《数据结构与算法设计》
课程设计总结

学号：**2050525**

姓名：陈开煦

专业：计算机科学与技术

2022 年 8 月

目录

1 算法实现设计说明

1.1 题目

按照相应选题规则，我选了算法实现题的第 5 题《二叉树》，题目描述如下：

二叉树，完成：

(1) 建立一棵二叉树，并对它进行先序、中序、后序遍历，

(2) 统计树中的叶子结点个数；

(3) 分别对它进行先序、中序、后序线索化；

(4) 实现先序、中序线索树的遍历；

显示该树和线索化后的树（此要求可视情况选择是否完成）。

1.2 软件功能

该软件名为 *BiTreeThreadTraverse*，是一个包含了二叉树线索化及遍历功能的演示程序。具体包含以下功能:

- 根据用户在 *spinbox* 控件中的选择，建立层数为 2-6 的满二叉树。
- 根据用户在程序自动建立的满二叉树结点上的点击事件进行结点的删除，以满足用户建立不同二叉树的需求。

  实现方式: 在此程序中，我将树结点在 GUI 界面中用一个 QLabel 的方式表示，用 *QMap* 这样一个基于红黑树的结构，将用户可以直观看到的 label 同内存中的 *treeNode* 结构体绑定，在用户点击的同时进行内存中结点的删除操作。
- 返回二叉树的结点个数

  实现方式: 采用递归的方式，若当前结点为空，则返回数量 0，作为递归终点；否则返回左子树和右子树的结点数和 + 1。
- 先序遍历二叉树: 并在 GUI 界面中通过动画形式演示遍历过程，并打印遍历结果。

  实现方式: 当遍历到某节点时，将该结点对应的 *QLabel* 的颜色加深，并同时将程序暂停 0.3 秒左右，更直观生动地演示遍历过程。

- 中序遍历二叉树: 同样采用递归地方式，并在 GUI 界面中以动画的方式演示遍历过程，打印遍历结果。
- 后序遍历二叉树: 和先序和中序类似，采用递归的方式遍历并在 GUI 中演示遍历过程，打印遍历结果。
- 先序线索化二叉树: 在递归先序遍历的过程中将结点中空置的左孩子结点指向当前结点的前驱（先序遍历），右孩子结点指向当前结点的后继结点。并在 GUI 界面通过红色箭头和蓝色箭头来进行可视化显示，其中红色箭头表示指向前驱结点，蓝色箭头表示指向后继结点。
- 中序线索化二叉树: 在中序遍历的过程中将遍历到的每个结点的空置的左孩子指针指向中序遍历的前驱结点，空置的右孩子指针指向中序遍历的后继结点，并在 GUI 界面中进行可视化演示。
- 后序线索化二叉树: 同中序线索化和先序线索化类似，在后序遍历的过程中将遍历到的每个结点的空置的左孩子指针指向后序遍历的前驱结点，空置的右孩子指针指向后序遍历的后继结点，并在图形界面中通过不同颜色的箭头进行演示。
- 先序线索化遍历: 由于已经建立了线索信息，对二叉树的先序遍历不需要再以递归的方式进行，而是充分利用部分结点中的前驱/后继信息进行遍历。
- 中序线索化遍历: 首先从根结点开始不断访问当前结点的左子树，当找到一个不存在左子树的结点时，将其作为遍历的起点。之后根据部分结点中存储的线索信息进行中序非递归遍历。

通过软件功能介绍可以看出，本程序较好地完成了题目的所有要求。

1.3 设计思想

由于题目要求进行图形化界面的演示，基于自身兴趣和基础，我选择 Qt 框架进行开发，Qt 是一款使用较为广泛的跨平台开发框架。首先，我通过网课和一些相关技术书籍对 Qt 框架进行了一段时间系统的学习，对 Qt 框架特性，信号槽机制和常用控件都有了一定的了解。

1.3.1 界面间的切换关系

在开发过程中，我采取先整体再局部，自顶向下的设计方式，从用户使用的角度出发将程序分为了多个功能界面，再依次考虑这些功能界面之间的数据传递，切换等功能。

从用户进入程序开始到各项功能的演示需要经历以下几个界面的切换，每个界面基本对应一个继承自 *QMainWindow* 的类，这些类的切换关系如下图所示:

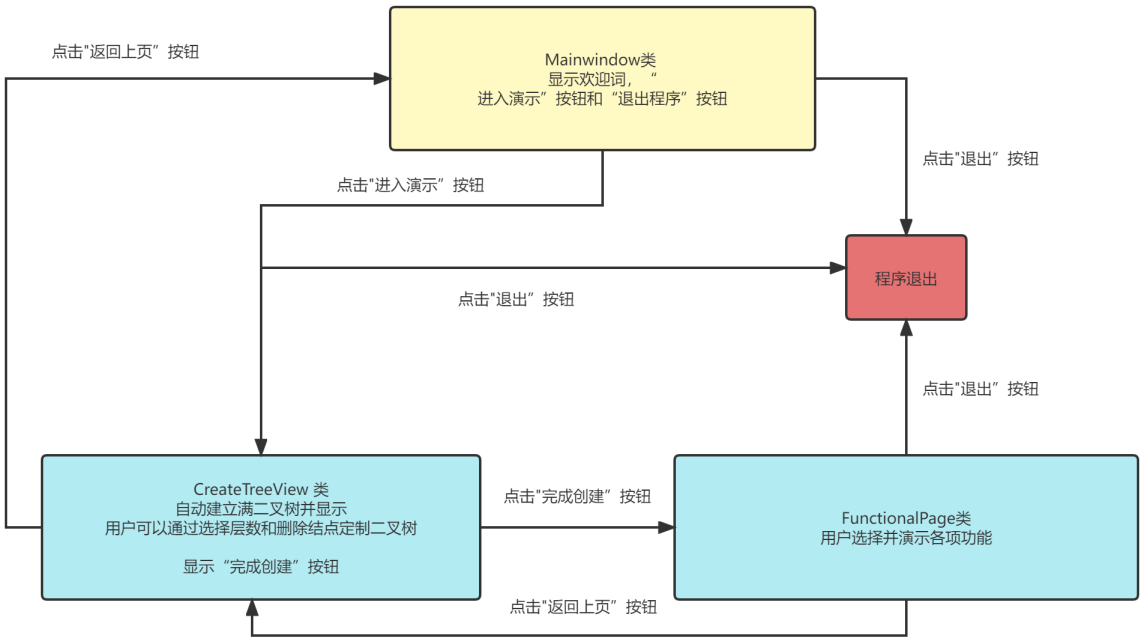


图 1: 程序各个页面的切换关系

1.3.2 各对象之间的继承/组合/委托关系

本程序采用面向对象的思想进行设计，除去 Qt 框架提供的各种控件对象之外包含 5 个新创建的类，下面是这 5 个类的一些基本情况:

- MainWindow 类，用于显示用户欢迎页，继承自 *QMainWindow* 类，内部组合有一个 *createTreeView* 对象。
- CreateTreeView 类, 用于显示创建二叉树的页面，继承自 *QMainWindow* 类，内部组合有一个 *BinaryTree* 类。
- BinaryTree 类，代表一棵二叉树，内部定义了 *treeNode* 结点结构体，并提供了用于先序/中序/后序遍历、线索化的将近 10 个接口。除此之外，还组合了 *QMap* 对象用于实现内部结点和 GUI 中显示的 *QLabel* 之间的联系。
- NodeLabel 类，用于在 GUI 上显示一个二叉树结点，继承自 *QPushbutton* 类，为了显示对应结点在满二叉树中的编号，还组合了 *QLabel* 类。
- FunctionalPage 类，用于演示各项功能，继承自 *QMainWindow* 类，同时组合了一个 *BinaryTree* 的对象指针以实现委托的效果。

上述 5 个类的继承/组合关系可总结为如下的关系图:

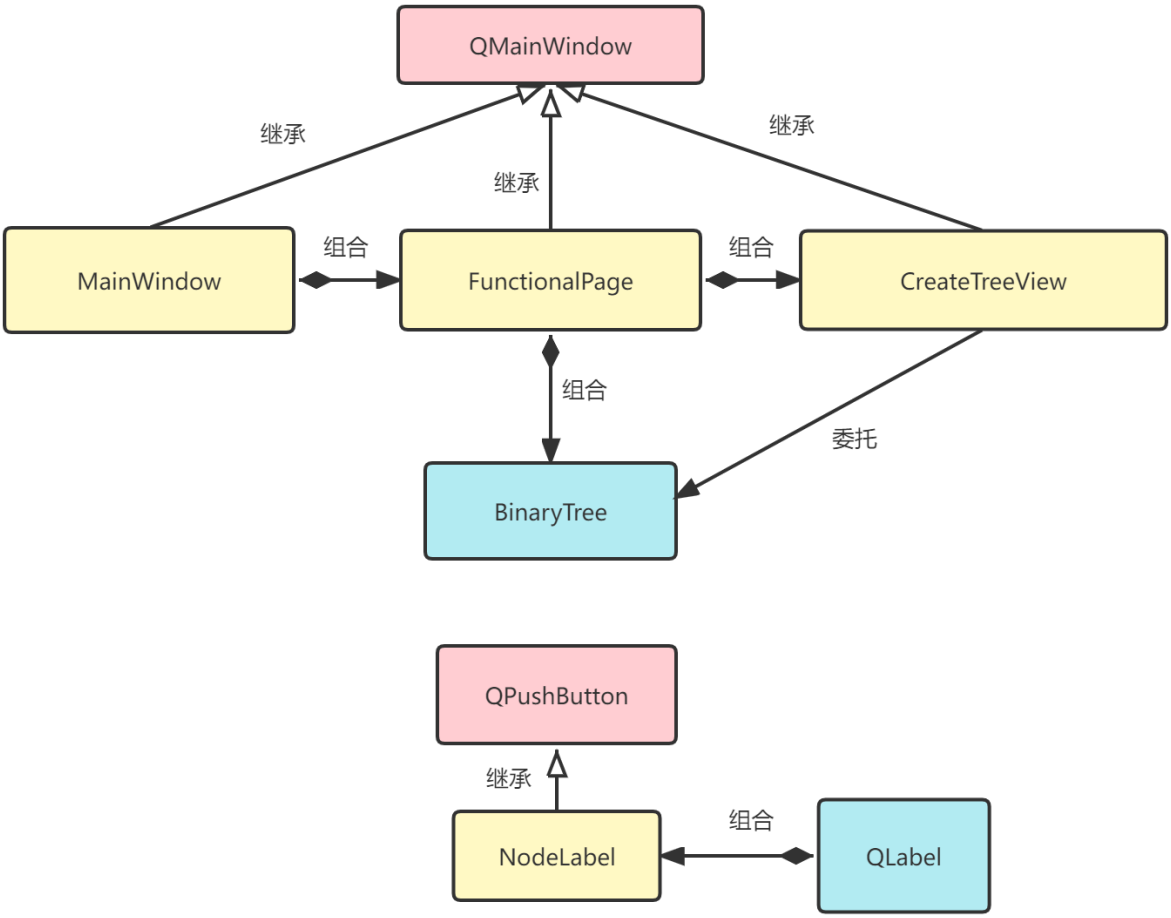


图 2: 程序各类之间的继承/组合关系

1.3.3 重要功能的算法实现

- 根据层次遍历建立满二叉树并为结点编号:

  首先建立一个队列，并将根结点标为 0 号，加入队列中。当队列不为空且当前树中结点树小于目标二叉树的结点个数时，执行以下步骤:

  1. 从队列中弹出一个结点，并新建其的两个子节点并按顺序标号。
  2. 当两个子节点不属于满二叉树的叶子结点层时，将其放入队列中。

- 先序遍历二叉树:

  采用递归的方式，起初将根结点设为当前结点。当前结点为空结点时，返回上一层，不是空结点时打印结点数据，并递归调用该函数先序遍历当前结点的左子树和右子树。

- 中序遍历二叉树:

  同样采用递归的方式，开始时将根结点设为当前结点。当前结点为空结点时，

返回上一层；不是空节点时，先递归调用该函数遍历左子树，再打印结点数据，最后递归调用该函数遍历右子树。

- 后序遍历二叉树:

  与先序和中序相同，采用递归的方式，开始时将根结点设为当前结点。当前结点为空结点时，返回上一层；不是空节点时，先递归调用该函数遍历左子树，然后递归调用该函数遍历右子树，最后打印结点数据。

- 先序线索化二叉树

  首先将当前结点设为根结点，并设立一个 pre 结点作为当前结点的前驱，初始值为空。

  当前结点为空时，返回。

  当前结点不为空时，执行下列步骤:

  1. 当前结点的左孩子为空时，将其标签设为线索并指向 pre。
  2. 当 pre 不为空且 pre 的右孩子为空时，将其右孩子标签设为线索并指向当前结点。
  3. 将当前结点设为 pre。
  4. 假如当前结点的左孩子存在，则递归调用此函数线索化左子树。
  5. 假如当前结点的右孩子存在，则递归调用此函数线索化右子树。

- 中序线索化二叉树

  和先序线索化类似，首先将当前结点设为根结点，并设立一个 pre 结点作为当前结点的前驱，初始值为空。

  当前结点为空时，返回。

  当前结点不为空时，执行以下步骤:

  1. 假如当前结点的左孩子存在，则递归调用此函数线索化左子树。
  2. 当前结点的左孩子为空时，将其标签设为线索并指向 pre。
  3. 当 pre 不为空且 pre 的右孩子为空时，将其右孩子标签设为线索并指向当前结点。
  4. 将当前结点设为 pre。
  5. 假如当前结点的右孩子存在，则递归调用此函数线索化右子树。

- 后序线索化二叉树

  同先序和中序类似，采用递归的方式进行线索化。首先将当前结点设为根结点，并设立一个 pre 结点作为当前结点的前驱，初始值为空。

  当前结点为空时，返回。

当前结点不为空时，执行以下步骤:

1. 假如当前结点的左孩子存在，则递归调用此函数线索化左子树。
2. 假如当前结点的右孩子存在，则递归调用此函数线索化右子树。
3. 当前结点的左孩子为空时，将其标签设为线索并指向 pre。
4. 当 pre 不为空且 pre 的右孩子为空时，将其右孩子标签设为线索并指向当前结点。
5. 将当前结点设为 pre。

- 先序遍历线索二叉树

  设置根结点为当前结点 p，p 不为空时，循环执行以下操作:

  若当前结点 p 的左孩子存在，将 p 设置为 p 的左孩子。否则，将 p 设置为 p 的右指针指向的结点。

- 中序遍历线索二叉树

  首先，从根结点开始沿二叉树的左支遍历，直到找到一个不存在左孩子的结点，将其作为当前结点 p。当 p 不为空时，循环执行以下操作:

  当 p 的右孩子不为空时，将 p 设置为 p 的右孩子结点，然后不断沿着 p 的左支遍历，将第一个不存在左孩子的结点作为新的 p。

  否则，将 p 设置为 p 的右指针指向的结点。

- 计算当前二叉树中的结点数目

  采用递归的方式，将根结点设置为当前结点。

  当前结点为空时，返回 0；否则返回左子树结点数目和右子树结点数目的和再加上 1 的结果。

1.4 逻辑结构和物理结构

本程序的核心数据结构是二叉树。

二叉树的逻辑结构: 二叉树是 $n(n \geq 0)$ 个结点的有限集合，该集合或者为空集合，或者由一个根结点和两棵互不相交的、分别称为根结点的左子树和右子树的二叉树组成。

