原.文件

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* AutoLibrary.cpp \*

\* 自助图书馆 \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

# pragma warning(disable:4786)

# include<iostream>

# include<cstring>

# include<string>

# include<fstream>

# include<map>

# include"global.h"

# include"date.h"

# include"BookData.h"

# include"ReaderData.h"

# include"database.h"

# include"Librarian.h"

# include"reader.h"

using namespace std;

int main()

{

Database<BookData>BookBase("book.dat");

Database<ReaderData>ReaderBase("readers.dat");

char choice = 'X';

while (!(choice == 'E' || choice == 'e'))

{

cout << "(L)ibrarian entry,(R)eader entry,Re(t)urn book,(E)xit:";

cin >> choice;

Librarian \* mgr = NULL;

Reader \* rdr = NULL;

switch (choice)

{

//===============管理员入口================//

case 'L':case 'l':

mgr = new Librarian(101, "yjc");

choice = mgr->login();

while (!(choice == 'E' || choice == 'e'))

{

cout << "(A)dd reader,Add(B)ook,(Q)uery Reader,"

<< "(P)ut book to shelf.(E)xit:";

cin >> choice;

switch (choice)

{

long ID;

char name[40];

case 'A':case 'a':

cout << "Give a reader PIN and input a name:";

cin >> ID;

cin.ignore();

cin.get(name, 40, '\n');

ReaderBase.Insert(ReaderData(ID, name));

break;

case 'B':case 'b':

cout << "Input abook ID and name:";

cin >> ID;

cin.ignore();

cin.get(name, 40, '\n');

BookBase.Insert(BookData(ID, name));

break;

case 'Q':case 'q':

cout << "Input a reader's PIN:";

cin >> ID;

if (ReaderBase.Query(ID) == NULL)

cout << "NO such a reader!" << endl;

else

ReaderBase.Query(ID)->ShowData();

break;

case 'P':case 'p':

cout << "Input a book ID:";

cin >> ID;

if (BookBase.Query(ID) == NULL)

cout << "No such a book!" << endl;

else

BookBase.Query(ID)->SetState(SHELF);

break;

case 'S':case 's':

ReaderBase.ShowAllData();

BookBase.ShowAllData();

break;

case 'E':case 'e':

break;

default:

cout << "Unavailable function!\n";

}

}

delete mgr;

choice = 'x';

break;

//-------------------管理员退出---------------------

//===================读者入口=======================

case 'R':case 'r':

long PIN, BookID;

int i, t1, t2;

char name[40];

for (i = 0; i < 3; ++i)

{

cout << "Input PIN:";

cin >> PIN;

if (ReaderBase.Query(PIN) != NULL)

{

rdr = new Reader(PIN);

break;

}

}

if (i == 3)

{

cout << "Check in failed!" << endl;

choice = 'E';

}

while (!(choice == 'E' || choice == 'e'))

{

cout << "(A)dd book to cart,check(0)ut,(Q)uery book by name,"

<< "(L)ist my books (E)xit:";

cin >> choice;

switch (choice)

{

case 'A':case 'a':

cout << "Input a book ID:";

cin >> BookID;

if ((BookBase.Query(BookID) != NULL)

&& (BookBase.Query(BookID)->GetState() == SHELF))

if (rdr->AddBook(BookID))

BookBase.Query(BookID)->SetState(LIB);

break;

case 'O':case 'o':

t1 = rdr->GetNum();

t2 = ReaderBase.Query(PIN)->GetNum();

//在读者的图书配号

if (t1 > 0 && t2 < LIMIT)

{

cout << PIN << " " << ReaderBase.Query(PIN)->GetName()

<< "book list" << endl;

for (i = 0; i < t1&&i < (LIMIT - t2); ++i)

{

BookID = ReaderBase.Query(PIN)->BorrowBook(rdr->CheckOut());

BookBase.Query(BookID)->SetState(READER);

BookBase.Query(BookID)->SetPIN(PIN);

cout << i + 1 << "\t"

<< BookBase.Query(BookID)->GetName() << endl;

}

Date().display();

break;

case 'Q':case 'q':

cout << "Input a book name(part):";

cin.ignore();

cin.get(name, 40, '\n');

if (BookBase.QueryName(name) != NULL)

BookBase.QueryName(name)->ShowData();

break;

case 'L':case 'l':

ReaderBase.Query(PIN)->ShowData();

break;

case 'c':case 'C':

rdr->ShowCart();

break;

case 'e':case 'E':

break;

default:

cout << "Unavailable function!\n";

