第一章 使用的数据结构

程序无非就是数据结构+算法。而数据结构就是程序中用来组织数据的方法。本次编写的图书管理系统用的是一个比较简单的数据结构：链表。

首先，先确定图书管理系统中需要存储的数据。

1.     书，书的属性有：书名，作者，类型，介绍，还有为了唯一标记一本书，给他一个id。

2.     用户：姓名，密码，还有id。

3.     用户借阅的书：人的id，书的id，借阅时间。

因为要用链表要储存这些数据，所以要从这些数据中抽象一个父类出来作为链表中的节点（总不能为它们每一个都建立一个链表吧，那样维护起来太难），我们发现，每一个数据都有一个共同的属性，那就是id，因此，建立一个只有id属性的类作为它们的基类，在链表中使用这个基类作为节点。

关于这几个类的实现，都是一些比较简单的操作。除了Compare方法需要讲一下。Compare方法是用来比较的函数，在本项目中的作用就是判断当前节点是否为想要找的节点。BookType类，UserType类通过id属性来判断，UserBookType类通过bookId属性来判断，因此需要重写Compare函数。

下面开始建立链表，链表中储存的节点是NoteType类型，也就是BookType，UserType，UserBookType的父类，在使用到具体子类的时候可以强转到具体的子类。

LinerList的方法：

1. Init函数：从外存中（本项目是数据库）中读取数据来初始化链表。
2. Save函数：用来保存在程序中被修改的数据，通过链表头节点的id值来判断是添加还是删除：1表示添加，调用SaveAdd函数；-1表示删除，调用SaveDelete函数，
3. SaveAdd函数：把链表中的数据添加到数据库中。
4. SaveDelete函数：把链表中的数据从数据库中删掉。
5. AddNode函数：把node节点添加到链表中。
6. DeleteNode函数：把链表中节点属性id的值为id的节点删掉

LinerList的属性：

1. List：该属性是链表的头节点。
2. Length：用来记录链表长度
3. db：这是一个QSqlDatabase类型的静态变量，本项目中所有的数据都存储在同一个数据库中，为了方便之后的读入读出操作，就设置了一个静态变量来保存数据库对象，这样在初始化一次之后，就不用再重复进行初始化操作了。注：注意类中静态变量的初始化方法。要在类体外进行初始化。

链表的实现比较简单，这里大家看一下代码就行，没什么好讲的。

需要注意的一点是，在Qt中使用MySql时，要在配置文件（.pro文件）中加一句QT += sql

第二章 把基本的数据结构具体为项目所需要的数据结构

链表只是基本的数据结构，在项目中要使用的是储存书本信息的链表，储存用户信息的链表，储存用户的书的信息的链表，因此要对LinerList进一步具体实现才能得到项目所需要的数据结构。本项目要使用的三个数据结构

1. Book： 储存书本信息的链表。
2. User： 储存用户信息的链表
3. UserBook： 储存用户的书的链表

都是继承自LinerList基类，并具体实现了LinerList的Init方法，SaveAdd方法，SaveDelete方法。因为三个类基本类似，只是细节上的实现有稍微差别，本次就以User类为例，来介绍一下Init，SaveAdd，SaveDelete，三个方法的具体实现。

Init方法的具体实现：

1. 先用open函数打开数据库
2. 声明一个QSqlQuery类型的变量，给类可以执行MySql语句，同时会保存执行的结果。
3. 使用exec方法查询users表中的所有信息。
4. 使用next方法遍历查询结果，并把每一条记录转换成项目定义的数据结构储存起来。
5. 关闭数据库（用完数据库一定要关闭）。

