**《面向对象程序课程设计》课程设计报告**

班级：7班 学号：55180713 姓名：肖子健

1. **设计任务分析**

设计微X功能，实现开通，登录，好友管理，群管理，基本信息管理等功能。技术层次达到支持对象层次，抽象封装层次，面向对象层次，优化提高层次

1. **设计方案**

采用句柄类，减小文件之间的联系。并且分成三类文件

第一类是数据文件(BasicData, QQData, WCData, WBData, QQ, WC, WB, group)并且接口都是数据成员，没有操作。

第二类是管理文件(userMana, creatAcc, creatAccImpl, log, logImpl, goupMana, goupManaImpl, friendMana, friendManaImpl)其中除userMana之外都含有一个句柄类

第三类是文件管理文件(Files)

第四类是连接类(Connect)

1. **详细设计**

**BasicDataXZJ.hpp**

#ifndef BasicDataXZJ\_hpp

#define BasicDataXZJ\_hpp

#include <iostream>

#include <vector>//一定要包含

struct Day

{

Day(){}

explicit Day(long d):val(d){}

long val;

};

struct Month

{

Month(){}

explicit Month(long m):val(m){}

long val;

};

struct Year

{

Year(){}

explicit Year(long y):val(y){}

long val;

};

class Time

{

public:

Time(){}

explicit Time(const Year& y, const Month& m, const Day& d):\_y(y),\_m(m),\_d(d){}

void show()const{std::cout<<\_y.val<<"年"<<\_m.val<<"月"<<\_d.val<<"日";}

void year(long y){\_y.val=y;}

void month(long m){\_m.val=m;}

void day(long d){\_d.val=d;}

const long year()const{return \_y.val;}

const long month()const{return \_m.val;}

const long day()const{return \_d.val;}

bool operator==(const Time& time)const;

bool operator!=(const Time& time)const;

private:

Year \_y;

Month \_m;

Day \_d;

};

std::ostream& operator<<(std::ostream& os, const Time& t);

**class** BasicDataXZJ

{

public:

BasicDataXZJ(){}

explicit BasicDataXZJ(const long id, const long password,

const std::string& name, const std::string& home,

const Time& birth, const Time& t\_age)

:\_password(password),\_ID(id),\_name(name),\_home(home),\_birth(birth),\_t\_age(t\_age){}

virtual ~BasicDataXZJ(){}

const long ID()const{return \_ID;}

long ID(){return \_ID;}

const long password()const{return \_password;}

const std::string& name()const{return \_name;}

const std::string& home()const{return \_home;}

const Time& birth()const{return \_birth;}

const Time& t\_age()const{return \_t\_age;}

std::vector<long>& groNum(){return \_groNum;}

std::vector<long>& tempgroNum(){return \_tempgroNum;}

long& changepassword(){return \_password;}

const bool operator==(const BasicDataXZJ& qd);

const bool operator!=(const BasicDataXZJ& qd);

const bool operator==(BasicDataXZJ& qd)const;

const bool operator!=(BasicDataXZJ& qd)const;

const bool operator==(const BasicDataXZJ& qd)const;

const bool operator!=(const BasicDataXZJ& qd)const;

const bool operator==(BasicDataXZJ& qd);

const bool operator!=(BasicDataXZJ& qd);

private:

long \_ID;

long \_password;

std::string \_name;

std::string \_home;

std::vector<long> \_groNum;

std::vector<long> \_tempgroNum;

Time \_birth;

Time \_t\_age;

};

#endif

**BasicDataXZJ.cpp**

#include "BasicDataXZJ.hpp"

bool Time::operator==(const Time& time)const

{

if(this->\_y.val!=time.\_y.val||this->\_m.val!=time.\_m.val||this->\_d.val!=time.\_d.val)

return 0;

return 1;

}

bool Time::operator!=(const Time& time)const

{

return !(\*this==time);

}

std::ostream& operator<<(std::ostream& os, const Time& t)

{

os<<t.year()<<"年"<<t.month()<<"月"<<t.day()<<"日";

return os;

}

const bool BasicDataXZJ::operator==(const BasicDataXZJ& qd)

{

return static\_cast<const BasicDataXZJ&>(\*this)==qd;

}

const bool BasicDataXZJ::operator!=(const BasicDataXZJ& qd)

{

return !(\*this==qd);

}

const bool BasicDataXZJ::operator==(BasicDataXZJ& qd)

{

return \*this==static\_cast<const BasicDataXZJ&>(qd);

}

const bool BasicDataXZJ::operator!=(BasicDataXZJ& qd)

{

return !(\*this==qd);

}

const bool BasicDataXZJ::operator==(BasicDataXZJ& qd)const

{

return \*this==static\_cast<const BasicDataXZJ&>(qd);

}

const bool BasicDataXZJ::operator!=(BasicDataXZJ& qd)const

{

return !(\*this==qd);

}

const bool BasicDataXZJ::operator==(const BasicDataXZJ& qd)const

{

if(this->ID()!=qd.ID()&&this->home()!=qd.home()&&this->name()!=qd.name()&&this->t\_age()==qd.t\_age()&&this->birth()==qd.birth())

return 0;

return 1;

}

const bool BasicDataXZJ::operator!=(const BasicDataXZJ& qd)const

{

return !(\*this==qd);

}

**QQDataXZJ.hpp**

#ifndef QQDataXZJ\_hpp

#define QQDataXZJ\_hpp

#include <iostream>

#include "BasicDataXZJ.hpp"

class QQDataXZJ:public BasicDataXZJ

{

public:

QQDataXZJ():BasicDataXZJ(){}

explicit QQDataXZJ(const std::string& name, const std::string& home,

const Time& birth, const Time& t\_age,

long id, long password, long wc=0, long wb=0, long qqNum=0)

:BasicDataXZJ(id,password,name,home,birth,t\_age),\_wc(wc),\_wb(wb),\_qqNum(qqNum){}

explicit QQDataXZJ(std::\_\_wrap\_iter<const std::shared\_ptr<QQDataXZJ> \*> i):

BasicDataXZJ((\*i)->ID(),

(\*i)->password(),

(\*i)->name(),

(\*i)->home(),

(\*i)->birth(),

(\*i)->t\_age()),\_wc((\*i)->wc()),\_wb((\*i)->wb()),\_qqNum((\*i)->qqNum()){}

explicit QQDataXZJ(const std::shared\_ptr<QQDataXZJ>& i):

BasicDataXZJ(i->ID(),

i->password(),

i->name(),

i->home(),

i->birth(),

i->t\_age()),\_wc(i->wc()),\_wb(i->wb()),\_qqNum(i->qqNum()){}

explicit QQDataXZJ(const QQDataXZJ& qd)

:BasicDataXZJ(qd.ID(),qd.password(),qd.name(),qd.home(),qd.birth(),qd.t\_age()),\_wc(qd.wc()),\_wb(qd.wb()),\_qqNum(qd.qqNum()){}

explicit QQDataXZJ(const BasicDataXZJ& qd)

:BasicDataXZJ(qd.ID(),qd.password(),qd.name(),qd.home(),qd.birth(),qd.t\_age()),\_wc(0),\_wb(0),\_qqNum(0){}

virtual ~QQDataXZJ(){}

long& qqNum(){return \_qqNum;}

long& wc(){return \_wc;}

long& wb(){return \_wb;}

const long qqNum()const{return \_qqNum;}

const long wc()const{return \_wc;}

const long wb()const{return \_wb;}

std::vector<std::shared\_ptr<QQDataXZJ>>& friendList(){return \_friendList;}

std::vector<std::shared\_ptr<QQDataXZJ>>& askfriendList(){return \_askfriendList;}

std::vector<long>& friendlistnum(){return \_friendlistnum;}

std::vector<long>& askfriendlistnum(){return \_askfriendlistnum;}

const bool operator==(const QQDataXZJ& qd);

const bool operator!=(const QQDataXZJ& qd);

const bool operator==(QQDataXZJ& qd)const;

const bool operator!=(QQDataXZJ& qd)const;

const bool operator==(const QQDataXZJ& qd)const;

const bool operator!=(const QQDataXZJ& qd)const;

const bool operator==(QQDataXZJ& qd);

const bool operator!=(QQDataXZJ& qd);

private:

long \_qqNum=0;

long \_wc=0;

long \_wb=0;

std::vector<std::shared\_ptr<QQDataXZJ>> \_friendList;

std::vector<long> \_friendlistnum;

std::vector<std::shared\_ptr<QQDataXZJ>> \_askfriendList;

std::vector<long> \_askfriendlistnum;

};

#endif

**QQDataXZJ.cpp**

#include "QQDataXZJ.hpp"

const bool QQDataXZJ::operator==(const QQDataXZJ& qd)

{

return static\_cast<const QQDataXZJ&>(\*this)==qd;

}

const bool QQDataXZJ::operator!=(const QQDataXZJ& qd)

{

return !(\*this==qd);

}

const bool QQDataXZJ::operator==(QQDataXZJ& qd)

{

return \*this==static\_cast<const QQDataXZJ&>(qd);

}

const bool QQDataXZJ::operator!=(QQDataXZJ& qd)

{

return !(\*this==qd);

}

const bool QQDataXZJ::operator==(QQDataXZJ& qd)const

{

return \*this==static\_cast<const QQDataXZJ&>(qd);

}

const bool QQDataXZJ::operator!=(QQDataXZJ& qd)const

{

return !(\*this==qd);

}

const bool QQDataXZJ::operator==(const QQDataXZJ& qd)const

{

if(this->ID()!=qd.ID()&&this->home()!=qd.home()&&

this->name()!=qd.name()&&this->t\_age()==qd.t\_age()&&

this->birth()==qd.birth()&&\_qqNum==qd.qqNum()&&\_wc==qd.wc()&&\_wb==qd.wb())

return 0;

return 1;

}

const bool QQDataXZJ::operator!=(const QQDataXZJ& qd)const

{

return !(\*this==qd);

}

**WBDataXZJ.hpp**

#ifndef WBDataXZJ\_hpp

#define WBDataXZJ\_hpp

#include <iostream>

#include "BasicDataXZJ.hpp"

class WBDataXZJ:public BasicDataXZJ

{

public:

WBDataXZJ():BasicDataXZJ(){}

explicit WBDataXZJ(const std::string& name, const std::string& home,

const Time& birth, const Time& t\_age,

long id, long password, long qq=0, long wc=0, long wbNum=0)

:BasicDataXZJ(id,password,name,home,birth,t\_age),\_qq(qq),\_wc(wc),\_wbNum(wbNum){}

explicit WBDataXZJ(std::\_\_wrap\_iter<const std::shared\_ptr<WBDataXZJ> \*> i):

BasicDataXZJ((\*i)->ID(),

(\*i)->password(),

(\*i)->name(),

(\*i)->home(),

(\*i)->birth(),

(\*i)->t\_age()),\_qq((\*i)->qq()),\_wc((\*i)->wc()),\_wbNum((\*i)->wbNum()){}

explicit WBDataXZJ(const std::shared\_ptr<WBDataXZJ>& i):

BasicDataXZJ(i->ID(),

i->password(),

i->name(),

i->home(),

i->birth(),

i->t\_age()),\_qq(i->qq()),\_wc(i->wc()),\_wbNum(i->wbNum()){}

explicit WBDataXZJ(const WBDataXZJ& wb)

:BasicDataXZJ(wb.ID(),wb.password(),wb.name(),wb.home(),wb.birth(),wb.t\_age()),\_qq(wb.qq()),\_wc(wb.wc()),\_wbNum(wb.wbNum()){}

explicit WBDataXZJ(const BasicDataXZJ& wb)

:BasicDataXZJ(wb.ID(),wb.password(),wb.name(),wb.home(),wb.birth(),wb.t\_age()),\_qq(0),\_wc(0),\_wbNum(0){}

virtual ~WBDataXZJ(){}

long& qq(){return \_qq;}

long& wc(){return \_wc;}

long& wbNum(){return \_wbNum;}

const long qq()const{return \_qq;}

const long wc()const{return \_wc;}

const long wbNum()const{return \_wbNum;}

std::vector<std::shared\_ptr<WBDataXZJ>>& friendList(){return \_friendList;}

std::vector<std::shared\_ptr<WBDataXZJ>>& askfriendList(){return \_askfriendList;}

std::vector<long>& friendlistnum(){return \_friendlistnum;}

std::vector<long>& askfriendlistnum(){return \_askfriendlistnum;}

const bool operator==(const WBDataXZJ& qd);

const bool operator!=(const WBDataXZJ& qd);

const bool operator==(WBDataXZJ& qd)const;

const bool operator!=(WBDataXZJ& qd)const;

const bool operator==(const WBDataXZJ& qd)const;

const bool operator!=(const WBDataXZJ& qd)const;

const bool operator==(WBDataXZJ& qd);

const bool operator!=(WBDataXZJ& qd);

private:

long \_wbNum=0;

long \_qq=0;

long \_wc=0;

std::vector<std::shared\_ptr<WBDataXZJ>> \_friendList;

std::vector<long> \_friendlistnum;

std::vector<std::shared\_ptr<WBDataXZJ>> \_askfriendList;

std::vector<long> \_askfriendlistnum;

};

#endif

**WBDataXZJ.cpp**

#include "WBDataXZJ.hpp"

const bool WBDataXZJ::operator==(const WBDataXZJ& qd)

{

return static\_cast<const WBDataXZJ&>(\*this)==qd;

}

const bool WBDataXZJ::operator!=(const WBDataXZJ& qd)

{

return !(\*this==qd);

}

const bool WBDataXZJ::operator==(WBDataXZJ& qd)

{

return \*this==static\_cast<const WBDataXZJ&>(qd);

}

const bool WBDataXZJ::operator!=(WBDataXZJ& qd)

{

return !(\*this==qd);

}

const bool WBDataXZJ::operator==(WBDataXZJ& qd)const

{

return \*this==static\_cast<const WBDataXZJ&>(qd);

}

const bool WBDataXZJ::operator!=(WBDataXZJ& qd)const

{

return !(\*this==qd);

}

const bool WBDataXZJ::operator==(const WBDataXZJ& qd)const

{

if(this->ID()!=qd.ID()&&this->home()!=qd.home()&&

this->name()!=qd.name()&&this->t\_age()==qd.t\_age()&&

this->birth()==qd.birth()&&\_wbNum==qd.wbNum()&&\_qq==qd.qq()&&\_wc==qd.wc())

return 0;

return 1;

}

const bool WBDataXZJ::operator!=(const WBDataXZJ& qd)const

{

return !(\*this==qd);

}

**WCDataXZJ.hpp**

#ifndef WCDataXZJ\_hpp

#define WCDataXZJ\_hpp

#include <iostream>

#include "BasicDataXZJ.hpp"

class WCDataXZJ:public BasicDataXZJ

{

public:

WCDataXZJ():BasicDataXZJ(){}

explicit WCDataXZJ(const std::string& name, const std::string& home,

const Time& birth, const Time& t\_age,

long id, long password, long qq=0, long wb=0)

:BasicDataXZJ(id,password,name,home,birth,t\_age),\_qq(qq),\_wb(wb){}

explicit WCDataXZJ(std::\_\_wrap\_iter<const std::shared\_ptr<WCDataXZJ> \*> i):

BasicDataXZJ((\*i)->ID(),

(\*i)->password(),

(\*i)->name(),

(\*i)->home(),

(\*i)->birth(),

(\*i)->t\_age()),\_qq((\*i)->qq()),\_wb((\*i)->wb()){}

explicit WCDataXZJ(const std::shared\_ptr<WCDataXZJ>& i):

BasicDataXZJ(i->ID(),

i->password(),

i->name(),

i->home(),

i->birth(),

i->t\_age()),\_qq(i->qq()),\_wb(i->wb()){}

explicit WCDataXZJ(const WCDataXZJ& wc)

:BasicDataXZJ(wc.ID(),wc.password(),wc.name(),wc.home(),wc.birth(),wc.t\_age()),\_qq(wc.qq()),\_wb(wc.wb()){}

explicit WCDataXZJ(const BasicDataXZJ& wc)

:BasicDataXZJ(wc.ID(),wc.password(),wc.name(),wc.home(),wc.birth(),wc.t\_age()),\_qq(0),\_wb(0){}

virtual ~WCDataXZJ(){}

long& qq(){return \_qq;}

long& wb(){return \_wb;}

const long qq()const{return \_qq;}

const long wb()const{return \_wb;}

std::vector<std::shared\_ptr<WCDataXZJ>>& friendList(){return \_friendList;}

std::vector<std::shared\_ptr<WCDataXZJ>>& askfriendList(){return \_askfriendList;}

std::vector<long>& friendlistnum(){return \_friendlistnum;}

std::vector<long>& askfriendlistnum(){return \_askfriendlistnum;}

const bool operator==(const WCDataXZJ& qd);

const bool operator!=(const WCDataXZJ& qd);

const bool operator==(WCDataXZJ& qd)const;

const bool operator!=(WCDataXZJ& qd)const;

const bool operator==(const WCDataXZJ& qd)const;

const bool operator!=(const WCDataXZJ& qd)const;

const bool operator==(WCDataXZJ& qd);

const bool operator!=(WCDataXZJ& qd);

private:

long \_qq=0;

long \_wb=0;

std::vector<std::shared\_ptr<WCDataXZJ>> \_friendList;

std::vector<long> \_friendlistnum;

std::vector<std::shared\_ptr<WCDataXZJ>> \_askfriendList;

std::vector<long> \_askfriendlistnum;

};

#endif

**WCDataXZJ.cpp**

#include "WCDataXZJ.hpp"

const bool WCDataXZJ::operator==(const WCDataXZJ& qd)

{

return static\_cast<const WCDataXZJ&>(\*this)==qd;

}

const bool WCDataXZJ::operator!=(const WCDataXZJ& qd)

{

return !(\*this==qd);

}

const bool WCDataXZJ::operator==(WCDataXZJ& qd)

{

return \*this==static\_cast<const WCDataXZJ&>(qd);

}

const bool WCDataXZJ::operator!=(WCDataXZJ& qd)

{

return !(\*this==qd);

}

const bool WCDataXZJ::operator==(WCDataXZJ& qd)const

{

return \*this==static\_cast<const WCDataXZJ&>(qd);

}

const bool WCDataXZJ::operator!=(WCDataXZJ& qd)const

{

return !(\*this==qd);

}

const bool WCDataXZJ::operator==(const WCDataXZJ& qd)const

{

if(this->ID()!=qd.ID()&&this->home()!=qd.home()&&

this->name()!=qd.name()&&this->t\_age()==qd.t\_age()&&

this->birth()==qd.birth()&&\_qq==qd.qq()&&\_wb==qd.wb())

return 0;

return 1;

