《家谱管理系统》分析报告

——数据结构课程大作业报告

小组成员：张嘉伟 2019302141095、马薪宇 2019302080309

一、需求分析

《家谱管理系统》程序的设计目的，是为了解决中国传统家谱不易保存、不易修改、不易统计的缺陷。利用计算机程序，可以实现在计算机上存储、管理、查看家谱的相关信息。

目标功能：

1.建立家谱：在计算机上建立树状家谱结构。要求用户友好，便于不懂计算机的人使用，因此需要制作图形用户界面、支持鼠标操作，操作逻辑与市面常规操作系统一致。

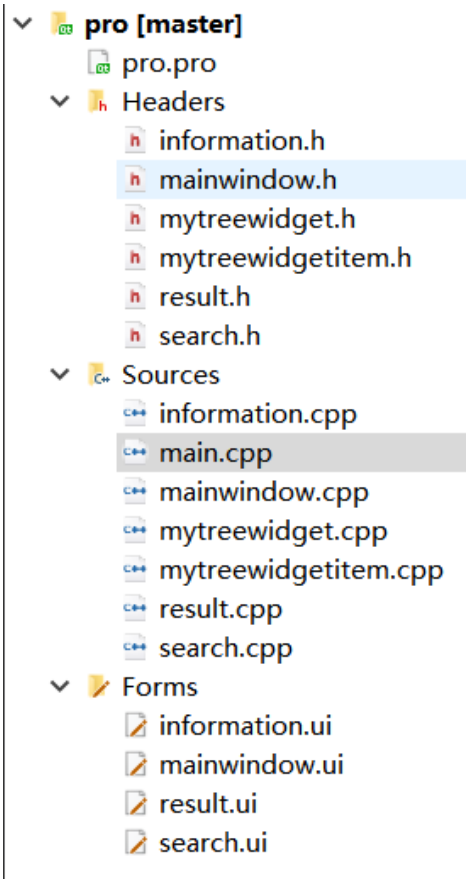
2.管理信息：方便对家谱成员进行信息管理。其中要求实现家谱成员的信息修改、成员的插入与删除，信息要求包括一个成员的基本信息，包括：姓名、性别、出生日期、死亡日期、出生地点、身高等。要求支持鼠标操作，操作逻辑简单、与市面常规操作系统一致。

3.统计信息：为了满足对一整个家族的研究，系统要求支持对家谱中所有成员信息进行统计，包括对医学研究有价值的信息，如：年龄、身高等。要求实现鼠标操作，操作逻辑简单、与市面常规操作系统一致。

4.存储信息：家谱管理要求实现信息存储，方便信息的重复查看、使用。要求包括的功能有：打开程序自动进行信息的初始化，将计算机内存储的数据自动初始化为家谱数据；完成信息的修改后可以实现保存，将家谱信息保存至计算机。

二、项目设计

1.总体设计：



项目的头文件、源文件及ui文件

在家谱管理系统项目中，我们创建了四个ui界面、六个头文件及七个源文件。

其中四个ui界面分别用于录入信息、主窗体的显示、统计值信息的显示、依据输入姓名搜索目标信息。

头文件中：

mainwindow.h 中声明 MainWindow类 ， 包括将Item的相关信息输出到MainWindow.ui中相应标签中、点击搜索/统计按钮，弹出相应窗口、进行搜索等；相关函数与信号和槽的连接在 mainwindow.cpp 中实现；

information.h 中声明 Information类 ， 包括录入的相关信息，信息的返回等；相关函数在 information.cpp 中实现；

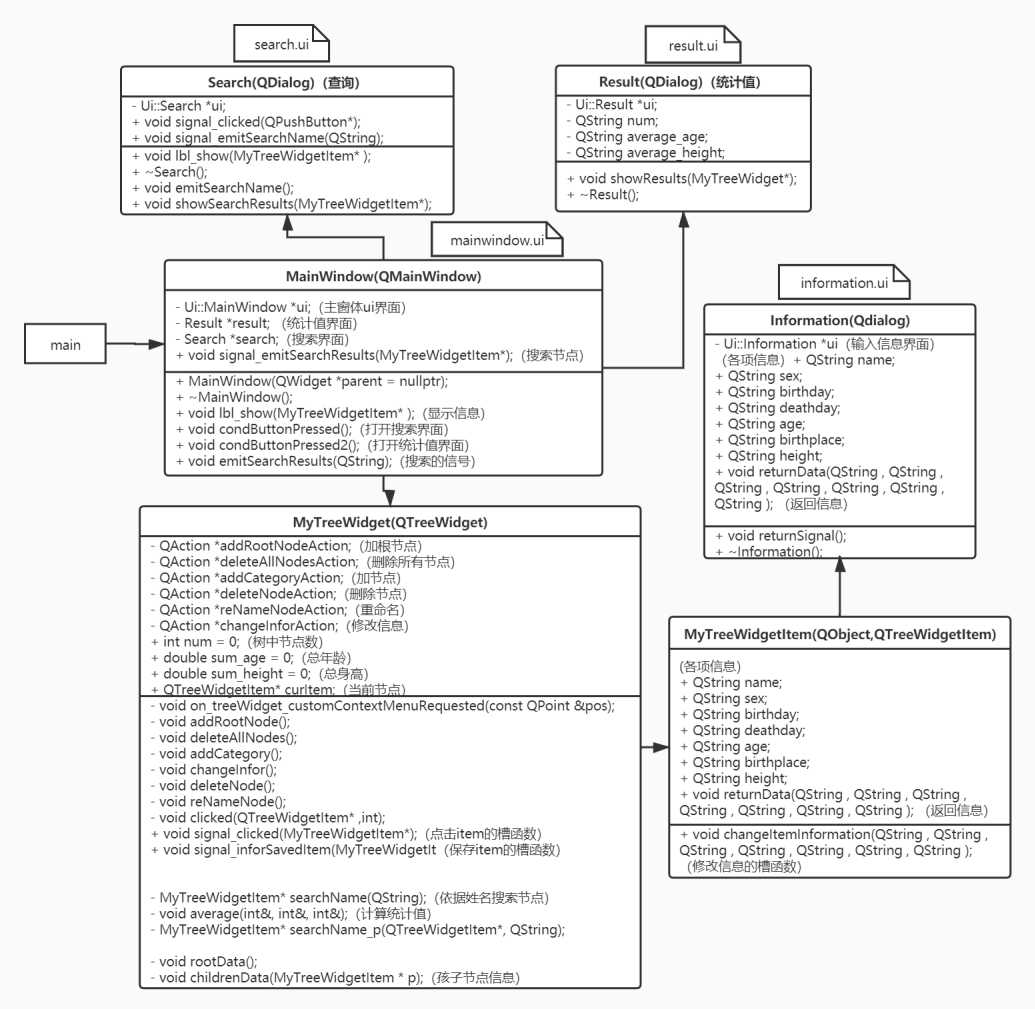
mytreewidget.h 中声明了 MyTreeWidget类，包括：点击按钮显示结点的信息、新建结点、删除结点、对结点进行信息的修改、重命名结点等功能函数的声明；相关函数与信号和槽的连接在 mytreewidget.cpp 中实现；

mytreewidgetitem.h 中声明了 MyTreeWidgetItem类，包括修改结点信息的函数声明；相关函数在 mytreewidgetitem.cpp 中实现；

result.h 中声明了 Result类，其中有全部结点的信息的统计值，如总结点数、平均年龄、平均身高，并声明了输出统计值的函数；相关函数与信号和槽的连接在 result.cpp、mytreewidget.cpp 中实现；