二叉树的物理结构: 在本程序中采用链式存储结构，利用二叉链表的方式将父节点和它的两个孩子结点关联在一起。

程序中二叉树的类为 *BinaryTree*, 其定义的代码如下:

```
1 class BinaryTree:public QObject
2 {
```

```
    Q_OBJECT
private:
    int layerNum=0; //二叉树层数
public:
    typedef struct treeNode  //结点结构体
    {
        int data=0;  //标签，满二叉树中的顺序
        treeNode * lchild=nullptr; //左孩子指针
        treeNode * rchild=nullptr; //右孩子指针
        treeNode * parent=nullptr; //指向父亲
        bool ltag=0;    //线索化标志
        bool rtag=0;    //线索化标志
        bool isLeaf=true;
        treeNode(const int i):data(i) //结点构造
        { }
        treeNode(){}
    } treeNode;

    BinaryTree(const int layNum=4); //注意此处默认满二叉树
    int getLayer()
    {
        return layerNum;
    }
    void buildTree();  //层次遍历法建立满二叉树
    int getNodeNum(treeNode*r);  //返回结点数目

    void preOrderTraverse();  //先序遍历
    void preOrderTraverseKernel(treeNode*r);  //先序遍历

    void midOrderTraverse();
    void midOrderTraverseKernel(treeNode*r);  //中序遍历

    void postOrderTraverse(); //后序遍历
    void postOrderTraverseKernel(treeNode*r);  //后序遍历

    void clearThread();//清理线索化痕迹
    void preOrderThread();// 先序线索化
    void preOrderThreadKernel(treeNode *root,treeNode * &pre);
    void midOrderThread(); //中序线索化
    void midOrderThreadKernel(treeNode *root,treeNode * &pre);
    void postOrderThread();
     void postOrderThreadKernel(treeNode *root,treeNode * &pre);

    void preOrderThreadTraverse(); //先序线索化遍历
    void midOrderThreadTraverse();  //中序线索化遍历
    ~BinaryTree();
    void destroy();
    void setLayer(int i);
    treeNode *root=nullptr;  //二叉树根结点
```

```
52
53 signals:
54     void signal_buildtree();
55     void signal_traverse(treeNode* node);
56 };
```

1.5 开发平台

本程序开发平台的各项参数如下:

- 内存 : 16GB
- 操作系统 : Windows10 家庭中文版
- CPU : Intel Core i5-10210U CPU
- 开发语言 : C++ (C++ 11 标准及以上)
- 开发框架 : Qt 5.9.9
- 集成开发环境 : Qt Creator 4.11.0 (Community)
- 编译器 : MinGW 32bit
- 运行环境 : Debug 版本和 Release 版本使用 QCreator 集成开发环境可以正常编译运行，使用 Qt 5.9.9 的 windeployqt 工具打包后的可执行文件基本可在 windows 环境的机型下正常运行。

1.6 系统的运行结果分析说明

1.6.1 调试与开发过程

在调试过程中，我主要使用 QCreator 集成开发环境自带的 gdb 调试器，通过设置断点，步进等方式进行调试、并在右侧的窗口中通过搜索查看相关变量来辅助调试。

除此之外，通过程序打印一些相关变量的值也是辅助调试的好方法，Qt 的 *QDebug* 头文件也提供的相应的函数和类。

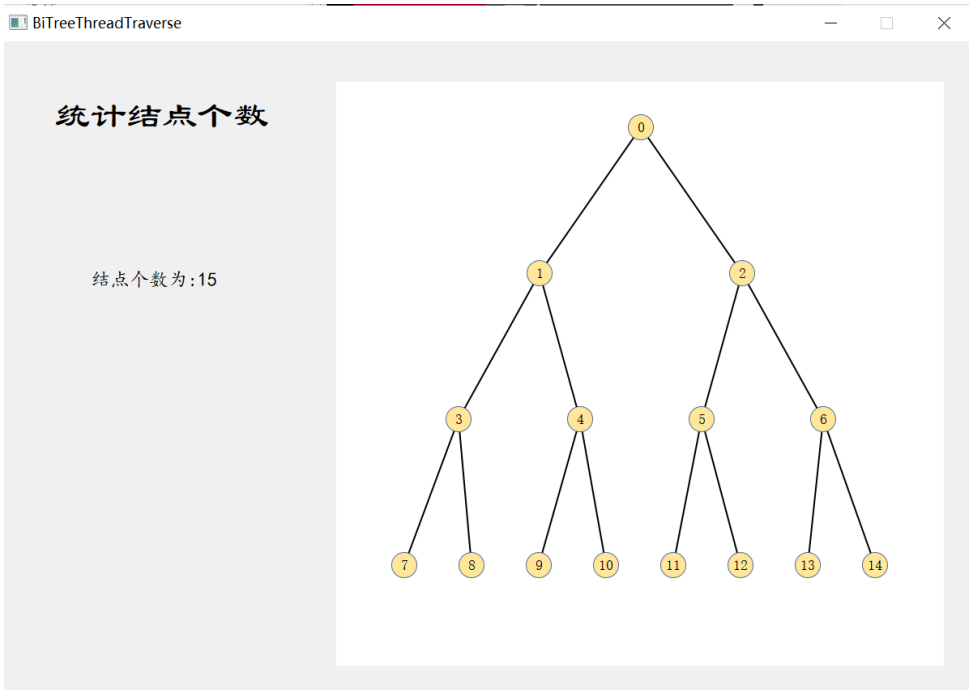
在开发前，我首先通过画出简要的类图，对象图大体明确了程序的页面切换，类之间的继承/组合关系以及怎么处理 GUI 和内部树形结构之间的关系。

在开发过程中，我采用模块化的方式，将程序拆分为不同的类或者模块，定义好不同模块间数据传递的方式以及接口，之后只需要在每个模块内部进行实现即可。整体的进展还是比较顺利的，对框架以及控件不熟悉的地方也基本都在 *Assistent.exe* 帮助程序中查找到了解决方案。

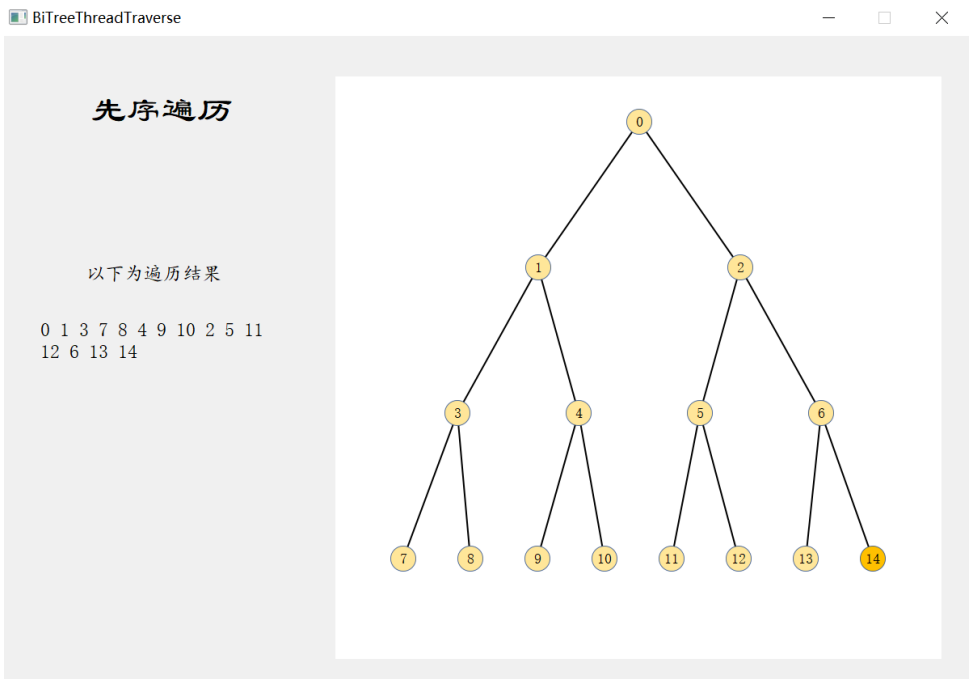
1.6.2 软件的成果分析

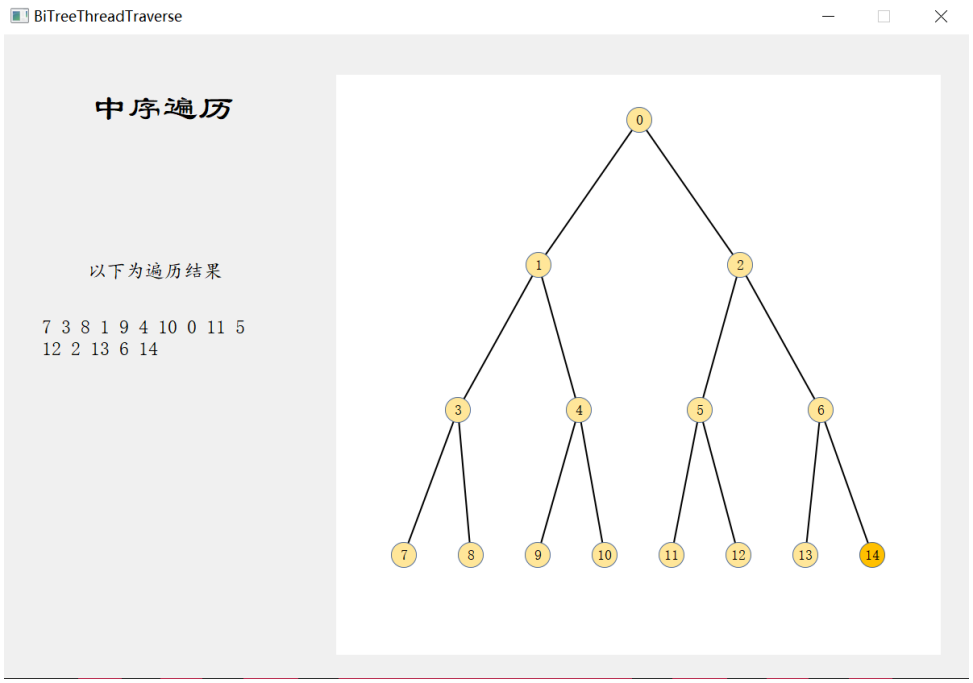
首先是正确性，在多组测试下程序的输出结果均与预期相同。这里以层数为 4 的满二叉树为例:

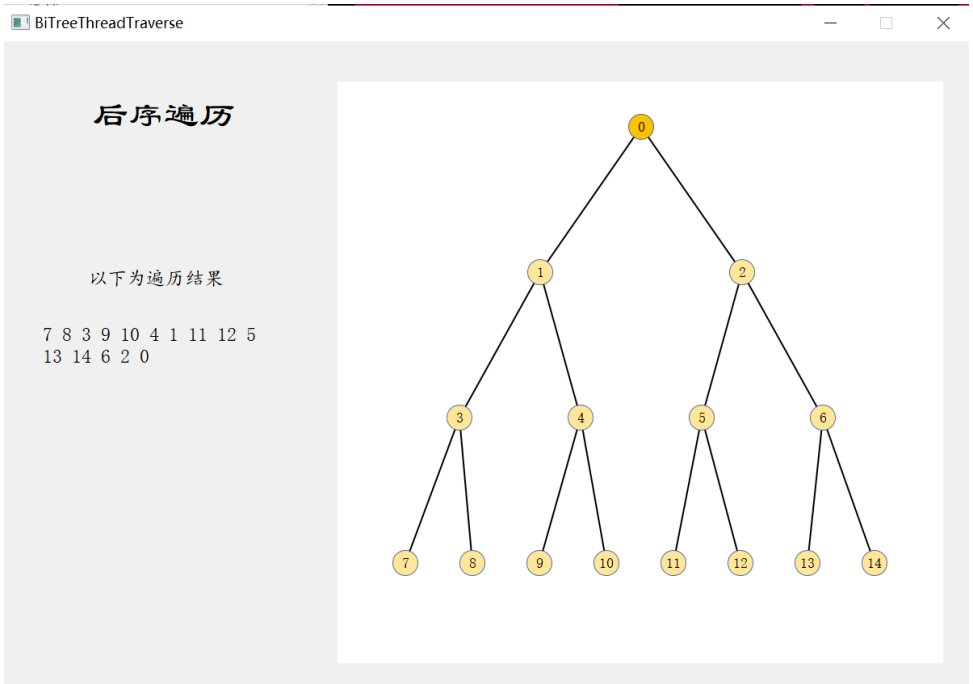
首先是结点数目, 可以看到输出 15 个结点:



然后是先序，中序，后序遍历的结果:





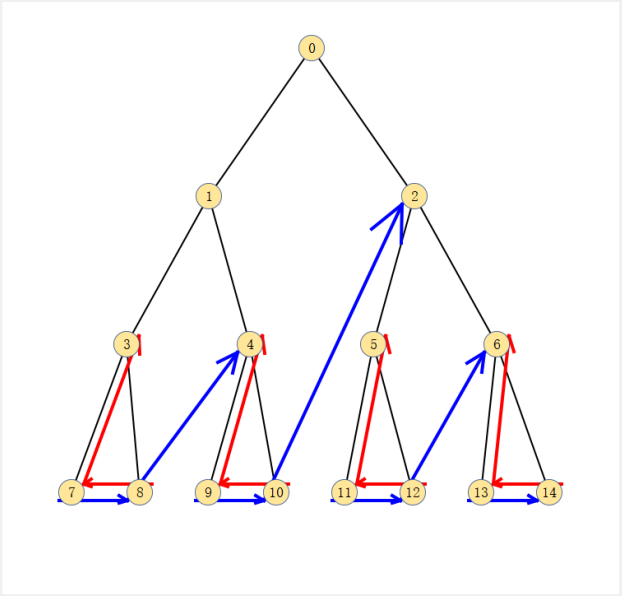


最后是先序、中序、后序线索化以及对先序、中序线索化二叉树进行遍历的结果:

BiTreeThreadTraverse

先序线索化

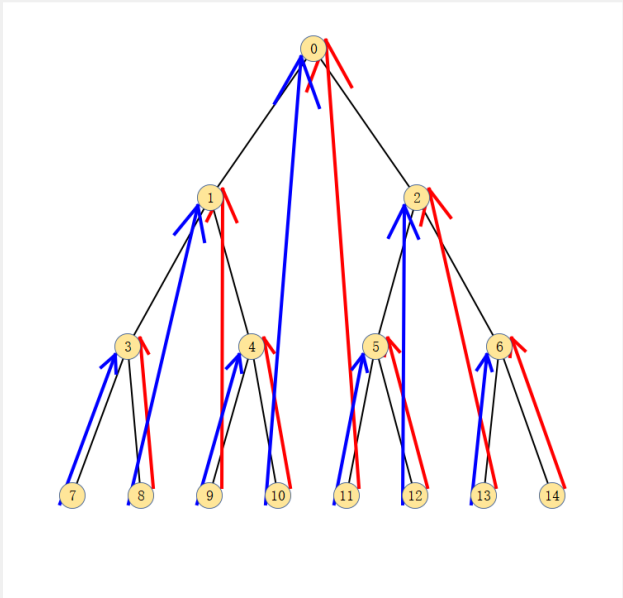
右侧为线索化结果
红:前驱 蓝:后继



BiTreeThreadTraverse

中序线索化

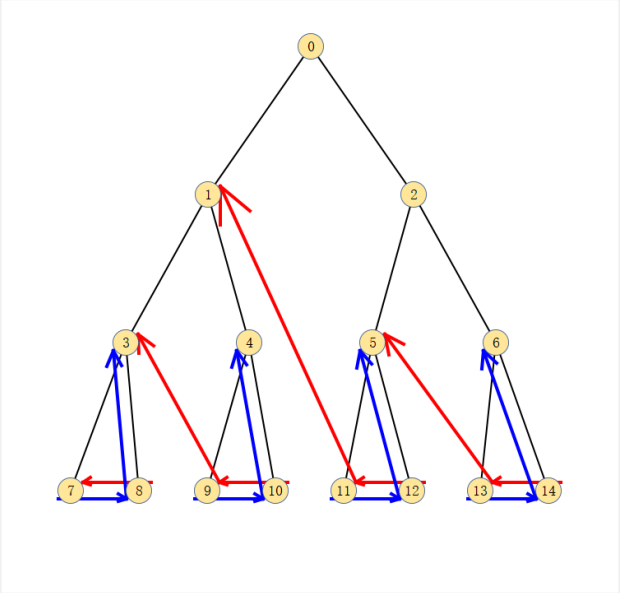
右侧为线索化结果
红:前驱 蓝:后继



BiTreeThreadTraverse

后序线索化

右侧为线索化结果
红:前驱 蓝:后继

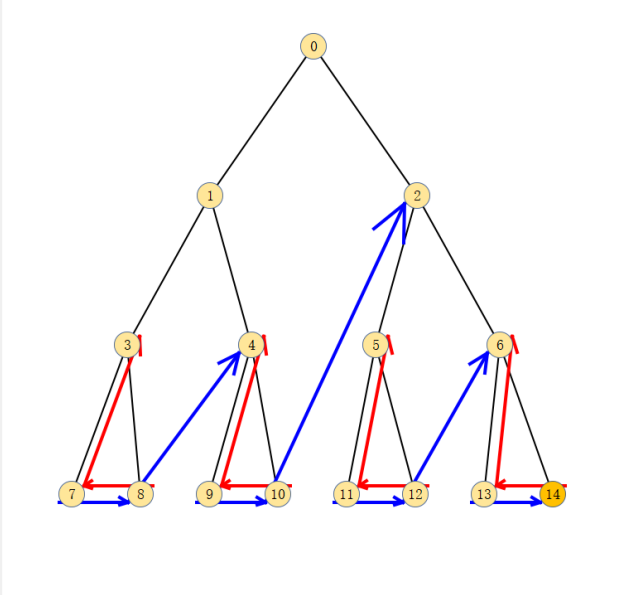


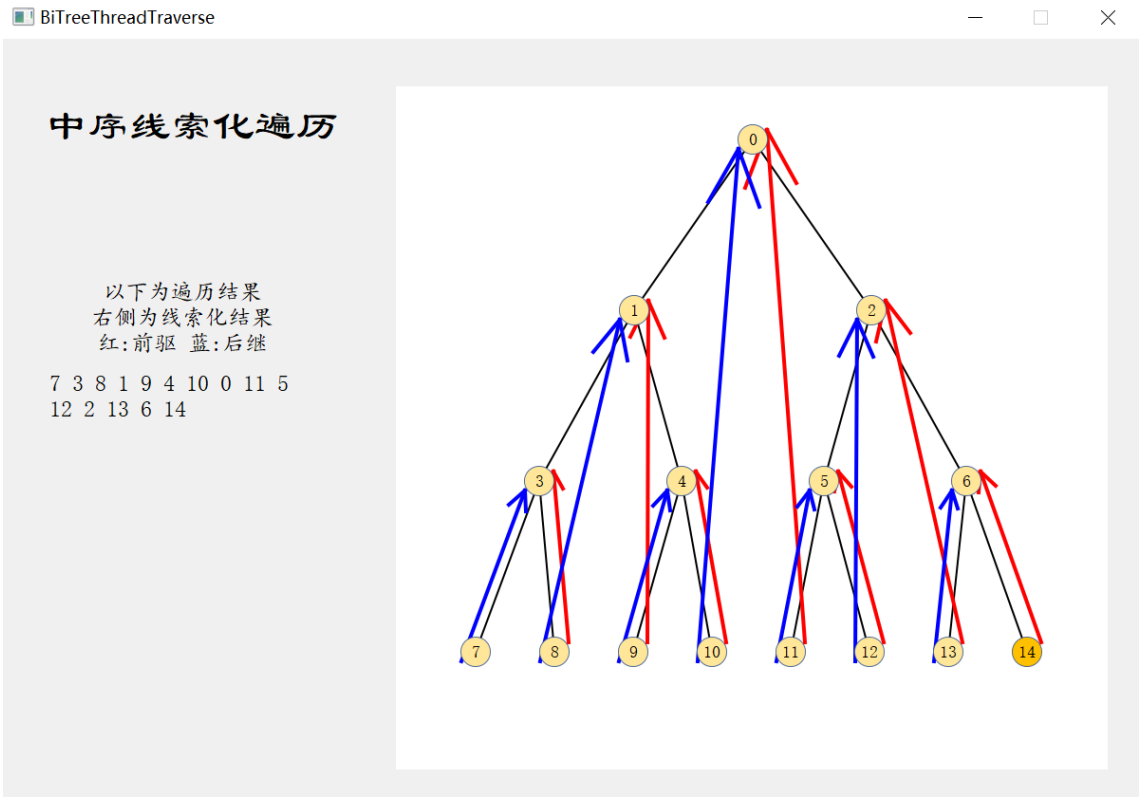
BiTreeThreadTraverse

先序线索化遍历

以下为遍历结果
右侧为线索化结果
红:前驱 蓝:后继

0 1 3 7 8 4 9 10 2 5 11
12 6 13 14



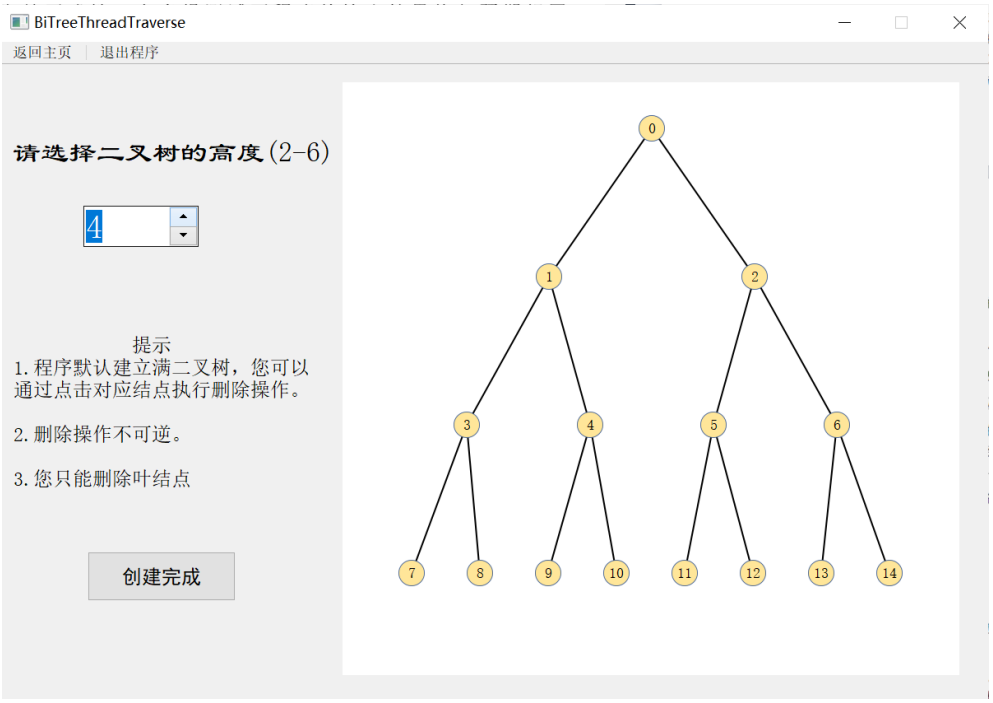


接下来是稳定性，首先我将打包好的程序发给了多个好友，经过他们的反应，程序在不同的 windows 机器上都可以得到正确的结果。

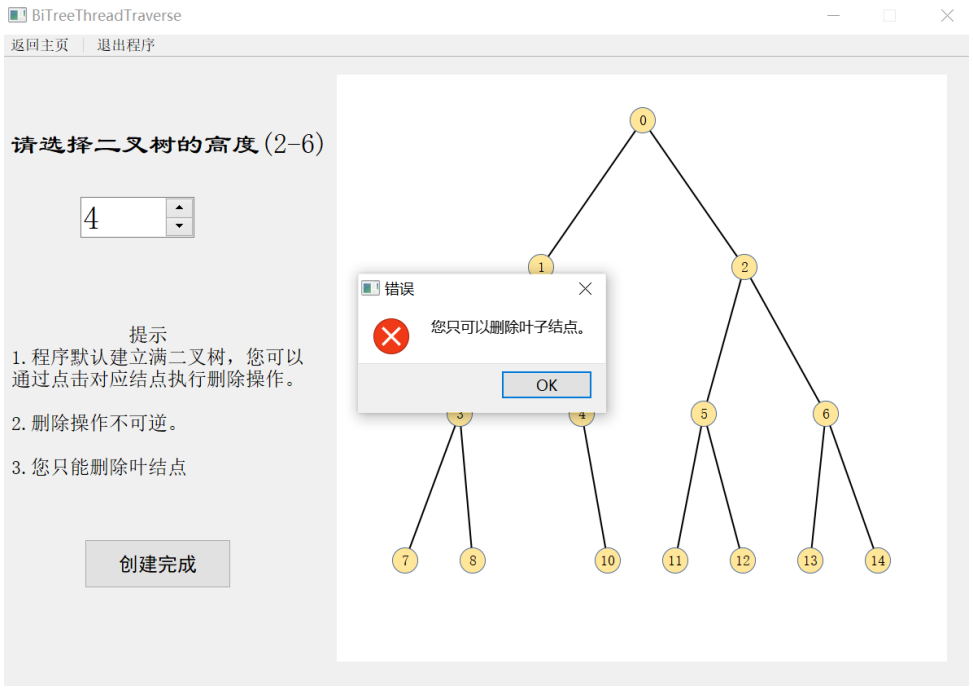
接下来我测试了程序在不同分辨率下的稳定性，通过以下几幅图可以看出，程序在多种分辨率下都可以正常地显示并运行。

最后是容错能力，本程序具有很强地容错能力，通过一系列措施有力地防止了用户的误操作。

首先，在二叉树的建立中，通过 *spinbox* 控件输入层数，保证了用户只能选择 2-6 之间的整数:



其次，本程序通过自动创建满二叉树，用户通过点击结点删除的方式来定制二叉树，规避了传统的通过输入先序遍历结果的方式来输入，使建立二叉树更加方便，也更不容易出错。同时，程序对用户通过点击删除结点的过程也进行了检查，使得用户只可以删除叶结点、无法删除非叶节点和根结点，有力地保证了二叉树建立的正确性:

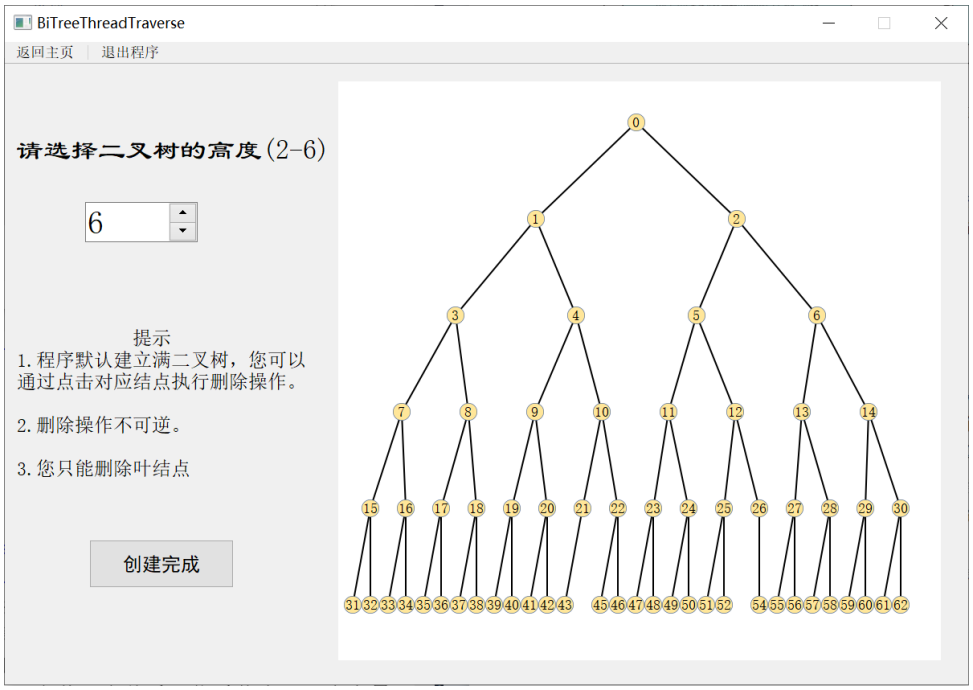


最后，当用户进入了功能演示界面后，我取消了通过点击删除结点的信号-槽连接，确保在演示过程中二叉树结构的稳定。

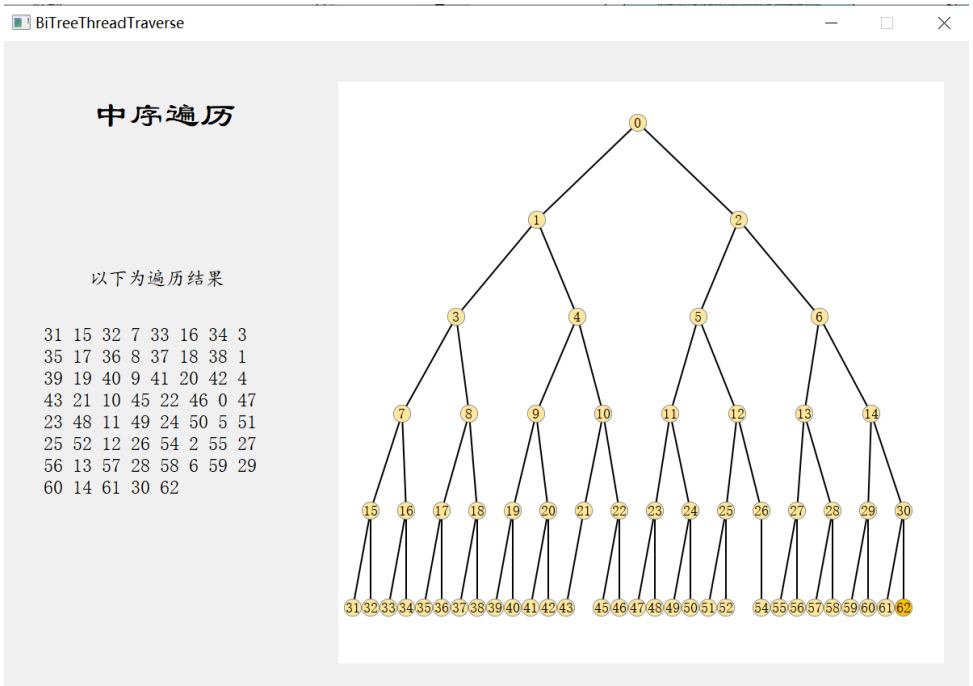
1.6.3 运行结果

这里将再给出一组运行结果来表现程序的普适性和正确性（以删去了两个结点的 6 层二叉树为例）。

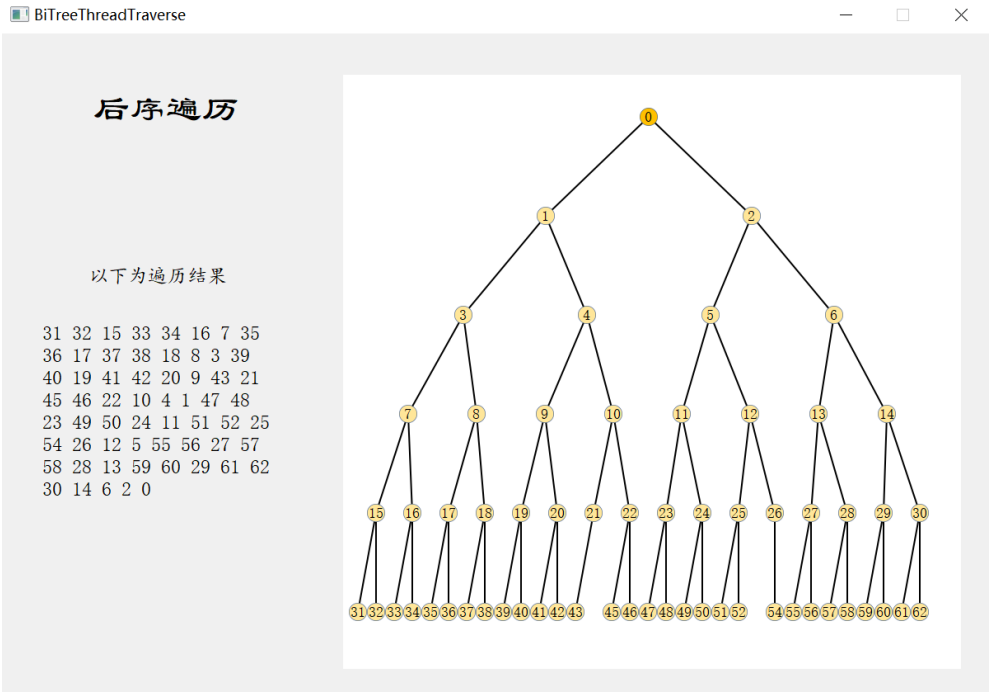
创建二叉树界面



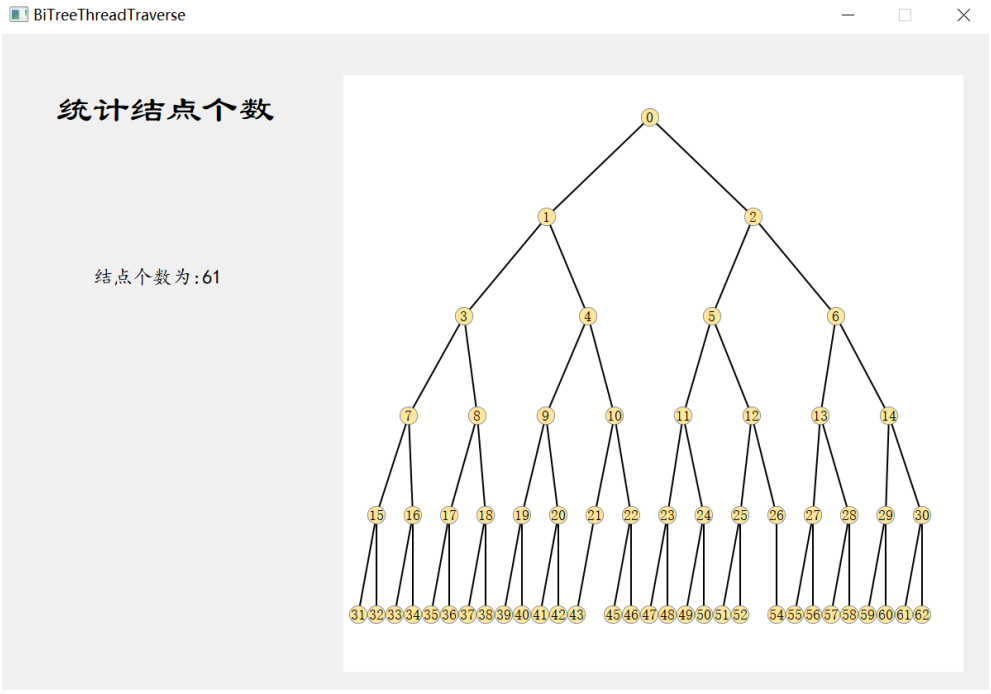
中序遍历界面



后序遍历界面



统计结点数界面



1.7 操作说明

1. 首先，打开 2050525_ 陈开煦 _ 计算机科学与技术 _ 执行程序 中的算法实现题 文件夹, 其中的 *BiTreeThreadTraverse.exe* 即为可执行文件。