}

const bool WCDataXZJ::operator!=(const WCDataXZJ& qd)const

{

return !(\*this==qd);

}

**ConnectXZJ.hpp**

#ifndef ConnectXZJ\_hpp

#define ConnectXZJ\_hpp

#include <iostream>

#include "Files.hpp"//确保ID或者号码存在

enum method{ID, number};

class Connect//微博ID连接QQID，微信ID连接QQ号码，微信ID连接微博ID

{

public:

Connect():

pF(std::make\_shared<Files>()),

pQ(std::make\_shared<QQXZJ>(pF->QQ())),

pWC(std::make\_shared<WCXZJ>(pF->WC())),

pWB(std::make\_shared<WBXZJ>(pF->WB())){}

~Connect(){}

friend long ConnectQQ(const method me, const long digital);//连接的种类和相应的号码

friend long ConnectWC(const method me, const long digital);

friend long ConnectWB(const method me, const long digital);

private:

const bool searchQQID(const long id);

const bool searchWBID(const long id);

const bool searchWCID(const long id);

const bool searchQQNum(const long num);

const bool searchWBNum(const long num);

std::shared\_ptr<Files> pF=nullptr;

std::shared\_ptr<QQXZJ> pQ=nullptr;

std::shared\_ptr<WCXZJ> pWC=nullptr;

std::shared\_ptr<WBXZJ> pWB=nullptr;

};

long ConnectQQ(const method me, const long digital);

long ConnectWC(const method me, const long digital);

long ConnectWB(const method me, const long digital);

#endif

**ConnectXZJ.cpp**

#include "ConnectXZJ.hpp"

long ConnectQQ(const method me, long digital)

{

std::shared\_ptr<Connect> con=std::make\_shared<Connect>();

std::string mee("ID");

do{

switch (me) {

case ID:

while(!con->searchQQID(digital))

{

std::cout<<"没有这个QQID，重输:";

std::cin>>digital;

}

return digital;

break;

case number:

while(!con->searchQQNum(digital))

{

std::cout<<"没有这个QQ号，重输:";

std::cin>>digital;

}

return digital;

default:

std::cout<<"只能选择输入ID或者number"<<std::endl

<<"重输:";

std::cin>>mee;

break;

}

}while(mee!="ID"||mee!="number");

return 0;

}

long ConnectWC(const method me, long digital)

{

std::shared\_ptr<Connect> con=std::make\_shared<Connect>();

std::string mee("ID");

do{

switch (me) {

case ID:

while(!con->searchWCID(digital))

{

std::cout<<"没有这个微信ID，重输:";

std::cin>>digital;

}

return digital;

break;

case number:

default:

std::cout<<"只能选择输入ID"<<std::endl

<<"重输:";

std::cin>>mee;

break;

}

}while(mee!="ID");

return 0;

}

long ConnectWB(const method me, long digital)

{

std::shared\_ptr<Connect> con=std::make\_shared<Connect>();

std::string mee("ID");

do{

switch (me) {

case ID:

while(!con->searchWBID(digital))

{

std::cout<<"没有这个微博ID，重输:";

std::cin>>digital;

}

return digital;

break;

case number:

while(!con->searchWBNum(digital))

{

std::cout<<"没有这个微博号，重输:";

std::cin>>digital;

}

return digital;

default:

std::cout<<"只能选择输入ID或者number"<<std::endl

<<"重输:";

std::cin>>mee;

break;

}

}while(mee!="ID"||mee!="number");

return 0;

}

const bool Connect::searchQQID(const long id)

{

for(auto each=(pQ->people())->cbegin(); each!=(pQ->people())->cend(); ++each)

if((each->get())->ID()==id)

return 1;

return 0;

}

const bool Connect::searchWBID(const long id)

{

for(auto each=(pWB->people())->cbegin(); each!=(pWB->people())->cend(); ++each)

if((each->get())->ID()==id)

return 1;

return 0;

}

const bool Connect::searchWCID(const long id)

{

for(auto each: \*pWB->people())

if(each->ID()==id)

return 1;

return 0;

}

const bool Connect::searchQQNum(const long num)

{

for(auto each: \*pQ->people())

if(each->qqNum()==num)

return 1;

return 0;

}

const bool Connect::searchWBNum(const long num)

{

for(auto each: \*pWB->people())

if(each->wbNum()==num)

return 1;

return 0;

}

**groupXZJ.hpp**

#ifndef groupXZJ\_hpp

#define groupXZJ\_hpp

#include <iostream>

#include <vector>

#include "QQDataXZJ.hpp"

#include "WBDataXZJ.hpp"

#include "WCDataXZJ.hpp"

class groupXZJ

{

public:

groupXZJ():

\_tempqqGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_tempwcGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_tempwbGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_qqGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_wcGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_wbGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_askJoinQQGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_askJoinWCGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_askJoinWBGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_askJointempQQGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_askJointempWCGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_askJointempWBGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_qqGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_wcGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_wbGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()){}

explicit groupXZJ(long i):

\_groupNum(i),

\_tempqqGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_tempwcGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_tempwbGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_qqGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_wcGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_wbGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_askJoinQQGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_askJoinWCGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_askJoinWBGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_askJointempQQGro(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_askJointempWCGro(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_askJointempWBGro(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_qqGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_wcGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_wbGroAdministrators(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()){}

const long groupNum()const{return \_groupNum;}

long& groupNum(){return \_groupNum;}

const std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& qqGro()const{return \_qqGro;}

const std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& wbGro()const{return \_wbGro;}

const std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& wcGro()const{return \_wcGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& qqGro(){return \_qqGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& wbGro(){return \_wbGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& wcGro(){return \_wcGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& tempqqGro(){return \_tempqqGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& tempwbGro(){return \_tempwbGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& tempwcGro(){return \_tempwcGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& askJoinQQGroList(){return \_askJoinQQGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& askJoinWBGroList(){return \_askJoinWBGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& askJoinWCGroList(){return \_askJoinWCGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& askJointempQQGroList(){return \_askJointempQQGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& askJointempWBGroList(){return \_askJointempWBGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& askJointempWCGroList(){return \_askJointempWCGro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& qqGroAdministrators(){return \_qqGroAdministrators;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& wbGroAdministrators(){return \_wbGroAdministrators;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& wcGroAdministrators(){return \_wcGroAdministrators;}

~groupXZJ(){}

private:

long \_groupNum;

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_tempqqGro;

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_qqGro;

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_askJoinQQGro;

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_askJointempQQGro;

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_qqGroAdministrators;

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_tempwbGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_wbGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_askJoinWBGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_askJointempWBGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_wbGroAdministrators;

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_tempwcGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_wcGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_askJoinWCGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_askJointempWCGro;

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_wcGroAdministrators;

};

#endif

**QQXZJ.hpp**

#ifndef QQXZJ\_hpp

#define QQXZJ\_hpp

#include <iostream>

#include <vector>

#include "QQDataXZJ.hpp"

#include "groupXZJ.hpp"

class QQXZJ

{

public:

QQXZJ():

\_people(std::make\_shared<std::vector<std::shared\_ptr<QQDataXZJ>>>()),

\_tempdisgro(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>()),

\_groupMana(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>())

{

}

explicit QQXZJ(std::shared\_ptr<QQXZJ> pQ):

\_people(pQ->people()),

\_groupMana(pQ->groupMana()),

\_tempdisgro(pQ->tempdisgro()){}

explicit QQXZJ(const QQXZJ& pQ):

\_people(pQ.people()),

\_groupMana(pQ.groupMana()),

\_tempdisgro(pQ.tempdisgro()){}

const std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& people()const{return \_people;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& groupMana()const{return \_groupMana;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro()const{return \_tempdisgro;}

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& people(){return \_people;}

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro(){return \_tempdisgro;}

~QQXZJ(){}

private:

std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>> \_people=nullptr;

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_tempdisgro=nullptr;//临时讨论组

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_groupMana=nullptr;//初始化时别忘记设置群号

};

#endif

**WBXZJ.hpp**

#ifndef WBXZJ\_hpp

#define WBXZJ\_hpp

#include <iostream>

#include <vector>

#include "WBDataXZJ.hpp"

#include "groupXZJ.hpp"

class WBXZJ

{

public:

WBXZJ():

\_people(std::make\_shared<std::vector<std::shared\_ptr<WBDataXZJ>>>()),

\_tempdisgro(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>()),

\_groupMana(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>())

{

}

explicit WBXZJ(std::shared\_ptr<WBXZJ> pWB):

\_people(pWB->people()),

\_groupMana(pWB->groupMana()),

\_tempdisgro(pWB->tempdisgro()){}

explicit WBXZJ(const WBXZJ& pWB):

\_people(pWB.people()),

\_groupMana(pWB.groupMana()),

\_tempdisgro(pWB.tempdisgro()){}

const std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& people()const{return \_people;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& groupMana()const{return \_groupMana;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro()const{return \_tempdisgro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& people(){return \_people;}

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro(){return \_tempdisgro;}

~WBXZJ(){}

private:

std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>> \_people=nullptr;

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_tempdisgro=nullptr;//临时讨论组

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_groupMana=nullptr;//初始化时别忘记设置群号

};

#endif

**WCXZJ.hpp**

#ifndef WCXZJ\_hpp

#define WCXZJ\_hpp

#include <iostream>

#include <vector>

#include "WCDataXZJ.hpp"

#include "groupXZJ.hpp"

class WCXZJ

{

public:

WCXZJ():

\_people(std::make\_shared<std::vector<std::shared\_ptr<WCDataXZJ>>>()),

\_tempdisgro(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>()),

\_groupMana(std::make\_shared<std::vector<std::shared\_ptr<groupXZJ>>>())

{

}

explicit WCXZJ(std::shared\_ptr<WCXZJ> pWC):

\_people(pWC->people()),

\_groupMana(pWC->groupMana()),

\_tempdisgro(pWC->tempdisgro()){}

explicit WCXZJ(const WCXZJ& pWC):

\_people(pWC.people()),

\_groupMana(pWC.groupMana()),

\_tempdisgro(pWC.tempdisgro()){}

const std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& people()const{return \_people;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& groupMana()const{return \_groupMana;}

const std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro()const{return \_tempdisgro;}

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& people(){return \_people;}

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>>& tempdisgro(){return \_tempdisgro;}

~WCXZJ(){}

private:

std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>> \_people=nullptr;

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_tempdisgro=nullptr;//临时讨论组

std::shared\_ptr<std::vector<std::shared\_ptr<groupXZJ>>> \_groupMana=nullptr;//初始化时别忘记设置群号

};

#endif

**userManaXZJ.hpp**

#ifndef userManaXZJ\_hpp

#define userManaXZJ\_hpp

#include <iostream>

#include "Files.hpp"

#include "QQDataXZJ.hpp"

#include "WCDataXZJ.hpp"

#include "WBDataXZJ.hpp"

#include "creatAccXZJ.hpp"

#include "logXZJ.hpp"

#include "groupManaXZJ.hpp"

#include "friendManaXZJ.hpp"

enum type{qq, wc, wb};

class userManaXZJ

{

public:

explicit userManaXZJ();//调用文件写入QQ，微信，微博数据

~userManaXZJ(){}

private:

type \_t;

void CreatAccount();

void Log();

void ManageGroup();

void ManageFriends();

void mainOperation();

std::shared\_ptr<QQDataXZJ> pQD=nullptr;//当前登录状态

std::shared\_ptr<WCDataXZJ> pWCD=nullptr;//当前登录状态

std::shared\_ptr<WBDataXZJ> pWBD=nullptr;//当前登录状态

std::shared\_ptr<QQXZJ> pQ=nullptr;

std::shared\_ptr<WCXZJ> pWC=nullptr;

std::shared\_ptr<WBXZJ> pWB=nullptr;

std::shared\_ptr<Files> pF=nullptr;

std::shared\_ptr<creatAccXZJ> pcA=nullptr;//管理创建账户

std::shared\_ptr<logXZJ> pl=nullptr;//管理登录

std::shared\_ptr<groupManaXZJ> pgM=nullptr;//管理群

std::shared\_ptr<friendManaXZJ> pfM=nullptr;//管理好友

};

#endif

**userManaXZJ.cpp**

#include "userManaXZJ.hpp"

userManaXZJ::userManaXZJ()

{

pF=(std::make\_shared<Files>());

pQ=(pF->QQ());

pWB=(pF->WB());

pWC=(pF->WC());

std::cout<<"你有两个选择 1.创建账户 2.登录";

long i, judge1;

do {

judge1=1;

std::cin>>i;

switch (i) {

case 1:

{

CreatAccount();

std::cout<<std::endl;

break;

}

case 2:

{

Log();

break;

}

default:

{

std::cout<<"只能输入1或者2哦";

judge1=0;

break;

}

}

}while(judge1==0);

std::cout<<"好了，登录成功，你现在有这些功能"<<std::endl;

mainOperation();

std::cout<<"1.继续操作 2.保存退出"<<std::endl;

long j;

do{

do {

std::cin>>j;

judge1=1;

switch (j) {

case 1:

mainOperation();

break;

case 2:

{

pF->Save(pQ);

pF->Save(pWC);

pF->Save(pWB);

exit(1);

}

default:

{

std::cout<<"只能输入1，2哦";

judge1=0;

break;

}

}

}while(judge1==0);

std::cout<<"1.继续操作 2.保存退出";

std::cin>>j;

}while(1);

}

void userManaXZJ::CreatAccount()

{

const std::shared\_ptr<creatAccXZJ> pcA=std::make\_shared<creatAccXZJ>(pF);

std::cout<<"创建什么账户: 1.qq 2.wechet 3.webo";

long i,j=0;

std::cin>>i;

do {

j=0;

switch (i) {

case 1:

{

pQD=pcA->creatQQAcc();

(pQ->people())->push\_back(pQD);

break;

}

case 2:

{

pWCD=pcA->creatWCAcc();

(pWC->people())->push\_back(pWCD);

break;

}

case 3:

{

pWBD=pcA->creatWBAcc();

(pWB->people())->push\_back(pWBD);

break;

}

default:

std::cout<<"重新输入:";

std::cin>>i;

j=1;

break;

}

}while(j);

}

void userManaXZJ::Log()

{

pl=(std::make\_shared<logXZJ>(pF));

std::cout<<"你要登录 1.QQ 2.微信 3.微博";

long i, judge;

do {

judge=1;

std::cin>>i;

switch (i) {

case 1:

{

\_t=qq;

std::cout<<"输入你的QQ号\_\_";

long j;

std::cin>>j;

pQD=pl->logQQ(j);

break;

}

case 2:

{

\_t=wc;

std::cout<<"输入你的微信ID\_\_";

long j;

std::cin>>j;

pWCD=pl->logWC(j);

break;

}

case 3:

{

\_t=wb;

std::cout<<"输入你的微博号\_\_";

long k;

std::cin>>k;

pWBD=pl->logWB(k);

break;

}

default:

{

std::cout<<"只能输入1，2，3，4哦";

judge=0;

break;

}

}

}while(judge==0);

}

void userManaXZJ::ManageGroup()

{

std::cout<<"选择模式啦"<<std::endl<<std::endl

<<"模式1"<<std::endl

<<"可以创建群,申请加群,退出群,进入群,搜索群的成员,设置临时讨论组,管理群。而管理群是以群主为核心来进行管理的"<<std::endl<<std::endl

<<"选择模式2"<<std::endl

<<"可以创建群,推荐加群,退出群,进入群,搜索群的成员,管理群。而管理群是只有群主是特权账号"<<std::endl<<std::endl

<<"输入你的选择\_\_";

switch (\_t) {

case qq:

pgM=std::make\_shared<groupManaXZJ>(pF, pQD);

break;

case wc:

pgM=std::make\_shared<groupManaXZJ>(pF, pWCD);

break;

case wb:

pgM=std::make\_shared<groupManaXZJ>(pF, pWBD);

break;

}

long i, judge;

do {

judge=1;

std::cin>>i;

switch (i) {

case 1:

pgM->ModeOne();

break;

case 2:

pgM->ModeTwo();

break;

default:

{

std::cout<<"只能选择模式1或者模式2哦";

judge=0;

break;

}

}

}while(judge==0);

}

void userManaXZJ::ManageFriends()

{

long judge1, judge2;

std::cout<<"你有这些管理好友的功能"<<std::endl

<<"1.添加好友 2.推荐好友 3.查询好友 4.删除好友 5.同意或者拒绝其他好友添加";

long i;

switch (\_t) {

case qq:

pfM=std::make\_shared<friendManaXZJ>(pF, pQD);

break;

case wc:

pfM=std::make\_shared<friendManaXZJ>(pF, pWCD);

break;

case wb:

pfM=std::make\_shared<friendManaXZJ>(pF, pWBD);

break;

default:

break;

}

do {

std::cin>>i;

judge1=1;

switch (i) {

case 1:

{

pfM->addFriend();

break;

}

case 2:

{

judge2=1;

std::cout<<"推荐什么好好友 1.QQ 2.微信 3.微博";

long j;

do {

std::cin>>j;

switch (j) {

case 1:

pfM->recoQQFriend();

break;

case 2:

pfM->recoWCFriend();

break;

case 3:

pfM->recoWBFriend();

break;

default:

{

std::cout<<"只能输入1,2,3哦";

judge2=0;

break;

}

}

}while(judge2==0);

break;

}

case 3:

{

std::cout<<"查询 1.所有好友 2.按照名字查询 3.ID查询";

long j;

do {

std::cin>>j;

judge2=1;

switch (j) {

case 1:

pfM->searchFriend();

break;

case 2:

{

std::cout<<"输入好友名字";

std::string name;

std::cin>>name;

pfM->searchFriend(name);

break;

}

case 3:

{

std::cout<<"输入好友ID";

long id;

std::cin>>id;

pfM->searchFriend(id);

break;

}

default:

{

std::cout<<"只能输入1,2,3哦";

judge2=0;

break;

}

}

}while(judge2==0);

break;

}

case 4:

{

pfM->searchFriend();

std::cout<<"输入删除哪个好友";

long hy;

std::cin>>hy;

switch (\_t) {

case qq:

pfM->deleteFriend(\*((pQD->friendList()).begin()+hy-1));

break;

case wc:

pfM->deleteFriend(\*((pWCD->friendList()).begin()+hy-1));

break;

case wb:

pfM->deleteFriend(\*((pWBD->friendList()).begin()+hy-1));

break;

}

break;

}

case 5:

{

pfM->agreeFriend();

break;

}

default:

{

std::cout<<"只能输入1～5哦";

judge1=0;

break;