SaveAdd方法的具体实现：

1. 使用open方法打开数据库
2. 遍历链表中的每一个节点，把节点信息转换成MySql的insert语句，并执行
3. 使用close方法关闭数据库

SaveDelete方法的具体实现：

1. 打开数据库
2. 遍历链表中的每一个节点，把节点信息转换成MySql的delete语句，并执行。
3. 关闭数据库

Book类，UserBook类的实现同上相类似。

第三章 登陆界面

现在，项目所需要的数据结构已经介绍完了，可以开始搭建图书管理系统的图形界面了，先从登陆界面开始。

1. 首先，新建一个项目，选择Qt Widgets Application项目，名称改为LibraryManageSystem，点下一步，再点下一步，基类选QWidget，类名改为Login。(注意路径和项目名不要出现空格和中文，如果添加一个文件后，编译报错，可以先清理项目，重新构建项目，执行qmake试试)。
2. 双击项目Forms文件夹下的login.ui文件，设计登陆界面。Qt为我们提供了许多方便的控件，直接使用就行了，Login界面使用了两个label，两个lineEdit，两个pushButton。控件上显示的文字只要双击控件就能更改，控件的变量名通过右击控件，选择改变对象名称来修改。下面给出login界面的控件显示文字和控件的变量名。
3. Label：名称：“用户名”，变量名：在程序中没有用到，可不修改。
4. Label：名称：“密码”，变量名：在程序中没有用到，可不修改
5. Line Edit：名称：无，变量名：nameLineEdit。
6. Line Edit：，名称：无，变量名：passwordLineEdit。
7. Push Button：名称：“登陆”，变量名：loginPushButton。
8. Push Button：名称：“退出”，变量名：closePushButton。
9. Login类的实现并不复杂，首先做一下初始化操作，再添加一个槽函数，实现用户点击登陆按钮后的登陆操作就行了。先给大家简单介绍信号与槽，大家只要简单了解一下就会使用了（具体的原理请看专业书籍，这里只讲应用）。以一个按钮为例，用户单击这个按钮后，会发出一个clicked信号，这时可以设置一个函数来接受这个信号，进行一些操作，这个接受的函数就叫槽，信号与槽通过connect函数来告诉系统把信号和槽连接起来

下面来介绍Login类的实现细节：

1. 在构造函数里先初始化user变量，把两个按钮的信号与相应的槽连接起来
2. LoginSystemSlot槽函数是自己定义的槽，主要实现登陆过程，在函数中先获得用户输入用户名和密码，然后与数据库中的信息一个一个进行比较，如果相等，再判断用户名是否为管理员，是管理员则进入管理员界面，是用户则进入用户界面，如果数据库中所有信息都比较完后，还没有相等的，表明用户输入错误，这时要提示用户输入错误。
3. 因为这里要用到用户界面和管理界面，我们先再这里把两个界面建立出来，具体的实现，后面章节再讲。

右击项目，添加新文件，选择Qt设计界面类，界面模板选择Widget，类名改为UserInterface；同样方法再建一个AdminInterface。把两个类的头文件包含进login.cpp 文件中

1. 判断用户输入正确后，就建立相应的界面，并把它显示出来。例如：如用用户名是管理员，AdminInterface\* w = new AdminInterface(); w->show();，注意一定要调用show函数，否则界面不显示。
2. 如果编译时报错：login.obj : error LNK2019: 无法解析的外部符号……，可以尝试点构建->清理所有项目->重新构建所有项目->执行qmake。如果程序没错，就可以运行了。

第四章 用户界面

我们在上一章已经建立了一个用户界面，不过并没有编写实现，因为登陆的是一个用户，所以在登陆成功，打开用户界面时，要传一个参数过来以确定是哪个用户。因此要修改一下UserInterface类的构造函数，当然在登陆界面中，建立用户界面的函数也要修改。

用户界面的构造函数添加了一个int类型的id，用来确定用户的身份。在new一个UserInterface变量时，要把登陆的用户的id穿进去，变量不仅要调用show函数让界面显示，还要调用ShowUserBooks让用户已借阅的书显示。

1. 用户界面共添加了四个属性

1.    userBook属性：储存当前用户已借阅的书

2.    book属性：储存所有书的信息

3.    borrowBook属性：储存当前操作借阅的书

4.    returnBook属性：储存当前操作还掉的书

1. 用户界面的功能实现如下：

1.    在Tree Widget显示用用户已借阅的书

2.    点击借书按钮，打开借书界面

3.    点击还书按钮，打开还书界面

4.    双击每一行书，会显示书的详细信息

1. 用户界面需要四个槽:

1.    显示用户已借阅书籍的槽

2.    当用户点击借书按钮，打开借书界面的槽。

3.    当用户点击还书按钮，打开还书界面的槽。

4.    当用户双击每一行书，显示书本详细信息的槽。

1. 设计用户界面，双击userinterface.ui，用户界面用了两个Push Button，一个Tree Widget。

1.    Push Button： 显示文字：“借书”，变量名：borrowPushButton。

2.    Push Button：显示文字：“还书“，变量名：returnPushButton。

3.    Tree Widget：显示文字：无，变量名：因为只有一个，可不该。

右击Tree Widget，选择编辑项目，在列那一栏，点+号，可以添加Tree Widget的列。

1. 实现具体的函数：
2. 在构造函数中，先初始化各个熟悉的值，userBook要根据登陆界面传来的ld进行初始化。用户界面有三个槽需要连接，borrowPushButton的clicked信号，与用户界面BorrowBookSlot槽连接；returnPushButton的clicked信号，与用户界面的ReturnBookSlot槽连接；treeWidget的双击信号itemDoubleClicked信号与ShowBookInfoSlot槽连接

在这里简单介绍一下connect函数，connect函数共有四个参数，第一个参数是发送信号的对象的指针，第二个参数对象是发送的信号，第三个参数是接受信号的对象的指针，第四个参数是槽函数。信号的参数与槽的参数一一对应。

1. 在ShowAllUserBooks槽函数中，先清空treeWidget，然后遍历userBook链表，把链表的每一个节点的信息转换成QTreeWidgetItem对象，插入到treeWidget中。

第五章 显示图书详细信息界面

上一章已经把双击的信号与显示详细信息的槽连接了，但是因为没有建显示图书详细信息的界面，所以函数是空的，这一章，我们将编写一个显示图书详细信息的界面。

为了程序的简洁性，我们只给书这个类添加了四个属性：名称，作者，类型，介绍。这样可以让我们把注意力集中到实现逻辑上，而非实现细节。

现在开始编写显示图书详细信息的界面：

1. 新建一个Qt设计师界面类，界面模板选择Widget，类名改为BookInfo。
2. 双击bookinfo.ui开始设计界面，添加三个Label，一个Text Browser。

三个Label都不用该显示文字，要根据具体的书籍信息来显示文字。变量名依次改为：nameLabel，authorLabel，categoryLabel，Text Browser变量名改为introductionTextBrowser。

1. 类的实现很简单，只要在构造函数里，根据传进来的BookType类型变量值修改四个控件的显示文字就行了。

第六章 借书与还书（一）

到目前为止，我们已经可以显示用户借阅的书，显示书的详细信息。但是图书管理最重要的两个功能还没有做，那就是借书和还书。

在做这两个界面之前，我们先看一下实现后的界面，大家看一下能不能发现什么问题。

在上面的四个界面，我们发现，这四个界面都是有相同的控件构成的：一个Tree Widget，两个Push Button。C++代码非常注重代码的重用性，为这四个四个相似的界面，编写四个几乎相同的代码，程序就会显得非常笨重。因此，我们把这四个界面的相似的东西抽线出来，做一个基类，而借书，还书等界面继承此类，这样就不用重复编码了。

下面开始编码：

1. 新建一个Qt设计师界面类，界面模板选择Widget，类名就用Form。
2. 设计界面，界面共有一个Tree Widget，两个Push Button：
3. Tree Widget的变量名不用修改。
4. Push Button：显示文字：“确定”，变量名：confirmPushButton。
5. Push Button：显示文字：“取消“，变量名：cancelPushButton。

这个基类只是负责提供界面，没有什么要实现的逻辑。

确定按钮的功能要在子类里实现，取消按钮的功能都一样，在基类实现即可。

第七章 借书与还书（二）

上一章我们建了一个基类用来作为还书界面与借书界面的基类，这一章，我们就来具体实现还书界面和借书界面。

首先我们来思考一下借书和还书的流程,，借书的时候打开借书界面，选择要借的书，然后点确定按钮；还书的时候，打开还书界面，选择要换的书，然后点确定。两个操作在实现方面也很类似，借书操作要把借的书记录下来，最后保存到数据库中，还书操作要把还的书记录下来，最后从数据库中把还的书删掉。