名称	修改日期	类型	大小
iconengines	2022-09-02 19:46	文件夹	
imageformats	2022-09-02 19:46	文件夹	
platforms	2022-09-02 19:46	文件夹	
translations	2022-09-02 19:46	文件夹	
BiTreeThreadTraverse.exe	2022-09-02 19:41	应用程序	484 KB
D3Dcompiler_47.dll	2014-03-11 18:54	应用程序扩展	3,386 KB
libEGL.dll	2019-12-04 4:49	应用程序扩展	28 KB
libgcc_s_dw2-1.dll	2015-12-29 6:25	应用程序扩展	118 KB
libGLESV2.dll	2019-12-04 4:49	应用程序扩展	2,748 KB
libstdc++-6.dll	2015-12-29 6:25	应用程序扩展	1,505 KB
libwinpthread-1.dll	2015-12-29 6:25	应用程序扩展	78 KB
opengl32sw.dll	2016-06-14 21:08	应用程序扩展	15,621 KB
Qt5Core.dll	2022-09-02 19:46	应用程序扩展	6,004 KB
Qt5Gui.dll	2019-12-04 4:49	应用程序扩展	6,086 KB
Qt5Svg.dll	2019-12-04 5:00	应用程序扩展	358 KB
Qt5Widgets.dll	2019-12-04 4:49	应用程序扩展	6,219 KB

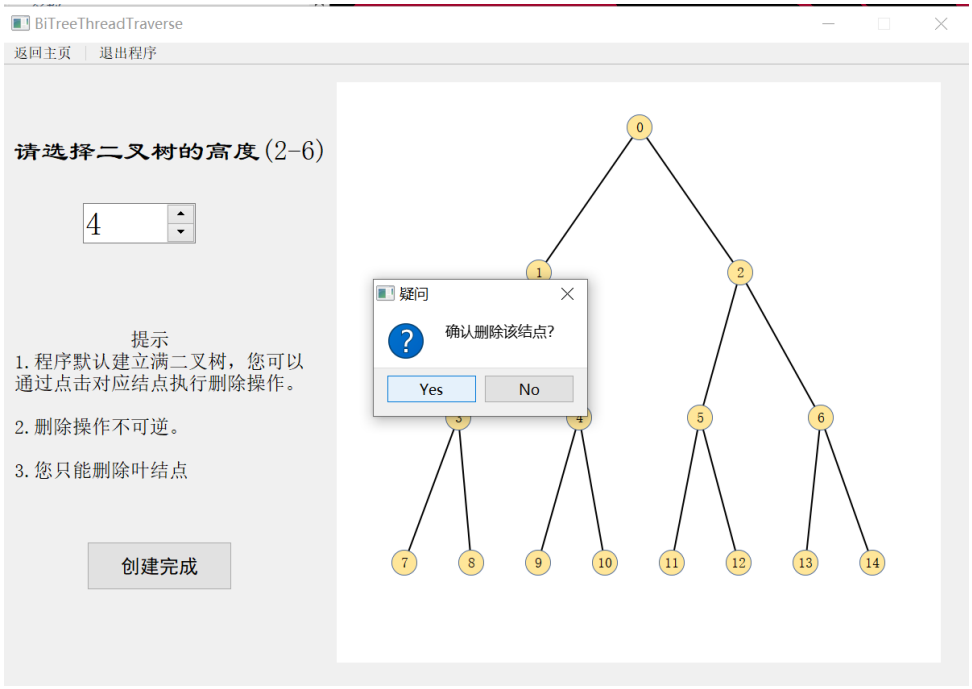
2. 点击进入程序后，显示如下界面，点击“创建二叉树”即可进入创建界面。

3. 用户可以通过界面左侧的按钮选择二叉树的层数 (2-6), 右侧画板中会默认建立对应层数的满二叉树。

   除此之外，用户可以通过点击右侧画板上的对应结点对相应结点执行删除操作。此处需要特别注意: 只允许对叶子结点执行删除操作，并且不可以删除根结点，否则程序会弹出相应警告对话框。

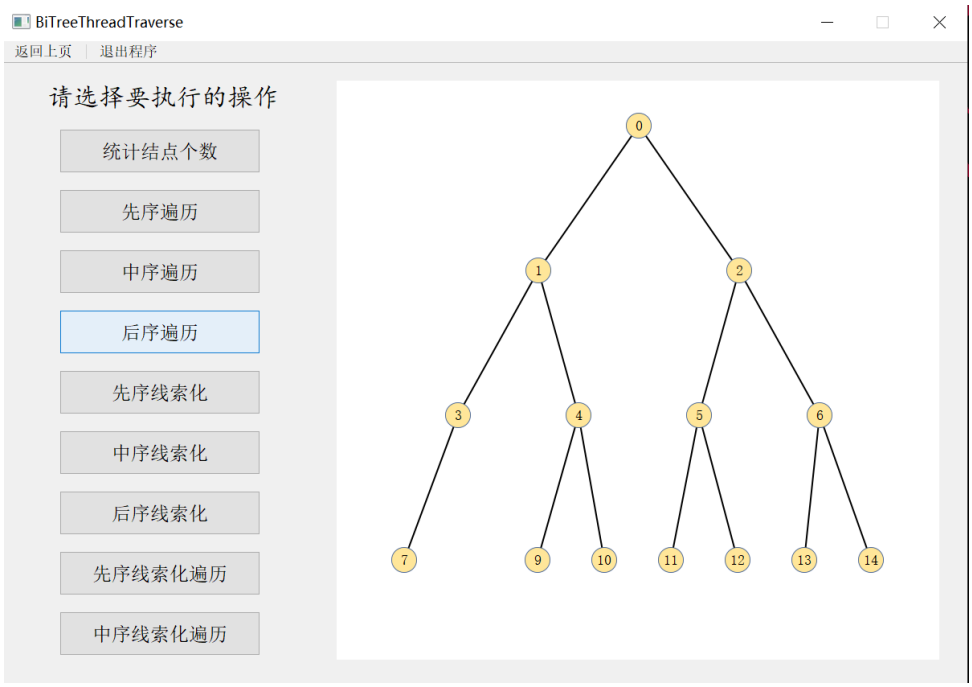
   二叉树创建完成之后，点击左侧面板的" 创建完成" 按钮即可进入到功能演示界面。

同时，用户也可以通过点击上方的" 返回主页”按钮返回欢迎页。



4. 进入到功能演示界面之后，用户可以通过点击左侧面板上不同功能的按钮实现不同功能的演示效果。用户也可以通过点击上方的" 返回上页" 返回创建二叉树的界面。

   此处需要注意的是，每项功能演示结束后才会出现下方的" 返回" 按钮，用户可通过此按钮返回功能演示界面。



2 综合应用设计说明

2.1 题目

按照相应选题规则，我选了综合应用题的第 5 题《社会关系网络》，难度为 3 颗星，题目描述如下：

在某社会关系网络系统中，一个人属性包括所在地区，就读的各级学校、工作单位等，每个人有一众好友，并可以根据个人兴趣及社会活动加入到某些群组。现需设计一算法，从该社会关系网络中某一人出发，寻找其可能认识的人。例如根据两个人共同好友的数量及所在群组的情况，来发现可能认识的人；通过就读学校发现可能认识的同学。

- 通过图形化界面，显示某一人的关系网络。
- 寻找某一人可能认识的人（不是其好友），并查看这些人与其关联度（共同好友数）。
- 根据可能认识的关联度对这些人进行排序。

2.2 软件功能

该软件名为 *SocialNetwork*，实现了一个不超过 25 个成员和 18 个群组的社会关系网络的添加，查找，关联度排序等功能，具体包括以下功能:

- 初始化一个含有 3 个成员，8 个群组的无向图。
- 用户可以通过点击相关按钮添加成员，并添加和既有结点的朋友关系以及和既有群组的包含关系。
- 用户可以通过点击相关按钮查看当前社会关系图中的群组数量以及其所包含的成员。
- 用户可以通过点击相关按钮添加群组，并为该群组选定一些成员。
- GUI 界面左侧画板中会实时显示当前的社会关系网络图，认识的两个人之间以直线相连。
- 用户可以通过点击相关按钮查看社会关系网络中某成员的详细信息，并查看网络中可能认识的人以及和可能认识的人的关联度和共同群组数量，同时，左侧 GUI 界面中会将当前成员、当前成员认识的人、当前成员可能认识的人以不同的颜色标出，更加直观。

。通过软件功能介绍可以看出，本程序较好地完成了题目的所有要求。

2.3 设计思想

由于个人兴趣和相关基础等因素，我选择 Qt 框架进行开发。Qt 框架对于面向对象变成具有良好的支持性，其大多数控件诸如 *QMainWindow*,*QWidget*,*QDialog* 等都是类，而且支持继承、组合等面向对象开发方法，并提供了大量可供 *override* 的虚函数。因此，本程序采用面向对象的设计思想。下面，我将从程序运行时的流程，程序中各类之间的继承、组合关系以及部分模块的实现方法 3 个方面来介绍本程序的设计思想。

2.3.1 程序运行流程

程序运行流程如下图所示:

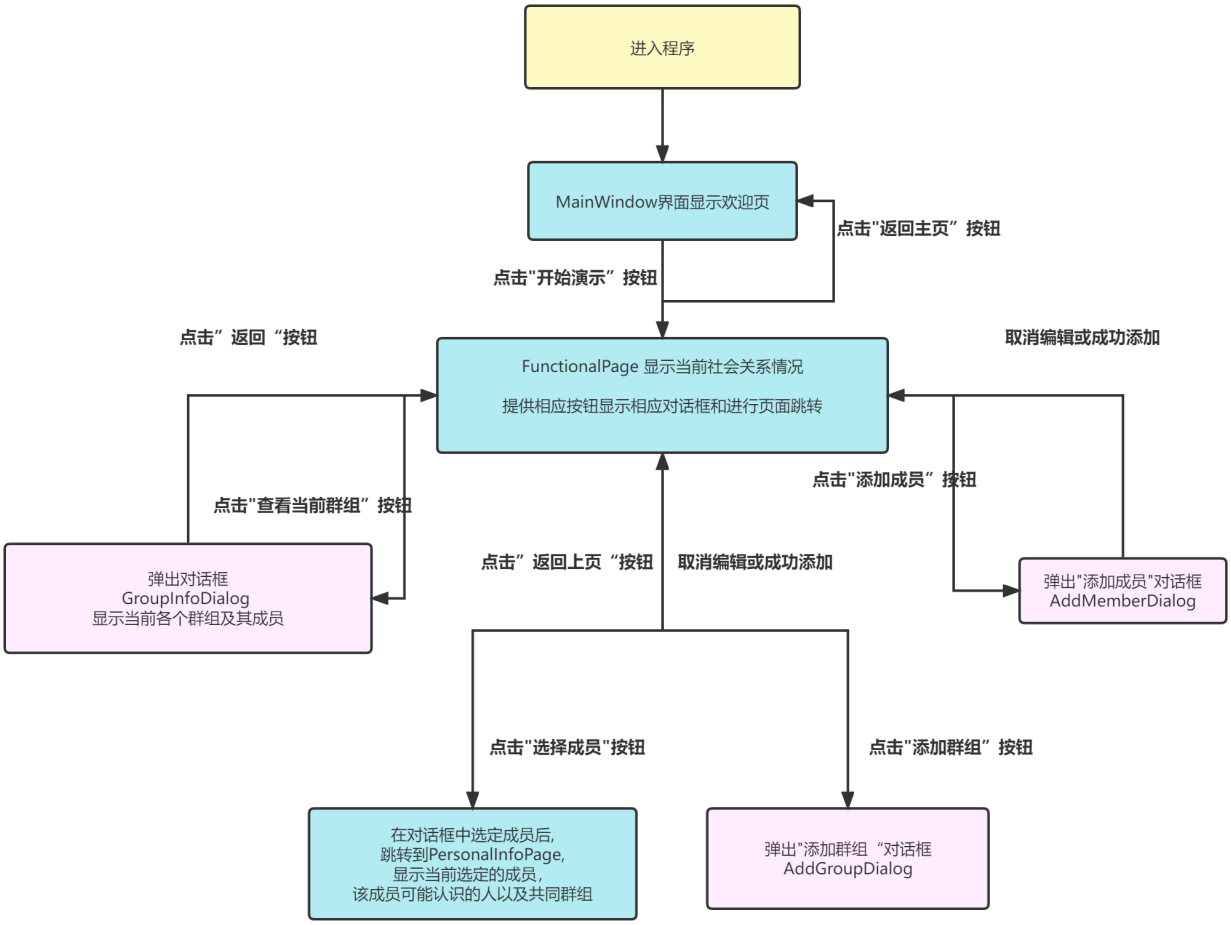


图 3: 程序运行流程图

2.3.2 程序中各类的继承、组合关系

本程序采用面向对象的思想，一共设计了 7 个类, 分别是:

- MainWindow 类: 继承自 *QMainWindow* 类，用于显示欢迎页。
- FunctionalPage 类: 继承自 *QMainWindow* 类，委托有一个 *InnerGraph* 类的指针。
- AddMemberDialog 类: 继承自 *QDialog* 类，用于显示添加新成员的对话框。
- AddGroupDialog 类: 继承自 *QDialog* 类，用于显示添加新群组的对话框。
- GroupInfoDialog 类: 继承自 *QDialog* 类，用于显示当前群组及其成员的对话框。
- InnerGraph 类: 继承自 *QObject* 类，用于以邻接矩阵的形式存储一个表示关系网络的无向图。
- PersonalInfoPage 类: 继承自 *QMainWindow* 类，用于显示某特定成员的社会关系，包括可能认识的人及共同群组数量，委托有一个 *InnerGraph* 类的指针。

上述 7 个类之间的继承，组合关系如下图所示:

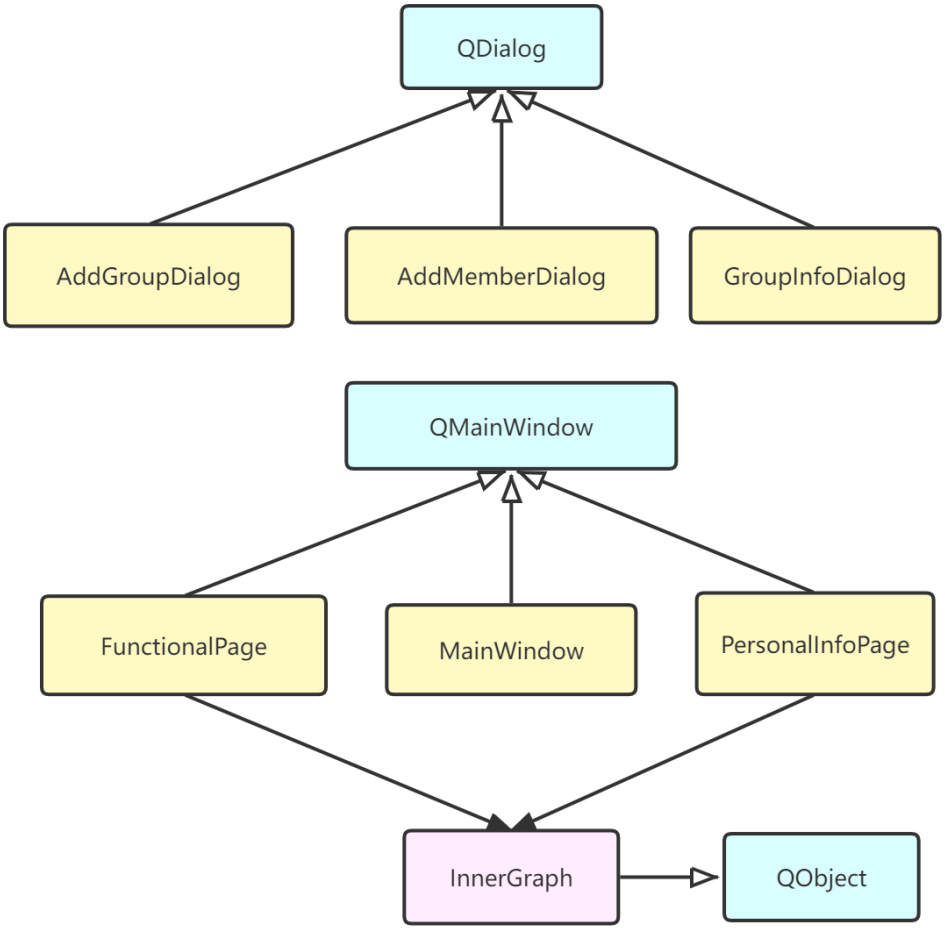


图 4: 各类的继承、组合关系

2.3.3 重要模块的实现思路

- 如何查找某成员可能认识的人及关联度

  我们假设当前要查找成员 A 可能认识的人。遍历成员列表，每当发现一个 A 不认识的成员 B 时，执行以下操作:

  1. 遍历 A 的所有认识的人，假设 A 与 C 相互认识
  2. 遍历 C 认识的人，假设 C 与 B 相互认识，则 A 和 B 之间的关联度提高 1。

  这个过程类似于最短路径问题的迪杰斯特拉算法的第一步，时间复杂度为 $O(n)$。

- 如何查找两个成员共同群组数量

  为了使这一过程的事件复杂度尽可能低，我将每个群组定义为一个结构体，这个结构体里面包含一个布尔数组，数组的长度即为当前社会关系网络中成员的数量，假如位置 i 的值为 1。即代表第 i 个成员属于这个群组，反之亦然。

  有了上述的结构，当查找两个成员的共同群组时，只需要遍历群组列表，在每个群组对应的成员数组中查看两个待查成员相应位置的值是否均为 1，这个过程的时间复杂度为 $O(1)$，故查找两个成员共同群组数量的时间复杂度为 $O(m)$。

2.4 逻辑结构和物理结构

本程序的核心部分在于表示社会关系网络的无向图。

逻辑结构: 用无向图的逻辑结构表示社会关系网络。

图是由顶点的有穷非空集合和顶点之间边的集合组成，通常表示为 $G(V,E)$，其中 G 表示一个图，V 是图 G 中顶点的集合，E 是图 G 中边的集合。

若顶点 V_i 和 V_j 之间没有方向，则为无向边，如果图中任意两个顶点之间的边都是无向边，则称该图为无向图。

物理结构: 在本程序中，由于社会关系网络中的成员数量被限制在较小的范围内，而且彼此之间的联系较为紧密，故采用邻接矩阵的方式来存储无向图。邻接矩阵的存储结构属于顺序存储结构，可以实现随机存取。

2.5 开发平台

本程序开发平台的各项参数如下:

- 内存 : 16GB

- 操作系统 : Windows10 家庭中文版
- CPU : Intel Core i5-10210U CPU
- 开发语言 : C++ (C++ 11 标准及以上)
- 开发框架 : Qt 5.9.9
- 集成开发环境 : Qt Creator 4.11.0 (Community)
- 编译器 : MinGW 32bit
- 运行环境：Debug 版本和 Release 版本使用 QCreator 集成开发环境可以正常编译运行，使用 Qt 5.9.9 的 windeployqt 工具打包后的可执行文件基本可在 windows 环境的机型下正常运行。

2.6 系统的运行结果分析说明

2.6.1 调试与开发过程

在调试过程中，我依然采用的是 QCreater 集成开发环境下的 gdb 调试器，通过设置断点、步进、搜索查看中间变量等方法来调试。由于有了算法实现题的经验，在综合应用题的开发过程中，我的错误明显少了很多。

除了使用集成开发环境提供的调试器，打印相关变量的值来查看程序运行的正确性也是辅助我们调试的好方法。Qt 框架中也提供了相应的 *QDebug* 库。

在开发过程中，我首先通过画类图和对象图明确了程序中各个页面的切换关系，运行流程；接下来定义每个模块/类的方法和接口，明确模块间数据传递的方式；最后在每个模块内部进行实现。

2.6.2 软件的成果分析

首先是程序的正确性，在多组测试案例上，程序均运行良好，与预期完全一致。

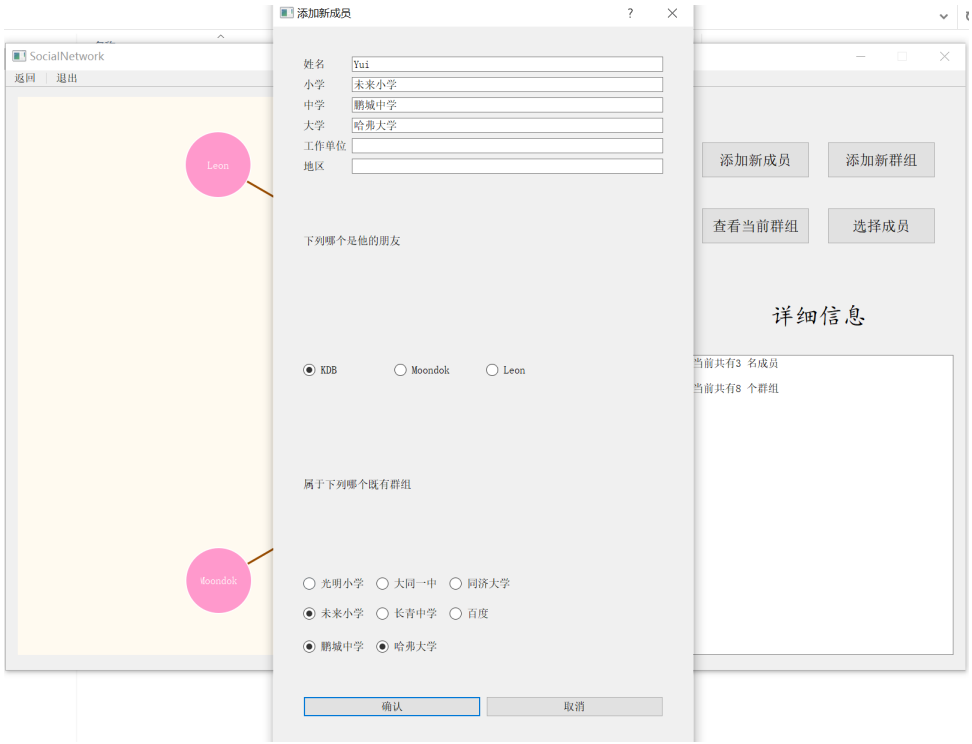
为了优化演示体验，社会关系网络预先建立了三个成员和八个群组，分别是:

- KDB：光明小学、大同一中、同济大学
- Moondok：未来小学、长青中学、百度
- Leon：鹏城中学、哈弗大学、百度

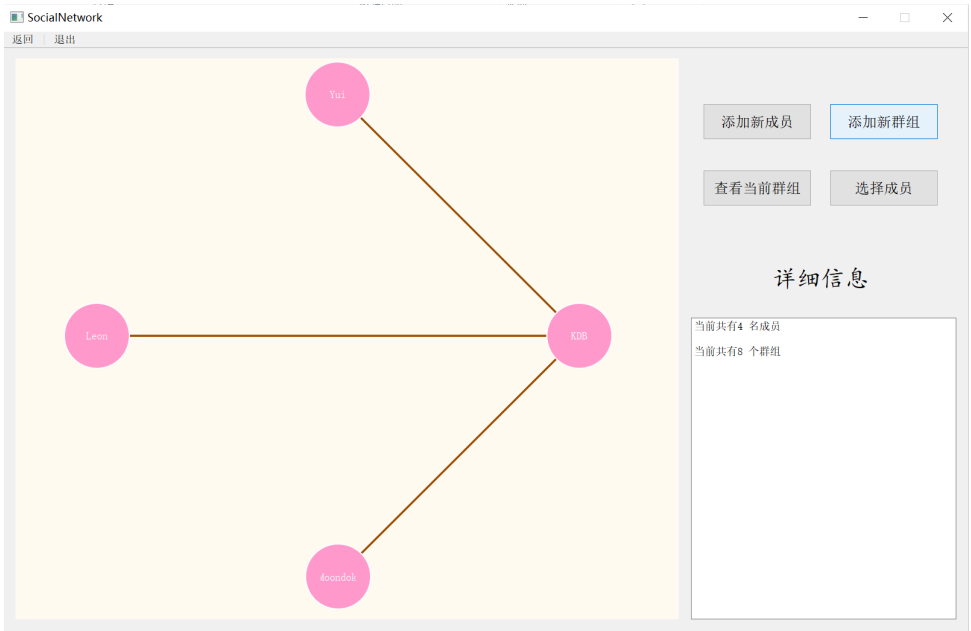
其中 Moondok 和 Leon 都与 KDB 是朋友关系，但他们彼此之间不认识。

为说明程序的正确性，我向社会关系网络中添加了 Yui 和 Hu 两个人，其中 Yui 属于未来小学、哈弗大学和鹏城中学，和 KDB 是好友；Hu 属于北京大学和光明小学，和 Moondok、Yui 是好友。

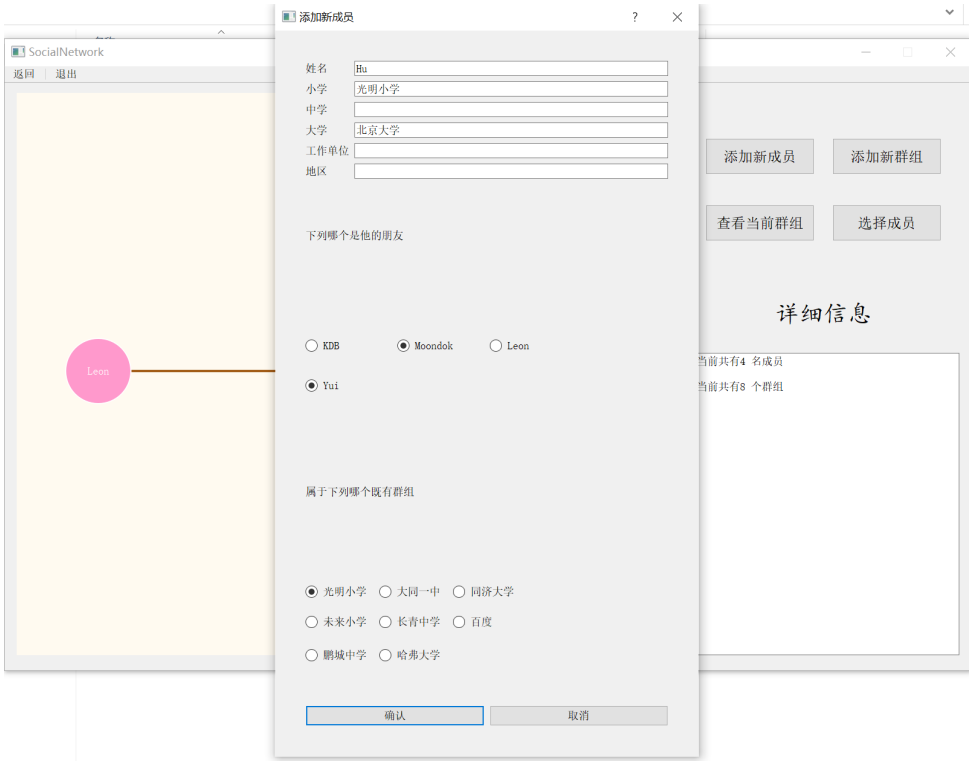
点击右侧面板的" 添加新成员" 按钮，首先添加 Yui:



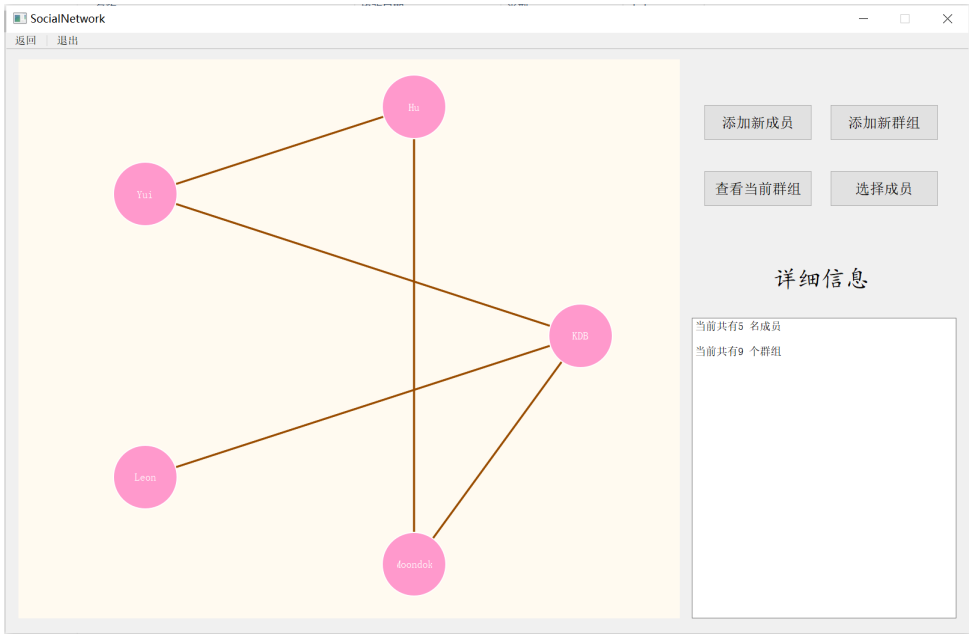
添加完 Yui 后，左侧画板变成了下面这样:



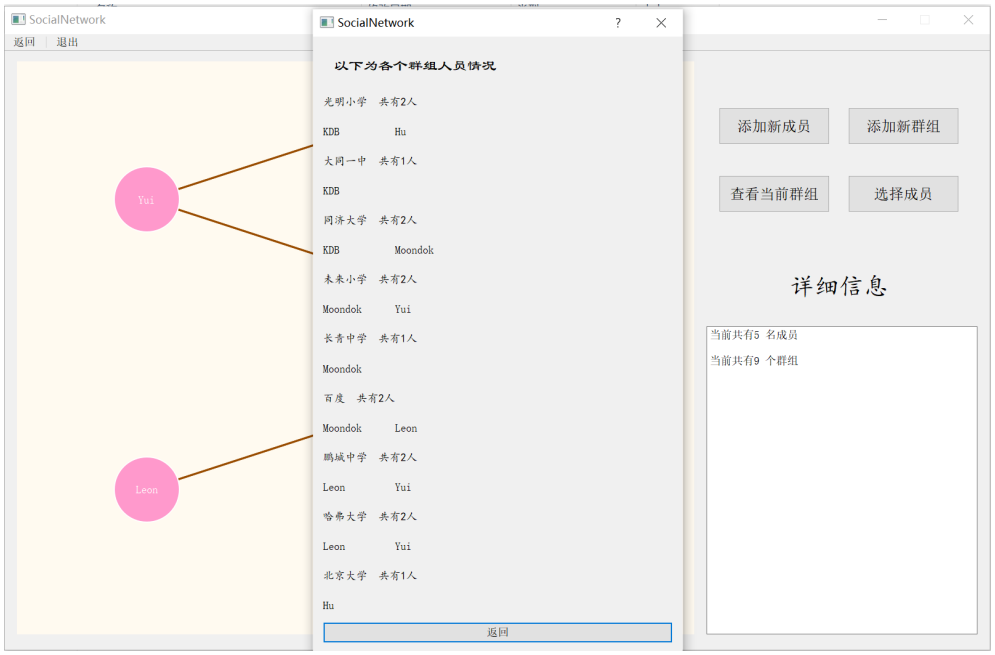
接下来添加 Hu:



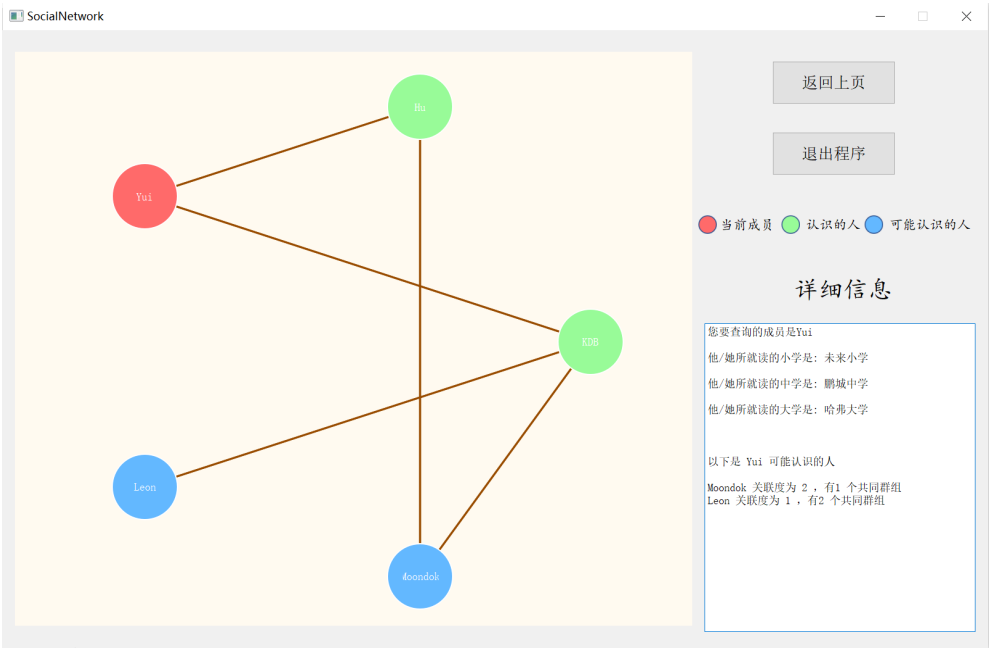
添加 Hu 之后，左侧画板变成了下图这样子:



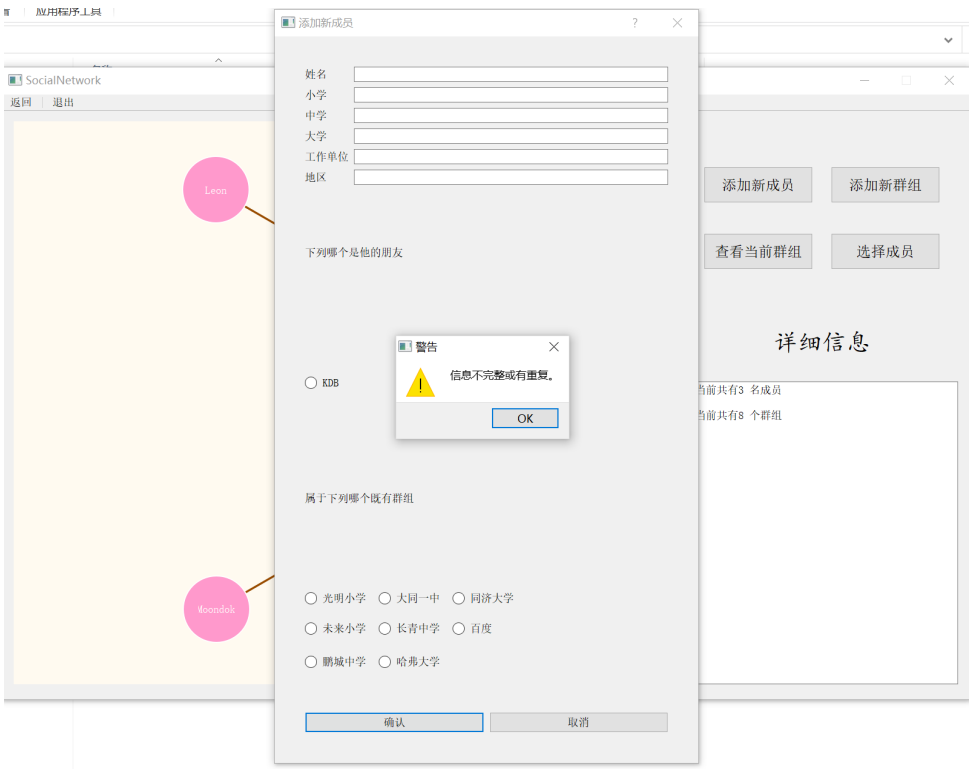
点击右侧面板的" 查看当前群组" 按钮，可以发现增添了新群组" 北京大学"，既有群组中也正确添加了新成员:



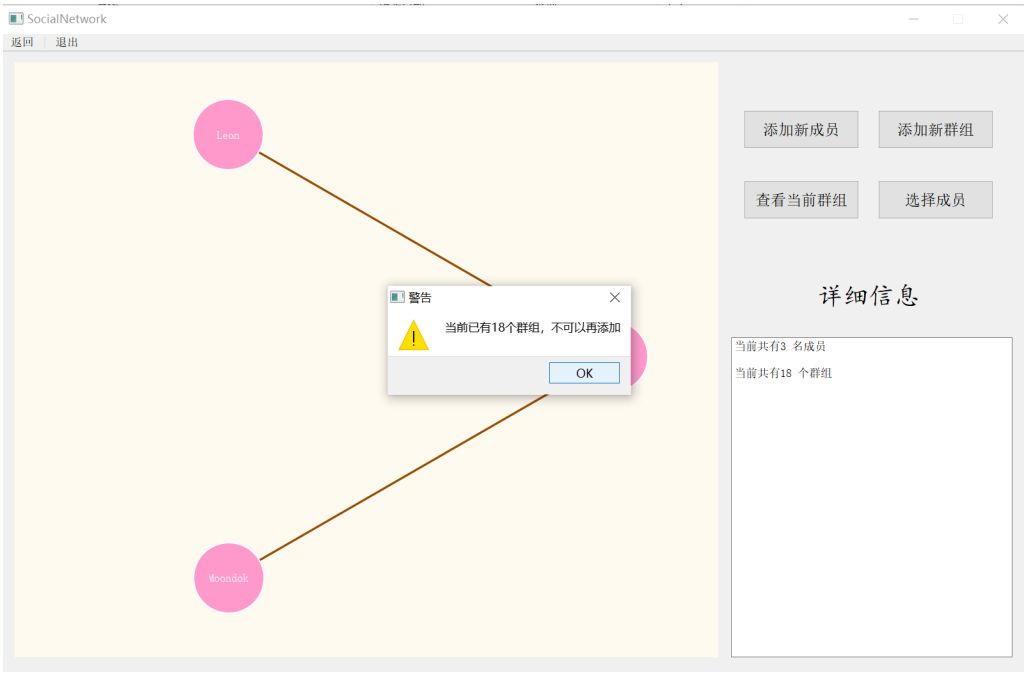
点击右侧面板的" 选择成员" 按钮选择 Yui，进入到成员详细信息页面，可以看到 Yui 和其他成员的关联度和共同群组数也都正确显示:



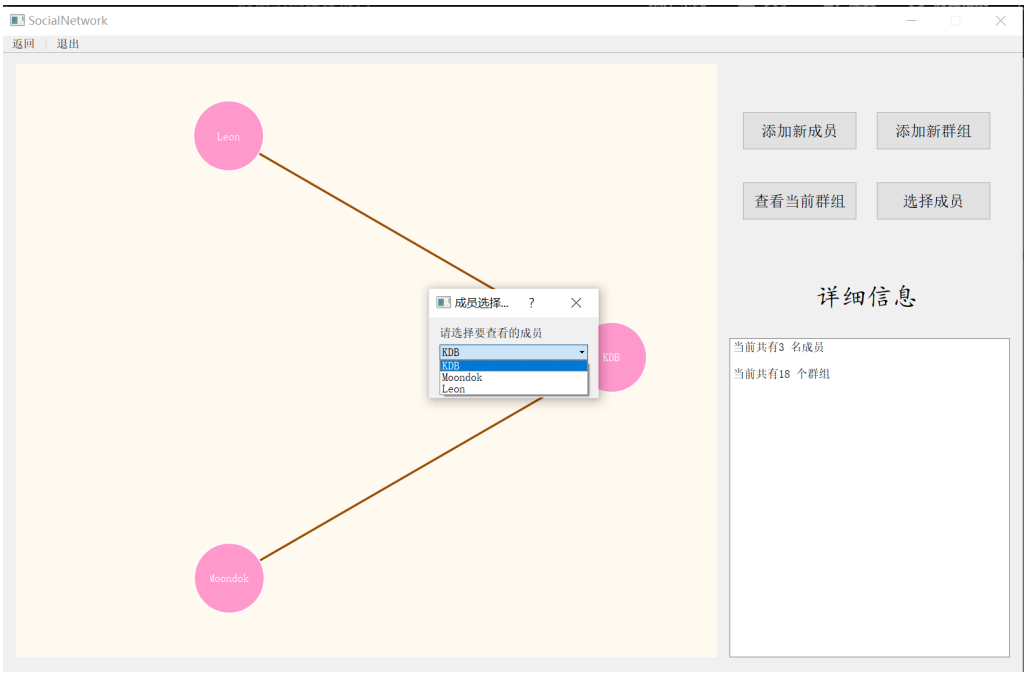
本程序具有良好的容错性，首先，为了防止用户输入重复信息或者不完整 (无成员名，群组名) 的信息，我在添加群组和添加成员的对话框中添加了对用户输入的检查，当名称一栏为空时程序会弹出警告对话框。



除此之外，在用户添加成员时，假如当前既有成员的数量达到 25，则无法继续添加成员，程序会弹出警告对话框。添加群组时，假如当前既有群组数量达到 18，则同样无法添加，程序会弹出警告对话框。



最后，在选择成员查看详细信息时，我采用 *QInputDialog* 的 *getItem* 方法，用选择的方式代替输入，降低了出错率。



2.6.3 运行结果

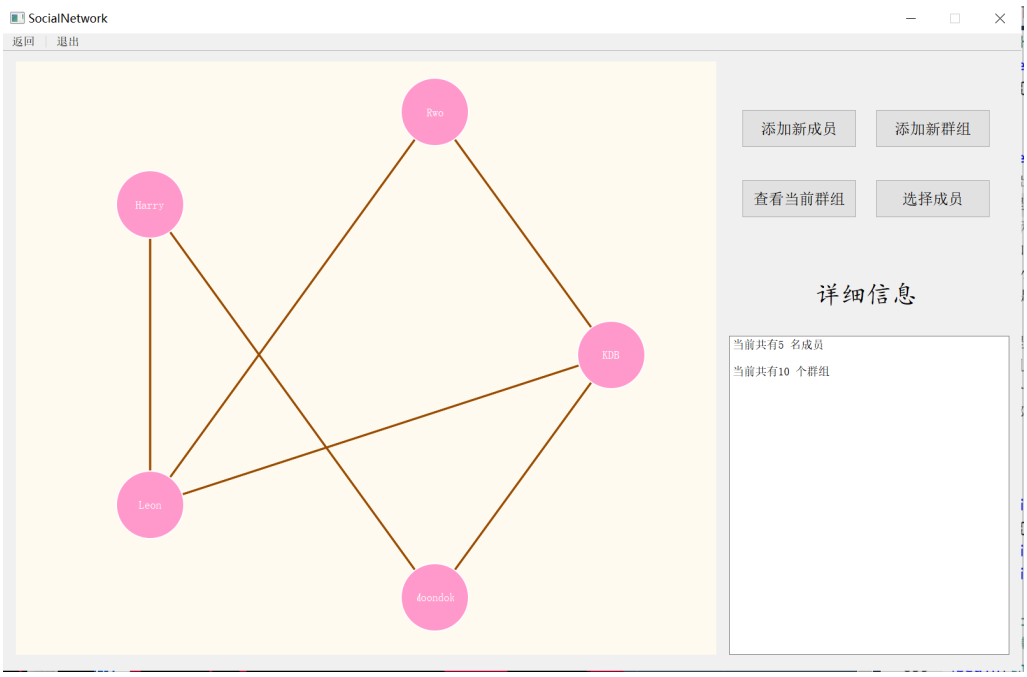
这里再给出一组程序运行的案例。

在程序既有的 3 个成员的基础上再添加两个成员 Harry 和 Rwo, 其中 Harry 和 Moondok、Leon 是好友，属于光明小学和清华大学；Rwo 和 KDB、Leon 是好友，

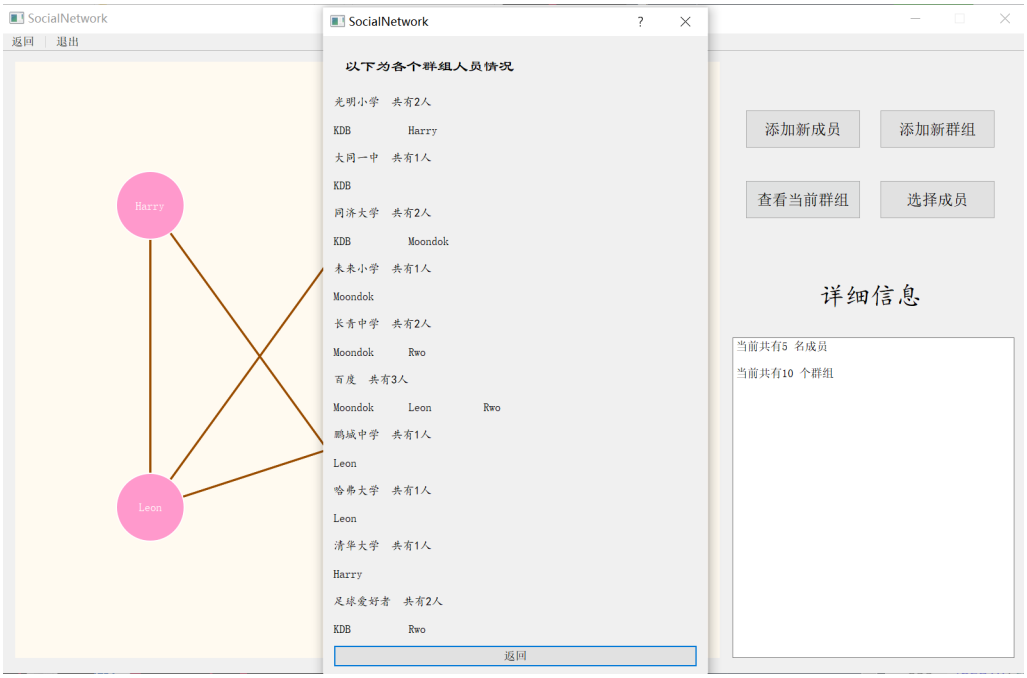
属于长青中学和百度。

在程序既有的 8 个群组上再添加 1 个群组“足球爱好者"，包含 KDB 和 Rwo 两个人。

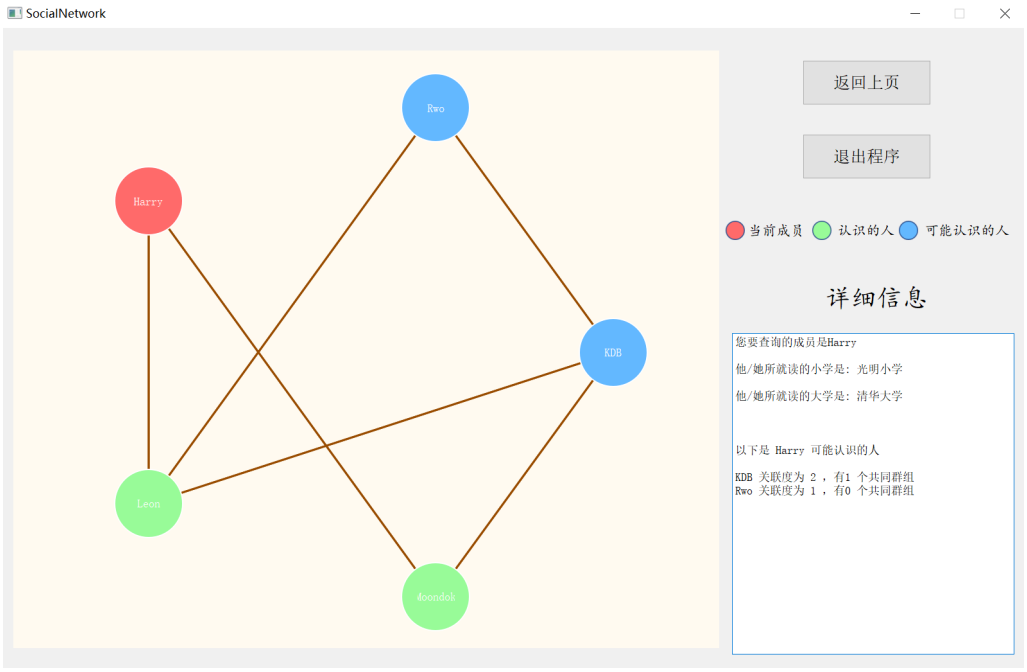
全部添加完的界面如下图所示:



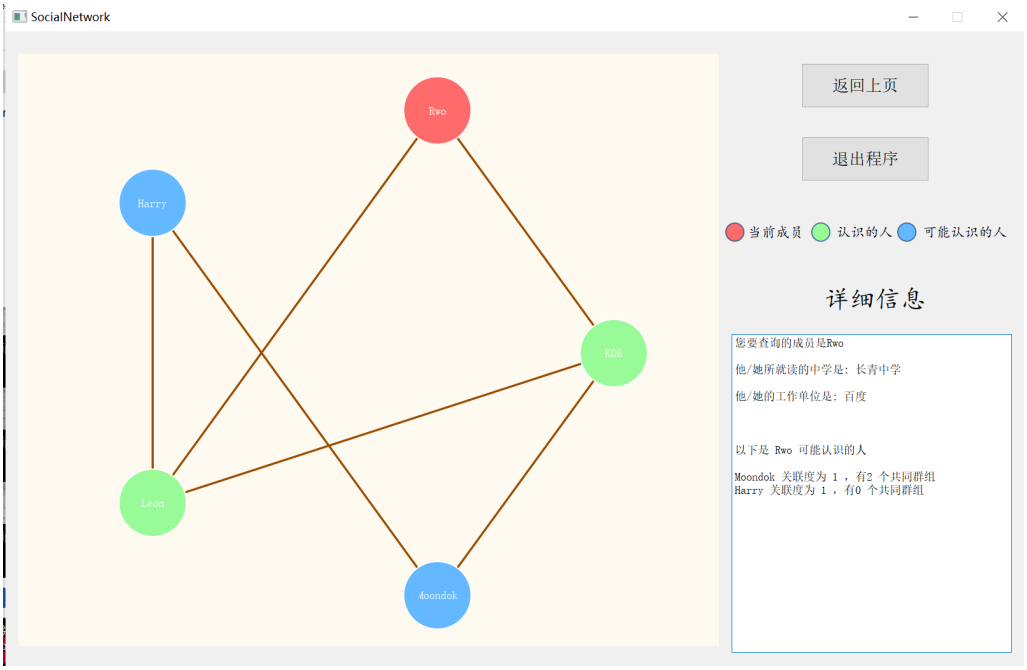
接下来查看当前群组，" 清华大学" 和" 足球爱好者" 都已经被添加进群组列表。



接下来查看"Harry" 的详细信息，和 KDB 的两个共同好友是 Leon 和 Moondok，共同群组是" 光明小学"; 和 Rwo 的共同好友是 Leon, 无共同群组:



最后查看"Rwo" 的详细信息，和 Monndok 的共同好友是 KDB，共同群组是" 长青中学" 和" 百度"; 和 Harry 的共同好友是 Leon, 无共同群组。



2.7 操作说明

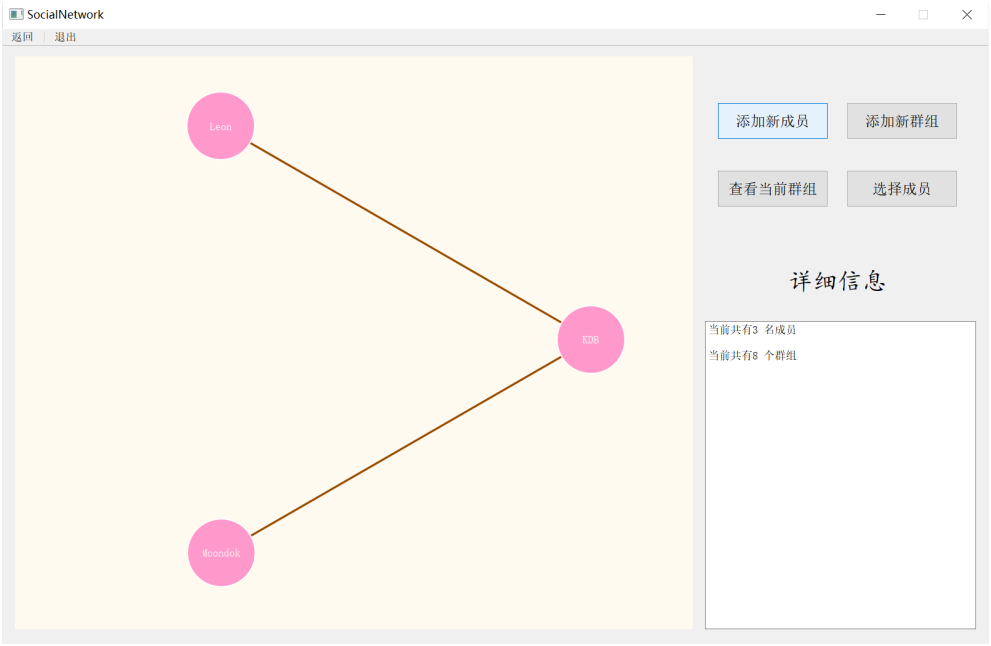
1. 首先，打开 2050525_ 陈开煦 _ 计算机科学与技术 _ 执行程序 中的综合应用题 文件夹, 其中的 *SocialNetwork.exe* 即为可执行文件。

名称	修改日期	类型	大小
iconengines	2022-09-02 13:44	文件夹	
imageformats	2022-09-02 13:44	文件夹	
platforms	2022-09-02 13:44	文件夹	
translations	2022-09-02 13:44	文件夹	
D3Dcompiler_47.dll	2014-03-11 18:54	应用程序扩展	3,386 KB
libEGL.dll	2019-12-04 4:49	应用程序扩展	28 KB
libgcc_s_dw2-1.dll	2015-12-29 6:25	应用程序扩展	118 KB
libGLESV2.dll	2019-12-04 4:49	应用程序扩展	2,748 KB
libstdc++-6.dll	2015-12-29 6:25	应用程序扩展	1,505 KB
libwinpthread-1.dll	2015-12-29 6:25	应用程序扩展	78 KB
opengl32sw.dll	2016-06-14 21:08	应用程序扩展	15,621 KB
Qt5Core.dll	2022-09-02 13:44	应用程序扩展	6,004 KB
Qt5Gui.dll	2019-12-04 4:49	应用程序扩展	6,086 KB
Qt5Svg.dll	2019-12-04 5:00	应用程序扩展	358 KB
Qt5Widgets.dll	2019-12-04 4:49	应用程序扩展	6,219 KB
SocialNetwork.exe	2022-09-02 13:40	应用程序	493 KB

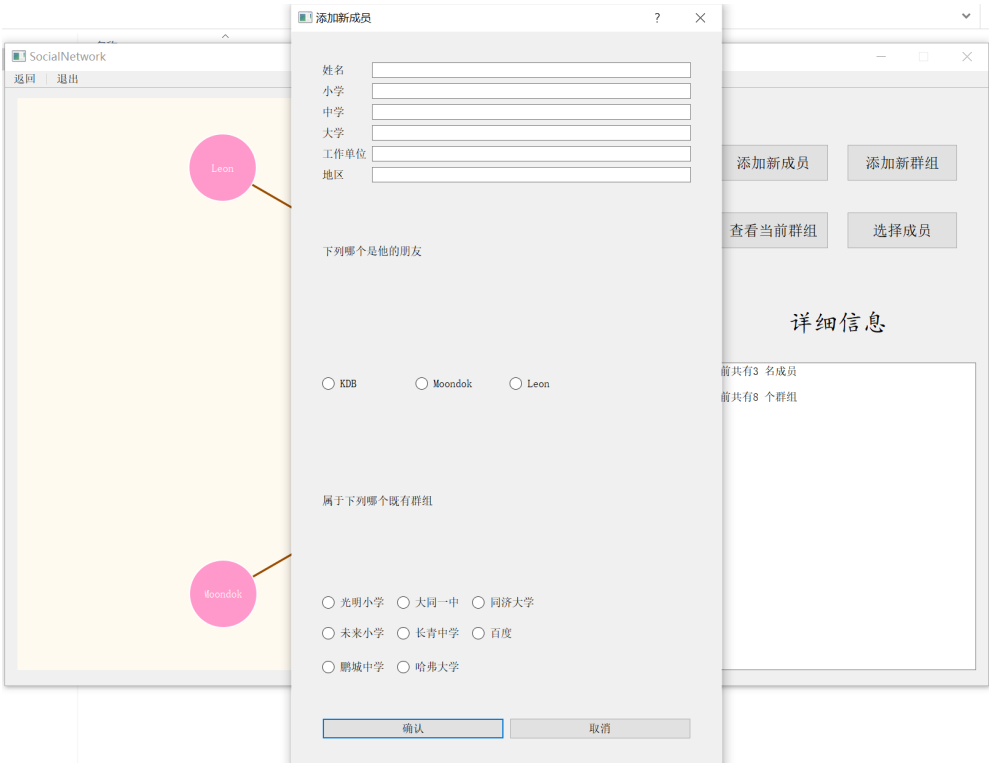
2. 进入程序后显示如下欢迎界面，点击" 开始演示" 按钮即可进入演示界面。

3. 演示界面的左侧是展示当前社会关系图的画板，右下方显示当前成员的数量和群组的数量。

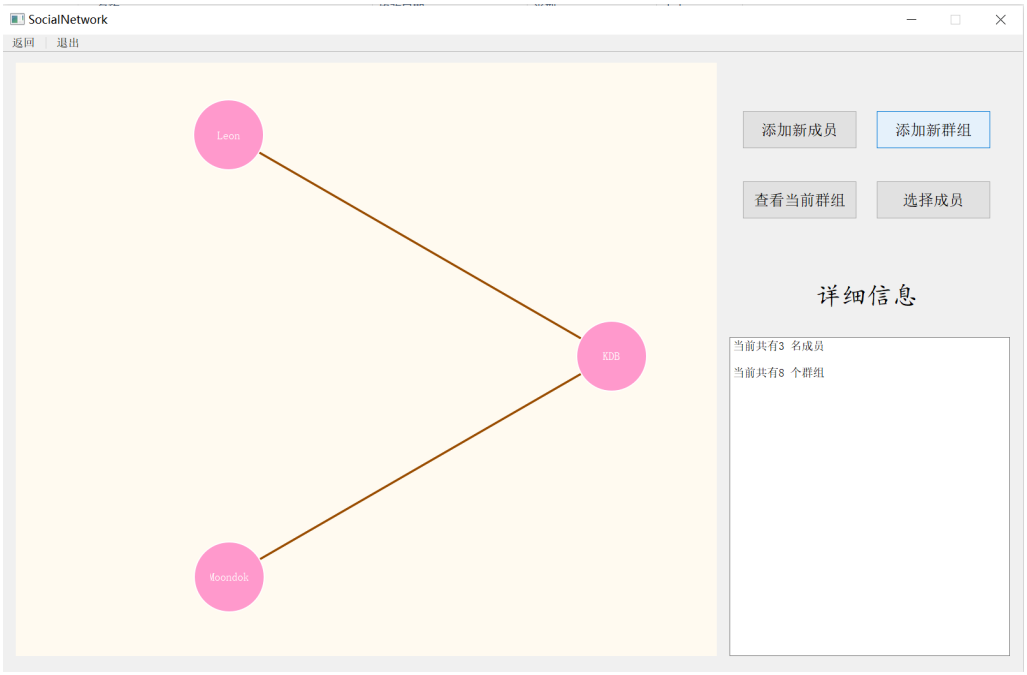
   在演示界面中点击" 添加成员" 按钮，程序会弹出添加成员的对话框。当前成员的数量大于等于 25 时将无法添加成员。



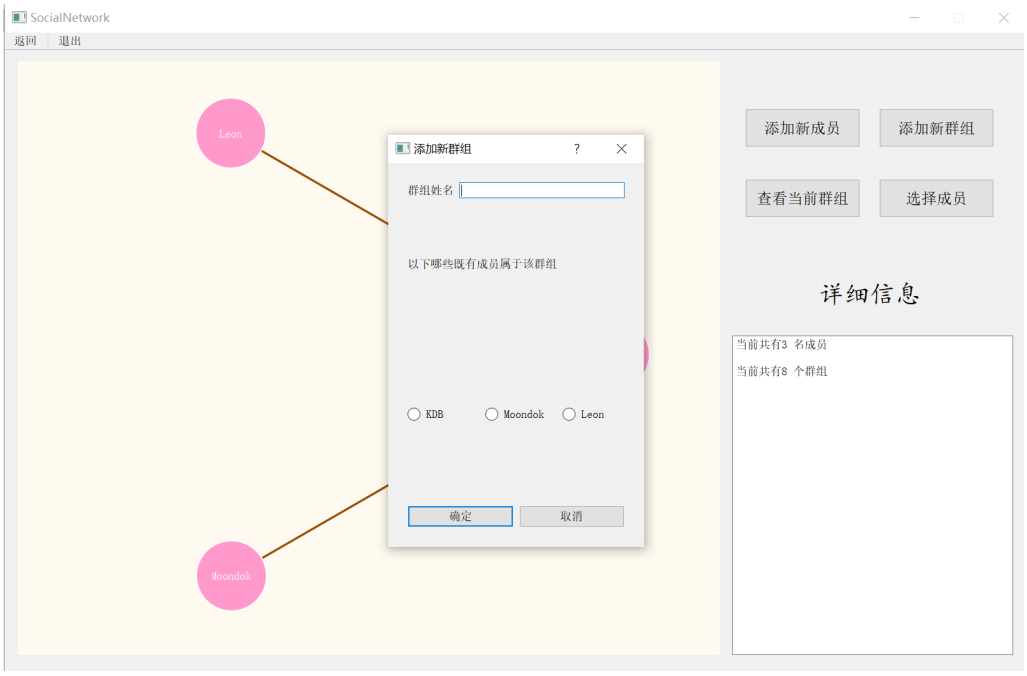
4. 在添加成员对话框中可以选择新成员和既有成员以及既有群组的关系，新成员的名字不能为空。