}

}

}while(judge1==0);

}

void userManaXZJ::mainOperation()

{

long judge2;

std::cout<<"1.有关群的操作 2.有关好友的操作 3.保存退出";

long j;

do {

judge2=1;

std::cin>>j;

switch (j) {

case 1:

ManageGroup();

break;

case 2:

ManageFriends();

break;

case 3:

{

pF->Save(pQ);

pF->Save(pWB);

pF->Save(pWC);

exit(1);

break;

}

default:

{

std::cout<<"只能输入1或者2或者3哦";

judge2=0;

break;

}

}

}while(judge2==0);

}

**creatAcc.hpp**

#ifndef creatAccXZJ\_hpp

#define creatAccXZJ\_hpp

#include <iostream>

class creatAccImplXZJ;

class QQDataXZJ;

class WBDataXZJ;

class WCDataXZJ;

class Files;

class creatAccXZJ

{

public:

creatAccXZJ(const std::shared\_ptr<Files>& \_pF);

const std::shared\_ptr<QQDataXZJ> creatQQAcc();

const std::shared\_ptr<WBDataXZJ> creatWBAcc();

const std::shared\_ptr<WCDataXZJ> creatWCAcc();

~creatAccXZJ(){}

private:

std::shared\_ptr<creatAccImplXZJ> pcAI=nullptr;

};

#endif

**creatAcc.cpp**

#include "creatAccXZJ.hpp"

#include "creatAccImplXZJ.hpp"

creatAccXZJ::creatAccXZJ(const std::shared\_ptr<Files>& \_pF)

{

pcAI=(std::make\_shared<creatAccImplXZJ>(\_pF));

}

const std::shared\_ptr<QQDataXZJ> creatAccXZJ::creatQQAcc()

{

return pcAI->creatQQAcc();

}

const std::shared\_ptr<WCDataXZJ> creatAccXZJ::creatWCAcc()

{

return pcAI->creatWCAcc();

}

const std::shared\_ptr<WBDataXZJ> creatAccXZJ::creatWBAcc()

{

return pcAI->creatWBAcc();

}

**creatAccImpl.hpp**

#ifndef creatAccImplXZJ\_hpp

#define creatAccImplXZJ\_hpp

#include <iostream>

#include <ctime>

#include "Files.hpp"

#include "ConnectXZJ.hpp"

#include "QQDataXZJ.hpp"

#include "WBDataXZJ.hpp"

#include "WCDataXZJ.hpp"

enum choose{qq, wechet, webo};

class creatAccImplXZJ

{

public:

creatAccImplXZJ(const std::shared\_ptr<Files>& pF);

const std::shared\_ptr<QQDataXZJ> creatQQAcc();

const std::shared\_ptr<WBDataXZJ> creatWBAcc();

const std::shared\_ptr<WCDataXZJ> creatWCAcc();

~creatAccImplXZJ(){}

private:

time\_t rawtime;

struct tm \*ptminfo;

std::shared\_ptr<Files> \_pF=nullptr;

std::shared\_ptr<QQXZJ> \_pQ=nullptr;

std::shared\_ptr<WBXZJ> \_pWB=nullptr;

std::shared\_ptr<WCXZJ> \_pWC=nullptr;

};

#endif

**creatAccImpl.cpp**

#include "creatAccImplXZJ.hpp"

creatAccImplXZJ::creatAccImplXZJ(const std::shared\_ptr<Files>& pF)

{

\_pF=(pF);

\_pQ=(std::make\_shared<QQXZJ>(\_pF->QQ()));

\_pWC=(std::make\_shared<WCXZJ>(\_pF->WC()));

\_pWB=(std::make\_shared<WBXZJ>(\_pF->WB()));

}

const std::shared\_ptr<QQDataXZJ> creatAccImplXZJ::creatQQAcc()

{

time(&rawtime);

ptminfo = localtime(&rawtime);

srand(static\_cast<unsigned>(time(0)));

//年 ptminfo->tm\_year + 1900

//月 ptminfo->tm\_mon + 1

//日 ptminfo->tm\_mday

// cout << rand() % 100 << endl;

long \_password, password;

std::cout<<"设置密码\_\_";std::cin>>\_password;

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

while(\_password!=password)

{

std::cout<<"密码有误";

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

}

std::string \_name;

std::cout<<"输入你想起的名字\_\_";std::cin>>\_name;

std::cout<<std::endl;

std::string \_home;

std::cout<<"输入你的家乡\_\_";std::cin>>\_home;

std::cout<<std::endl;

long i,j,k;

std::cout<<"输入你的出生日期: 年\_\_";std::cin>>i;

Year y(i);

std::cout<<" 月\_\_";std::cin>>j;

Month m(j);

std::cout<<" 日\_\_";std::cin>>k;

Day d(k);

Time \_birth(y, m, d);

std::cout<<std::endl;

Time \_t\_age(Year(ptminfo->tm\_year + 1900), Month(ptminfo->tm\_mon + 1), Day(ptminfo->tm\_mday));//根据系统时间

long \_ID;//随机生成，不能重复

long \_qqNum;//随机生成，不能重复

long \_wb=0, \_wc=0;

std::cout<<"是否想要绑定 是Y/y, 否N/n"<<std::endl;

char cho;

long judge1, judge2, judge3, judge4=1;//判断是否输入错误

long yby=0;//是否继续绑定

do {

judge1=1;

std::cin>>cho;

switch (cho) {

case 'Y':

case 'y':

{

do {

do {

std::cout<<"绑定 a.微博 b.微信\_\_";

char ch;std::cin>>ch;

judge2=0;

switch (ch) {

case 'a':

{

std::cout<<"绑定 a.微博ID b.微博账号\_\_";

char c;

std::cin>>c;

do {

judge3=0;

switch (c) {

case 'a':

std::cout<<"输入ID号\_\_";

long idNum;

std::cin>>idNum;

\_wb=ConnectWB(ID, idNum);

break;

case 'b':

std::cout<<"输入账号\_\_";

long Number;

std::cin>>Number;

\_wb=ConnectWB(number, Number);

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>c;

judge3=1;

break;

}

}while(judge3);

break;

}

case 'b':

{

std::cout<<"输入绑定的微信ID号码\_\_";

long id;

std::cin>>id;

\_ID=ConnectWC(ID, id);

\_wc=\_ID;

judge4=0;

break;

}

default:

{

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>ch;

judge2=1;

break;

}

}

}while(judge2);

std::cout<<"还要继续绑定吗 1.要 2.不要";

std::cin>>yby;

}while(yby==1);

break;

}

case 'N':

case 'n':

break;

default:

{

std::cout<<"只能输入Y,y,N,n哦";

judge1=0;

break;

}

}

}while(judge1==0);

if(judge4!=0)

{

\_ID=rand()%1000000;

if(!(\_pQ->people()->empty()))

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i){

if(\_ID==(\*i)->ID()){

do {

\_ID=rand()%1000000;

}while(\_ID==(\*i)->ID());

}

}

}

\_qqNum=rand()%10000000000;

if(!(\_pQ->people())->empty())

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i){

if(\_qqNum==(\*i)->qqNum()){

do {

\_qqNum=rand()%10000000000;

}while(\_qqNum==(\*i)->qqNum());

}

}

std::cout<<"\*\*创建成功\*\*"<<std::endl;

std::cout<<"QQ号是:"<<\_qqNum<<std::endl;

return std::make\_shared<QQDataXZJ>(\_name, \_home, \_birth, \_t\_age, \_ID, password, \_wc, \_wb, \_qqNum);

}

const std::shared\_ptr<WBDataXZJ> creatAccImplXZJ::creatWBAcc()

{

time(&rawtime);

ptminfo = localtime(&rawtime);

srand(static\_cast<unsigned>(time(0)));

long \_password, password;

std::cout<<"设置密码\_\_";std::cin>>\_password;

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

while(\_password!=password)

{

std::cout<<"密码有误";

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

}

std::string \_name;

std::cout<<"输入你想起的名字\_\_";std::cin>>\_name;

std::cout<<std::endl;

std::string \_home;

std::cout<<"输入你的家乡\_\_";std::cin>>\_home;

std::cout<<std::endl;

long i,j,k;

std::cout<<"输入你的出生日期: 年\_\_";std::cin>>i;

Year y(i);

std::cout<<" 月\_\_";std::cin>>j;

Month m(j);

std::cout<<" 日\_\_";std::cin>>k;

Day d(k);

Time \_birth(y, m, d);

std::cout<<std::endl;

Time \_t\_age(Year(ptminfo->tm\_year + 1900), Month(ptminfo->tm\_mon + 1), Day(ptminfo->tm\_mday));//根据系统时间

long \_ID;//随机生成，不能重复

long \_wbNum;//随机生成，不能重复

long \_qq=0, \_wc=0;

std::cout<<"是否想要绑定 是Y/y, 否N/n";

char cho;

long judge1=0, judge2=0, judge3=0, judge4=1;//判断是否输入错误

long yby;//是否继续绑定

do {

judge1=0;

std::cin>>cho;

switch (cho) {

case 'Y':

case 'y':

{

do {

do {

std::cout<<"绑定 a.QQ b.微信\_\_";

char ch;std::cin>>ch;

judge2=0;

switch (ch) {

case 'a':

std::cout<<"绑定 a.QQID b.QQ账号\_\_";

char c;

std::cin>>c;

do {

judge3=0;

switch (c) {

case 'a':

std::cout<<"输入ID号\_\_";

long idNum;

std::cin>>idNum;

\_qq=ConnectQQ(ID, idNum);

break;

case 'b':

std::cout<<"输入账号\_\_";

long Number;

std::cin>>Number;

\_qq=ConnectQQ(number, Number);

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>c;

judge3=1;

break;

}

}while(judge3);

break;

case 'b':

std::cout<<"输入绑定的微信ID号码\_\_";

long id;

std::cin>>id;

\_ID=ConnectWC(ID, id);

\_wc=\_ID;

judge4=0;

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>ch;

judge2=1;

break;

}

}while(judge2);

std::cout<<"还要继续绑定吗 1.要 2.不要";

std::cin>>yby;

}while(yby);

break;

}

case 'N'://直接生成ID和号码

case 'n':

break;

default:

std::cout<<"只能输入Y,y,N,n哦";

judge1=1;

break;

}

}while(judge1);

if(judge4!=0)

{

\_ID=rand()%1000000;

for(auto i=(\_pWB->people())->cbegin(); i!=(\_pWB->people())->cend(); ++i){

if(\_ID==(\*i)->ID()){

do {

\_ID=rand()%1000000;

}while(\_ID==(\*i)->ID());

}

}

}

\_wbNum=rand()%10000000000;

for(auto i=(\_pWB->people())->cbegin(); i!=(\_pWB->people())->cend(); ++i){

if(\_wbNum==(\*i)->wbNum()){

do {

\_wbNum=rand()%10000000000;

}while(\_wbNum==(\*i)->wbNum());

}

}

std::cout<<"\*\*创建成功\*\*";

std::cout<<"微博号是:";

std::cout<<\_wbNum;

return std::make\_shared<WBDataXZJ>(\_name, \_home, \_birth, \_t\_age, \_ID, password, \_qq, \_wc, \_wbNum);

}

const std::shared\_ptr<WCDataXZJ> creatAccImplXZJ::creatWCAcc()

{

time(&rawtime);

ptminfo = localtime(&rawtime);

srand(static\_cast<unsigned>(time(0)));

long \_password, password;

std::cout<<"设置密码\_\_";std::cin>>\_password;

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

while(\_password!=password)

{

std::cout<<"密码有误";

std::cout<<std::endl;

std::cout<<"再次输入密码\_\_";std::cin>>password;

std::cout<<std::endl;

}

std::string \_name;

std::cout<<"输入你想起的名字\_\_";std::cin>>\_name;

std::cout<<std::endl;

std::string \_home;

std::cout<<"输入你的家乡\_\_";std::cin>>\_home;

std::cout<<std::endl;

long i,j,k;

std::cout<<"输入你的出生日期: 年\_\_";std::cin>>i;

Year y(i);

std::cout<<" 月\_\_";std::cin>>j;

Month m(j);

std::cout<<" 日\_\_";std::cin>>k;

Day d(k);

Time \_birth(y, m, d);

std::cout<<std::endl;

Time \_t\_age(Year(ptminfo->tm\_year + 1900), Month(ptminfo->tm\_mon + 1), Day(ptminfo->tm\_mday));//根据系统时间

long \_ID;//随机生成，不能重复

long \_qq=0, \_wb=0;

std::cout<<"是否想要绑定 是Y/y, 否N/n";

char cho; std::cin>>cho;

long judge1=0, judge2=0, judge3=0;//判断是否输入错误

long yby;//是否继续绑定

do {

judge1=0;

switch (cho) {

case 'Y':

case 'y':

{

do {

do {

std::cout<<"绑定 a.QQ b.微博\_\_";

char ch;std::cin>>ch;

judge2=0;

switch (ch) {

case 'a':

std::cout<<"绑定 a.QQID b.QQ账号\_\_";

char c;

std::cin>>c;

do {

judge3=0;

switch (c) {

case 'a':

std::cout<<"输入ID号\_\_";

long idNum;

std::cin>>idNum;

\_qq=ConnectQQ(ID, idNum);

break;

case 'b':

std::cout<<"输入账号\_\_";

long Number;

std::cin>>Number;

\_qq=ConnectQQ(number, Number);

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>c;

judge3=1;

break;

}

}while(judge3);

break;

case 'b':

std::cout<<"绑定 a.微博ID b.微博账号\_\_";

char cc;

std::cin>>cc;

do {

judge3=0;

switch (cc) {

case 'a':

std::cout<<"输入ID号\_\_";

long idNum;

std::cin>>idNum;

\_wb=ConnectWB(ID, idNum);

break;

case 'b':

std::cout<<"输入账号\_\_";

long Number;

std::cin>>Number;

\_wb=ConnectWB(number, Number);

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>cc;

judge3=1;

break;

}

}while(judge3);

break;

default:

std::cout<<"只能输入a,b哦"<<std::endl<<"重新输入\_\_";

std::cin>>ch;

judge2=1;

break;

}

}while(judge2);

std::cout<<"还要继续绑定吗 1.要 2.不要";

std::cin>>yby;

}while(yby);

break;

}

case 'N'://直接生成ID和号码

case 'n':

break;

default:

std::cout<<"只能输入Y,y,N,n哦";

judge1=1;

break;

}

}while(judge1);

\_ID=rand()%1000000;

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i){

if(\_ID==(\*i)->ID()){

do {

\_ID=rand()%1000000;

}while(\_ID==(\*i)->ID());

}

}

std::cout<<"\*\*创建成功\*\*";

std::cout<<"微信ID是:";

std::cout<<\_ID;

return std::make\_shared<WCDataXZJ>(\_name, \_home, \_birth, \_t\_age, \_ID, password, \_qq, \_wb);

}

**logXZJ.hpp**

#ifndef logXZJ\_hpp

#define logXZJ\_hpp

#include <iostream>

class logImplXZJ;

class QQDataXZJ;

class WBDataXZJ;

class WCDataXZJ;

class Files;

class logXZJ

{

public:

logXZJ(const std::shared\_ptr<Files>& \_pF);//调用文件写入QQ，微信，微博数据

const std::shared\_ptr<QQDataXZJ> logQQ(long accnum);//简单确认之后其他自动登录

const std::shared\_ptr<WBDataXZJ> logWB(long accnum);

const std::shared\_ptr<WCDataXZJ> logWC(long accnum);

const std::shared\_ptr<QQDataXZJ> nopasswordlogQQ(long accnum);

const std::shared\_ptr<WBDataXZJ> nopasswordlogWB(long accnum);

const std::shared\_ptr<WCDataXZJ> nopasswordlogWC(long accnum);

const std::shared\_ptr<QQDataXZJ> pQD()const;

const std::shared\_ptr<WBDataXZJ> pWBD()const;

const std::shared\_ptr<WCDataXZJ> pWCD()const;

~logXZJ(){}

private:

std::shared\_ptr<logImplXZJ> plI=nullptr;

};

#endif

**logXZJ.cpp**

#include "logXZJ.hpp"

#include "logImplXZJ.hpp"

logXZJ::logXZJ(const std::shared\_ptr<Files>& \_pF)

//调用文件写入QQ，微信，微博数据

{

plI=(std::make\_shared<logImplXZJ>(\_pF));

}

const std::shared\_ptr<QQDataXZJ> logXZJ::logQQ(long accnum)//简单确认之后其他自动登录

{

return plI->logQQ(accnum);

}

const std::shared\_ptr<WBDataXZJ> logXZJ::logWB(long accnum)

{

return plI->logWB(accnum);

}

const std::shared\_ptr<WCDataXZJ> logXZJ::logWC(long accnum)

{

return plI->logWC(accnum);

}

const std::shared\_ptr<QQDataXZJ> logXZJ::pQD()const

{

return plI->pQD();

}

const std::shared\_ptr<WBDataXZJ> logXZJ::pWBD()const

{

return plI->pWBD();

}

const std::shared\_ptr<WCDataXZJ> logXZJ::pWCD()const

{

return plI->pWCD();

}

const std::shared\_ptr<QQDataXZJ> logXZJ::nopasswordlogQQ(long accnum)

{

return plI->nopasswordlogQQ(accnum);

}

const std::shared\_ptr<WBDataXZJ> logXZJ::nopasswordlogWB(long accnum)

{

return plI->nopasswordlogWB(accnum);

}

const std::shared\_ptr<WCDataXZJ> logXZJ::nopasswordlogWC(long accnum)

{

return plI->nopasswordlogWC(accnum);

}

**logImplXZJ.hpp**

#ifndef logImplXZJ\_hpp

#define logImplXZJ\_hpp

#include <iostream>

#include "Files.hpp"

#include "QQDataXZJ.hpp"

#include "WBDataXZJ.hpp"

#include "WCDataXZJ.hpp"

class logImplXZJ

{

public:

explicit logImplXZJ(const std::shared\_ptr<Files>& \_pF) //调用文件写入QQ，微信，微博数据

{

pF=(\_pF);

\_pQ=(pF->QQ());

\_pWC=(pF->WC());

\_pWB=(pF->WB());