下面开始编写借书界面：

1. 新建一个C++类，类名为BorrowBook。
2. 让BorrowBook类public继承Form类，在BorrowBook.cpp 文件中添加#include”ui\_form.h”，并把Form类的ui属性改为protected，否则在BorrowBook类中访问不了ui变量。
3. 类的属性：
   1. userBook：储存用户已借阅的书，在完成借书操作后，会把借阅的书添加到userBook中去。
   2. borrowBook：用来储存借书操作中借阅的书，方便之后回写到数据库中。如果没有这个变量，在回写操作中就要遍历userBook以确定那本书是借的。
   3. book：用来储存了所有书的信息。
4. 类的方法：
   1. ShowAllBooks：显示所有的书，让用户来借
   2. BorrowBookSlot：用户在点击确定按钮后，进入此函数，记录借的书。

具体的实现：

1. 在构造函数中，先设置一下Tree Widget有几列，并为每一列设置一下标题。然后把确认按钮与BorrowBookSlot槽连接起来。
2. ShowAllBooks：遍历book链表，把所有的书的信息显示在Tree Widget中，每一行都要添加一个复选框，让用户来选择借阅的书。
3. BorrowBookSlot：遍历Tree Widget的每一行，检查复选框有没有被选中，如果被选中了，就添加到borrowBook中。注意要在类的声明中加Q\_OBJECT宏，否则无法在父类和子类间进行信号与槽的连接。

虽然借书的界面和基本逻辑都建立完了，但还是有两个问题：

1. 当用户点击确定按钮后，虽然会调用BorrowBookSLot函数，把借的书存到borrowBook链表里，但并没有存到数据库中。
2. 在用户点击确认按钮后，借书界面关闭，回到user界面，但user界面所显示的书并没有发生变化。

要解决两个问题也很简单：

1. Qt提供了一个函数closeEvent，这个函数可以拦截关闭事件，让我们可以自己处理关闭时间，首先在头文件borrowbook.h里添加一下函数的声明。然后在closeEvent函数里实现保存的操作。注意：这里有一个地方比较容易犯错，我们实现的链表在添加节点的函数（AddNode函数）中使用的参数是指针，当添加节点的时候，添加的节点不要同时添加到两个链表中，具体的分析过程我在另一篇博客中介绍了。

在函数中我们使用了Clear函数，这是新给LinerList新添加的函数。

1. 为了实现在用户借书之后刷新user界面，需要给user界面一个信号，当user界面接到这个信号时，就刷新界面。这里有两种方法：第一种方法是让借书界面关闭后自动销毁，这样就会产生一个destroyed的信号，user界面接到这个信号就刷新。第二个方法就是自己定义一个信号，不过这里并没有这个必要。

第八章 借书与还书

还书界面与借书界面类似，只不过是显示的时候不是显示所有书籍，而是显示用户已借阅的书籍。其他方面都类似。

类的声明：

1. 新建一个C++类，类名为ReturnBook，public继承Form类，在returnbook.cpp中添加#include “ui\_form.h”，否则不能使用ui
2. 类的属性：
   1. userBook：储存用户已借阅的书。
   2. returnBook：储存在还书界面还的书。
   3. book：储存所有书的信息
3. 类的方法：
   1. ShowAllUserBooks：显示所有用户已借阅的书，让用户还书。
   2. ReturnBookSlot：当用户点击确定按钮后，进入此函数，把用户还的书添加到returnBook链表中去。以便以后从数据库中删掉用户还的书。注意：在类的声明中添加Q\_OBJECT，并执行一下qmake，否则父类与子类的信号无法传递。
   3. closeEvent：Qt提供的虚函数，用来拦截close事件，让用户自己处理。在此函数中，我们将提示用户是否保存操作，是，则把用户还的书添加到userBook链表中，并把书从数据库中删掉，否，则不进行操作，把returnBook链表清空，返回use界面。