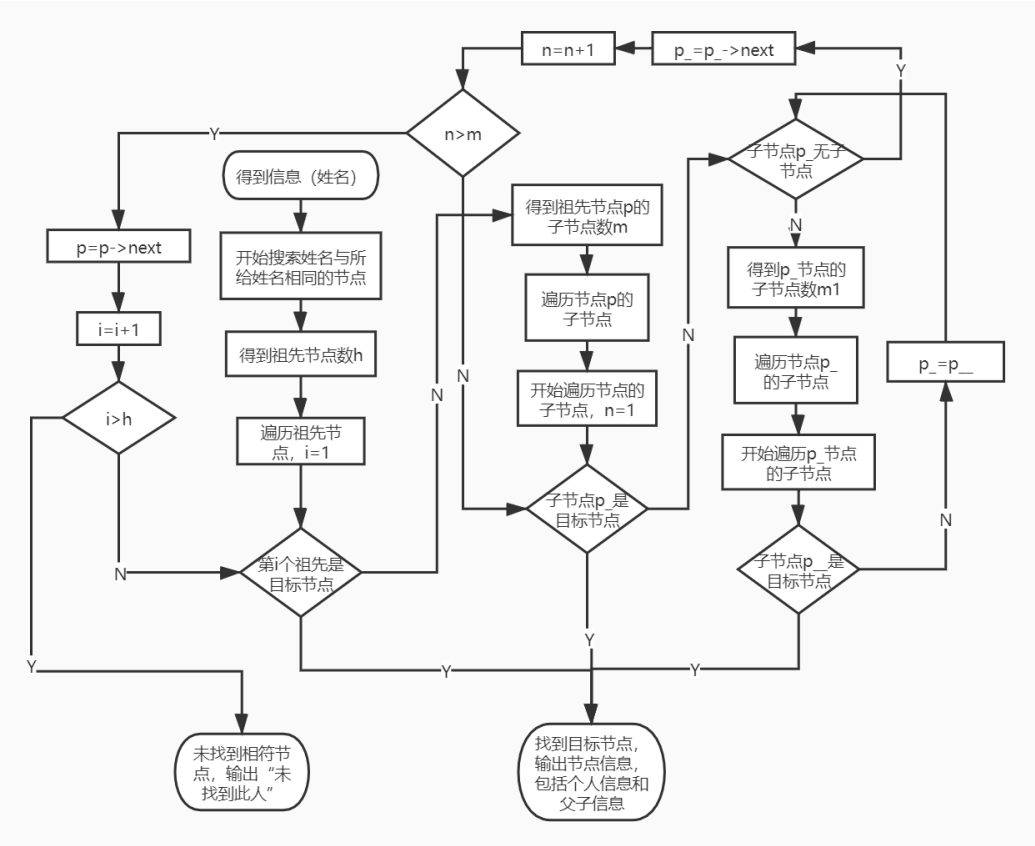
search.h 中声明了 Search类，包括依据输入的姓名进行搜索的函数声明，和搜索结点信息输出函数的声明；相关函数与信号和槽的连接在 search.cpp 、mytreewidget.cpp 中实现。

main.cpp 中的main函数作为程序的起点，运行程序。



项目设计的类图

2.算法设计与分析：



查找算法的流程图

```
MyTreeWidgetItem* MyTreeWidget::searchName(QString name)
{
    MyTreeWidgetItem* p;
    int count;
    count = topLevelItemCount();
    for(int i=0;i<count;i++)  //依次访问所有根节点
    {
        p = (MyTreeWidgetItem*)topLevelItem(i);
        if(p->name == name)  //若该结点为目标结点，则返回该结点
            return p;
        else
            p=searchName_p(p, name); //否则递归调用

        if(p != NULL) return p; //若递归后在此根节点下得到了目标结点，返回该结点；否则访问下一个根节点
    }
    return NULL;  //全部访问结束返回NULL
}

MyTreeWidgetItem* MyTreeWidget::searchName_p(QTreeWidgetItem * p, QString name)
{
    int count = p->childCount();  //获取当前结点的子代个数
    MyTreeWidgetItem* p_child;
    for(int i=0;i<count;i++)     //依次访问各个子代
    {
        //MyTreeWidgetItem* p_child = (MyTreeWidgetItem*)p->child(i);
        p_child = (MyTreeWidgetItem*)p->child(i);
        if(p_child->name == name)  //若找到所需结点，就返回该结点
        return p_child;
        else return searchName_p(p_child,name); //否则递归调用
    }
    return NULL;  //全部查找完没有找到，返回空指针
}
```