}

const std::shared\_ptr<QQDataXZJ> logQQ(long accnum);

const std::shared\_ptr<WBDataXZJ> logWB(long accnum);

const std::shared\_ptr<WCDataXZJ> logWC(long accnum);

const std::shared\_ptr<QQDataXZJ> nopasswordlogQQ(long accnum);

const std::shared\_ptr<WBDataXZJ> nopasswordlogWB(long accnum);

const std::shared\_ptr<WCDataXZJ> nopasswordlogWC(long accnum);

const std::shared\_ptr<QQDataXZJ> pQD()const{return \_pQD;}

const std::shared\_ptr<WBDataXZJ> pWBD()const{return \_pWBD;}

const std::shared\_ptr<WCDataXZJ> pWCD()const{return \_pWCD;}

~logImplXZJ(){}

private:

std::shared\_ptr<Files> pF=nullptr;

std::shared\_ptr<QQXZJ> \_pQ=nullptr;

std::shared\_ptr<WBXZJ> \_pWB=nullptr;

std::shared\_ptr<WCXZJ> \_pWC=nullptr;

std::shared\_ptr<QQDataXZJ> \_pQD=nullptr;

std::shared\_ptr<WBDataXZJ> \_pWBD=nullptr;

std::shared\_ptr<WCDataXZJ> \_pWCD=nullptr;

const std::shared\_ptr<QQDataXZJ> findQQperson(long accnum)const;

const std::shared\_ptr<WBDataXZJ> findWBperson(long accnum)const;

std::\_\_wrap\_iter<const std::shared\_ptr<WCDataXZJ> \*> extracted() const;

const std::shared\_ptr<WCDataXZJ> findWCperson(long accnum)const;

};

#endif

**logImplXZJ.cpp**

#include "logImplXZJ.hpp"

const std::shared\_ptr<QQDataXZJ> logImplXZJ::logQQ(long accnum)

{

while(1){

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i)

{

if((\*i)->qqNum()==accnum)

{

long j=0;

do {

std::cout<<"输入密码\_\_";

std::cin>>j;

std::cout<<std::endl;

if(j!=(\*i)->password())

std::cout<<"密码错误";

}while(j!=(\*i)->password());

if((\*i)->wb()!=0)

\_pWBD=findWBperson((\*i)->wb());

if((\*i)->wc()!=0)

\_pWCD=findWCperson((\*i)->wc());

return \*i;

}

}

std::cout<<"没找到用户, 重新输入账户";

std::cin>>accnum;

}

}

const std::shared\_ptr<WBDataXZJ> logImplXZJ::logWB(long accnum)

{

while(1){

for(auto i=(\_pWB->people())->cbegin(); i!=(\_pWB->people())->cend(); ++i)

{

if((\*i)->wbNum()==accnum)

{

long j=0;

do {

std::cout<<"输入密码:";

std::cin>>j;

std::cout<<std::endl;

}while(j!=(\*i)->password());

if((\*i)->qq()!=0)

\_pQD=findQQperson((\*i)->qq());

if((\*i)->wc()!=0)

\_pWCD=findWCperson((\*i)->wc());

return \*i;

}

}

std::cout<<"没找到用户, 重新输入账户";

std::cin>>accnum;

}

return nullptr;

}

const std::shared\_ptr<WCDataXZJ> logImplXZJ::logWC(long accnum)

{

while(1){

for(auto i=(\_pWC->people())->cbegin(); i!=(\_pWC->people())->cend(); ++i)

{

if((\*i)->ID()==accnum)

{

long j=0;

do {

std::cout<<"输入密码:";

std::cin>>j;

std::cout<<std::endl;

}while(j!=(\*i)->password());

if((\*i)->qq()!=0)

\_pQD=findQQperson((\*i)->qq());

if((\*i)->wb()!=0)

\_pWBD=findWBperson((\*i)->wb());

return \*i;

}

}

std::cout<<"没找到用户, 重新输入账户";

std::cin>>accnum;

}

return nullptr;

}

const std::shared\_ptr<QQDataXZJ> logImplXZJ::findQQperson(long accnum)const

{

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i)

{

if((\*i)->qqNum()==accnum)

return \*i;

if((\*i)->ID()==accnum)

return \*i;

}

std::cout<<"没有这个QQ号/ID";

return nullptr;

}

const std::shared\_ptr<WBDataXZJ> logImplXZJ::findWBperson(long accnum)const

{

for(auto i=(\_pWB->people())->cbegin(); i!=(\_pWB->people())->cend(); ++i)

{

if((\*i)->wbNum()==accnum)

return \*i;

if((\*i)->ID()==accnum)

return \*i;

}

std::cout<<"没有这个微博号/ID";

return nullptr;

}

const std::shared\_ptr<WCDataXZJ> logImplXZJ::findWCperson(long accnum)const

{

for(auto i=(\_pWC->people())->cbegin(); i!=(\_pWC->people())->cend(); ++i)

if((\*i)->ID()==accnum)

return \*i;

std::cout<<"没有这个微信ID";

return nullptr;

}

const std::shared\_ptr<QQDataXZJ> logImplXZJ::nopasswordlogQQ(long accnum)

{

for(auto i=(\_pQ->people())->cbegin(); i!=(\_pQ->people())->cend(); ++i)

{

if((\*i)->qqNum()==accnum)

{

if((\*i)->wb()!=0)

\_pWBD=findWBperson((\*i)->wb());

if((\*i)->wc()!=0)

\_pWCD=findWCperson((\*i)->wc());

return \*i;

}

}

return nullptr;

}

const std::shared\_ptr<WBDataXZJ> logImplXZJ::nopasswordlogWB(long accnum)

{

for(auto i=(\_pWB->people())->cbegin(); i!=(\_pWB->people())->cend(); ++i)

{

if((\*i)->wbNum()==accnum)

{

if((\*i)->qq()!=0)

\_pQD=findQQperson((\*i)->qq());

if((\*i)->wc()!=0)

\_pWCD=findWCperson((\*i)->wc());

return \*i;

}

}

std::cout<<"没找到这个人";

return nullptr;

}

const std::shared\_ptr<WCDataXZJ> logImplXZJ::nopasswordlogWC(long accnum)

{

for(auto i=(\_pWC->people())->cbegin(); i!=(\_pWC->people())->cend(); ++i)

{

if((\*i)->ID()==accnum)

{

if((\*i)->qq()!=0)

\_pQD=findQQperson((\*i)->qq());

if((\*i)->wb()!=0)

\_pWBD=findWBperson((\*i)->wb());

return \*i;

}

}

return nullptr;

}

**groupManaXZJ.hpp**

#ifndef groupManaXZJ\_hpp

#define groupManaXZJ\_hpp

#include <iostream>

class groupManaImplXZJ;

class QQDataXZJ;

class WCDataXZJ;

class WBDataXZJ;

class Files;

class groupManaXZJ

{

public:

explicit groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& pQD);//调用文件写入QQ，微信，微博数据

explicit groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& pWCD);

explicit groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& pWBD);

void ModeOne();

void ModeTwo();

~groupManaXZJ(){}

private:

std::shared\_ptr<groupManaImplXZJ> pgMI=nullptr;

};

#endif

**groupManaXZJ.cpp**

#include "groupManaXZJ.hpp"

#include "groupManaImplXZJ.hpp"

groupManaXZJ::groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& pQD)//调用文件写入QQ，微信，微博数据

{

pgMI=(std::make\_shared<groupManaImplXZJ>(\_pF, pQD));

}

groupManaXZJ::groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& pWCD)

{

pgMI=(std::make\_shared<groupManaImplXZJ>(\_pF, pWCD));

}

groupManaXZJ::groupManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& pWBD)

{

pgMI=(std::make\_shared<groupManaImplXZJ>(\_pF, pWBD));

}

void groupManaXZJ::ModeOne()

{

pgMI->ModeOne();

}

void groupManaXZJ::ModeTwo()

{

pgMI->ModeTwo();

}

**groupManaImplXZJ.hpp**

#ifndef groupManaImplXZJ\_hpp

#define groupManaImplXZJ\_hpp

#include <iostream>

#include <ctime>

#include "Files.hpp"

#include "BasicDataXZJ.hpp"

#include "QQDataXZJ.hpp"

#include "WBDataXZJ.hpp"

#include "WCDataXZJ.hpp"

enum type{ qq, wc, wb};

enum managertype{ Owner, administrator, normal};

class groupManaImplXZJ

{

public: //调用文件写入QQ，微信，微博数据

explicit groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF);

explicit groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& user);

explicit groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& user);

explicit groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& user);

void ModeOne();//群可以申请加入，可以设置临时讨论组，群主和管理员一起管理群

void ModeTwo();//群只能推荐加入，不可以设置临时讨论组，群主管理群

~groupManaImplXZJ(){}

private://加群没有单独函数，因为加群的理念不一样，不会涉及到代码重复

//应用里面操作

void creatGro();

int inGro(long i);

void exitGro(long& i, std::shared\_ptr<QQDataXZJ>);//给出这个人的 位置 和 这个人 退出群

void exitGro(long& i, std::shared\_ptr<WCDataXZJ>);

void exitGro(long& i, std::shared\_ptr<WBDataXZJ>);

void showAllGroNum();

void showGroNum();

void showtempGroNum();

//在你进入的群里面操作

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<QQDataXZJ> \*>& p)const;//展示一个人

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WCDataXZJ> \*>& p)const;

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WBDataXZJ> \*>& p)const;

void showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& p)const;//展示一群人

void showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& p)const;

void showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& p)const;

void searchMem()const;//直接展示所有成员

void searchMem(const std::string& name)const;//查询成员，名字查询

void searchMem(const long id)const;//ID查询

void manageSearchMem();

void settempdisgro();

//群主和管理员的功能函数

bool isOwner();

bool isAdministrator();

void confirmPosition();

void Kickmem(long number);

void disSolvedGro();

void addMem();

void setAdministrator(long number);

type \_t;//登录的种类

managertype \_mt;//群中的地位

std::shared\_ptr<QQDataXZJ> pQD=nullptr;//当前使用的人

std::shared\_ptr<WCDataXZJ> pWCD=nullptr;

std::shared\_ptr<WBDataXZJ> pWBD=nullptr;

std::shared\_ptr<Files> pF=nullptr;

std::shared\_ptr<QQXZJ> pQ=nullptr;//里面有群

std::shared\_ptr<WBXZJ> pWB=nullptr;

std::shared\_ptr<WCXZJ> pWC=nullptr;

std::shared\_ptr<groupXZJ> \_qqGro=nullptr;//当前登录的群

std::shared\_ptr<groupXZJ> \_wbGro=nullptr;

std::shared\_ptr<groupXZJ> \_wcGro=nullptr;

};

#endif

**groupManaImplXZJ.cpp**

#include "groupManaImplXZJ.hpp"

groupManaImplXZJ::groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& user):

pF(\_pF),

pQ(std::make\_shared<QQXZJ>(pF->QQ())),

pWC(std::make\_shared<WCXZJ>(pF->WC())),

pWB(std::make\_shared<WBXZJ>(pF->WB())),

pQD(user),

\_qqGro(std::make\_shared<groupXZJ>())

{

\_t=qq;

}

groupManaImplXZJ::groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& user):

pF(\_pF),

pQ(std::make\_shared<QQXZJ>(pF->QQ())),

pWC(std::make\_shared<WCXZJ>(pF->WC())),

pWB(std::make\_shared<WBXZJ>(pF->WB())),

pWCD(user),

\_wcGro(std::make\_shared<groupXZJ>())

{

\_t=wc;

}

groupManaImplXZJ::groupManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& user):

pF(\_pF),

pQ(std::make\_shared<QQXZJ>(pF->QQ())),

pWC(std::make\_shared<WCXZJ>(pF->WC())),

pWB(std::make\_shared<WBXZJ>(pF->WB())),

pWBD(user),

\_wbGro(std::make\_shared<groupXZJ>())

{

\_t=wb;

}

void groupManaImplXZJ::ModeOne()

{

std::cout<<"选择你的操作 1.创建群 2.申请加群 3.退出群 4.进入群";

//群管理1.踢人(群主，管理员) 2.解散群（群主） 3.同意网友进群或者拒绝(群主，管理员) 4.设置管理员(群主) 5.(设置临时讨论组)

long i;

do {

std::cin>>i;

switch (i) {

case 1:

{

creatGro();

break;

}

case 2:

{

int judge=0;

showAllGroNum();

std::cout<<"想加哪个群";

long u;

switch (\_t) {

case qq:

{

do {

judge=0;

std::cin>>u;

for(auto ii=(pQ->groupMana())->begin(); ii!=(pQ->groupMana())->end(); ++ii)

if((\*ii)->groupNum()==u)

{

judge=1;

((\*ii)->askJoinQQGroList())->push\_back(pQD);

}

if(judge==0)

std::cout<<"没有这个群"<<std::endl;

}while(judge==0);

break;

}

case wb:

{

do {

judge=0;

std::cin>>u;

for(auto ii=(pWB->groupMana())->begin(); ii!=(pWB->groupMana())->end(); ++ii)

if((\*ii)->groupNum()==u)

{

judge=1;

((\*ii)->askJoinWBGroList())->push\_back(pWBD);

}

if(judge==0)

std::cout<<"没有这个群"<<std::endl;

}while(judge==0);

break;

}

case wc:

{

do {

judge=0;

std::cin>>u;

for(auto ii=(pWC->groupMana())->begin(); ii!=(pWC->groupMana())->end(); ++ii)

if((\*ii)->groupNum()==u)

{

judge=1;

((\*ii)->askJoinWCGroList())->push\_back(pWCD);

}

if(judge==0)

std::cout<<"没有这个群"<<std::endl;

}while(judge==0);

break;

}

}

break;

}

case 3:

{

std::cout<<"你有这些群:"<<std::endl;

showGroNum();

std::cout<<"你有这些临时讨论组"<<std::endl;

showtempGroNum();

std::cout<<"输入要退出的群号;"<<std::endl;

long gron;

std::cin>>gron;

switch (\_t) {

case qq:

exitGro(gron, pQD);

break;

case wb:

exitGro(gron, pWCD);

break;

case wc:

exitGro(gron, pWBD);

break;

}

break;

}

case 4:

{

int judge=1;

std::cout<<"群"<<std::endl;

showGroNum();

std::cout<<"讨论组"<<std::endl;

showtempGroNum();

std::cout<<"输入你要进入的群号";

long num;

std::cin>>num;

int ju=inGro(num);

if(ju==1)

{

std::cout<<"你可以有下面的操作 1.管理群 2.搜索群成员\_\_ 3.设置临时讨论组";

int choose;

do {

std::cin>>choose;

switch (choose) {

case 1:

{

if(\_mt==normal)

std::cout<<"你不是群主或者管理员哦"<<std::endl;

else

{

if(isOwner())//群主

{

std::cout<<"你是群主"<<std::endl;

std::cout<<"群管理 1.踢人 2.解散群 3.同意网友进群或者拒绝 4.设置管理员";

long cho;

do {

std::cin>>cho;

switch (cho) {

case 1:

{

std::vector<std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>>::size\_type count=0;

searchMem();

switch (\_t) {

case qq:

count=(\_qqGro->qqGro()->size());

break;

case wc:

count=(\_wcGro->wcGro())->size();

break;

case wb:

count=(\_wbGro->wbGro())->size();

break;

}

std::cout<<"输入你要踢出的人的编号";

while(1)

{

long bh;

std::cin>>bh;

if(bh<=0||bh>count)

{

std::cout<<"没有这个人";

std::cin>>bh;

}

else

Kickmem(bh-1);

}

break;

}

case 2:

{

disSolvedGro();

break;

}

case 3:

{

addMem();

break;

}

case 4:

{

searchMem();

std::cout<<"输入你要设置的人的编号";

long bh;

std::cin>>bh;

setAdministrator(bh-1);

break;

}

default:

{

std::cout<<"只能输入1～4哦";

cho=0;

break;

}

}

}while(cho==0);

}

else if(isAdministrator())

{

std::cout<<"你是管理员"<<std::endl;

std::cout<<"群管理 1.踢人 2.同意网友进群或者拒绝";

long cho;

do {

std::cin>>cho;

switch (cho) {

case 1:

{

searchMem();

std::cout<<"输入你要踢出的人的编号";

long bh;

std::cin>>bh;

Kickmem(bh-1);

break;

}

case 2:

{

addMem();

break;

}

default:

{

std::cout<<"只能输入1～2哦";

cho=0;

break;

}

}

}while(cho==0);

}

}

break;

}

case 2:

{

manageSearchMem();

break;

}

case 3:

{

settempdisgro();

break;

}

default:

{

std::cout<<"只能输入1或者2哦";

judge=0;

break;

}

}

}while(judge==0);

break;

}

}

}

}while(i==0);

}

void groupManaImplXZJ::ModeTwo()

{

std::cout<<"选择你的操作 1.创建群 2.推荐加群 3.退出群 4.进入群";

//管理群:1.踢人(群主) 2.解散群(群主) 3.同意网友进群或者拒绝(群主)

long i;

do {

std::cin>>i;

switch (i) {

case 1:

{

creatGro();

break;

}

case 2:

{

srand(static\_cast<unsigned>(time(0)));

switch (\_t) {

case qq:

{

long count=0;

long select;

std::vector<long> groname;

for(auto i=(pQ->groupMana())->cbegin(); i!=(pQ->groupMana())->cend(); ++i)

{

++count;

groname.push\_back((\*i)->groupNum());

}

long j=0;

do {

select=rand()%count;

for(auto i=(pQD->groNum()).cbegin(); i!=(pQD->groNum()).cend(); ++i)

if(\*i==groname[select])

j=1;

}while(j);

std::cout<<"推荐你加群"<<select<<std::endl

<<"1.加群 2.不加群";

long k;

do {

std::cin>>k;

switch (k) {

case 1:

for(auto i=(pQ->groupMana())->begin(); i!=(pQ->groupMana())->end(); ++i)

if((\*i)->groupNum()==groname[select])

((\*i)->askJoinQQGroList())->push\_back(pQD);

break;

case 2:

break;

default:

std::cout<<"只能输入1或者2哦";

k=0;

break;

}

}while(k==0);

break;

}

case wb:

{

long count=0;

long select;

std::vector<long> groname;

for(auto i=(pWB->groupMana())->cbegin(); i!=(pWB->groupMana())->cend(); ++i)

{

++count;

groname.push\_back((\*i)->groupNum());

}

long j=0;

do {

select=rand()%count;

for(auto i=(pWBD->groNum()).cbegin(); i!=(pWBD->groNum()).cend(); ++i)

if(\*i==groname[select])

j=1;

}while(j);

std::cout<<"推荐你加群"<<select<<std::endl

<<"1.加群 2.不加群";

long k;

do {

std::cin>>k;

switch (k) {

case 1:

for(auto i=(pWB->groupMana())->begin(); i!=(pWB->groupMana())->end(); ++i)

if((\*i)->groupNum()==groname[select])