}

}

delete mgr;

choice = 'X';

break;

//--------------------------------读者退出----------------------------

//===============================还书入口=============================

case 't':case 'T':

cout << "Input a book ID:";

cin >> BookID;

if (BookBase.Query(BookID) != NULL)

{

BookBase.Query(BookID)->SetState(LIB);

if ((PIN = BookBase.Query(BookID)->GetPIN()) > 0)

ReaderBase.Query(PIN)->ReturnBook(BookID);

}

break;

//-------------------------------退出还书入口------------------------

case 'E':case 'e':

break;

default:

cout << "Unavailable function!\n";

}

}

return 0;

}

}

Readerdata.h

class ReaderData :public object

{

protected:

long PIN;

char name[20];

long BookList[LIMIT];

Date BorrowDate[LIMIT];

int num;

public:

ReaderData(int PIN, const char\*name)

{

SetID(PIN);

SetName(name);

num = 0;

for (int i = 0; i < LIMIT; ++i)

BookList[i] = 0;

}

ReaderData()

{

PIN = 0;

num = 0;

for (int i = 0; i < LIMIT; ++i)

BookList[i] = 0;

}

void SetID(long PIN){ this->PIN = PIN; }

void SetName(const char\* i\_name){ strcpy\_s(name, i\_name); }

long GetID(){ return PIN; }

const char \* GetName(){ return name; }

int GetNum(){ return num; }

long BorrowBook(long);

void ShowData();

long ReturnBook(long);

};

long ReaderData::ReturnBook(long BookID)

{

for (int i = 0; i < LIMIT; ++i)

if (BookList[i] == BookID)

{

BookList[i] = 0;

--num;

return BookID;

}

return 0;

}

void ReaderData::ShowData()

{

cout << PIN << "\t" << name << endl;

for (int i = 0; i < LIMIT; ++i)

{

if (BookList[i])

{

cout << i + 1 << ":" << BookList[i] << "\t";

BorrowDate[i].display();

}

}

}

long ReaderData::BorrowBook(long BookID)

{

for (int i = 0; i < LIMIT; ++i)

if (BookList[i] == 0)

{

BookList[i] = BookID;

BorrowDate[i] = Date();

num++;

return BookID;

}

cout << "Book reach the limit!" << endl;

return 0;

}

Reader.h

class Reader :public object

{

private:

long PIN;

long BookID[LIMIT];

int num;

public:

Reader(long PIN = 0, const int num = 0) :PIN(PIN), num(num){}

int AddBook(const long BookID)

{

if (num < LIMIT)

{

this->BookID[num] = BookID;

cout << "Book" << BookID << "added!" << endl;

num++;

return 1;

}

else

cout << "Cart full!" << endl;

return 0;

}

long CheckOut(){

--num;

return BookID[num];

}

void ShowCart()

{

for (int i = 0; i < num; ++i)

cout << BookID[i] << endl;

}

int GetNum(){ return num; }

};

Librarian.h

class Librarian : public object

{

private:

long ID;

char name[20];

char passwd[9];

public:

Librarian(long ID, const char \* name) :ID(ID)

{

strcpy\_s(this->name, name);

strcpy\_s(passwd, "abc");

}

char login()

{

char pw[9];

for (int i = 0; i < 3; ++i)

{

cout << "Enter password:";

cin >> pw;

if (strcmp(pw, passwd) == 0)

return 'X';

}

cout << "Login fail!" << endl;

return 'E';

}

};

global.h

const int LIMIT = 3;

enum State{ READER, LIB, SHELF };

//

class object{};

date.h

# include<ctime>

using namespace std;

class Date : public object

{

int year, month, day;

int DayOfMonth(int y, int m)const;

public:

Date()

{

/\*time\_t curtime = time(NULL);

tm tim = \* localtime(&curtime);

day = tim.tm\_mday;

month = tim.tm\_mon + 1;

year = tim.tm\_year + 1900;\*/

struct tm t; //tm结构指针

time\_t now; //声明time\_t类型变量

time(&now); //获取系统日期和时间

localtime\_s(&t, &now); //获取当地日期和时间

day = t.tm\_mday;

month = t.tm\_mon + 1;

year = t.tm\_year + 1900;

}

Date(int y, int m, int d)

{

if ((y <= 0) || (m <= 0 || m > 12) || (d <= 0 || d > DayOfMonth(y, m)))