类的实现：

1. 在构造函数中，先设置一下界面的显示，然后初始化变量，最后连接一下槽函数。
2. ShowAllUserBooks：遍历useBook链表，根据书的id，从book链表中找到书的详细信息，并显示到界面上。
3. ReturnBookSlot：遍历Tree Widget，检查复选框是否被选中，如果被选中，则把这本书添加到returnBook链表中。
4. closeEvent：closeEvent：弹出一个对话框，询问用户是否保存操作，是，则把还的书从userBook链表中删掉，并把returnBook链表中的信息从数据库中删掉，否，则清空returnBook链表，返回user界面。

第八章 管理员界面

到上一章为止，我们已经实现了登录界面，用户界面，在用户界面中还实现了借书界面和还书界面。这一章，我们将介绍管理员界面。在Login界面中，当用户输入用户名和密码，按下登录按钮后，会进入LoginSystemSLot函数，在函数中我们把用户输入的用户名，密码和数据库中存储的用户名，密码，一个一个地比较，如果有相同的，则说明用户的输入正确，这时候再判断输入的用户名是管理员还是普通用户，在本项目中，为了方便起见，直接通过用户名为“root”来判断是否是管理员。

在实现Login界面时，为了能体现登录的效果，我们已经建立了一个名为AdminInterface的Qt设计师界面类，界面模板为Widget，类名是AdminInterface。

在实现Login界面时，为了能体现登录的效果，我们已经建立了一个名为AdminInterface的Qt设计师界面类，界面模板为Widget，类名是AdminInterface。

首先现在设计一下管理员界面的界面：

管理员界面的主要控件是一个Tab Widget，在第一个Tab我们在里面显示所有书籍，提供添加书籍，删除书籍的功能。在第二个Tab我们在里面显示所有用户，并提供添加用户和删除用户的功能。

Tab1中有一个Tree Widget，两个Push Button

1. Tree Widtet：显示文字：无，变量名：bookTreeWidget。
2. Push Button：显示文字：“添加书籍”，变量名：addBookPushButton。
3. Push Button：显示文字：“删除书籍”，变量名：deleteBookPushButton。

Tab2与Tab1一样，也是一个Tree Widget，两个Push Button：

1. Tree Widget：显示文字：无，变量名：userTreeWidget。
2. Push Button：显示文字：“添加用户”，变量名：addUserPushButton。
3. Push Button：显示文字：“删除用户”，变量名：deleteUserPushButton。

下面开始介绍AdminInterface的声明：

类的属性：

1. user：储存所有用户的变量。
2. addUser：储存添加的用户。
3. deleteUser：储存删除的用户。
4. book：储存所有书的变量。
5. addBook：储存添加的书。
6. deleteBook：储存删除的书。

类的方法：

1. ShowBookInfoSlot：显示书的详细信息的槽函数，当用户双击一行书时，会打开一个界面，在这个界面上详细显示书的信息。
2. AddBookSlot：添加书的槽函数，当用户点击添加书籍按钮时，会打开添加书籍的界面。在界面中填写添加的书的详细信息。
3. DeleteBookSlot：删除书的槽函数，当用户点击删除书籍按钮时，会打开删除书籍界面，在界面中显示所有书籍，让用户选择要删除的书。
4. AddUserSlot：添加用户槽函数，当用户点击添加用户按钮时，打开添加用户界面，用户在界面上输入用户的详细信息。
5. DeleteUserSlot：删除用户槽函数，当用户点击删除用户按钮时，打开删除用户界面，在界面上显示所有用户，让用户选择要删除的用户。
6. ShowAllBooks：在bookTreeWidget上显示所有的书。
7. ShowAllUsers：在userTreeWidget上显示所有用户。