((\*i)->askJoinWBGroList())->push\_back(pWBD);

break;

case 2:

break;

default:

std::cout<<"只能输入1或者2哦";

k=0;

break;

}

}while(k==0);

break;

}

case wc:

{

long count=0;

long select;

std::vector<long> groname;

for(auto i=(pWC->groupMana())->cbegin(); i!=(pWC->groupMana())->cend(); ++i)

{

++count;

groname.push\_back((\*i)->groupNum());

}

long j=0;

do {

select=rand()%count;

for(auto i=(pWCD->groNum()).cbegin(); i!=(pWCD->groNum()).cend(); ++i)

if(\*i==groname[select])

j=1;

}while(j);

std::cout<<"推荐你加群"<<select<<std::endl

<<"1.加群 2.不加群";

long k;

do {

std::cin>>k;

switch (k) {

case 1:

for(auto i=(pWC->groupMana())->begin(); i!=(pWC->groupMana())->end(); ++i)

if((\*i)->groupNum()==groname[select])

((\*i)->askJoinWCGroList())->push\_back(pWCD);

break;

case 2:

break;

default:

std::cout<<"只能输入1或者2哦";

k=0;

break;

}

}while(k==0);

break;

}

}

break;

}

case 3:

{

std::cout<<"你有这些群:"<<std::endl;

showGroNum();

std::cout<<"你有这些临时讨论组"<<std::endl;

showtempGroNum();

std::cout<<"输入要退出的群号;";

long gron;

std::cin>>gron;

switch (\_t) {

case qq:

exitGro(gron, pQD);

break;

case wb:

exitGro(gron, pWCD);

break;

case wc:

exitGro(gron, pWBD);

break;

}

break;

}

case 4:

{

int judge=1;

std::cout<<"群"<<std::endl;

showGroNum();

std::cout<<"讨论组"<<std::endl;

showtempGroNum();

long num;

std::cin>>num;

inGro(num);

std::cout<<"你可以有下面的操作 1.管理群 2.搜索群成员";

int choose;

do{

std::cin>>choose;

switch (choose) {

case 1:

{

if(!isOwner())

std::cout<<"你不是群主哦";

else

{

std::cout<<"你是群主"<<std::endl;

std::cout<<"管理群: 1.踢人 2.解散群 3.同意网友进群或者拒绝";

long cho;

std::cin>>cho;

do {

switch (cho) {

case 1:

{

searchMem();

std::cout<<"输入你要踢出的人的编号";

long bh;

std::cin>>bh;

Kickmem(bh-1);

break;

}

case 2:

{

disSolvedGro();

break;

}

case 3:

{

addMem();

break;

}

default:

{

std::cout<<"只能输入1～3哦";

cho=0;

break;

}

}

}while(cho==0);

}

break;

}

case 2:

manageSearchMem();

break;

default:

{

std::cout<<"只能输入1或者2哦";

judge=0;

break;

}

}

}while(judge==0);

break;

}

default:

{

std::cout<<"只能输入1～4哦";

i=0;

break;

}

}

}while(i==0);

}

void groupManaImplXZJ::creatGro()

{

std::cout<<"你想创建的群号是多少\_\_"<<std::endl;

long gronum;

int judge;

switch (\_t) {

case qq:

do

{

judge=0;

std::cin>>gronum;

for(auto i=(pQ->groupMana())->cbegin(); i!=(pQ->groupMana())->cend(); ++i)

if((\*i)->groupNum()==gronum)

{

std::cout<<"群号重复";

judge=1;

}

if(judge==0)

{

\_qqGro=std::make\_shared<groupXZJ>(gronum);

(\_qqGro->qqGro())->push\_back(pQD);

(pQD->groNum()).push\_back(gronum);

(pQ->groupMana())->push\_back(\_qqGro);//qq里面增加一个群

}

}while(judge==1);

break;

case wb:

do

{

judge=0;

std::cin>>gronum;

for(auto i=(pWB->groupMana())->cbegin(); i!=(pWB->groupMana())->cend(); ++i)

if((\*i)->groupNum()==gronum)

{

std::cout<<"群号重复";

judge=1;

}

if(judge==0)

{

\_wbGro=std::make\_shared<groupXZJ>(gronum);

(\_wbGro->wbGro())->push\_back(pWBD);

(pWBD->groNum()).push\_back(gronum);

(pWB->groupMana())->push\_back(\_wbGro);//qq里面增加一个群

}

}while(judge==1);

break;

case wc:

do

{

judge=0;

std::cin>>gronum;

for(auto i=(pWC->groupMana())->cbegin(); i!=(pWC->groupMana())->cend(); ++i)

if((\*i)->groupNum()==gronum)

{

std::cout<<"群号重复";

judge=1;

}

if(judge==0)

{

\_wcGro=std::make\_shared<groupXZJ>(gronum);

(\_wcGro->wcGro())->push\_back(pWCD);

(pWCD->groNum()).push\_back(gronum);

(pWC->groupMana())->push\_back(\_wcGro);//qq里面增加一个群

}

}while(judge==1);

break;

}

std::cout<<"\*\*创建成功\*\*"<<std::endl;

}

int groupManaImplXZJ::inGro(long num)

{

long judge=0;

switch (\_t) {

case qq:

{

do

{

if(!(pQD->groNum()).empty())

{

for(auto i=(pQD->groNum()).cbegin(); i!=(pQD->groNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pQ->groupMana())->cbegin(); j!=(pQ->groupMana())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_qqGro=\*j;

return 1;

}

judge=1;

}

for(auto i=(pQD->tempgroNum()).cbegin(); i!=(pQD->tempgroNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pQ->tempdisgro())->cbegin(); j!=(pQ->tempdisgro())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_qqGro=\*j;

return 1;

}

judge=1;

}

if(judge==0)

{

std::cout<<"没有这个群号";

std::cin>>num;

}

}

else

return 0;

}while(judge==0);

break;

}

case wb:

{

do

{

if(!(pWBD->groNum()).empty())

{

for(auto i=(pWBD->groNum()).cbegin(); i!=(pWBD->groNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pWB->groupMana())->cbegin(); j!=(pWB->groupMana())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_wbGro=\*j;

return 1;

}

judge=1;

}

for(auto i=(pWBD->tempgroNum()).cbegin(); i!=(pWBD->tempgroNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pWB->tempdisgro())->cbegin(); j!=(pWB->tempdisgro())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_wbGro=\*j;

return 1;

}

judge=1;

}

if(judge==0)

{

std::cout<<"没有这个群号";

std::cin>>num;

}

}

else

return 0;

}while(judge==0);

break;

}

case wc:

{

do

{

if(!(pWCD->groNum()).empty())

{

for(auto i=(pWCD->groNum()).cbegin(); i!=(pWCD->groNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pWC->groupMana())->cbegin(); j!=(pWC->groupMana())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_wcGro=\*j;

return 1;

}

judge=1;

}

for(auto i=(pWCD->tempgroNum()).cbegin(); i!=(pWCD->tempgroNum()).cend(); ++i)

if(\*i==num)

{

for(auto j=(pWC->tempdisgro())->cbegin(); j!=(pWC->tempdisgro())->cend(); ++j)

if(num==(\*j)->groupNum())

{

\_wcGro=\*j;

return 1;

}

judge=1;

}

if(judge==0)

{

std::cout<<"没有这个群号";

std::cin>>num;

}

}

else

return 0;

}while(judge==0);

break;

}

}

return 0;

}

void groupManaImplXZJ::exitGro(long& i, std::shared\_ptr<QQDataXZJ> pqd)

{

long k;

do {

k=1;

for(auto j=(pqd->groNum()).begin(); j!=(pqd->groNum()).end(); ++j)//从人里面删除组号

if(\*j==i)

{

j=(pQD->groNum()).erase(j);

k=0;

break;

}

if(k==0)

{

for(auto j=(pQ->groupMana())->begin(); j!=(pQ->groupMana())->end(); ++j)//从组里面删除人

if((\*j)->groupNum()==i)

for(auto l=((\*j)->qqGro())->begin(); l!=((\*j)->qqGro())->end(); ++l)

if(\*\*l==\*pqd)

{

l=((\*j)->qqGro())->erase(l);

break;

}

for(auto j=(pQ->tempdisgro())->begin(); j!=(pQ->tempdisgro())->end(); ++j)

if((\*j)->groupNum()==i)

for(auto l=((\*j)->tempqqGro())->begin(); l!=((\*j)->tempqqGro())->end(); ++l)

if(\*\*l==\*pqd)

{

l=((\*j)->tempqqGro())->erase(l);

break;

}

}

if(k)

{

std::cout<<"没有这个群号哦，重新输入叭\_\_";

std::cin>>i;

}

}while(k);

}

void groupManaImplXZJ::exitGro(long& i, std::shared\_ptr<WCDataXZJ> pwcd)

{

long k;

do {

k=1;

for(auto j=(pwcd->groNum()).begin(); j!=(pwcd->groNum()).end(); ++j)//从人里面删除组号

if(\*j==i)

{

j=(pWCD->groNum()).erase(j);

k=0;

break;

}

if(k==0)

{

for(auto j=(pWC->groupMana())->begin(); j!=(pWC->groupMana())->end(); ++j)//从组里面删除人

if((\*j)->groupNum()==i)

for(auto l=((\*j)->wcGro())->begin(); l!=((\*j)->wcGro())->end(); ++l)

if(\*\*l==\*pwcd)

{

l=((\*j)->wcGro())->erase(l);

break;

}

for(auto j=(pWC->tempdisgro())->begin(); j!=(pWC->tempdisgro())->end(); ++j)

if((\*j)->groupNum()==i)

for(auto l=((\*j)->tempwcGro())->begin(); l!=((\*j)->tempwcGro())->end(); ++l)

if(\*\*l==\*pwcd)

{

l=((\*j)->tempwcGro())->erase(l);

break;

}

}

if(k)

{

std::cout<<"没有这个群号哦，重新输入叭\_\_";

std::cin>>i;

}

}while(k);

}

void groupManaImplXZJ::exitGro(long& i, std::shared\_ptr<WBDataXZJ> pwbd)

{

long k;

do {

k=1;

for(auto j=(pwbd->groNum()).begin(); j!=(pwbd->groNum()).end(); ++j)//从人里面删除组号

if(\*j==i)

{

j=(pWBD->groNum()).erase(j);

k=0;

break;

}

if(k==0)

{

for(auto j=(pWB->groupMana())->begin(); j!=(pWB->groupMana())->end(); ++j)//从组里面删除人

if((\*j)->groupNum()==i)

for(auto l=((\*j)->wbGro())->begin(); l!=((\*j)->wbGro())->end(); ++l)

if(\*\*l==\*pwbd)

{

l=((\*j)->wbGro())->erase(l);

break;

}

for(auto j=(pWB->tempdisgro())->begin(); j!=(pWB->tempdisgro())->end(); ++j)

if((\*j)->groupNum()==i)

for(auto l=((\*j)->tempwbGro())->begin(); l!=((\*j)->tempwbGro())->end(); ++l)

if(\*\*l==\*pwbd)

{

l=((\*j)->tempwbGro())->erase(l);

break;

}

}

if(k)

{

std::cout<<"没有这个群号哦，重新输入叭\_\_";

std::cin>>i;

}

}while(k);

}

void groupManaImplXZJ::showAllGroNum()

{

switch (\_t) {

case qq:

for(auto j=(pQ->groupMana())->cbegin(); j!=(pQ->groupMana())->cend(); ++j)

std::cout<<(\*j)->groupNum()<<std::endl;

break;

case wc:

for(auto j=(pWC->groupMana())->cbegin(); j!=(pWC->groupMana())->cend(); ++j)

std::cout<<(\*j)->groupNum()<<std::endl;

break;

case wb:

for(auto j=(pWB->groupMana())->cbegin(); j!=(pWB->groupMana())->cend(); ++j)

std::cout<<(\*j)->groupNum()<<std::endl;

break;

}

}

void groupManaImplXZJ::showGroNum()

{

switch (\_t) {

case qq:

for(auto j=(pQD->groNum()).cbegin(); j!=(pQD->groNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

case wc:

for(auto j=(pWCD->groNum()).cbegin(); j!=(pWCD->groNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

case wb:

for(auto j=(pWBD->groNum()).cbegin(); j!=(pWBD->groNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

}

}

void groupManaImplXZJ::showtempGroNum()

{

switch (\_t) {

case qq:

for(auto j=(pQD->tempgroNum()).cbegin(); j!=(pQD->tempgroNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

case wc:

for(auto j=(pWCD->tempgroNum()).cbegin(); j!=(pWCD->tempgroNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

case wb:

for(auto j=(pWBD->tempgroNum()).cbegin(); j!=(pWBD->tempgroNum()).cend(); ++j)

std::cout<<(\*j)<<std::endl;

break;

}

}

void groupManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<QQDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"QQ号:"<<(p->get())->qqNum()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void groupManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WBDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"微博号:"<<(p->get())->wbNum()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void groupManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WCDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void groupManaImplXZJ::showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<QQDataXZJ>>>& p)const

{

long j=0;

for(auto i=p->cbegin(); i!=p->cend(); ++i)

{

std::cout<<++j<<"."<<std::endl;

std::cout

<<"名字:"<<(\*i)->name()<<std::endl

<<"QQ号:"<<(\*i)->qqNum()<<std::endl

<<"ID:"<<(\*i)->ID()<<std::endl

<<"家庭住址"<<(\*i)->home()<<std::endl

<<"生日:"<<(\*i)->birth()<<std::endl

<<"T龄:"<<(\*i)->t\_age()<<std::endl;

}

}

void groupManaImplXZJ::showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<WCDataXZJ>>>& p)const

{

long j=0;

for(auto i=p->cbegin(); i!=p->cend(); ++i)

{

std::cout<<++j<<"."<<std::endl;

std::cout

<<"名字:"<<(\*i)->name()<<std::endl

<<"微信ID:"<<(\*i)->ID()<<std::endl

<<"家庭住址"<<(\*i)->home()<<std::endl

<<"生日:"<<(\*i)->birth()<<std::endl

<<"T龄:"<<(\*i)->t\_age()<<std::endl;

}

}

void groupManaImplXZJ::showperson(const std::shared\_ptr<std::vector<std::shared\_ptr<WBDataXZJ>>>& p)const

{

long j=0;

for(auto i=p->cbegin(); i!=p->cend(); ++i)

{

std::cout<<++j<<"."<<std::endl;

std::cout

<<"名字:"<<(\*i)->name()<<std::endl

<<"微博号:"<<(\*i)->wbNum()<<std::endl

<<"ID:"<<(\*i)->ID()<<std::endl

<<"家庭住址"<<(\*i)->home()<<std::endl

<<"生日:"<<(\*i)->birth()<<std::endl

<<"T龄:"<<(\*i)->t\_age()<<std::endl;

}

}

void groupManaImplXZJ::searchMem()const

{

long j=0;

switch (\_t) {

case qq:

for(auto i=(\_qqGro->qqGro())->begin(); i!=(\_qqGro->qqGro())->end(); ++i)

{

std::cout<<(\*i)->name();

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

break;

case wb:

for(auto i=(\_wbGro->wbGro())->begin(); i!=(\_wbGro->wbGro())->end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

break;

case wc:

for(auto i=(\_wcGro->wcGro())->begin(); i!=(\_wcGro->wcGro())->end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

break;

}

}

void groupManaImplXZJ::searchMem(const std::string& name)const

{

switch (\_t) {

case qq:

for(auto i=(\_qqGro->qqGro())->begin(); i!=(\_qqGro->qqGro())->end(); ++i)

if((\*i)->name()==name)

{

showperson(i);

break;

}

break;

case wb:

for(auto i=(\_wbGro->wbGro())->begin(); i!=(\_wbGro->wbGro())->end(); ++i)

if((\*i)->name()==name)

{

showperson(i);

break;

}

break;

case wc:

for(auto i=(\_wcGro->wcGro())->begin(); i!=(\_wcGro->wcGro())->end(); ++i)

if((\*i)->name()==name)

{

showperson(i);

break;

}

break;

}

}

void groupManaImplXZJ::searchMem(const long id)const

{

switch (\_t) {

case qq:

for(auto i=(\_qqGro->qqGro())->begin(); i!=(\_qqGro->qqGro())->end(); ++i)

if((\*i)->ID()==id)

{

showperson(i);

break;

}

break;

case wb:

for(auto i=(\_wbGro->wbGro())->begin(); i!=(\_wbGro->wbGro())->end(); ++i)

if((\*i)->ID()==id)

{

showperson(i);

break;

}

break;

case wc:

for(auto i=(\_wcGro->wcGro())->begin(); i!=(\_wcGro->wcGro())->end(); ++i)

if((\*i)->ID()==id)

{

showperson(i);

break;

}

break;

}

}

void groupManaImplXZJ::manageSearchMem()

{

std::cout<<"可以 1.名字搜索 2.ID搜索 3.展示所有的人";

long s, judge=0;

do {

std::cin>>s;

judge=0;

switch (s) {

case 1:

{

std::cout<<"输入名字\_\_";

std::string name;

std::cin>>name;

searchMem(name);

break;

}

case 2:

{

std::cout<<"输入ID\_\_";

long id;

std::cin>>id;

searchMem(id);

break;

}

case 3:

{

searchMem();

break;

}

default:

{

std::cout<<"只能输入1，2，3哦";

std::cin>>s;

judge=1;

break;

}

}

}while(judge);

}

void groupManaImplXZJ::settempdisgro()

{

std::cout<<"设置临时讨论组的组号\_\_";

long gron;

std::cin>>gron;

std::shared\_ptr<groupXZJ> tempGro=std::make\_shared<groupXZJ>(gron);//建立讨论组

searchMem();

std::cout<<"选择你要邀请的人，输入8888结束";

long i;

switch (\_t) {

case qq:

{

do {

std::cin>>i;//push\_back---------std::shared\_ptr<QQDataXZJ>

if(i!=8888)

((tempGro.get())->tempqqGro())->push\_back((\*(\_qqGro->tempqqGro()))[i-1]);

}while(i!=8888);

(pQ->tempdisgro())->push\_back(tempGro);

break;

}

case wb:

{

do {

std::cin>>i;

if(i!=8888)

((tempGro.get())->tempwbGro())->push\_back((\*(\_wbGro->tempwbGro()))[i-1]);

}while(i!=8888);

(pWB->tempdisgro())->push\_back(tempGro);

break;

}

case wc:

{

do {

std::cin>>i;

if(i!=8888)

((tempGro.get())->tempwcGro())->push\_back((\*(\_wcGro->tempwcGro()))[i-1]);

}while(i!=8888);