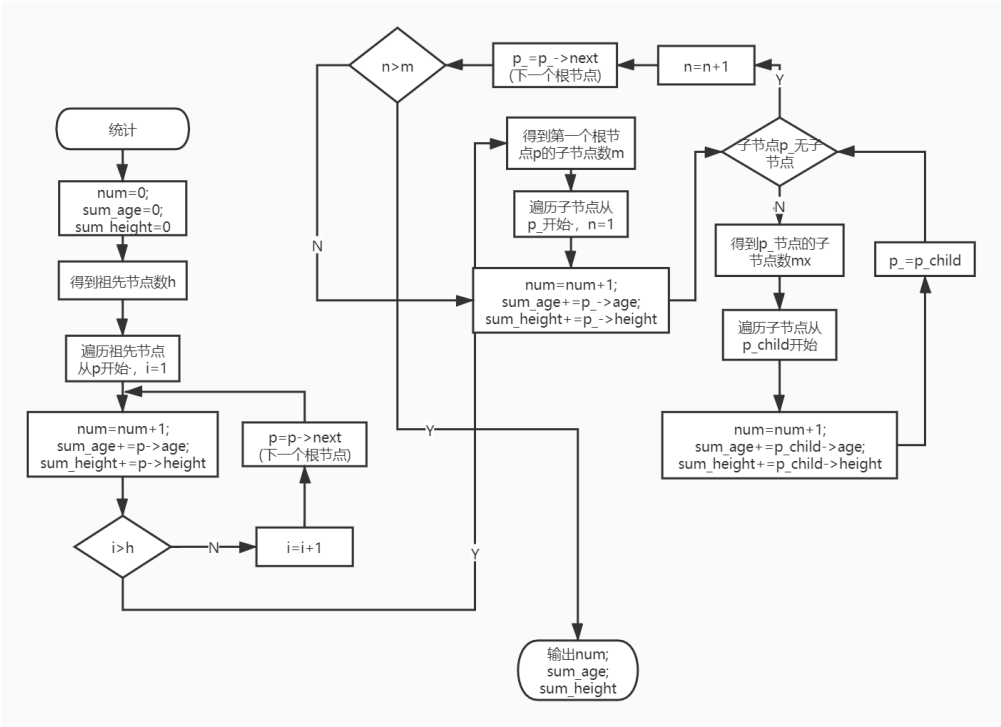
查找算法的代码部分

根据人名进行查找时，我们点击主窗口的查找按钮，在输入一个人名后，点击查找的按钮，进行查找。

查找的原理是利用for循环依次访问根结点，取结点的人名信息，与输入的人名进行比对，如果相等，则找到了目标结点，返回此结点。

如果在根结点中没找到，就依次遍历根结点的子结点，先获得当前结点的子结点个数，用for循环访问各个子代，如果找到了目标结点，返回此结点，否则就对此函数进行递归调用，这样可以在找到目标结点前一直遍历，当遍历所有结点完我们仍没有找到目标结点，则说明树中没有此人。

如果找到此人，会输出他的父亲和孩子，同时输出他的个人信息；未找到此人，输出“未找到此人”。



统计算法的流程图

```
void MyTreeWidget:: rootData()
{
    MyTreeWidgetItem* p;
        int count;                          //顶层节点数
        count = topLevelItemCount();
        for(int i = 0; i < count; i ++)
        {
            p = (MyTreeWidgetItem*)topLevelItem(i);//第i个顶层节点
            num += 1;
            sum_age += p->age.toInt();
            sum_height += p->height.toInt();
            childrenData(p);                //把子节点加进来了
        }
        qDebug()<<"num = "<<num;
        qDebug()<<"sum_age = "<<sum_age;
        qDebug()<<"sum_height = "<<sum_height;
}

void MyTreeWidget:: childrenData(MyTreeWidgetItem *p)
{
    int count = p->childCount();    //p父节点的子代个数
        MyTreeWidgetItem * p_child;
        for (int i = 0; i < count; i++)//循环对子节点i进行数据统计
        {
            p_child = (MyTreeWidgetItem *)p->child(i);
            num += 1;
            sum_age += p_child->age.toDouble();
            sum_height += p_child->height.toDouble();
            childrenData(p_child);      //子代的子代进行数据统计
        }
}
```

统计算法的代码部分

根据树的数据统计时，我们点击主窗口中的统计按钮，在统计界面中，点击统计数据的按钮，进行数据的统计与输出，统计的数据有家谱中总人数、平均年龄、平均身高。

数据统计的原理是用for循环依次访问树中的结点，首先遍历每个根结点，每访问一个结点，对num进行加1、对sum_age进行累加、对sum_height进行累加。

依次遍历子结点，先获得当前结点的子结点个数，用for循环访问各个子代，对各个数据进行累加，再对子结点递归调用此函数，这样可以遍历所有除根结点之外的结点数据，和最初对根结点的数据一起组成了累计的数据。

再对数据进行处理，求均值，输出到相应的ui的label上。

3.用到的数据结构：

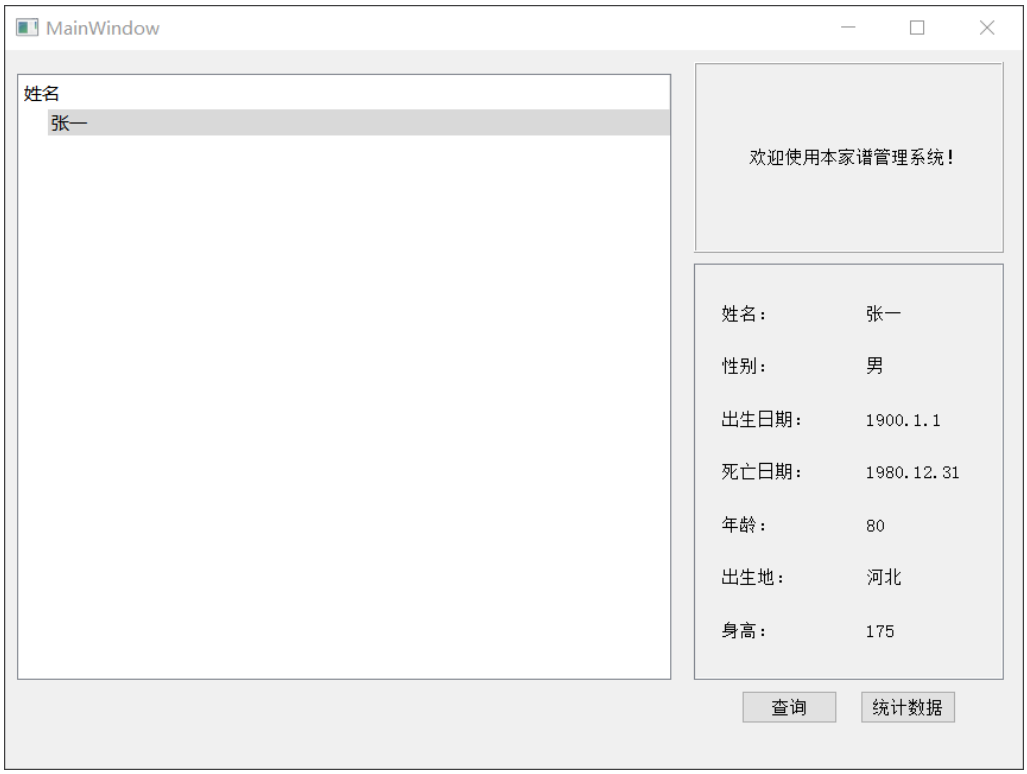
项目中用到了树形结构，一个结点可以和多个结点相连，Qt中的QTreeWidget运用的就是树形结构对数据进行存储，我们在项目中对树中结点进行遍历，用到的是深度优先的遍历方式，其形式是递归。

