
 coment = !coment;  
}  
  
**int** Record::get\_id(){ **return** id;}  
**int** Record::get\_author\_id() { **return** author\_id;}  
string Record::get\_name\_autor() { **return** name\_autor;}  
**bool** Record::get\_coment() { **return** coment;}  
  
std::ostream& **operator**<< (std::ostream &out, **const** Record &rec) {  
 out <<**" "**<< **"<"** << rec.title << **">"** << endl;  
 out << **" "** << rec.date;  
 out << **" "** << rec.name\_autor<<**":"**<<endl;  
 out << **" "** << rec.text << endl << endl;  
 out << **" Комментарии:"** << endl;  
 **for** (**auto** com: rec.comments) {  
 out << com;  
 }  
 **return** out;  
}  
  
MList<Commentaire> \*Record::getCommends(){  
 **return** &comments;  
}  
**int** Record::get\_unique\_com\_id(){  
 **return** rand() ;  
};  
**void** Record::add\_comment(**int** id\_user, string name\_user){  
 string text;  
 cout<<**"Введите коммментарий"**<<endl;  
 cin >> text;  
  
 Commentaire comment = Commentaire(id\_user, name\_user, text, get\_unique\_com\_id());;  
 comments.push\_front(comment);  
}

Commentaire.h

#ifndef **RGZ\_COMMENT\_H**#define **RGZ\_COMMENT\_H**#include **<string>**#include **<time.h>**#include **<stdio.h>  
  
using namespace** std;  
**class** Commentaire {  
 **int** author\_id;  
 string text;  
 string date;  
 string name\_author;  
 **int** id\_com;  
**public**:  
 Commentaire(**int** id, string name, string text, **int** id\_com){  
 **this**->author\_id = id;  
 **this**->text = text;  
 **this**->name\_author=name;  
 time\_t seconds = time(**NULL**);  
 tm\* timeinfo = localtime(&seconds);  
 **this**->date = asctime(timeinfo);  
 **this**->id\_com=id\_com;  
 };  
 string get\_name\_autor();  
 **int** get\_author\_id();  
 string get\_date();  
 string get\_text();  
  
 **friend** std::ostream& **operator**<< (std::ostream &out, **const** Commentaire &comm);  
};  
  
  
#endif *//RGZ\_COMMENT\_H*

Commentaire.cpp

#include **"Commentaire.h"**#include **<iostream>  
  
  
int** Commentaire::get\_author\_id() {  
 **return** author\_id;  
}  
string Commentaire::get\_date() {  
 **return** date;  
}  
string Commentaire::get\_text() {  
 **return** text;  
}  
  
string Commentaire::get\_name\_autor() {  
 **return** name\_author;  
}  
std::ostream& **operator**<< (std::ostream &out, **const** Commentaire &comm) {  
 out << **" "**<<comm.name\_author << **":"**<<**"\n"**<< **" "**<< comm.date << **" "** << comm.text << endl;  
 **return** out;  
}

Blog.h

#ifndef **RGZ\_DATABASE\_H**#define **RGZ\_DATABASE\_H**#include **"User.h"**#include **"listM.hpp"  
  
class** Blog{  
 **bool** check\_password(string password);  
 string cin\_password(string s);  
 string cin\_name(string s);  
 **void** show\_line();  
 **int** get\_unique\_id();  
 **void** add\_new\_user();  
 **bool** check\_nickname(string name);  
 **int** check\_authorization(string login, string password);  
 **int** authorization();  
 **int** registrtion();  
 **void** edit\_user(User u);  
 **void** look\_user\_blog();  
 **void** del\_user(User user);  
 **void** print\_all\_record();  
 User \*login();  
 **void** interface(User \*user);  
 **void** print\_info\_user(User u);  
**public**:  
 **static** Blog& getInstance() {  
 **static** Blog instance;  
 **return** instance;  
 }  
 MList<User> users;  
 Blog(){};  
 **void** main\_loop();  
  
};  
  
  
#endif *//RGZ\_DATABASE\_H*

Blog.cpp

**void** Blog::look\_user\_blog(){  
 string name;  
 cout << **"Введите именя пользователя чей блог вы хотите посмотреть"** << endl;  
 cin >> name;  
 **for** (**auto** user:users)  
 **if** (user.get\_name()==name) {  
 cout << user;  
 user.print\_record\_user();  
 **return**;  
 }  
 cout<<**" Пользователь не найден"**<<endl;  
}  
  