(pWC->tempdisgro())->push\_back(tempGro);

break;

}

}

}

bool groupManaImplXZJ::isOwner()

{

switch (\_t) {

case qq:

if(\*((\_qqGro->qqGro())->front())==\*pQD)

return true;

else

return false;

break;

case wb:

if(\*((\_wbGro->wbGro())->front())==\*pWBD)

return true;

else

return false;

break;

case wc:

if(\*((\_wcGro->wcGro())->front())==\*pWCD)

return true;

else

return false;

break;

}

}

bool groupManaImplXZJ::isAdministrator()

{

switch (\_t) {

case qq:

{

for(auto i=(\_qqGro->qqGroAdministrators())->cbegin(); i!=(\_qqGro->qqGroAdministrators())->cend(); ++i)

if(\*\*i==\*pQD)

return true;

return false;

break;

}

case wb:

{

for(auto i=(\_wbGro->wbGroAdministrators())->cbegin(); i!=(\_wbGro->wbGroAdministrators())->cend(); ++i)

if(\*\*i==\*pWBD)

return true;

return false;

break;

}

case wc:

{

for(auto i=(\_wcGro->wcGroAdministrators())->cbegin(); i!=(\_wcGro->wcGroAdministrators())->cend(); ++i)

if(\*\*i==\*pWCD)

return true;

return false;

break;

}

}

}

void groupManaImplXZJ::confirmPosition()

{

if(isOwner())

\_mt=Owner;

else if(isAdministrator())

\_mt=administrator;

else

\_mt=normal;

switch (\_mt) {

case Owner:

std::cout<<"你是本群的群主哦";

break;

case administrator:

std::cout<<"你是本群的管理员哦";

break;

case normal:

std::cout<<"你是本群的普通成员哦";

break;

}

}

void groupManaImplXZJ::Kickmem(long number)//number是人在组中的序号

{

switch (\_t) {

case qq:

{

//把人中的组号移除

for(auto i=((\*((\_qqGro->qqGro())->begin()+number))->groNum()).begin(); i!=((\*((\_qqGro->qqGro())->begin()+number))->groNum()).end(); ++i)

if(\*i==\_qqGro->groupNum())

{

((\*((\_qqGro->qqGro())->begin()+number))->groNum()).erase(i);

break;

}

//把人从组中移除

auto i=\_qqGro->qqGro()->begin()+number;

(\_qqGro->qqGro())->erase(i);

break;

}

case wb:

{

//把人中的组号移除

for(auto i=((\*((\_wbGro->wbGro())->begin()+number))->groNum()).begin(); i!=((\*((\_wbGro->wbGro())->begin()+number))->groNum()).end(); ++i)

if(\*i==\_wbGro->groupNum())

{

((\*((\_wbGro->wbGro())->begin()+number))->groNum()).erase(i);

break;

}

//把人从组中移除

auto i=\_wbGro->wbGro()->begin()+number;

(\_wbGro->wbGro())->erase(i);

break;

}

case wc:

{

//把人中的组号移除

for(auto i=((\*((\_wcGro->wcGro())->begin()+number))->groNum()).begin(); i!=((\*((\_wcGro->wcGro())->begin()+number))->groNum()).end(); ++i)

if(\*i==\_wcGro->groupNum())

{

((\*((\_wcGro->wcGro())->begin()+number))->groNum()).erase(i);

break;

}

//把人从组中移除

auto i=\_wcGro->wcGro()->begin()+number;

(\_wcGro->wcGro())->erase(i);

break;

}

}

}

void groupManaImplXZJ::disSolvedGro()

{

switch (\_t) {

case qq:

{

//先从人中删除组号,然后从容器末尾向前删除成员

for(auto i=(\_qqGro->qqGro())->rbegin();i!=(\_qqGro->qqGro())->rend(); ++i)

{

for(auto j=((\*i)->groNum()).begin(); j!=((\*i)->groNum()).end(); ++j)

{

if(\*j==\_qqGro->groupNum())

{

((\*i)->groNum()).erase(j);//先从人中删除组号

break;

}

}

}

while((\_qqGro->qqGro())->size()!=0)

(\_qqGro->qqGro())->pop\_back();

break;

}

case wb:

{

for(auto i=(\_wbGro->wbGro())->rbegin();i!=(\_wbGro->wbGro())->rend(); ++i)

{

for(auto j=((\*i)->groNum()).begin(); j!=((\*i)->groNum()).end(); ++j)

{

if(\*j==\_wbGro->groupNum())

{

((\*i)->groNum()).erase(j);//先从人中删除组号

break;

}

}

}

while((\_wbGro->wbGro())->size()!=0)

(\_wbGro->wbGro())->pop\_back();

break;

}

case wc:

{

for(auto i=(\_wcGro->wcGro())->rbegin();i!=(\_wcGro->wcGro())->rend(); ++i)

{

for(auto j=((\*i)->groNum()).begin(); j!=((\*i)->groNum()).end(); ++j)

{

if(\*j==\_wcGro->groupNum())

{

((\*i)->groNum()).erase(j);//先从人中删除组号

break;

}

}

}

while((\_wcGro->wcGro())->size()!=0)

(\_wcGro->wcGro())->pop\_back();

break;

}

}

}

void groupManaImplXZJ::addMem()

{

switch (\_t) {

case qq:

{

std::cout<<"有这些人想要加入你管理的群"<<std::endl;

showperson(\_qqGro->askJoinQQGroList());

std::cout<<"输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

long xh;

std::cin>>xh;

while(xh!=-1||!(\_qqGro->askJoinQQGroList())->empty())

{

xh-=1;

(\_qqGro->qqGro())->push\_back(\*((\_qqGro->askJoinQQGroList())->begin()+xh));

auto i=(\_qqGro->askJoinQQGroList())->begin()+xh;

(\_qqGro->askJoinQQGroList())->erase(i);

if((\_qqGro->askJoinQQGroList())->empty())

break;

else

{

showperson(\_qqGro->askJoinQQGroList());

std::cout<<"继续输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

std::cin>>xh;

}

}

break;

}

case wb:

{

std::cout<<"有这些人想要加入你管理的群";

showperson(\_wbGro->askJoinWBGroList());

std::cout<<"输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

long xh;

std::cin>>xh;

while(xh!=-1||!(\_wbGro->askJoinWBGroList())->empty())

{

xh-=1;

(\_wbGro->wbGro())->push\_back(\*((\_wbGro->askJoinWBGroList())->begin()+xh));

auto i=(\_wbGro->askJoinWBGroList())->begin()+xh;

(\_wbGro->askJoinWBGroList())->erase(i);

if((\_wbGro->askJoinWBGroList())->empty())

break;

else

{

showperson(\_wbGro->askJoinWBGroList());

std::cout<<"继续输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

std::cin>>xh;

}

}

break;

}

case wc:

{

std::cout<<"有这些人想要加入你管理的群";

showperson(\_wcGro->askJoinWCGroList());

std::cout<<"输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

long xh;

std::cin>>xh;

while(xh!=-1||!(\_wcGro->askJoinWCGroList())->empty())

{

xh-=1;

(\_wcGro->wcGro())->push\_back(\*((\_wcGro->askJoinWCGroList())->begin()+xh));

auto i=(\_wcGro->askJoinWCGroList())->begin()+xh;

(\_wcGro->askJoinWCGroList())->erase(i);

if((\_wcGro->askJoinWCGroList())->empty())

break;

else

{

showperson(\_wcGro->askJoinWCGroList());

std::cout<<"继续输入你同意加入的人的序号,输入-1结束"<<std::endl<<std::endl<<std::endl;

std::cin>>xh;

}

}

break;

}

}

}

void groupManaImplXZJ::setAdministrator(long number)

{

switch (\_t) {

case qq:

{

std::cout<<"你的群里有这些人:"<<std::endl;

showperson(\_qqGro->qqGro());

std::cout<<"输入你想设置为管理员的号码";

long hm;

std::cin>>hm;

(\_qqGro->qqGroAdministrators())->push\_back(\*((\_qqGro->qqGro())->begin()+hm-1));

break;

}

case wb:

{

std::cout<<"你的群里有这些人:"<<std::endl;

showperson(\_wbGro->wbGro());

std::cout<<"输入你想设置为管理员的号码";

long hm;

std::cin>>hm;

(\_wbGro->wbGroAdministrators())->push\_back(\*((\_wbGro->wbGro())->begin()+hm-1));

break;

}

case wc:

{

std::cout<<"你的群里有这些人:"<<std::endl;

showperson(\_wcGro->wcGro());

std::cout<<"输入你想设置为管理员的号码";

long hm;

std::cin>>hm;

(\_wcGro->wcGroAdministrators())->push\_back(\*((\_wcGro->wcGro())->begin()+hm-1));

break;

}

}

}

**friendManaXZJ.hpp**

#ifndef friendManaXZJ\_hpp

#define friendManaXZJ\_hpp

#include <iostream>

class friendManaImplXZJ;

class QQDataXZJ;

class WBDataXZJ;

class WCDataXZJ;

class Files;

class friendManaXZJ

{

public:

explicit friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& pQD);

explicit friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& pWBD);

explicit friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& pWCD);

void addFriend();

void addFriend(std::shared\_ptr<QQDataXZJ> p);

void addFriend(std::shared\_ptr<WCDataXZJ> p);

void addFriend(std::shared\_ptr<WBDataXZJ> p);

void deleteFriend(std::shared\_ptr<QQDataXZJ> p);

void deleteFriend(std::shared\_ptr<WCDataXZJ> p);

void deleteFriend(std::shared\_ptr<WBDataXZJ> p);

void searchFriend()const;//直接展示所有好友

void searchFriend(const std::string& name)const;//查询好友，名字查询

void searchFriend(const long id)const;//ID查询

void recoQQFriend();//推荐所有好友

void recoWBFriend();

void recoWCFriend();

void agreeFriend();

~friendManaXZJ(){}

private:

std::shared\_ptr<friendManaImplXZJ> pfMI=nullptr;

};

#endif

**friendManaXZJ.cpp**

#include "friendManaXZJ.hpp"

#include "friendManaImplXZJ.hpp"

friendManaXZJ::friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& pQD)

{

pfMI=(std::make\_shared<friendManaImplXZJ>(\_pF, pQD));

}

friendManaXZJ::friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& pWBD)

{

pfMI=(std::make\_shared<friendManaImplXZJ>(\_pF, pWBD));

}

friendManaXZJ::friendManaXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& pWCD)

{

pfMI=(std::make\_shared<friendManaImplXZJ>(\_pF, pWCD));

}

void friendManaXZJ::addFriend()

{

pfMI->addFriend();

}

void friendManaXZJ::addFriend(std::shared\_ptr<QQDataXZJ> p)

{

pfMI->addFriend(p);

}

void friendManaXZJ::addFriend(std::shared\_ptr<WCDataXZJ> p)

{

pfMI->addFriend(p);

}

void friendManaXZJ::addFriend(std::shared\_ptr<WBDataXZJ> p)

{

pfMI->addFriend(p);

}

void friendManaXZJ::deleteFriend(std::shared\_ptr<QQDataXZJ> p)

{

pfMI->deleteFriend(p);

}

void friendManaXZJ::deleteFriend(std::shared\_ptr<WCDataXZJ> p)

{

pfMI->deleteFriend(p);

}

void friendManaXZJ::deleteFriend(std::shared\_ptr<WBDataXZJ> p)

{

pfMI->deleteFriend(p);

}

void friendManaXZJ::searchFriend()const//直接展示所有好友

{

pfMI->searchFriend();

}

void friendManaXZJ::searchFriend(const std::string& name)const//查询好友，名字查询

{

pfMI->searchFriend(name);

}

void friendManaXZJ::searchFriend(const long id)const//ID查询

{

pfMI->searchFriend(id);

}

void friendManaXZJ::recoQQFriend()//推荐所有好友

{

pfMI->recoQQFriend();

}

void friendManaXZJ::recoWBFriend()

{

pfMI->recoWBFriend();

}

void friendManaXZJ::recoWCFriend()

{

pfMI->recoWCFriend();

}

void friendManaXZJ::agreeFriend()

{

pfMI->agreeFriend();

}

**friendManaImplXZJ.hpp**

#ifndef friendManaImplXZJ\_hpp

#define friendManaImplXZJ\_hpp

#include <iostream>

#include "Files.hpp"

#include "BasicDataXZJ.hpp"

#include "QQDataXZJ.hpp"//修改，删除，查询

#include "WBDataXZJ.hpp"

#include "WCDataXZJ.hpp"

#include "logXZJ.hpp"//需要自动登录所有微x

#include "Files.hpp"

enum type{qq,wc,wb};

class friendManaImplXZJ

{

public:

explicit friendManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<QQDataXZJ>& pqd)

{

pF=(\_pF);

pl=(std::make\_shared<logXZJ>(\_pF));

pQ=(pF->QQ());

pWC=(pF->WC());

pWB=(pF->WB());

pQD=(pl->nopasswordlogQQ(pqd->qqNum()));//有问题

pWBD=(pl->pWBD());

pWCD=(pl->pWCD());

uzing=(qq);

}

explicit friendManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WBDataXZJ>& pwb)

{

pF=(\_pF);

pl=(std::make\_shared<logXZJ>(\_pF));

pQ=(pF->QQ());

pWC=(pF->WC());

pWB=(pF->WB());

pWBD=(pl->nopasswordlogWB(pwb->wbNum()));

pQD=(pl->pQD());

pWCD=(pl->pWCD());

uzing=(wb);

}

explicit friendManaImplXZJ(const std::shared\_ptr<Files>& \_pF, const std::shared\_ptr<WCDataXZJ>& pwc)

{

pF=(\_pF);

pl=(std::make\_shared<logXZJ>(\_pF));

pQ=(pF->QQ());

pWC=(pF->WC());

pWB=(pF->WB());

pWCD=(pl->nopasswordlogWC(pwc->ID()));

pWBD=(pl->pWBD());

pQD=(pl->pQD());

uzing=(wc);

}

void addFriend();

void addFriend(std::shared\_ptr<QQDataXZJ> p);

void addFriend(std::shared\_ptr<WCDataXZJ> p);

void addFriend(std::shared\_ptr<WBDataXZJ> p);

void deleteFriend(std::shared\_ptr<QQDataXZJ> p);

void deleteFriend(std::shared\_ptr<WCDataXZJ> p);

void deleteFriend(std::shared\_ptr<WBDataXZJ> p);

void searchFriend()const;//直接展示所有好友

void searchFriend(const std::string& name)const;//查询好友，名字查询

void searchFriend(const long id)const;//ID查询

void recoQQFriend();//登录除QQ外其他app

void recoWBFriend();

void recoWCFriend();

void agreeFriend();

~friendManaImplXZJ(){}

private:

type uzing;//正在登录使用的微x

std::shared\_ptr<logXZJ> pl=nullptr;

std::shared\_ptr<Files> pF=nullptr;

std::shared\_ptr<QQDataXZJ> pQD=nullptr;

std::shared\_ptr<WBDataXZJ> pWBD=nullptr;

std::shared\_ptr<WCDataXZJ> pWCD=nullptr;

std::shared\_ptr<QQXZJ> pQ=nullptr;

std::shared\_ptr<WBXZJ> pWB=nullptr;

std::shared\_ptr<WCXZJ> pWC=nullptr;

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<QQDataXZJ> \*>& p)const;

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WBDataXZJ> \*>& p)const;

void showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WCDataXZJ> \*>& p)const;

};

//只有被声明为const的成员函数才能被一个const类对象调用

#endif

**friendManaImplXZJ.cpp**

#include "friendManaImplXZJ.hpp"

void friendManaImplXZJ::addFriend(std::shared\_ptr<QQDataXZJ> p)

{

(p->askfriendList()).push\_back(std::make\_shared<QQDataXZJ>(\*pQD.get()));

}

void friendManaImplXZJ::addFriend(std::shared\_ptr<WCDataXZJ> p)

{

(p->askfriendList()).push\_back(std::make\_shared<WCDataXZJ>(\*pWCD.get()));

}

void friendManaImplXZJ::addFriend(std::shared\_ptr<WBDataXZJ> p)

{

(p->askfriendList()).push\_back(std::make\_shared<WBDataXZJ>(\*pWBD.get()));

}

void friendManaImplXZJ::deleteFriend(std::shared\_ptr<QQDataXZJ> p)

{

for(auto j=(p->friendList()).begin(); j!=(p->friendList()).end(); ++j)//从对方把自己删除

if(\*j==pQD)

{

std::cout<<(\*j)->name();

j=(p->friendList()).erase(j);

--j;

}

for(auto i=(pQD->friendList()).begin(); i!=(pQD->friendList()).end(); ++i)//从自己把对方删除

if(\*(\*i)==\*p)

{

i=(pQD->friendList()).erase(i);

--i;

}

}

void friendManaImplXZJ::deleteFriend(std::shared\_ptr<WCDataXZJ> p)

{

for(auto j=(p->friendList()).begin(); j!=(p->friendList()).end(); ++j)//从对方把自己删除

if(\*j==pWCD)

{

j=(p->friendList()).erase(j);

--j;

}

for(auto i=(pWCD->friendList()).begin(); i!=(pWCD->friendList()).end(); ++i)//从自己把对方删除

if(\*(\*i)==\*p)

{

i=(pWCD->friendList()).erase(i);

--i;

}

}

void friendManaImplXZJ::deleteFriend(std::shared\_ptr<WBDataXZJ> p)

{

for(auto j=(p->friendList()).begin(); j!=(p->friendList()).end(); ++j)//从对方把自己删除

if(\*j==pWBD)

{

j=(p->friendList()).erase(j);

--j;

}

for(auto i=(pWBD->friendList()).begin(); i!=(pWBD->friendList()).end(); ++i)//从自己把对方删除

if(\*(\*i)==\*p)

{

i=(pWBD->friendList()).erase(i);

--i;

}

}

void friendManaImplXZJ::searchFriend()const//直接展示所有好友

{

long j=0;

switch (uzing) {

case qq:

for(auto i=(pQD->friendList()).begin(); i!=(pQD->friendList()).end(); ++i)

{

std::cout<<++j;

showperson(i);

}

break;

case wb:

for(auto i=(pWBD->friendList()).begin(); i!=(pWBD->friendList()).end(); ++i)

{

std::cout<<++j;

showperson(i);

}

break;

case wc:

for(auto i=(pWCD->friendList()).begin(); i!=(pWCD->friendList()).end(); ++i)

{

std::cout<<++j;

showperson(i);

}

break;