利用树形结构，能把家谱中父子关系表达得很清晰，利用深度优先的遍历方式使遍历有逻辑地进行，使结点能按顺序地进行搜索。

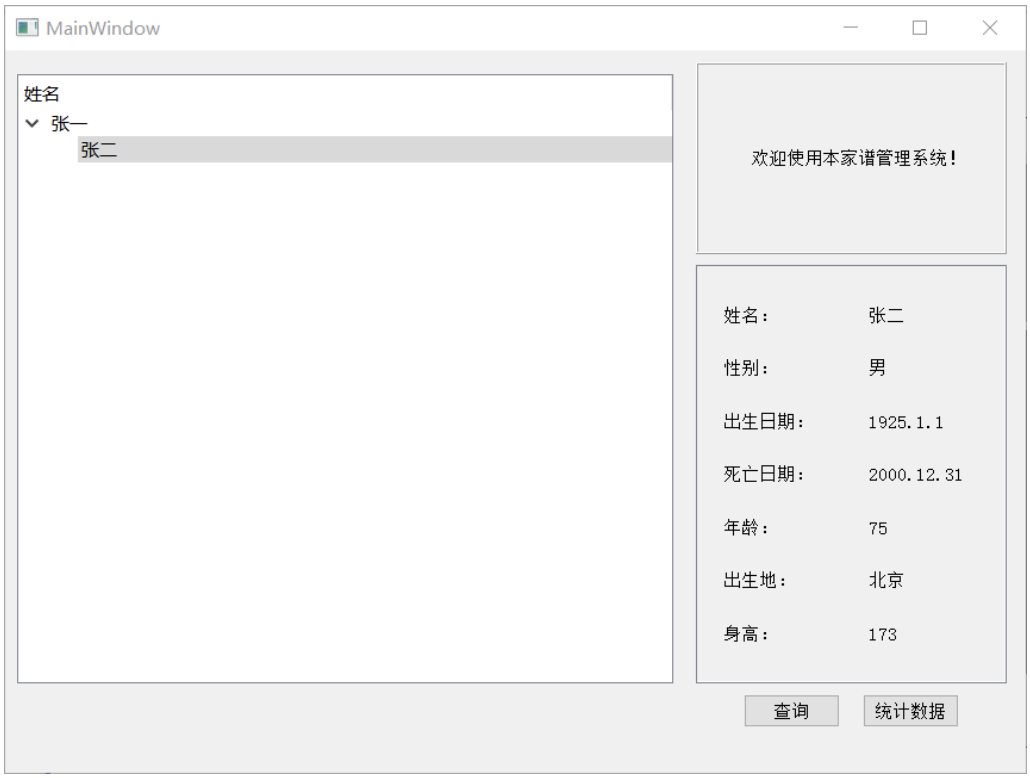
三、测试报告

1.合法数据测试

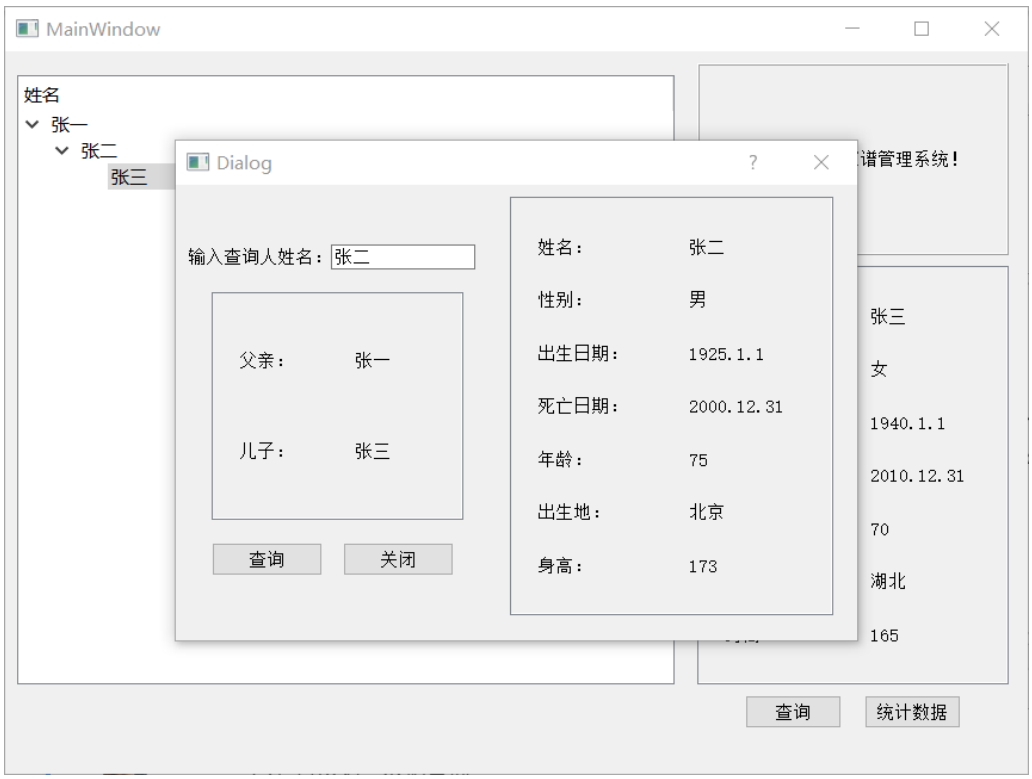
（1） 合法祖先结点的建立：



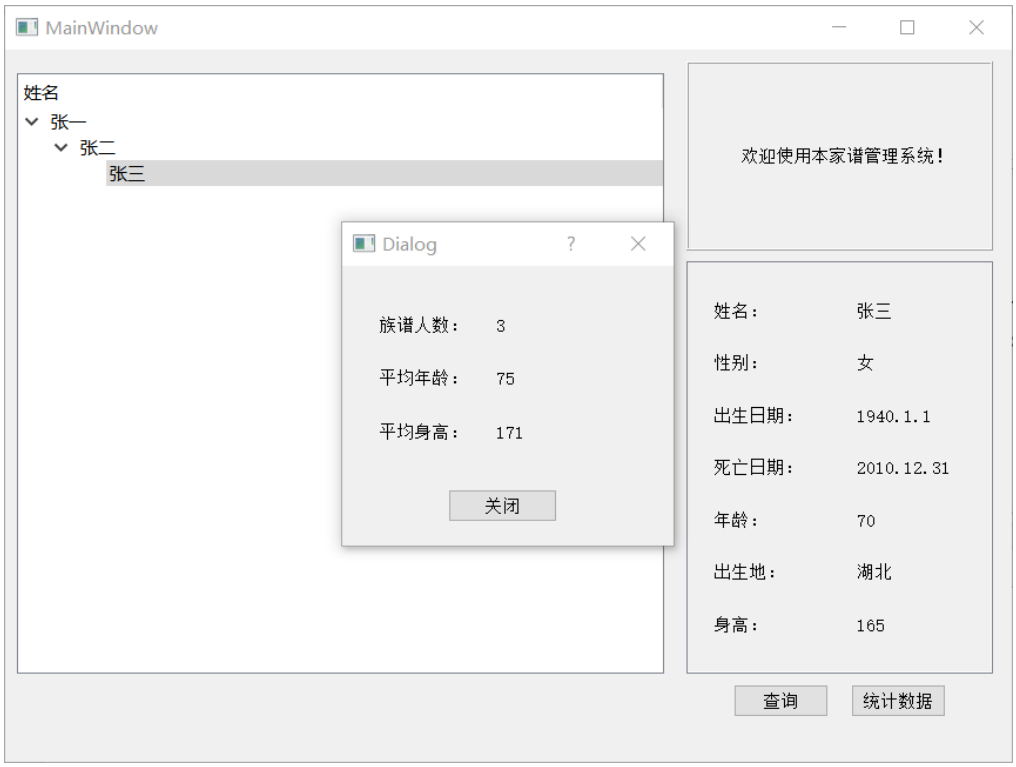
（2）合法子代结点的建立：