{

cout << "Invalid date,data has been set to 1900-1-1" << endl;

year = 1900;

month = day = 1;

}

}

virtual~Date(){}

int GetYear()const

{

return year;

}

int GetMonth()const

{

return month;

}

int GetDay()const

{

return day;

}

bool IsLeapyear()const

{

return year % 400 ? (year % 100 ? (year % 4 ? false : true) : false) : true;

}

bool IsLeapyear(const int y)const

{

return y % 400 ? (y % 100 ? (y % 4 ? false : true) : false) : true;

}

void display()const

{

std::cout << year << "-" << month << "-" << day << std::endl;

}

};

int Date::DayOfMonth(int y, int m)const

{

int d = 0;

switch (m)

{

case 1:case 3 : case 5 : case 7 : case 8 : case 10 : case 12 :

d = 31;

break;

case 4:case 6 : case 9 : case 11 :

d = 30;

break;

case 2:

d = 28 + IsLeapyear(y);

break;

}

return d;

}

Database.h

template<class T>

class Database :public object

{

private:

fstream File;

char FileName[40];

long FileLen, rec\_size;

typedef map<long, T, less<long>>mtype;

mtype RecMap;

public:

Database(const char\*FileName);

~Database(){ SaveMap(); }

void Insert(T&);

void Delete(long);

T \* Query(long);

T\* QueryName(const char \*);

void SaveMap();

void ShowAllData();

};

template<class T>

Database<T>::Database(const char \* FileName)

{

strcpy\_s(this->FileName, FileName);

File.open(FileName, ios::in | ios::binary);

rec\_size = sizeof(T);

if (File.is\_open()){

File.seekg(0,ios::end);

if ((FileLen = File.tellg()) > 0)

{

T Object;

File.seekg(0, ios::beg);

do{

File.read((char\*)&Object, rec\_size);

RecMap.insert(mtype::value\_type(Object.GetID(), Object));

} while (File.tellg() < FileLen);

}

File.close();

}

}

template<class T>

void Database<T>::SaveMap()

{

mtype::const\_iterator iter;

T Object;

File.open(FileName, ios::out | ios::binary | ios::trunc);

for (iter = RecMap.begin(); iter != RecMap.end(); ++iter)

File.write((char \*)& iter->second, rec\_size);

File.close();

}

template<class T>

void Database<T>::Insert(T & Object)

{

RecMap.insert(mtype::value\_type(Object.GetID(), Object)); //cout<<typeid(T).name()<<"inserted"<<endl;

}

template<class T>

T \* Database<T>::Query(long ObjID)

{

mtype::iterator iter;

iter = RecMap.find(ObjID);

if (iter == RecMap.end())

{

cout << ObjID << "not found!" << endl;

return NULL;

}

else

return &(iter->second);

}

template<class T>

T \* Database<T>::QueryName(const char \* ObjName)

{

mtype::iterator iter;

for (iter = RecMap.begin(); iter != RecMap.end(); ++iter)

if (strstr((iter->second).GetName(), ObjName) != NULL)

{

//cout<<"find a name:"<<(iter->second).zgetName()<<endl;

return &(iter->second);

}

cout << ObjName << "in" << typeid(T).name() << "not found!" << endl;

return NULL;

}

template<class T>

void Database<T>::Delete(long ObjID)

{

Query(ObjID);

RecMap.erase(ObjID);

}

template<class T>

void Database<T>::ShowAllData()

{

mtype::iterator iter;

T Object;

cout << "Data in" << typeid(T).name() << ":" << endl;

for (iter = RecMap.begin(); iter != RecMap.end(); ++iter)

{

(iter->second).ShowData();

}

}

BookData.h

/\*\*/

class BookData :public object

{

private:

long BookID;

char name[40];

State state;

long PIN;

public:

BookData(long BookID, const char\*name, State state = SHELF, long PIN = 0)

{

SetID(BookID);

SetName(name);

SetState(state);

SetPIN(PIN);

}

BookData(){ BookID = 0; PIN = 0; }

const long GetID(){ return BookID; }

const char\*GetName(){ return name; }

State GetState(){ return state; }

long GetPIN(){ return PIN; }

void ShowData(){ cout << BookID << "\t" << name << "\t" << state << "\t" << PIN << endl; }

void SetID(long BookID){ this->BookID = BookID; }

void SetName(const char \*i\_name){ strcpy\_s(name, i\_name); }

void SetState(State state) { this->state = state; }

void SetPIN(long PIN){ this->PIN = PIN; }

};