}

}

void friendManaImplXZJ::searchFriend(const std::string& name)const//查询好友，名字查询

{

switch (uzing) {

case qq:

for(auto i=(pQD->friendList()).begin(); i!=(pQD->friendList()).end(); ++i)

if((\*i)->name()==name)

showperson(i);

break;

case wb:

for(auto i=(pWBD->friendList()).begin(); i!=(pWBD->friendList()).end(); ++i)

if((\*i)->name()==name)

showperson(i);

break;

case wc:

for(auto i=(pWCD->friendList()).begin(); i!=(pWCD->friendList()).end(); ++i)

if((\*i)->name()==name)

showperson(i);

break;

}

}

void friendManaImplXZJ::searchFriend(const long id)const//ID查询

{

switch (uzing) {

case qq:

for(auto i=(pQD->friendList()).begin(); i!=(pQD->friendList()).end(); ++i)

if((\*i)->ID()==id)

showperson(i);

break;

case wb:

for(auto i=(pWBD->friendList()).begin(); i!=(pWBD->friendList()).end(); ++i)

if((\*i)->ID()==id)

showperson(i);

break;

case wc:

for(auto i=(pWCD->friendList()).begin(); i!=(pWCD->friendList()).end(); ++i)

if((\*i)->ID()==id)

showperson(i);

break;

}

}

void friendManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<QQDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"QQ号:"<<(p->get())->qqNum()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void friendManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WBDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"微博号:"<<(p->get())->wbNum()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void friendManaImplXZJ::showperson(const std::\_\_wrap\_iter<std::shared\_ptr<WCDataXZJ> \*>& p)const

{

std::cout

<<"名字:"<<(p->get())->name()<<std::endl

<<"ID:"<<(p->get())->ID()<<std::endl

<<"家庭住址"<<(p->get())->home()<<std::endl

<<"生日:"<<(p->get())->birth()<<std::endl

<<"T龄:"<<(p->get())->t\_age()<<std::endl;

}

void friendManaImplXZJ::recoQQFriend()

{

int judge=1;

switch (uzing) {

case wc:

{

if(!(pWCD->friendList()).empty())

{

for(auto i=(pWCD->friendList()).cbegin(); i!=(pWCD->friendList()).cend(); ++i)

if((\*i)->qq()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的QQ号是"<<(\*i)->qq()<<"(ID:"<<(\*i)->ID()<<")"<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWCD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的微信好友还没有绑定QQ";

}

else

std::cout<<"您的微信还没有好友"<<std::endl;

break;

}

case wb:

{

if(!(pWBD->friendList()).empty())

{

for(auto i=(pWBD->friendList()).cbegin(); i!=(pWBD->friendList()).cend(); ++i)

if((\*i)->qq()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的QQ号是"<<(\*i)->qq()<<"(ID:"<<(\*i)->ID()<<")"<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWCD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的微博好友还没有绑定QQ";

}

else

std::cout<<"您的微博还没有好友"<<std::endl;

break;

}

case qq:

std::cout<<"您当前登录的就是QQ"<<std::endl;

break;

}

}

void friendManaImplXZJ::recoWBFriend()

{

int judge=1;

switch (uzing) {

case qq:

{

if(!(pQD->friendList()).empty())

{

for(auto i=(pQD->friendList()).cbegin(); i!=(pQD->friendList()).cend(); ++i)

if((\*i)->wb()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的微博号是"<<(\*i)->wb()<<"(ID:"<<(\*i)->ID()<<")"<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pQD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的QQ好友还没有绑定微博";

}

else

std::cout<<"您的QQ还没有好友"<<std::endl;

break;

}

case wc:

{

if(!(pWCD->friendList()).empty())

{

for(auto i=(pWCD->friendList()).cbegin(); i!=(pWCD->friendList()).cend(); ++i)

if((\*i)->wb()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的微博号是"<<(\*i)->wb()<<"(ID:"<<(\*i)->ID()<<")"<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWCD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的微信好友还没有绑定微博";

}

else

std::cout<<"您的微信还没有好友"<<std::endl;

break;

}

case wb:

std::cout<<"您当前登录的就是微博"<<std::endl;

break;

}

}

void friendManaImplXZJ::recoWCFriend()

{

int judge=1;

switch (uzing) {

case qq:

{

if(!(pQD->friendList()).empty())

{

for(auto i=(pQD->friendList()).cbegin(); i!=(pQD->friendList()).cend(); ++i)

if((\*i)->wc()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的微微信ID是"<<(\*i)->ID()<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pQD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的QQ好友还没有绑定微信";

}

else

std::cout<<"您的QQ还没有好友"<<std::endl;

break;

}

case wb:

{

if(!(pWBD->friendList()).empty())

{

for(auto i=(pWBD->friendList()).cbegin(); i!=(pWBD->friendList()).cend(); ++i)

if((\*i)->wc()!=0)

{

judge=0;

std::cout<<"你的好友"<<(\*i)->name()<<"(ID:"<<(\*i)->ID()<<")"

<<"的微微信ID是"<<(\*i)->ID()<<std::endl<<std::endl;

}

if(judge==0)

{

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWBD->friendList()).begin()+i-1));

break;

}

}

}

else

std::cout<<"您的微博好友还没有绑定微信";

}

else

std::cout<<"您的微博还没有好友"<<std::endl;

break;

}

case wc:

std::cout<<"您当前登录的就是微信"<<std::endl;

}

}

void friendManaImplXZJ::addFriend()

{

int j=0;

switch (uzing) {

case qq:

{

for(auto i=(pQ->people())->begin(); i!=(pQ->people())->end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pQ->people())->begin()+i-1));

break;

}

}

break;

}

case wc:

{

for(auto i=(pWC->people())->begin(); i!=(pWC->people())->end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWC->people())->begin()+i-1));

break;

}

}

break;

}

case wb:

{

for(auto i=(pWB->people())->begin(); i!=(pWB->people())->end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"是否添加为好友，如果是添加的话输入序号，如果不添加输入-1";

long i;

std::cin>>i;

switch (i) {

case -1:

break;

default:

{

addFriend(\*((pWB->people())->begin()+i-1));

break;

}

}

break;

}

}

}

void friendManaImplXZJ::agreeFriend()

{

long j=0;

switch (uzing) {

case qq:

{

if(!(pQD->askfriendList()).empty())

{

for(auto i=(pQD->askfriendList()).begin(); i!=(pQD->askfriendList()).end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"输入同意的序号\_\_";

long k;

std::cin>>k;

auto i=(pQD->askfriendList()).begin();

(pQD->friendList()).push\_back(\*(i+k-1));//自己好友中加他

((\*(i+k-1))->friendList()).push\_back(pQD);//他的好友中加你

(pQD->askfriendList()).erase(i+k-1);

}

else

std::cout<<"没有好友要添加你"<<std::endl;

break;

}

case wb:

{

if(!(pWBD->askfriendList()).empty())

{

for(auto i=(pWBD->askfriendList()).begin(); i!=(pWBD->askfriendList()).end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"输入同意的序号\_\_";

long k;

std::cin>>k;

auto i=(pWBD->askfriendList()).begin();

(pWBD->friendList()).push\_back(\*(i+k-1));//自己好友中加他

((\*(i+k-1))->friendList()).push\_back(pWBD);//他的好友中加你

(pWBD->askfriendList()).erase(i+k-1);

}

else

std::cout<<"没有好友要添加你"<<std::endl;

break;

}

case wc:

{

if(!(pWCD->askfriendList()).empty())

{

for(auto i=(pWCD->askfriendList()).begin(); i!=(pWCD->askfriendList()).end(); ++i)

{

std::cout<<++j<<"."<<std::endl;

showperson(i);

}

std::cout<<"输入同意的序号\_\_";

long k;

std::cin>>k;

auto i=(pWCD->askfriendList()).begin();

(pWCD->friendList()).push\_back(\*(i+k-1));//自己好友中加他

((\*(i+k-1))->friendList()).push\_back(pWCD);//他的好友中加你

(pWCD->askfriendList()).erase(i+k-1);

}

else

std::cout<<"没有好友要添加你"<<std::endl;

break;

}

}

}

**Files.hpp**

#ifndef Files\_hpp

#define Files\_hpp

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include "QQXZJ.hpp"

#include "WBXZJ.hpp"

#include "WCXZJ.hpp"

//一个人一个文件(QQ+ID号为文件名)或者(WC+ID号为文件名)或者(WB+ID号为文件名)

//18个群文件,18个临时讨论组文件,存储群号和每个人的ID号

//外加QQ,WC,WB三个文件存储 所有人的ID号(用来识别人的),临时讨论组号,群号

class Files

{

public:

Files();

//save时清空文件，重新写入

void Save(const std::shared\_ptr<QQXZJ>& save);//保存QQ信息，顺便保存群的信息

void Save(const std::shared\_ptr<WBXZJ>& save);//保存微博信息，顺便保存群的信息

void Save(const std::shared\_ptr<WCXZJ>& save);//保存微信信息，顺便保存群的信息

const std::shared\_ptr<QQXZJ>& QQ(){return \_pQ;}

const std::shared\_ptr<WBXZJ>& WB(){return \_pWB;}

const std::shared\_ptr<WCXZJ>& WC(){return \_pWC;}

~Files(){}

private:

void Save(const std::shared\_ptr<QQDataXZJ>& save);//保存一个人的信息

void Save(const std::shared\_ptr<WBDataXZJ>& save);//保存一个人的信息

void Save(const std::shared\_ptr<WCDataXZJ>& save);//保存一个人的信息

void SaveQQGro(const std::shared\_ptr<groupXZJ>& save);

void SaveWBGro(const std::shared\_ptr<groupXZJ>& save);

void SaveWCGro(const std::shared\_ptr<groupXZJ>& save);

void SaveQQtempGro(const std::shared\_ptr<groupXZJ>& save);

void SaveWBtempGro(const std::shared\_ptr<groupXZJ>& save);

void SaveWCtempGro(const std::shared\_ptr<groupXZJ>& save);

void OutFileQQ();

void OutFileWC();

void OutFileWB();

std::ifstream outfile, qq, wc, wb;

std::ofstream infile, ifqq, ifwc, ifwb, ifperson;

std::istringstream instrstream;

std::ostringstream outstrstream;

std::shared\_ptr<QQXZJ> \_pQ=nullptr;

std::shared\_ptr<WBXZJ> \_pWB=nullptr;

std::shared\_ptr<WCXZJ> \_pWC=nullptr;

};

#endif

**Files.cpp**

#include "Files.hpp"

Files::Files()

{

OutFileQQ();

OutFileWB();

OutFileWC();

}

void Files::Save(const std::shared\_ptr<QQDataXZJ>& save)

{

std::stringstream ss;

ss<<save->ID();

std::string filename="QQ"+ss.str()+".txt";

try {

ifperson.open(filename);

if(!ifperson.is\_open())

throw 1;

ifperson<<save->ID()<<std::endl

<<save->qqNum()<<std::endl

<<save->password()<<std::endl

<<save->name()<<std::endl

<<save->home()<<std::endl;

if(!(save->groNum()).empty())

for(auto i=(save->groNum()).cbegin(); i!=(save->groNum()).cend(); ++i)

{

ifperson<<\*i<<" ";

}

else

ifperson<<"0";

ifperson<<std::endl;

if(!(save->tempgroNum()).empty())

for(auto i=(save->tempgroNum()).cbegin(); i!=(save->tempgroNum()).cend(); ++i)

{

ifperson<<\*i<<" ";

}

else

ifperson<<"0";

ifperson<<std::endl;

ifperson<<(save->birth()).year()<<" "<<(save->birth()).month()<<" "<<(save->birth()).day()<<std::endl

<<(save->t\_age()).year()<<" "<<(save->t\_age()).month()<<" "<<(save->t\_age()).day()<<" "<<std::endl

<<save->wc()<<std::endl

<<save->wb()<<std::endl;

// <<"好友们的ID是:";

if(!(save->friendList()).empty())

for(auto i=(save->friendList()).cbegin(); i!=(save->friendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

// ifperosn<<"要添加你为好友的用户的ID是:";

if(!(save->askfriendList()).empty())

for(auto i=(save->askfriendList()).cbegin(); i!=(save->askfriendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson.close();

} catch (...) {

std::cout<<"打开文件失败";

}

}

void Files::Save(const std::shared\_ptr<WBDataXZJ>& save)

{

std::stringstream ss;

ss<<save->ID();

std::string filename="WB"+ss.str()+".txt";

try {

ifperson.open(filename);//隐含截断文件

if(!ifperson.is\_open())

throw 1;

ifperson<<save->ID()<<std::endl

<<save->wbNum()<<std::endl

<<save->password()<<std::endl

<<save->name()<<std::endl

<<save->home()<<std::endl;

// <<"加入的群号:";

if(!(save->groNum()).empty())

for(auto i=(save->groNum()).cbegin(); i!=(save->groNum()).cend(); ++i)

ifperson<<\*i<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

if(!(save->tempgroNum()).empty())

for(auto i=(save->tempgroNum()).cbegin(); i!=(save->tempgroNum()).cend(); ++i)

ifperson<<\*i<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

ifperson<<(save->birth()).year()<<" "<<(save->birth()).month()<<" "<<(save->birth()).day()<<" "<<std::endl

<<(save->t\_age()).year()<<" "<<(save->t\_age()).month()<<" "<<(save->t\_age()).day()<<" "<<std::endl

<<save->wc()<<std::endl

<<save->qq()<<std::endl;

// <<"好友们的ID是:";

if(!(save->friendList()).empty())

for(auto i=(save->friendList()).cbegin(); i!=(save->friendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

// ifperson<<"要添加你为好友的用户的ID是:";

if(!(save->askfriendList()).empty())

for(auto i=(save->askfriendList()).cbegin(); i!=(save->askfriendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::Save(const std::shared\_ptr<WCDataXZJ>& save)

{

std::stringstream ss;

ss<<save->ID();

std::string filename="WC"+ss.str()+".txt";

try {

ifperson.open(filename);//隐含截断文件

if(!ifperson.is\_open())

throw 1;

ifperson<<save->ID()<<std::endl

<<save->password()<<std::endl

<<save->name()<<std::endl

<<save->home()<<std::endl;

// <<"加入的群号:";

if(!(save->groNum()).empty())

for(auto i=(save->groNum()).cbegin(); i!=(save->groNum()).cend(); ++i)

ifperson<<\*i<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

if(!(save->tempgroNum()).empty())

for(auto i=(save->tempgroNum()).cbegin(); i!=(save->tempgroNum()).cend(); ++i)

ifperson<<\*i<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

ifperson<<(save->birth()).year()<<" "<<(save->birth()).month()<<" "<<(save->birth()).day()<<" "<<std::endl

<<(save->t\_age()).year()<<" "<<(save->t\_age()).month()<<" "<<(save->t\_age()).day()<<" "<<std::endl

<<save->qq()<<std::endl

<<save->wb()<<std::endl;

// <<"好友们的ID是:";

if(!(save->friendList()).empty())

for(auto i=(save->friendList()).cbegin(); i!=(save->friendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson<<std::endl;

// ifperson<<"要添加你为好友的用户的ID是:";

if(!(save->askfriendList()).empty())

for(auto i=(save->askfriendList()).cbegin(); i!=(save->askfriendList()).cend(); ++i)

ifperson<<(\*i)->ID()<<" ";

else

ifperson<<"0";

ifperson.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::SaveQQGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("QQGro"+ss.str()+".txt");

try {

ifqq.open(filename);

if(!ifqq.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->qqGro())->empty())

for(auto i=(save->qqGro())->cbegin(); i!=(save->qqGro())->cend(); ++i)

ifqq<<(\*i)->ID()<<" ";

ifqq<<std::endl;

if(!(save->askJoinQQGroList())->empty())

for(auto i=(save->askJoinQQGroList())->cbegin(); i!=(save->askJoinQQGroList())->cend(); ++i)

{

ifqq<<(\*i)->ID()<<" ";

}

ifqq.close();

} catch (...) {

std::cout<<"打开文件失败";

}

}

void Files::SaveWBGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("WBGro"+ss.str()+".txt");

try {

ifwb.open(filename);

if(!ifwb.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->wbGro())->empty())

for(auto i=(save->wbGro())->cbegin(); i!=(save->wbGro())->cend(); ++i)

ifwb<<(\*i)->ID()<<" ";

ifwb<<std::endl;

if(!(save->askJoinWBGroList())->empty())

for(auto i=(save->askJoinWBGroList())->cbegin(); i!=(save->askJoinWBGroList())->cend(); ++i)

ifwb<<(\*i)->ID()<<" ";

ifwb.close();

} catch (...) {

std::cout<<"打开文件失败";

}

}

void Files::SaveWCGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("WCGro"+ss.str()+".txt");

try {

ifwc.open(filename);

if(!ifwc.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->wcGro())->empty())

for(auto i=(save->wcGro())->cbegin(); i!=(save->wcGro())->cend(); ++i)

ifwc<<(\*i)->ID()<<" ";

ifwc<<std::endl;

if(!(save->askJoinWCGroList())->empty())

for(auto i=(save->askJoinWCGroList())->cbegin(); i!=(save->askJoinWCGroList())->cend(); ++i)

ifwc<<(\*i)->ID()<<" ";

ifwc.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::SaveQQtempGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("QQtempGro"+ss.str()+".txt");

try {

ifqq.open(filename);

if(!ifqq.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->tempqqGro())->empty())

for(auto i=(save->tempqqGro())->cbegin(); i!=(save->tempqqGro())->cend(); ++i)

ifqq<<(\*i)->ID()<<" ";

ifqq<<std::endl;

if(!(save->askJointempQQGroList())->empty())

for(auto i=(save->askJointempQQGroList())->cbegin(); i!=(save->askJointempQQGroList())->cend(); ++i)

ifqq<<(\*i)->ID()<<" ";

ifqq.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::SaveWBtempGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("WBtempGro"+ss.str()+".txt");

try {

ifwb.open(filename);

if(!ifwb.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->tempwbGro())->empty())

for(auto i=(save->tempwbGro())->cbegin(); i!=(save->tempwbGro())->cend(); ++i)

ifwb<<(\*i)->ID()<<" ";

ifwb<<std::endl;

if(!(save->askJointempWBGroList())->empty())

for(auto i=(save->askJointempWBGroList())->cbegin(); i!=(save->askJointempWBGroList())->cend(); ++i)

ifwb<<(\*i)->ID()<<" ";

ifwb.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::SaveWCtempGro(const std::shared\_ptr<groupXZJ>& save)

{

std::stringstream ss;

ss<<save->groupNum();

std::string filename("WCtempGro"+ss.str()+".txt");

try {

ifwc.open(filename);

if(!ifwc.is\_open())

throw 1;