（3）合法结点数据查询：

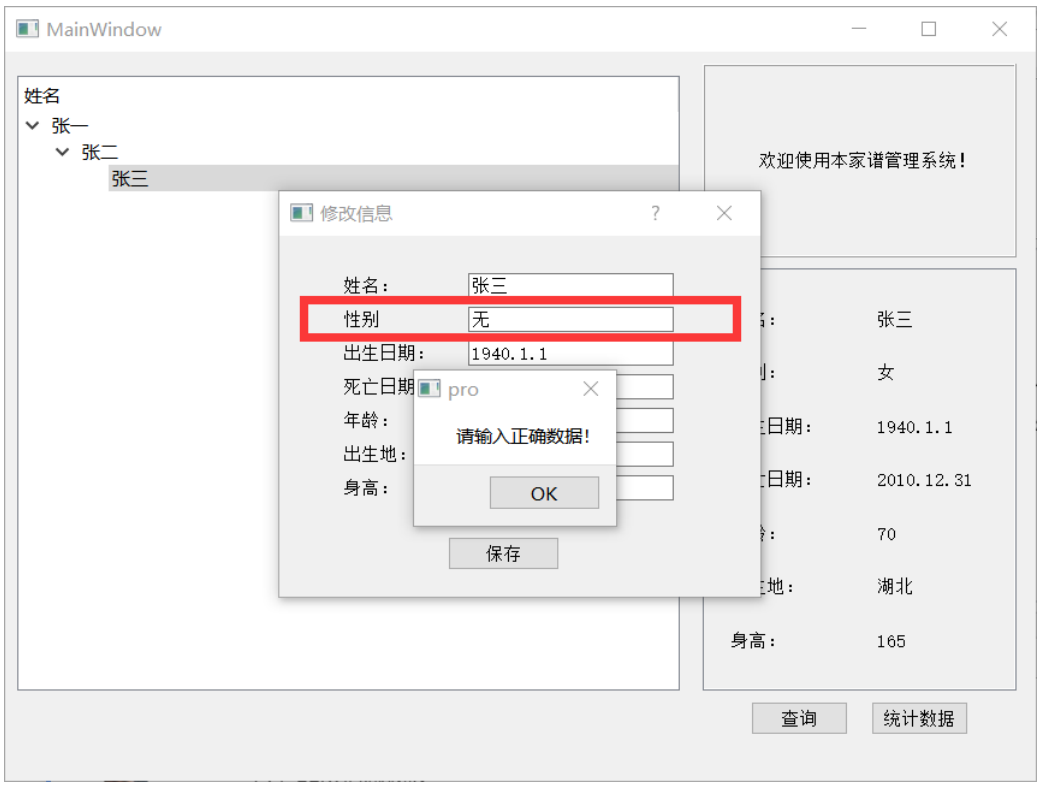


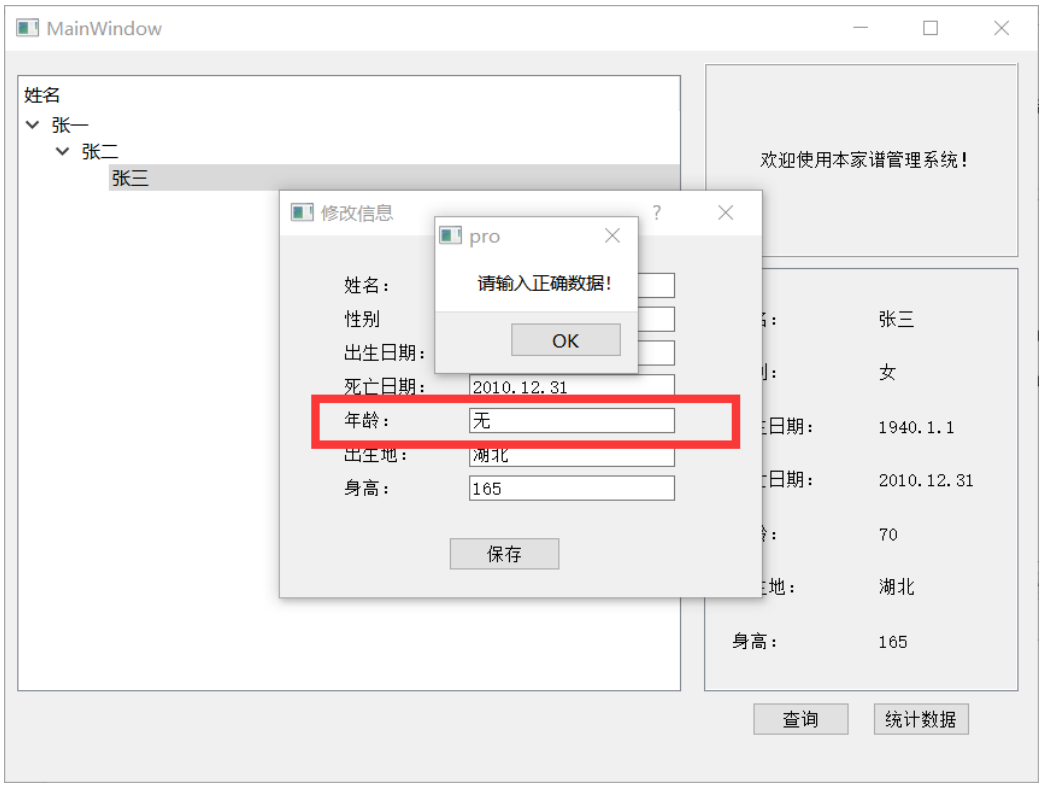
（4）合法数据统计：

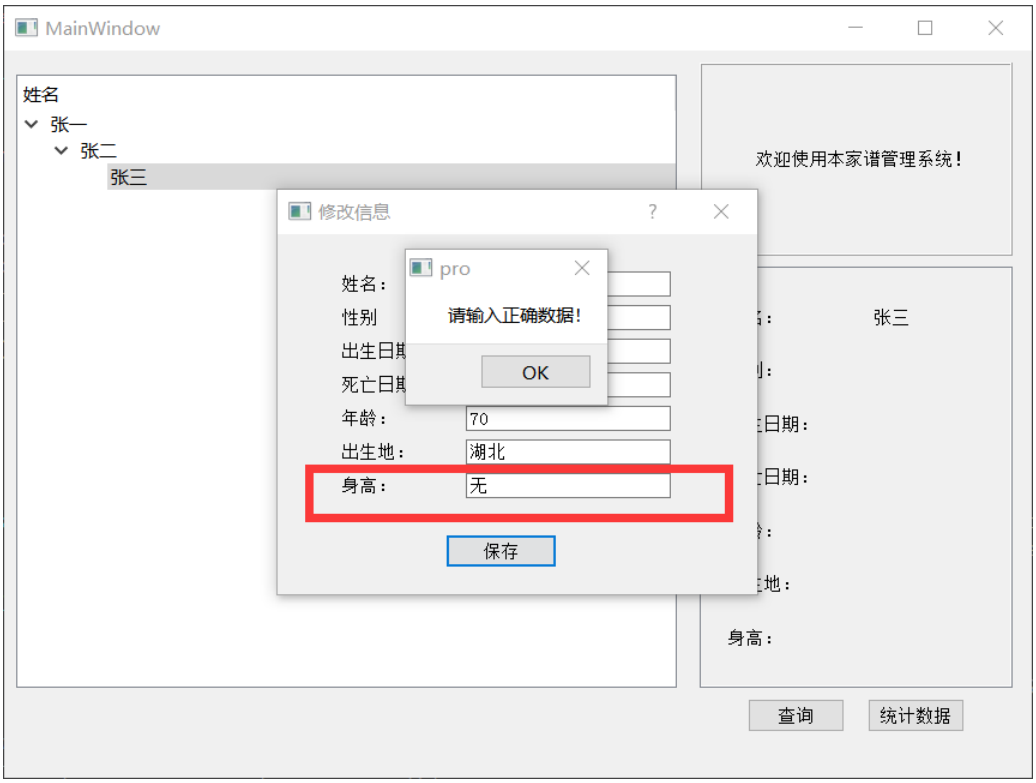


2.非法数据测试

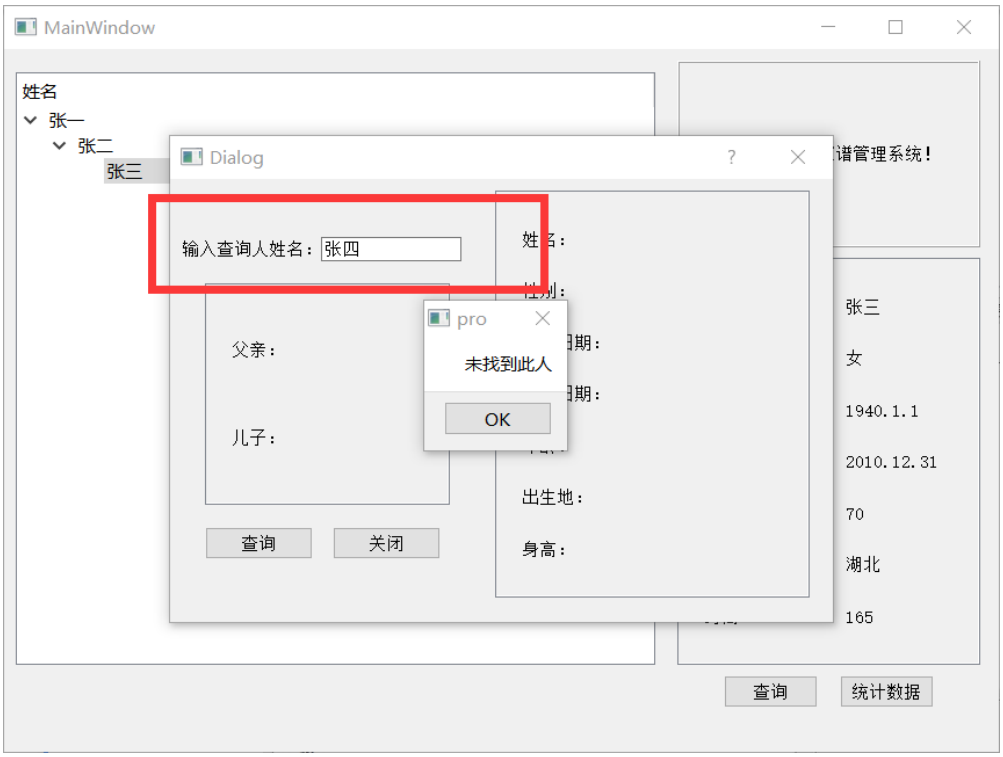
（1）非法信息修改（性别、年龄、身高）：




MainWindow
姓名
张一
张二
张三
修改信息
pro
请输入正确数据!
OK
姓名：
性别
出生日期：
死亡日期：
2010.12.31
年龄：
无
出生地：
湖北
身高：
165
保存
欢迎使用本家谱管理系统!
张三
女
1940.1.1
2010.12.31
70
湖北
165
查询
统计数据


MainWindow
姓名
张一
张二
张三
修改信息
pro
请输入正确数据!
OK
姓名：
性别
年龄：
70
出生地：
湖北
身高：
无
保存
欢迎使用本家谱管理系统!
张三
身高：
查询
统计数据