类的实现：

1. 在构造函数中，先设置一下窗口标题，Tab Widget的两个tab的标题，然后初始化一下变量，最后连接一下信号与槽。注意addBook，addUser的头节点的id要设为1，表示链表中的信息要存到数据库中，deleteBook，deleteUser的头节点的id要设为-1，表示链表中的信息要从数据中删掉。这四个变量都不需要调用Init函数，Init函数是从数据库中读取信息来初始化变量的，而这四个变量是用来回写数据库的。
2. ShowBookInfoSlot：该函数的作用是申请一个BookInfo界面，并让这个界面显示出来，BookInfo界面已经在之前的内容中实现过了，这里只要new一个BookInfo对象，并调用show函数就行了。
3. AddBookSlot：等AddBook界面实现后，再来实现。
4. DeleteBookSlot：等DeleteBook界面实现后，再来实现。
5. AddUserSlot：等AddUser界面实现后，再来实现。
6. DeleteUserSlot：等DeleteUser界面实现后，再来实现。
7. ShowAllBooks：该函数的作用是在Book Tree Widget上显示所有的书，只要遍历book链表，把每个节点的信息转换成QTreeWidgetItem对象，插入Tree Widget即可。
8. ShowAllUsers：该函数的作用是在User Tree Widget上显示所有的用户，只要遍历user链表，把每个节点的信息转换成QTreeWidgetItem对象，插入Tree Widget即可。

第十章 添加书籍

添加书籍界面的设计思想很简单，是在界面上显示几个输入框，让用户填写书籍的信息，在用户点确认按钮后，需要判断用户是否填写了信息

新建一个Qt设计师界面类，界面模板选Widget，类名改为AddBook。

下面开始添加书籍界面的界面设计，借书界面有五个Label用于提示用户输入什么，四个LineEdit和一个TextEdit用于接收用户的输入，两个Push Button：

1. Label：显示文字：“书籍编号”，变量名：没用到，可不该。
2. Label：显示文字：“书籍名称”，变量名：没用到，可不该。
3. Label：显示文字：“书籍作者”，变量名：没用到，可不该。
4. Label：显示文字：“书籍类型”，变量名：每用到，可不该。
5. Label：显示文字：“书籍介绍”，变量名：没用到，可不该。
6. LineEdit：显示文字：无，变量名：idLineEdit。
7. LineEdit：显示文字：无，变量名：nameLineEdit。
8. LineEdit：显示文字：无，变量名：authorLineEdit。
9. LineEdit：显示文字：无，变量名：categoryLineEidt。
10. TextEdit：显示文字：无，变量名：introductionTextEdit。
11. Push Button：显示文字：“确定”，变量名：confirmPushButton。
12. Push Button：显示文字：“取消”，变量名：cancelPushButton。

类的声明：

类的属性：

1. book： 储存所有的书的信息。
2. addBook：储存添加的书信息。

类的方法：

1. closeEvent：拦截关闭事件，询问用户是否保存操作。
2. AddBookSlot：用户点确定按钮后，保存用户输入的书信息。

类的实现：

1. 构造函数：设置窗口标题，初始化变量，连接信号与槽。
2. AddBookSlot：首先判断用户是否有输入，这里只判断了用户是否输入了书的id，然后new一个BookType类型的变量，根据用户的输入来给变量的属性赋值，最后把该变量插入addBook链表。
3. closeEvent：先弹出一个对话框，询问用户是否保存操作，用户点Save按钮后，就把用户添加的书插入book链表，并保存到数据库中，用户点Cancel按钮后，则清空addBook链表。

第十一章 删除书籍

删除书籍界面的实现与还书界面的实现非常相似。

新建一个C++类，雷名为DeleteBook，public继承自Form，在类声明中添加Q\_OBJECT，在deletebook.cpp 中添加ui\_form.h头文件，执行一下qmake重新构建一下makefile文件，否则qt会找不到文件。

类的声明：

类的属性：

1. book：储存所有的书的信息。
2. deleteBook：储存删除的书的信息。

类的方法：

1. ShowAllBooks：显示所有的书，让用户选择要删除的书。
2. DeleteBookSlot：进行删除书的操作。
3. closeEvent：拦截关闭事件，自定义界面关闭时要进行的操作。

类的实现：

1. 先设置一下窗口的标题，Tree Widget的项数量，以及项名称。然后初始化一下变量。最后把信号与槽连接起来。
2. ShowAllBooks：遍历book链表，把每一个节点的信息转换成QTreeWidgetItem对象，插入Tree Widget中。每一行都要在第0项设置一个复选框，让用户进行选择操作。
3. DeleteBookSlot：遍历Tree Widget的每一行，检查每一行的复选框是否被选中，如果被选中，就把选中的书添加进deleteBook链表，因为时删除操作，所以只要记录书的id就可以了。把书添加进deleteBook后还要把复选框置回未选中状态。
4. closeEvent：拦截关闭事件，提示用户是否保存操作，如果用户点击Save按钮，把删除的书从book链表中删掉，同样要在数据库中删掉。如果用户点击Cancel按钮，把deleteBook链表清空即可。