**bool** Blog::check\_password(string password) {  
 **return** (password.length() >= 8);  
}  
  
  
**void** Blog::print\_all\_record(){  
 MList<Record> record\_blog;  
 **if**(users.size()==0){  
 cout << **"Записей пока нет"**<<endl;  
 **return**;  
 }  
 **for** (**auto** user:users) {  
 MList<Record> ur=user.getRecords();  
 **for** (**int** i=0; i<ur.size();i++) {  
 record\_blog.push\_back(ur[i]);  
 }  
 }  
 record\_blog.sort([](Record &d1, Record &d2) { **return** d1.get\_date() > d2.get\_date(); });  
 **int** n=0;  
 **int** v=1;  
 **while** (v) {  
 **for** (**int** i = n; (i < n + 10) && (i < record\_blog.size()); i++) {  
 cout << record\_blog[i];  
 }  
 show\_line();  
 cout<<(record\_blog.size()<10);  
 **if**(record\_blog.size()<10) **return**;  
 cout << **"Перейти на следующую страницу - 1"** << endl;  
 cout << **"Перейти на предыдущую страницу -2 "** << endl;  
 cout << **"Выйти из просмотра архива записей - 0"** << endl;  
 cin >> v;  
 **switch** (v) {  
 **case** 1:  
 n+=10;  
 **break**;  
 **case** 2:  
 n-=10;  
 **break**;  
 **default**:  
 **return**;  
 }  
 }  
}  
  
  
**void** Blog::edit\_user(User u) {  
 **int** v;  
 string s;  
 cout << **"Изменить пароль пароль - 1"** << endl;  
 cout << **"Изменить интересы - 2"** << endl;  
 cout << **"Изменить почту - 3"** << endl;  
 cout << **"Изменить телефон - 4"** << endl;  
 cout << **"Изменить день рождения - 5"** << endl;  
 cout << **"Изменить информацию о себе - 6"** << endl << endl;  
 cin >> v;  
 **switch** (v) {  
 **case** 1:  
 cout << **"Введите пароль"**<<endl;  
 cin >> s;  
 s=cin\_password(s);  
 u.set\_password(s);  
 **break**;  
 **case** 2:  
 cout << **"Введите свои интересы"**<<endl;  
 cin>>s;  
 u.set\_interests(s);  
 **break**;  
 **case** 3:  
 cout << **"Введите почту"**<<endl;  
 cin >> s;  
 u.set\_mail(s);  
 **break**;  
 **case** 4:  
 cout << **"Введите телефон"**<<endl;  
 cin >>s;  
 u.set\_phone(s);  
 **break**;  
 **case** 5:  
 cout << **"Введите день рождения"**<<endl;  
 u.set\_birthday(s);  
 **break**;  
 **case** 6:  
 cout << **"Введите информацию о себе"**<<endl;  
 cin >> s;  
 u.set\_info(s);  
 **break**;  
 **default**:  
 edit\_user(u);  
 **break**;  
 };  
  
}  
**void** Blog::show\_line() {  
 cout<<**"====================================================================================================="**<<endl;  
}  
  
**bool** Blog::check\_nickname(string name) {  
 **for** (**auto** user:users)  
 **if**(user.get\_name()==name) **return false**;  
 **return true**;  
}  
  
  
string Blog::cin\_name(string s){  
 **while**(!check\_nickname(s)){  
 cout << **"Этот никнейм уже занят"**<<endl;  
 cout<<**"Введите никнейм"**<<endl;  
 cin>>s;  
 }  
 **return** s;  
}  
  
**void** Blog::add\_new\_user(){  
 string name,password,interests,mail,phone,birthday,info;  
 cout << **"Введите никнейм"**<<endl;  
 cin>>name;  
 name=cin\_name(name);  
 cout<<**"Введите пароль"**;  
 cin>>password;  
 cin\_password(password);  
 cout << **"Введите свои интересы"**<<endl;  
 cin>>interests;  
 cout << **"Введите почту"**<<endl;  
 cin>>mail;  
 cout << **"Введите телефон"**<<endl;  
 cin>>phone;  
 cout << **"Введите день рождения"**<<endl;  
 cin>>birthday;  
 cout << **"Введите информацию о себе"**<<endl;  
 cin>>info;  
  
 User user = User(get\_unique\_id(), name, password, interests, mail, phone, birthday, info);  
 users.push\_back(user);  
};  
  