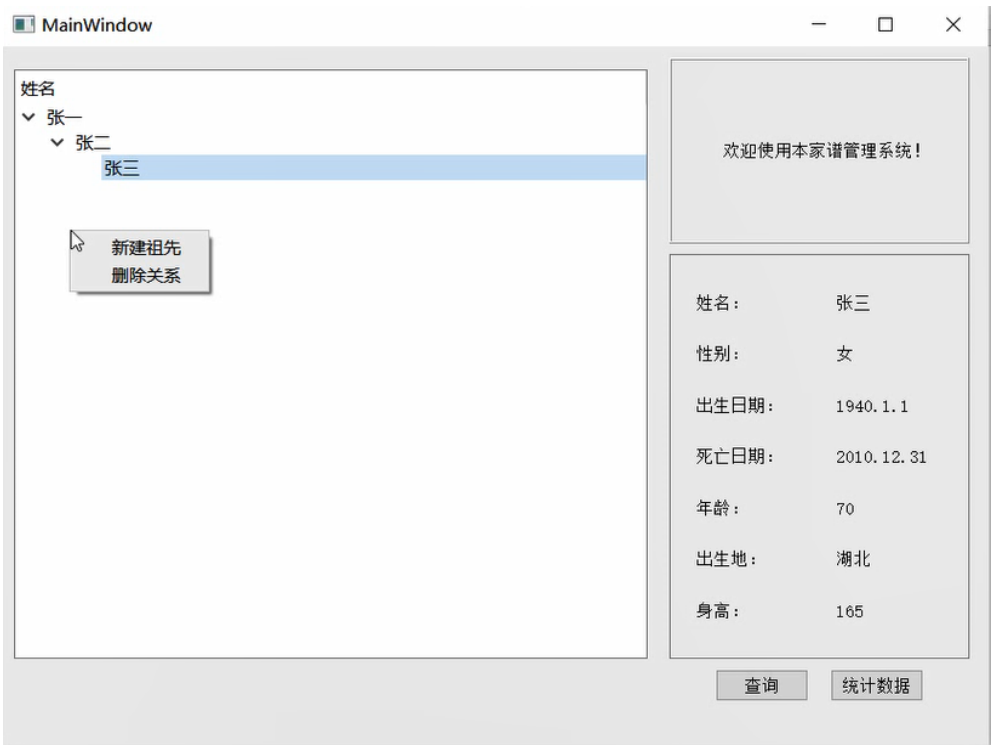
（2）无效成员查询



3.功能展示

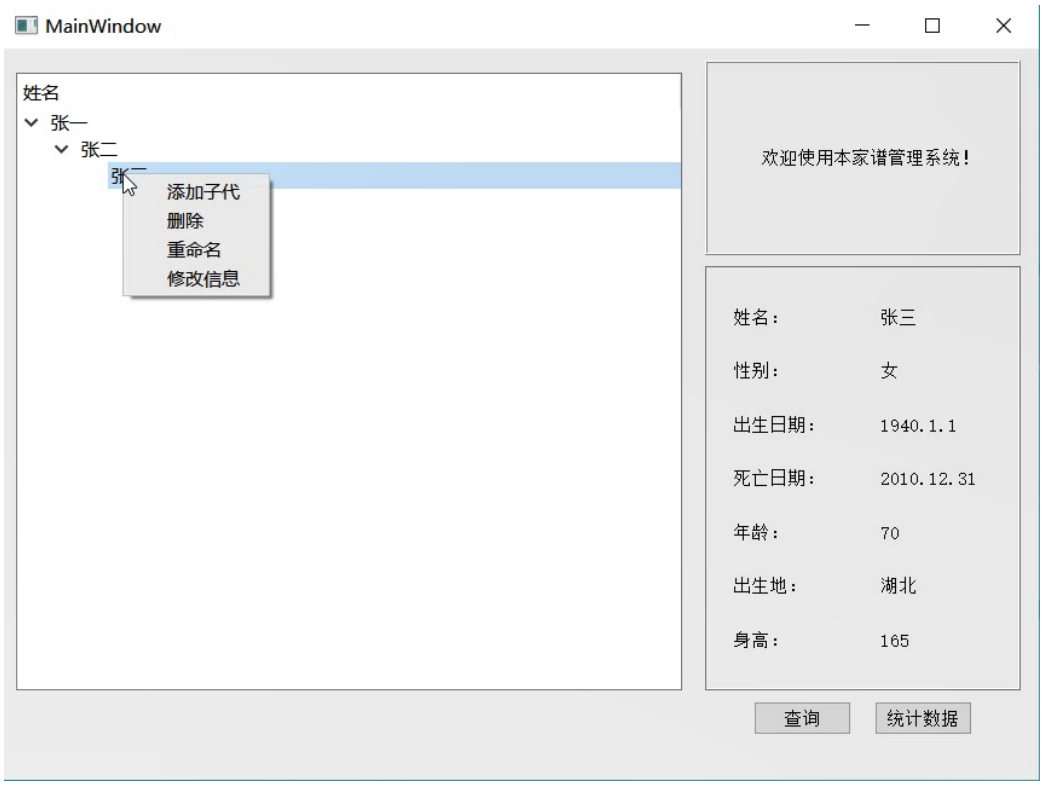
（1）祖先结点创建与删除

在空白区域单击鼠标右键，弹出选项



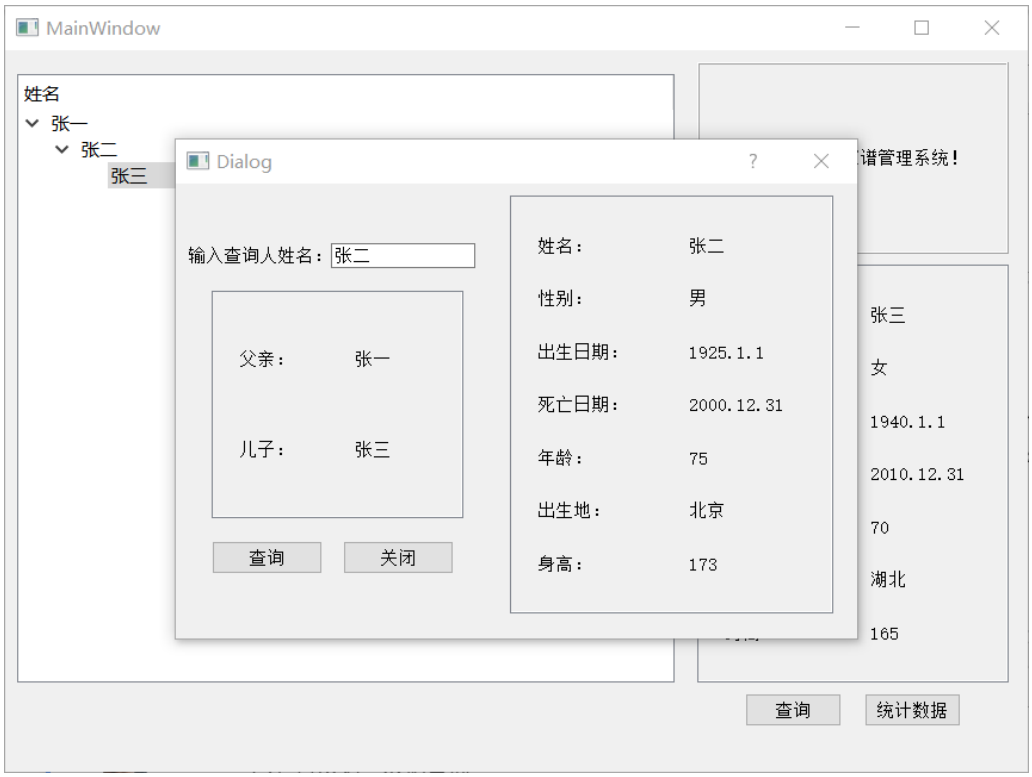
（2）子代结点创建与删除

选中结点单击鼠标右键，弹出选项

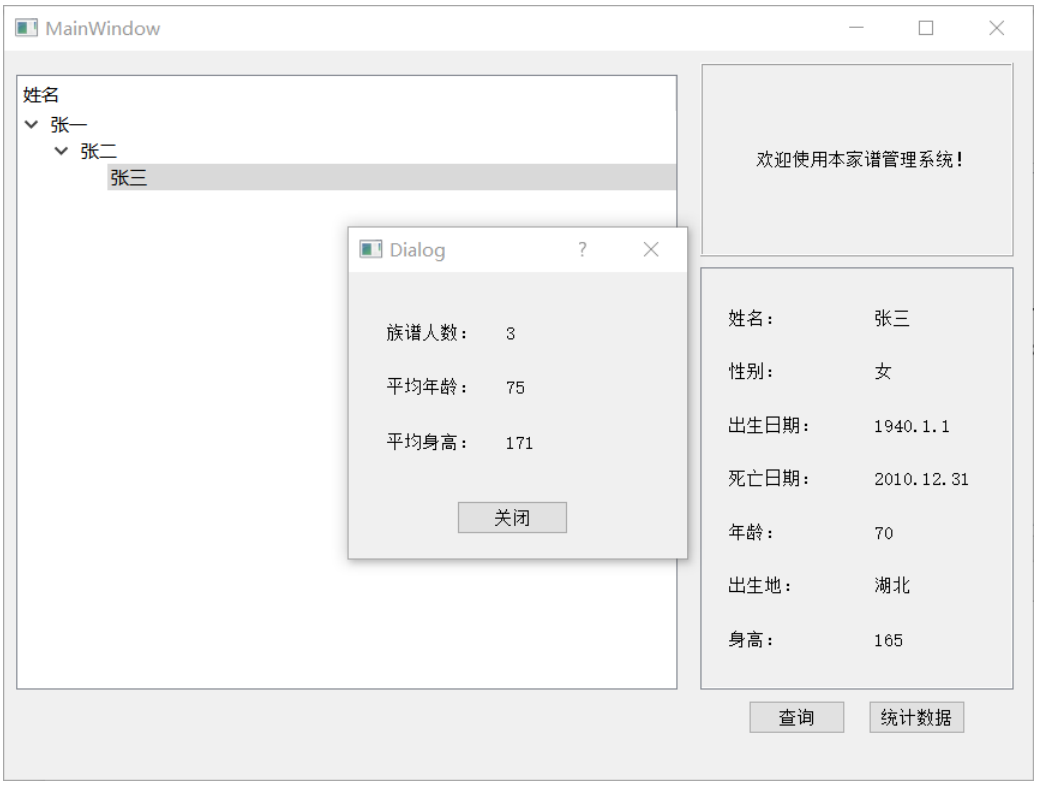


(3) 成员信息查询

点击右下方”查询“按钮



(4) 家谱成员数据统计



四、设计过程中遇到的问题及解决方法

问题一：图形用户界面的制作。

解决方法：我们小组之前并没有图形用户界面的开发经验，在接到这个大作业后，我们先是在网络上查找了有关图形用户界面的制作工具，了解到Qt功能强大、跨平台性能好的特点后，我们即选用了Qt作为我们的开发工具。并且在本学期通过网络课程+网络文档+图书的方式，初步学习了Qt开发的基本过程和主要功能，并在此基础上使用Qt完成了本次大作业。

问题二：Qt中树对象内部每个结点信息的存储。

解决方法：在创建每个成员信息作为结点时，发现把成员信息存储到Qt原有的--QTreeWidgetItem类中较为复杂，因此必须继承该类，派生一个包含成员信息的新的类--MyTreeWidgetItem。但是在派生过程中，由于对C++继承与派生机制掌握的不扎实，自己没有注意到很多之前使用的QTreeWidget对象中对QTreeWidgetItem的函数将不再适用于自己派生的对象。后来在网络上查询后了解到，编译器在编译时，会建立一个类的索引表，根据指针类型来确定指针指向对象成员的偏移量，所以对父类对象使用派生类对象指针不用进行类型转换，但是对派生类对象使用父类对象指针必须进行强制类型转换。

问题三：Qt树组件中遍历、查找的算法设计。

解决方法：Qt中树结构是封装成类的，无法直接通过指针获得父类或者子类结点，通过查询Qt的官方文档，找到了QTreeWidget和QTreeWidgetItem封装好的函数parent( )和child( )来返回父子结点的指针，以及childcount( )返回孩子结点个数。有了这些函数，就可以通过递归子树来实现遍历、查找功能。

五、尚未解决的问题及考虑应对的策略

本程序在设计**计算机存储功能**时遇到了困难。目标是用户在点击保存功能按钮时，将每个成员的信息输出至目标计算机硬盘目录上的文本文档中；而用户在重新打开此程序时，程序会自动读取目标计算机硬盘目录上的文本文档，实现通过文本文档数据初始化成员信息，从而实现程序的重复多次使用。

在实际编写程序时，发现对于此类非二叉树的更一般的树结构，不太容易通过较为简单的方式实现相同类型、规格数据初始化一棵树。在向一些同学请教、上网查找方法之后，我们还