第十二章 添加用户

添加用户的界面可以是和添加书籍的界面基本类似，只是有一点实现细节不同。

添加一个Qt设计师界面类，界面模板选择Widget，类名改为AddUser。

添加用户界面共有三个Label作为提示信息，三个LineEdit用于接受用户收入，两个Push Button。

1. Label：显示文字：“用户编号”，变量名：未用到，可不改。
2. Label：显示文字：“用户名”， 变量名：未用到，可不改。
3. Label：显示文字：“用户密码”， 变量名：未用到，可不改。
4. LineEdit：显示文字：无，变量名：idLineEdit。
5. LineEdit：显示文字：无，变量名：nameLineEdit。
6. LineEdit：显示文字：无，变量名：passwordLineEdit。
7. Push Button：显示文字：“确定”，变量名：confirmPushButton。
8. Push Button：显示文字，“取消“，变量名：cancelPushButton。

类的声明：

类的属性：

1. user：储存所有有用户的信息。
2. addUser：储存添加的用户的信息。

类的方法：

1. AddUserSlot：添加用户的槽函数，当用户点击确定按钮后，进入此函数，处理添加用户的具体操作。
2. closeEvent：拦截关闭事件，自定义界面关闭时要处理的事务。

类的实现：

1. 先设置一下窗口标题，然后初始化变量，最后连接一下信号与槽。
2. AddUserSlot：先判断是否输入了信息，没有就退出，如果输入了信息，把输入的信息转换成UserType对象，添加到addUser链表中去。
3. closeEvent：弹出对话框，询问用户是否保存操作，如果用户选择Save，把addUser中的信息添加到user链表中去，同时也把它们保存到数据库中去。如果用户选择Cancel，清空addUser链表即可。

如果出现错误：

admininterface.obj : error LNK2019: 无法解析的外部符号 "public: \_\_cdecl AddUser::AddUser(class User \*,class User \*,class QWidget \*)" (??0AddUser@@QEAA@PEAVUser@@0PEAVQWidget@@@Z)，该符号在函数 "public: void \_\_cdecl AdminInterface::AddUserSlot(void)" (?AddUserSlot@AdminInterface@@QEAAXXZ) 中被引用

debug\LibraryManageSystem.exe : fatal error LNK1120: 1 个无法解析的外部命令。

执行一下qmake就行了。

第十三章 删除用户

删除用户的界面与删除书籍界面的实现类似，只是会有一点细节差异。

添加一个C++类，类名为DeleteUser，public继承于Form，在类的声明中添加Q\_OBJECT宏，在deleteuser.cpp 中添加ui\_form.h头文件。执行一下qmake。

类的声明：

类的属性：

1. user：储存所有用户的信息。
2. deleteUser：储存删除的用户的信息。

类的方法：

1. ShowAllUsers：显示所有用户的信息，让用户选择要删除的对象。
2. DeleteUserSlot：把用户选择要删除的用户添加到deleteUser链表中去。
3. CloseEvent：拦截关闭事件，自定义一下在界面关闭时要处理的事务。

类的实现：

1. 在构造函数中，先设置一下窗口的标题，设置Tree Widget要显示几列信息，以及每一列的标题。然后初始化一下变量，最后连接一下信号与槽。
2. ShowAllUsers：把所有的用户的信息转换成QTreeWidgtItem对象，插入到Tree Widget中，在每一行的第0项，设置一个复选框，让用户选择要删除的对象。
3. closeEvent：拦截关闭信息，在界面关闭的时候，弹出一个对话框，询问用户是否保存操作，如果选择Save，把deleteUser联保中的信息从user链表中删掉，同时也从数据库中删掉。如果选择Cancel，只需清空deleteUser链表即可。