**int** Blog::get\_unique\_id(){  
 **return** users.size() ;  
};  
  
  
**void** Blog::interface(User \*user) {  
 **int** v;  
 **while** (v!=7){  
 cout << endl<<**"Действия с записями - 1"**<<endl;  
 cout << **"Посмотреть свой блог - 2"** << endl;  
 cout << **"Посмотреть архив запесей - 3"** << endl;  
 cout << **"Отредактировать данные о себе - 4"** << endl;  
 cout << **"Посмотреть чужой блог - 5 "** << endl;  
 cout << **"Удалить свой блог - 6 "** <<endl;  
 cout << **"Выйти из аккаунта - 7"** << endl << endl;  
 cin >> v;  
 **switch** (v){  
 **case** 1:  
 show\_line();  
 user->interface\_record(user);  
 **break**;  
 **case** 2:  
 show\_line();  
 print\_info\_user(\*user);  
 user->print\_record\_user();  
 **break**;  
 **case** 3:  
 show\_line();  
 print\_all\_record();  
 **break**;  
 **case** 4:  
 show\_line();  
 edit\_user(\*user);  
 **break**;  
 **case** 5:  
 show\_line();  
 look\_user\_blog();  
 **break**;  
 **case** 6:  
 show\_line();  
 del\_user(\*user);  
 **return**;  
 **break**;  
 **case** 7:  
 show\_line();  
 **break**;  
 **default**:  
 show\_line();  
 interface(user);  
 }  
 }  
}  
**int** Blog:: authorization() {  
 string login;  
 string password;  
 cout << **"Введите свой логин"** << endl;  
 cin >> login;  
 cout << **"Введите пароль"** << endl;  
 cin >> password;  
 **int** id = check\_authorization(login, password);  
 **if** (id != -1) {  
 **return** id;  
 } **else** {  
 cout << **"Пользователь не найден"** << endl;  
 cout << **"Попробовать еще раз - 1 "** << endl;  
 cout << **"Зарегистрироваться - 2 "** << endl;  
 cout << **"Назад - 3 "** << endl;  
 **int** v;  
 cin >> v;  
 **switch** (v) {  
 **case** 1:  
 **return** authorization();  
 **case** 2:  
 **return** registrtion();  
 **case** 3:  
 **break**;  
 **default**:  
 **return** authorization();  
 };  
 }  
 **return** 0;  
}  
  
User\* Blog:: login() {  
 **int** v;  
 **int** id;  
 cout << **"Войти - 1"** << endl;  
 cout << **"Зарегистрироваться - 2 "** << endl;  
 cin >> v;  
 **switch** (v)  
 {  
 **case** 1:  
 show\_line();  
 id=authorization();  
 **break**;  
 **case** 2:  
 show\_line();  
 id=registrtion();  
 **break**;  
 **default**:  
 show\_line();  
 **break**;  
  
 }  
 **return** &users[id];  
}  
  
**int** Blog::check\_authorization(string login, string password){  
 **for** (**auto** user:users)  
 **if** (login == user.get\_name() && password == user.get\_password())  
 **return** user.get\_id();  
 **return** -1;  
}  
  
**int** Blog::registrtion(){  
  
 string name, password, interest, mail,phone, birhday,info;  
 cout << **"Введите никнейм"**<<endl;  
 cin>>name;  
 name=cin\_name(name);  
 cout << **"Введите пароль"**<<endl;  
 cin >>password;  
 password=cin\_password(password);  
 cout << **"Введите свои интересы"**<<endl;  
 cin>>interest;  
 cout << **"Введите почту"**<<endl;  
 cin >> mail;  
 cout << **"Введите телефон"**<<endl;  
 cin>> phone;  
 cout << **"Введите день рождения"**<<endl;  
 cin>>birhday;  
 cout << **"Введите день информацию о себе"**<<endl;  
 cin>>info;  
 User user(get\_unique\_id(),name,password,interest,mail,phone,birhday,info);  
 users.push\_back(user);  
 **return** user.get\_id();  
}  
  
  
  
**void** Blog::main\_loop(){  
 **int** v=1;  
 **while** (v){  
 User \*user=login();  
 show\_line();  
 interface(user);  
 cout<<**"Продолжить работу?"**<<endl;  
 cout<<**"ДА - 1"**<<endl;  
 cout<<**"НЕТ - 0 "**<<endl;  
 cin>> v ;  
 }  
}

listM.hpp

#pragma once  
#ifndef **UPDATE\_H**#define **UPDATE\_H**#include **<list>**#include **<iterator>**#include **<stdio.h>**#include **<iostream>  
using namespace** std;  
  
**template** <**class** T>  
**class** MList : **public** list<T> {  
**public**:  
 T& **operator**[](**int** index) {  
 **if** (**this**->size() <= index) **throw** out\_of\_range(**"Out of range!"**);  
 **typename** MList<T>::iterator it;  
 it = **this**->begin();  
 advance(it, index);  
 **return** \*it;  
 }  
};  
#endif

Main.cpp

#include **<iostream>**#include **<string>**#include **<time.h>**#include **"Blog.h"  
using namespace** std;  
  
  
**int** main() {  
  
 Blog blog;  
 blog.getInstance();  
 blog.main\_loop();  
  
}