是没能够在有限的时间里完成这个很重要的功能，但是也有了一些想法。初步定下来的未来实现思路有两种。

思路一、使用”索引“

常规的通过数列初始化二叉树的方法对于家谱这种”多叉树“已经不适用，为此我们想到可以在数据存储时，为每个成员添加一个”索引“。可以在索引中标注其孩子结点的名字，在初始化此家谱时，初始化完成父亲结点后，通过索引找到孩子结点；建立好孩子结点后，通过孩子结点的索引继续查找孩子结点的孩子……通过递归方法，按照深度优先的方向建立树结构。

但是这种方法运行所消耗的时间可能会比较多，读取文件操作本来就是比较耗时的指令，遍历文件的次数又很多，在数量大的情况下，可能会花费较多时间。

思路二、使用数据库

在网络上查找相关的内容时，我们发现CSDN上有一名作者初始化树状结构时使用了数据库，通过数据库提供的丰富功能，方便地实现了数据的保存和树的初始化。但是我们小组两人都没有数据库操作知识作为基础，最后剩下的自学时间也不太够用。

没能完成如此重要的功能，我们的程序就只能算一个半成品，我们两个人都很不甘心。我们小组希望在寒假的时候，再花费一点时间将数据保存和初始化功能完善好，使该程序成为一个完整可用的家谱管理系统。

六、收获和心得

张嘉伟：自己在分工完成大作业时主要负责Qt界面、事件响应等方面的内容，所以首先是让我对Qt的使用变得更加熟练，对Qt的文件架构以及信号和槽的机理有了深刻的理解，以后可以进一步学习Qt的高级功能。

得益于Qt控件继承与派生的实现方式，在debug过程中让我对于面向对象以及C++的继承派生机制有了更深的感悟。在敲代码的过程中，自己也慢慢地感受到Qt一类绘制图形界面的工具的”工具性“，很多东西的使用需要的是查文档的能力，真正的内功其实还是它们基于的诸如C++继承派生等的特性，理解好这些才是计科专业真正重要的。

另外，和朋友完全独立地开发项目，也很大程度上地提高了自己的代码能力和开发信心，在之后的学习过程中自己也会努力做到多合作讨论、多动手敲代码，提高自己的学习效率和学习能力。

马薪宇：在这次大作业项目中，自己进行了部分ui界面的设计，并整理书写树形结构遍历的函数，完成大作业的过程中，促使我去学习运用Qt这个软件进行图形化界面的设计，大概学习了解了Qt基本的使用方法和查看帮助帮助文档工具的能力，也了解并实现了简单的信号和槽的连接。

在项目创立初期，我对项目的实现比较茫然，但事实证明只要沉下心去思考实践，总会有方法克服一个个问题，虽然我们的项目到最后因为时间原因没有来得及完善，但基本功能和树形结构中数据的遍历都做到了，我也在这次大作业的实现过程中学到很多实用的技能，慢慢培养起自己阅读代码的能力。同时自己也认识到合作完成项目过程中的和搭档交流的重要性，不仅能更合理地完成对项目的规划，同时团队一起出谋划策也能更好地缓解各方面的焦虑，使团队更好地合作。

之后也会多尝试项目的开发，提升自己的代码能力和计算思维，丰富自己的计算机专业素质。