// <<"群中成员们的ID号是:";

if(!(save->tempwcGro())->empty())

for(auto i=(save->tempwcGro())->cbegin(); i!=(save->tempwcGro())->cend(); ++i)

ifwc<<(\*i)->ID()<<" ";

ifwc<<std::endl;

if(!(save->askJointempWCGroList())->empty())

for(auto i=(save->askJointempWCGroList())->cbegin(); i!=(save->askJointempWCGroList())->cend(); ++i)

ifwc<<(\*i)->ID()<<" ";

ifwc.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::Save(const std::shared\_ptr<QQXZJ>& save)

{

try{

infile.open("QQ.txt");

if(!infile.is\_open())

throw 1;

// infile<<"所有人的ID:";

if(!(save->people())->empty())

for(auto i=(save->people())->cbegin(); i!=(save->people())->cend(); ++i)

{

Save(\*i);

infile<<(\*i)->ID()<<" ";

}

infile<<std::endl;

for(auto i=(save->groupMana())->cbegin(); i!=(save->groupMana())->cend(); ++i)

{

SaveQQGro(\*i);

infile<<(\*i)->groupNum()<<" ";

}

infile<<std::endl;

for(auto i=(save->tempdisgro())->cbegin(); i!=(save->tempdisgro())->cend(); ++i)

{

SaveQQtempGro(\*i);

infile<<(\*i)->groupNum()<<" ";

}

}

catch(long a){

std::cout<<"打开文件失败";

}

}

void Files::Save(const std::shared\_ptr<WBXZJ>& save)

{

try{

infile.open("WB.txt");

if(!infile.is\_open())

throw 1;

// infile<<"所有人的ID:";

if(!(save->people())->empty())

for(auto i=(save->people())->cbegin(); i!=(save->people())->cend(); ++i)

{

Save(\*i);

infile<<(\*i)->ID()<<" ";

}

infile<<std::endl;

// infile<<std::endl<<"所有群号:";

for(auto i=(save->groupMana())->cbegin(); i!=(save->groupMana())->cend(); ++i)

{

SaveWBGro((\*i));

infile<<(\*i)->groupNum()<<" ";

}

infile<<std::endl;

// infile<<std::endl<<"所有临时讨论组号:";

for(auto i=(save->tempdisgro())->cbegin(); i!=(save->tempdisgro())->cend(); ++i)

{

SaveWBtempGro(\*i);

infile<<(\*i)->groupNum()<<" ";

}

}catch(long a){

std::cout<<"打开文件失败";

}

}

void Files::Save(const std::shared\_ptr<WCXZJ>& save)

{

try {

infile.open("WC.txt");

if(!infile.is\_open())

throw 1;

// infile<<"所有人的ID:";

if(!(save->people())->empty())

for(auto i=(save->people())->cbegin(); i!=(save->people())->cend(); ++i)

{

Save(\*i);

infile<<(\*i)->ID()<<" ";

}

// infile<<std::endl<<"所有群号:";

for(auto i=(save->groupMana())->cbegin(); i!=(save->groupMana())->cend(); ++i)

{

SaveWCGro(\*i);

infile<<(\*i)->groupNum()<<" ";

}

// infile<<std::endl<<"所有临时讨论组号:";

for(auto i=(save->tempdisgro())->cbegin(); i!=(save->tempdisgro())->cend(); ++i)

{

SaveWCtempGro(\*i);

infile<<(\*i)->groupNum()<<" ";

}

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::OutFileQQ()

{

try {

\_pQ=std::make\_shared<QQXZJ>();

qq.open("QQ.txt");

if(!qq.is\_open())

throw 1;

std::stringstream qqss, wcss,wbss;

std::string qqs, wcs, wbs;

std::vector<long> gronum, tempgronum, personid;

//人

long id, year, month, day, tyear, tmonth, tday, wcid, wbnum, fid, askfid;

long password, number;

std::vector<long> friendid, askfriendid;

std::string name, home;

long a;

getline(qq,qqs);

qqss<<qqs;

while(qqss>>a)//人的ID

{

personid.push\_back(a);

}

for(auto i=personid.begin(); i!=personid.end(); ++i)

{

qqss.clear();

qqss.str("");

qqss<<\*i;

qqs="QQ"+qqss.str()+".txt";

outfile.open(qqs);

friendid.clear();;getline(outfile, qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

qqss>>id;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>number;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>password;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>name;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>home;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

long a;

gronum.clear();

while(qqss>>a)

{

gronum.push\_back(a);

}

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

tempgronum.clear();

while(qqss>>a)

{

tempgronum.push\_back(a);

}

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>year>>month>>day;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>tyear>>tmonth>>tday;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>wcid;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>wbnum;

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>fid;

friendid.clear();

if(fid!=0)

{

friendid.push\_back(fid);

while(qqss>>fid)

{

friendid.push\_back(fid);

}

}

qqss.clear();

qqss.str("");

friendid.clear();;getline(outfile, qqs);

qqss<<qqs;

qqss>>askfid;

askfriendid.clear();

if(askfid!=0)

{

askfriendid.push\_back(askfid);

while(qqss>>askfid)

{

askfriendid.push\_back(askfid);

}

}

qqss.clear();

qqss.str("");

std::shared\_ptr<QQDataXZJ> person=std::make\_shared<QQDataXZJ>(name, home, Time(Year(year), Month(month), Day(day)), Time(Year(tyear), Month(tmonth), Day(tday)), id, password, wcid, wbnum, number);

person->groNum()=gronum;//friendlist和askfriendlist等所有人都构建完之后在写入

person->tempgroNum()=tempgronum;

(person->friendlistnum())=friendid;

(person->askfriendlistnum())=askfriendid;

person->wc()=wcid;

person->wb()=wbnum;

(\_pQ->people())->push\_back(person);

outfile.close();

}//现在所有的人都齐了

for(auto i=(\_pQ->people())->begin(); i!=(\_pQ->people())->end(); ++i)//把friendlistnum和sakfriendlistnum的人都找到

{

if(!((\*i)->askfriendlistnum()).empty())

for(auto num=((\*i)->askfriendlistnum()).cbegin(); num!=((\*i)->askfriendlistnum()).cend(); ++num)

for(auto j=(\_pQ->people())->begin(); j!=(\_pQ->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->askfriendList()).push\_back(\*j);

if(!((\*i)->friendlistnum()).empty())

for(auto num=((\*i)->friendlistnum()).cbegin(); num!=((\*i)->friendlistnum()).cend(); ++num)

for(auto j=(\_pQ->people())->begin(); j!=(\_pQ->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->friendList()).push\_back(\*j);

}

getline(qq,qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

while(qqss>>a)//组号

{

(\_pQ->groupMana())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pQ->groupMana())->begin(); i!=(\_pQ->groupMana())->end(); ++i)//对于QQ中的每一个群

{

long j=(\*i)->groupNum();

qqss.clear();

qqss.str("");

qqss<<j;

qqs="QQGro"+qqss.str()+".txt";//群号

outfile.open(qqs);//根据群号打开文件

getline(outfile, qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

std::vector<long>().swap(personid);

while(qqss>>a)//群里每个人的ID

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ里面找然后压入这个群的人里面

for(auto k=(\_pQ->people())->begin(); k!=(\_pQ->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->qqGro())->push\_back(\*k);

break;

}

}

}

qqs="";

friendid.clear();;getline(outfile, qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

std::vector<long>().swap(personid);

while(qqss>>a)//群里每个人的ID

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ里面找然后压入这个群的人里面

for(auto k=(\_pQ->people())->begin(); k!=(\_pQ->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJoinQQGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

getline(qq,qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

while(qqss>>a)//临时讨论组号

{

(\_pQ->tempdisgro())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pQ->tempdisgro())->begin(); i!=(\_pQ->tempdisgro())->end(); ++i)//对于QQ中的每一个临时群

{

long j=(\*i)->groupNum();

qqss.clear();

qqss.str("");

qqss<<j;

qqs="QQGro"+qqss.str()+".txt";//群号

outfile.open(qqs);//根据群号打开文件

getline(outfile, qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

std::vector<long>().swap(personid);

while(qqss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pQ->people())->begin(); k!=(\_pQ->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->tempqqGro())->push\_back(\*k);

break;

}

}

}

getline(outfile, qqs);

qqss.clear();

qqss.str("");

qqss<<qqs;

std::vector<long>().swap(personid);

while(qqss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pQ->people())->begin(); k!=(\_pQ->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJointempQQGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

qq.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::OutFileWC()

{

try {

\_pWC=std::make\_shared<WCXZJ>();

wc.open("WC.txt");

if(!wc.is\_open())

throw 1;

std::stringstream qqss, wcss,wbss;

std::string qqs, wcs, wbs;

std::vector<long> gronum, tempgronum, personid;

//人

long id, year, month, day, tyear, tmonth, tday, qqnum, wbnum, fid, askfid;

long password, number;

std::vector<long> friendid, askfriendid;

std::string name, home;

long a;

getline(wc,wcs);

wcss<<wcs;

while(wcss>>a)//人的ID

{

personid.push\_back(a);

}

for(auto i=personid.begin(); i!=personid.end(); ++i)//先写到变量里

{

wcss.clear();

wcss.str("");

wcss<<\*i;

wcs="QQ"+wcss.str()+".txt";

outfile.open(wcs);

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>id;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>number;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>password;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>name;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>home;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

long a;

gronum.clear();

while(wcss>>a)

{

gronum.push\_back(a);

}

wcss.clear();

wcss.str("");

getline(outfile, wcs);

wcss<<wcs;

tempgronum.clear();

while(wcss>>a)

{

tempgronum.push\_back(a);

}

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>year>>month>>day;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>tyear>>tmonth>>tday;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>qqnum;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>wbnum;

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>fid;

friendid.clear();

if(fid!=0)

{

friendid.push\_back(fid);

while(wcss>>fid)

{

friendid.push\_back(fid);

}

}

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

wcss>>askfid;

askfriendid.clear();

if(askfid!=0)

{

askfriendid.push\_back(askfid);

while(wcss>>askfid)

{

askfriendid.push\_back(askfid);

}

}

std::shared\_ptr<WCDataXZJ> person=std::make\_shared<WCDataXZJ>(name, home, Time(Year(year), Month(month), Day(day)), Time(Year(tyear), Month(tmonth), Day(tday)), id, password, qqnum, wbnum);

person->groNum()=gronum;//friendlist和askfriendlist等所有人都构建完之后在写入

person->tempgroNum()=tempgronum;

(person->friendlistnum())=friendid;

(person->askfriendlistnum())=askfriendid;

person->qq()=qqnum;

person->wb()=wbnum;

(\_pWC->people())->push\_back(person);

outfile.close();

}//现在所有的人都齐了

for(auto i=(\_pWC->people())->begin(); i!=(\_pWC->people())->end(); ++i)//把friendlistnum和sakfriendlistnum的人都找到

{

if(!((\*i)->askfriendlistnum()).empty())

for(auto num=((\*i)->askfriendlistnum()).cbegin(); num!=((\*i)->askfriendlistnum()).cend(); ++num)

for(auto j=(\_pWC->people())->begin(); j!=(\_pWC->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->askfriendList()).push\_back(\*j);

if(!((\*i)->friendlistnum()).empty())

for(auto num=((\*i)->friendlistnum()).cbegin(); num!=((\*i)->friendlistnum()).cend(); ++num)

for(auto j=(\_pWC->people())->begin(); j!=(\_pWC->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->friendList()).push\_back(\*j);

}

getline(wc,wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

while(wcss>>a)//组号

{

(\_pWC->groupMana())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pWC->groupMana())->begin(); i!=(\_pWC->groupMana())->end(); ++i)//对于微信中的每一个群

{

long j=(\*i)->groupNum();

wcss.clear();

wcss.str("");

wcss<<j;

wcs="WCGro"+wcss.str()+".txt";//群号

outfile.open(wcs);//根据群号打开文件

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

std::vector<long>().swap(personid);

while(wcss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWC->people())->begin(); k!=(\_pWC->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->wcGro())->push\_back(\*k);

break;

}

}

}

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

std::vector<long>().swap(personid);

while(wcss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWC->people())->begin(); k!=(\_pWC->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJoinWCGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

getline(wc,wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

while(wcss>>a)//临时讨论组号

{

(\_pWC->tempdisgro())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pWC->tempdisgro())->begin(); i!=(\_pWC->tempdisgro())->end(); ++i)//对于微信中的每一个临时群

{

long j=(\*i)->groupNum();

wcss.clear();

wcss.str("");

wcss<<j;

wcs="WCGro"+wcss.str()+".txt";//群号

outfile.open(wcs);//根据群号打开文件

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

std::vector<long>().swap(personid);

while(wcss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWC->people())->begin(); k!=(\_pWC->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->tempwcGro())->push\_back(\*k);

break;

}

}

}

getline(outfile, wcs);

wcss.clear();

wcss.str("");

wcss<<wcs;

std::vector<long>().swap(personid);

while(wcss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWC->people())->begin(); k!=(\_pWC->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJointempWCGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

wc.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

void Files::OutFileWB()

{

try {

\_pWB=std::make\_shared<WBXZJ>();

wb.open("WB.txt");

if(!wb.is\_open())

throw 1;

std::stringstream qqss, wcss,wbss;

std::string qqs, wcs, wbs;

std::vector<long> gronum, tempgronum, personid;

//人

long id, password, number, year, month, day, tyear, tmonth, tday, wcid, qqnum, fid, askfid;

std::vector<long> friendid, askfriendid;

std::string name, home;

long a;

getline(wb,wbs);

wbss<<wbs;

while(wbss>>a)//人的ID

{

personid.push\_back(a);

}

for(auto i=personid.begin(); i!=personid.end(); ++i)

{

wbss.clear();

wbss.str("");

wbss<<\*i;

wbs="QQ"+wbss.str()+".txt";

outfile.open(wbs);

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>id;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>number;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>password;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>name;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>home;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

long a;

gronum.clear();

while(wbss>>a)

{

gronum.push\_back(a);

}

wbss.clear();

wbss.str("");

getline(outfile, wbs);

wbss<<wbs;

tempgronum.clear();

while(wbss>>a)

{

tempgronum.push\_back(a);

}

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>year>>month>>day;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>tyear>>tmonth>>tday;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>wcid;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>qqnum;

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>fid;

friendid.clear();

if(fid!=0)

{

friendid.push\_back(fid);

while(wbss>>fid)

{

friendid.push\_back(fid);

}

}

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

wbss>>askfid;

askfriendid.clear();

if(askfid!=0)

{

askfriendid.push\_back(askfid);

while(wbss>>askfid)

{

askfriendid.push\_back(askfid);

}

}

std::shared\_ptr<WBDataXZJ> person=std::make\_shared<WBDataXZJ>(name, home, Time(Year(year), Month(month), Day(day)), Time(Year(tyear), Month(tmonth), Day(tday)), id, password, qqnum, wcid, number);

person->groNum()=gronum;//friendlist和askfriendlist等所有人都构建完之后在写入

person->tempgroNum()=tempgronum;

(person->friendlistnum())=friendid;

(person->askfriendlistnum())=askfriendid;

person->qq()=qqnum;

person->wc()=wcid;

(\_pWB->people())->push\_back(person);

outfile.close();

}//现在所有的人都齐了

for(auto i=(\_pWB->people())->begin(); i!=(\_pWB->people())->end(); ++i)//把friendlistnum和sakfriendlistnum的人都找到

{

if(!((\*i)->askfriendlistnum()).empty())

for(auto num=((\*i)->askfriendlistnum()).cbegin(); num!=((\*i)->askfriendlistnum()).cend(); ++num)

for(auto j=(\_pWB->people())->begin(); j!=(\_pWB->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->askfriendList()).push\_back(\*j);

if(!((\*i)->friendlistnum()).empty())

for(auto num=((\*i)->friendlistnum()).cbegin(); num!=((\*i)->friendlistnum()).cend(); ++num)

for(auto j=(\_pWB->people())->begin(); j!=(\_pWB->people())->end(); ++j)

if((\*j)->ID()==\*num)

((\*i)->friendList()).push\_back(\*j);

}

getline(wb,wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

while(wbss>>a)//组号

{

(\_pWB->groupMana())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pWB->groupMana())->begin(); i!=(\_pWB->groupMana())->end(); ++i)//对于微博中的每一个群

{

long j=(\*i)->groupNum();

wbss.clear();

wbss.str("");

wbss<<j;

wbs="WBGro"+wbss.str()+".txt";//群号

outfile.open(wbs);//根据群号打开文件

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

std::vector<long>().swap(personid);

while(wbss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWB->people())->begin(); k!=(\_pWB->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->wbGro())->push\_back(\*k);

break;

}

}

}

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

std::vector<long>().swap(personid);

while(wbss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWB->people())->begin(); k!=(\_pWB->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJoinWBGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

getline(wb,wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

while(wbss>>a)//临时讨论组号

{

(\_pWB->tempdisgro())->push\_back(std::make\_shared<groupXZJ>(a));

}

for(auto i=(\_pWB->tempdisgro())->begin(); i!=(\_pWB->tempdisgro())->end(); ++i)//对于微博中的每一个群

{

long j=(\*i)->groupNum();

wbss.clear();

wbss.str("");

wbss<<j;

wbs="WBGro"+wbss.str()+".txt";//群号

outfile.open(wbs);//根据群号打开文件

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

std::vector<long>().swap(personid);

while(wbss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWB->people())->begin(); k!=(\_pWB->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->tempwbGro())->push\_back(\*k);

break;

}

}

}

getline(outfile, wbs);

wbss.clear();

wbss.str("");

wbss<<wbs;

std::vector<long>().swap(personid);

while(wbss>>a)

{

personid.push\_back(a);

}

if(!personid.empty())

{

for(auto j=personid.begin(); j!=personid.end(); ++j)

{//根据这个群中每个人的ID去QQ群里面找然后压入这个群的人里面

for(auto k=(\_pWB->people())->begin(); k!=(\_pWB->people())->end(); ++k)

if(\*j==(\*k)->ID())

{

((\*i)->askJointempWBGroList())->push\_back(\*k);

break;

}

}

}

outfile.close();

}

wb.close();

} catch (long a) {

std::cout<<"打开文件失败";

}

}

**四总结**

**花了两天时间看完Effictive C++然后重新打的课设，方法优化了很多，但是还是有很多不足**

**，我是全部功能打完之后在进行检查错误，特别繁琐，还要检查函数之间的调用关系，如果一步一步检查会轻松很多。然后打了两周，觉得身心疲惫，而且有的功能还没有实现，所以下次打代码一定要先打框架，然后打文件，然后一个函数一个函数检查。**