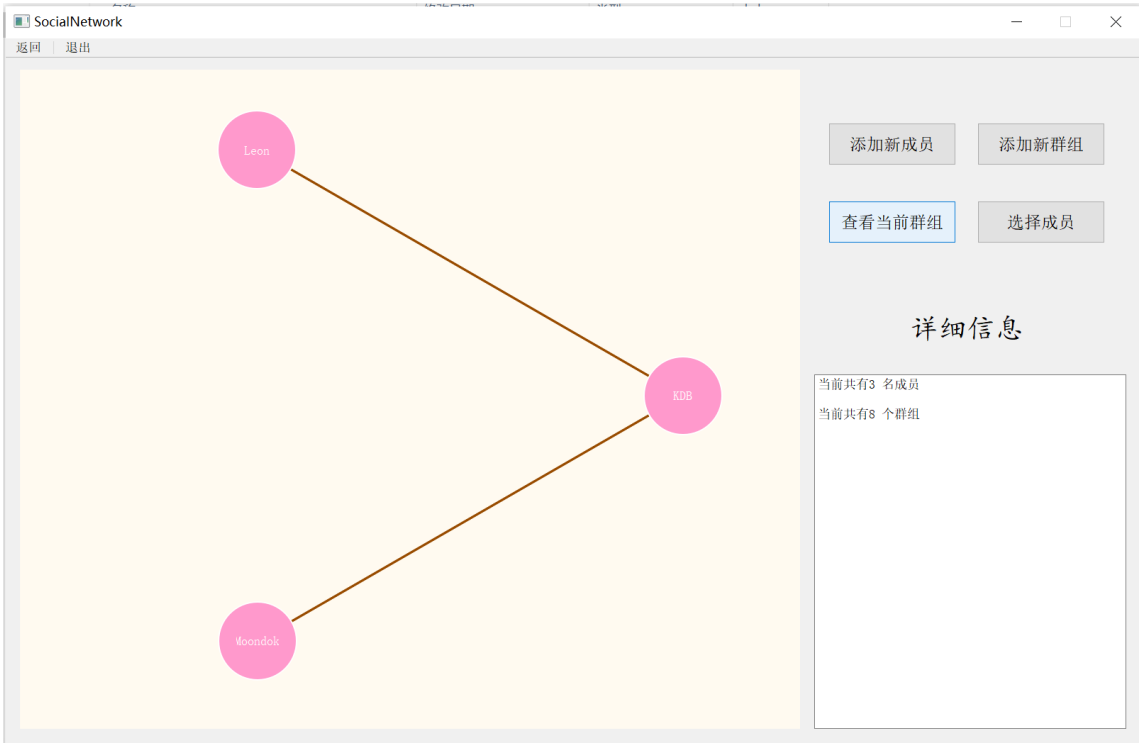
5. 在演示界面中点击" 添加群组" 按钮，程序会弹出添加群组的对话框。当前群组的数量大于等于 18 时将无法添加群组。

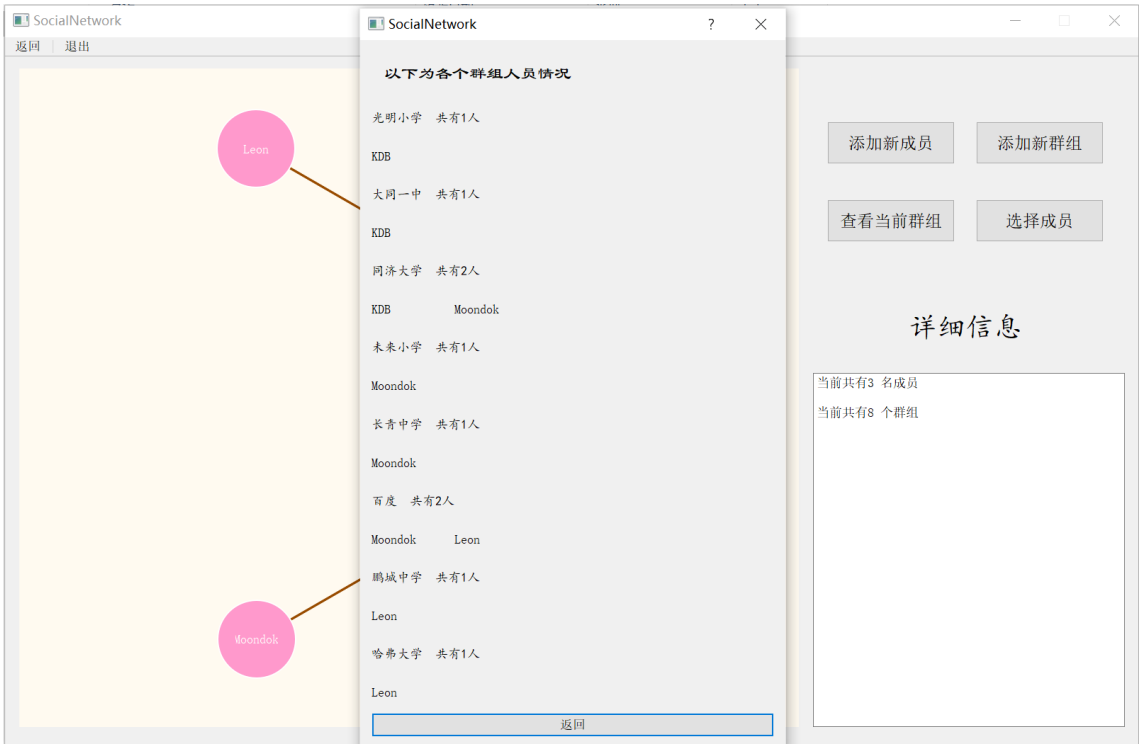


6. 在添加群组对话框中可以选择新群组和既有成员的关系。请注意新群组的名字不能为空。

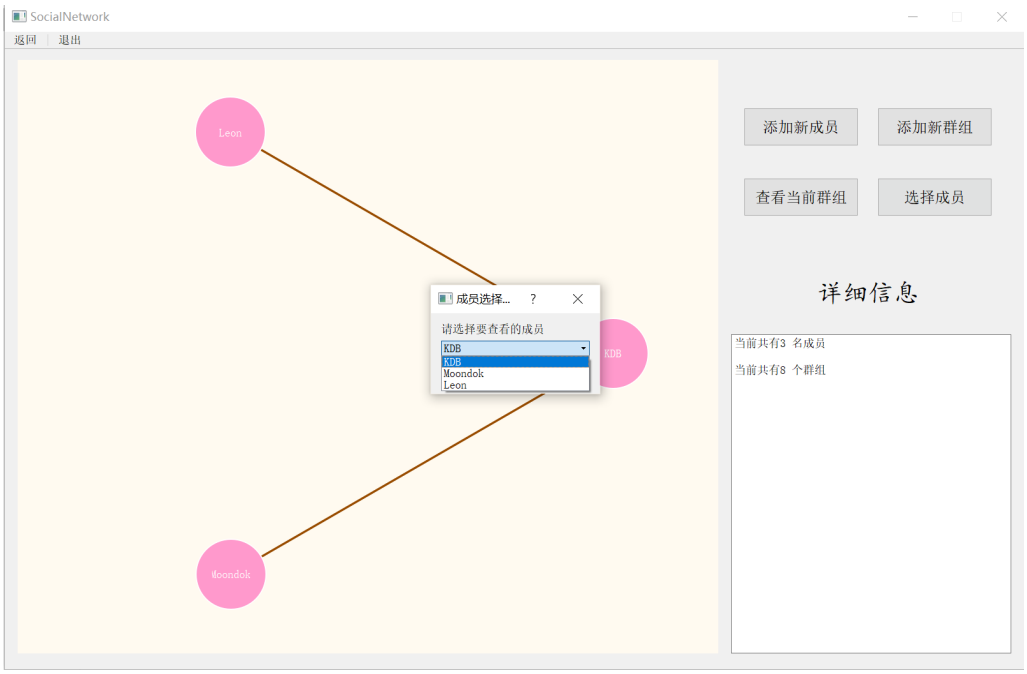


7. 在演示界面中点击" 查看当前群组" 按钮，程序会弹出对话框显示当前所有群组的成员信息。


SocialNetwork
返回
退出
Leon
KDB
Moondok
添加新成员
添加新群组
查看当前群组
选择成员
详细信息
当前共有3 名成员
当前共有8 个群组

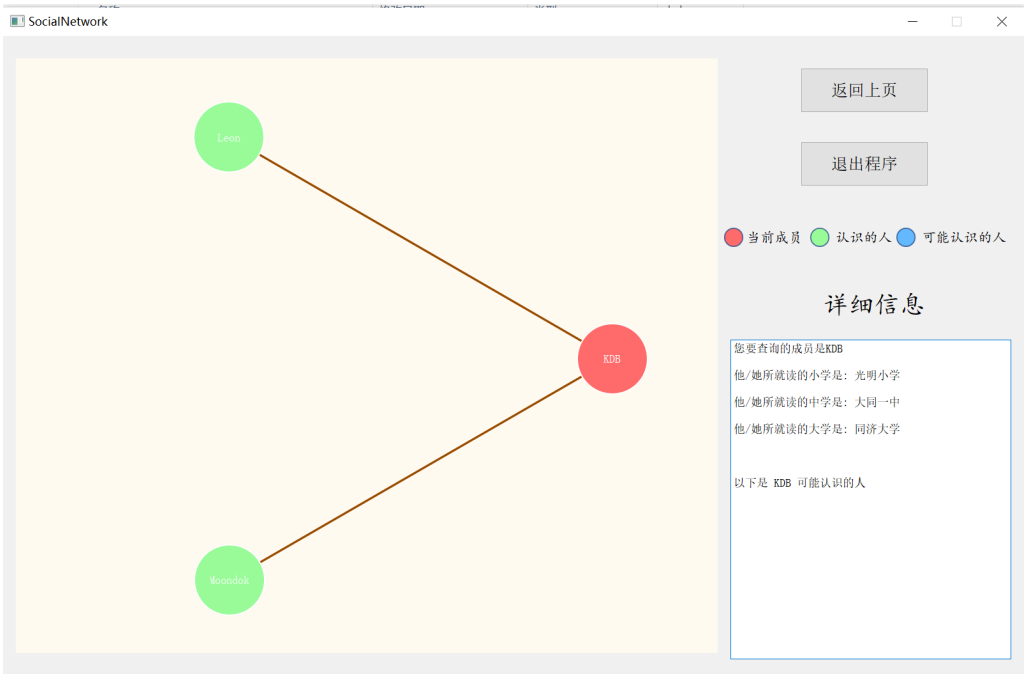

SocialNetwork
返回
退出
Leon
Moondok
添加新成员
添加新群组
查看当前群组
选择成员
详细信息
当前共有3 名成员
当前共有8 个群组
SocialNetwork
以下为各个群组人员情况
光明小学 共有1人
KDB
大同一中 共有1人
KDB
同济大学 共有2人
KDB Moondok
未来小学 共有1人
Moondok
长青中学 共有1人
Moondok
百度 共有2人
Moondok Leon
鹏城中学 共有1人
Leon
哈弗大学 共有1人
Leon
返回

8. 在演示界面中点击" 选择成员" 按钮，程序会弹出一个对话框，用户可以在该对话框中选择一个成员。选择确认后，界面会刷新并显示该成员的详细信息。



9. 在刷新后的界面中，左侧画板中以不同颜色显示当前选中的成员，该成员的好友以及可能认识的人。右侧“详细信息”一栏中会显示该成员和可能认识的人之间的关联度和共同群组数。

   同时，用户也可以点击右上方的" 返回上页" 以添加或查看其他成员。



3 实践总结

3.1 所做的工作

在本次课程设计中，为了更好地完成开发，我首先通过侯捷老师的 C++ 课程系统性地复习了 C++ 面向对象的知识。之后，我主要通过参考书和网课自学了 Qt Widget Appication 的基本架构以及一些常用控件。

在基本完成学习任务后，我分别对两个题目进行需求分析，数据结构设计，对象图、类图等蓝图的确定，模块的划分，实现以及最后的测试。在多台 Windows 操作系统的电脑下确定程序的正确性和稳定性后，将其打包成不需要配置 Qt 环境便可以直接运行的可执行文件。

最后，我对本次课程设计进行总结并完成了这篇报告。

3.2 总结与收获

本次课程设计历经的时间较长，在简要复习完 C++ 知识后，我便开始对 Qt 开发框架进行学习。起初，面对繁多的书籍以及网课资源，我有些畏怯心理。但随着学习的深入，当我对 QWidget 基本控件有了一定了解后，我便开始了设计和开发，走上了" 边学边做" 的路子。

在设计二叉树程序时，为了优化用户体验，我摒弃了传统的通过输入先序遍历序列建立二叉树的做法，采用了默认建立满二叉树，通过给予用户删除叶结点的权力来定制二叉树的方法。这种做法极大地方便了用户，却使我的设计遇到了一个很大的困难: 我希望实时地将内存中的树反馈到 Ui 界面上，同时可以通过用户的点击操作删除结点。起初我在将 Ui 界面中显示的 *QLabel* 放置在二叉树结点结构体中，这种方法可以很好地实现上述要求中的第二点，但面对第二点却束手无策。最终我将 *QLabel* 和二叉树结点一一关联起来，这才解决了这个问题。这个困难让我充分意识到开发带有 GUI 的程序时，处理好界面输出和内部结构的关系的重要性。有了算法实现题的经验，开发综合应用题的设计明显顺利了许多。

除了设计方面的问题，由于对 Qt 的部分控件不甚熟悉，我也犯了一些由于使用控件不当造成的错误，好在经过查阅网络以及书籍资料，遇到的问题基本得到了解决。

经过这次课程设计，我的能力在多个方面都得到了提升。首先是编程技能上面，经过学习和实践，我对 Qt 框架及其常用控件、绘图模型、事件截留、文件操作和信号槽机制等都有了一定的了解和掌握，扩展了我的 *skill set*。在实践的过程中，对于面向对象的设计思想也有了更深的了解和认识。

另外，通过这次课程设计，我的学习能力，独立发现问题、解决问题的能力也

得到了提高。

经过这次课程设计，我也有一些自己的感悟和体会。

要对知识和技术抱有热忱和信心。只有热爱自己的任务和工作，才能有足够的信心和动力把它做好。

万事开头难，只有勇于迈出探索的第一步，才有机会面对后面的挑战和惊喜。

整体而言，在这次课程设计中我学到了很多，自己的编程技巧、学习能力也得到了进步。衷心感谢老师的付出和同学们的帮助。没有你们的奉献，也就没有我的提升和进步，谢谢你们！

参考文献

[1] 陆文周.Qt5 开发及实例 (第 3 版)[M]. 北京: 电子工业出版社,2017.

[2] 王维波, 栗宝鹃, 侯春望.Qt 5.9 C++ 开发指南 [M]. 北京: 人民邮电出版社,2018.

[3] 霍亚飞.Qt Creator 快速入门 (第 3 版)[M]. 北京: 北京航空航天大学出版社,2017.

[4] Thomas H.Cormen,Charles E.Leiserson,Ronald L.Rivest,Clifford Stein. 算法导论 [M]. 北京: 机械工业出版社,2013.

[5] 严蔚敏, 吴伟民. 数据结构 (C 语言版)[M]. 北京: 清华大学出版社,2007.

[6] 查康. 精通 Git (第二版) [M] 北京: 人民邮电出版社,2017.

[7] Shijia Yin.Qt 值输入控件（QSpinBox 和 QDoubleSpinBox）[EB/OL].(2020-3-16)[2022-8-31].https://blog.csdn.net/YinShiJiaW/article/details/104896416.

[8] 阿里云.Qt 实用技巧：代码中 QIcon 缩放（QPixmap 的手动放大和 QIcon 自动缩小）[EB/OL].(2022-5-30)[2022-8-31].https://developer.aliyun.com/article/942332.

[9] oop4587.QT 通过起点、终点、弧度(方向)来绘制圆弧 [EB/OL].(2021-8-17)[2022-8-31].https://blog.csdn.net/oop4587/article/details/119758007.

[10] A 三三. 二叉线索树的先序、中序、后序的线索化及其遍历 [EB/OL].(2020-2-15)[2022-8-31].https://blog.csdn.net/m0_43456002/article/details/104268875.

[11] 胡乱 huluan. 数据结构图（二）社交网络 [EB/OL].(2020-2-22)[2022-8-31].https://blog.csdn.net/qq_44867435/article/details/104443790.

[12] 三石目.Qt5 开发：使用 windeployqt 打包发布 [EB/OL].(2019-7-3)[2022-8-31].https://blog.csdn.net/Stone_Wang_MZ/article/details/94591363.