**上 海 大 学**

**2019-2020学年冬季学期**

**《数据结构（2）》实验报告**

实 验 组 号： 01

上 课 老 师： 沈 俊

小 组 成 绩：

小组成员成绩表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 学号 | 姓名 | 贡献因子 | 成绩 |
| 1 | 17120193 | 张赛 | 20 |  |
| 2 | 18120189 | 林艺珺 | 20 |  |
| 3 | 18120206 | 徐奕婷 | 20 |  |
| 4 | 18120212 | 叶菁 | 20 |  |
| 5 | 18120221 | 陈思文 | 20 |  |

注：小组所有成员的贡献因子之和为100.

计算机工程与科学学院

2019年12月13日

实验一 树和森林

一．实验内容

（一）设计性实验：树的孩子表示法的设计与实现

1. 设计并实现树的孩子表示法，完成树的先根、后根和层次遍历。
2. 在树的孩子表示法中，设计并实现相应函数，求该树的度和叶子数。
3. 在树的孩子表示法中，设计并实现相应函数，求相应二叉树的高度和叶子数。

（二）综合性实验：家谱管理系统

家谱管理系统是查询家谱信息必不可少的一部分，利用家谱管理系统可以清楚的查询到家族成员的详细信息。成员的信息一般应包含以下内容：姓名、出生日期、婚姻状况（已婚、未婚等）、地址、目前状况（健在或身故）、死亡日期（若其已死亡）等，也可附加其它信息。系统要求设计合理的数据结构存储家谱中各成员的信息（一般定义结构体）和成员之间的关系。要求系统可以插入、查询、修改、删除等功能。

1. 从文件输入信息，建立初始家谱树。
2. 用恰当的形式显示家谱树。
3. 根据代号n，显示第n代所有人的姓名和人数。
4. 按照姓名查询，输出成员信息（包括其本人、父亲、孩子的信息，以及他在家谱中的代数）。
5. 按照出生日期查询成员名单。
6. 输入两人姓名，确定其关系。
7. 给某人添加孩子。
8. 删除某人（若其还有后代，则一并删除）。
9. 修改某人信息。

二．主要算法设计

（一）设计性实验：树的孩子表示法的设计与实现

1. 问题分析与算法设计

树的孩子表示法的设计：用一维数组顺序存储树中的各结点的信息，并将各结点的孩子信息组成一个单链表。孩子信息链中的每一个结点表示一个孩子结点，它由两个域组成，其中一个域表示该孩子结点在数组中的序号，另外一个域存储指向兄弟结点的指针。在结点数组中，每个元素包括结点的自身信息以及该结点的信息结点链表的头指针。

1. 设计并实现树的孩子表示法，完成树的先根、后根和层次遍历。

先根遍历：若树为空，遍历结束。否则：1）访问根节点。2）按照从左到右的顺序先根遍历根结点的每一棵子树。

后根遍历：若树为空，遍历结束。否则：1）按照从左到右的顺序先根遍历根结点的每一棵子树。2）访问根结点。

层次遍历：1）初始化队列，并将根结点入队。2）当队列非空时，取出对头结点p，转步骤3）；如果队列为空，结束遍历。3）访问取出的结点p；如果结点p有孩子，则依次将它们入队列。4）重复步骤2）、3），直到队列为空。

1. 在树的孩子表示法中，设计并实现相应函数，求该树的度和叶子数。

求树的度：1）从左到右顺序先根遍历根结点的每一棵子树。2）求子树的度数，与先前结点度数做比较，返回最大值。

求树叶子数：层次遍历每个结点，若其度数为0，则为叶子结点。

1. 在树的孩子表示法中，设计并实现相应函数，求相应二叉树的高度和叶子数。

1）求对应二叉树的高度

本功能的实现可以以普通二叉树的高度求法作为参考，在求普通二叉树的高的时候，是使用递归的方法，比较各个节点的左右字数的高度，得出高度较大的一颗子树，在该棵子树的高度上不断进行比较和加一操作来得到整棵树的高。

在树转换为二叉树后，对应的左右孩子变为了对应的第一个孩子和右兄弟结点，用同样的思想进行递归比较，便可得到对应二叉树的高。

2）求对应二叉树的叶子数

通过观察转化后的二叉树不难得到，二叉树中的叶子结点都是满足既没有第一个孩子结点也没有右兄弟结点的，所以，只需遍历保存所有结点信息的数组，其中没有孩子结点和兄弟结点的结点数量便是叶子数。

2. 函数功能设计

|  |  |
| --- | --- |
| ChildTree主要成员函数 | |
| ChildTree(); | 无参数构造函数 |
| ChildTree(const ElemType&e,int size=DEFAULT\_SIZE); | 以元素e为根构造函数 |
| ChildTree(ElemType items[],int parents[],int r,int n,int size=DEFAULT\_SIZE); | 给定元素构造函数 |
| virtual ~ChildTree(); | 析构函数 |
| int GetRoot()const; | 求根节点 |
| int NodeCount()const; | 求结点个数 |
| Status GetElem(int cur, ElemType &e) const; | 用e返回结点元素值 |
| int FirstChild(int cur) const; | 求第一个孩子结点 |
| int RightSibling(int cur) const; | 求第一个右兄弟 |
| int NodeDegree(int cur) const; | 结点度数 |
| void PreRootOrder(void (\*Visit)(const ElemType &))const; | 先根遍历 |
| void PostRootOrder(void(\*Visit)(const ElemType &))const; | 后根遍历 |
| void LevelOrder(void(\*Visit)(const ElemType&))const; | 层次遍历 |
| int Degree(int r)const; | 求树的度 |
| int Leaf()const; | 叶子数 |
| int bitreeLeaf() const; | 二叉树叶子数 |
| int bitreeHeight() const; | 二叉树高度 |

（二）综合性实验：家谱管理系统

1. 问题分析

本题是一个树的数据结构的实际应用管理系统设计。家谱作为典型的树型数据结构存储形式，借助课本上关于树的知识，选用孩子兄弟表示法来构造最基本的家谱树，以更快、更便利地实现要求在本家谱管理系统中实现的功能，尤其是第（3）条及第（6）条功能需求，使用双亲孩子表示法难以便捷地得到结果。

在原本孩子兄弟表示法的基础上，为便于寻找所需结点，增加对于特殊结点指针返回的函数；将输出信息的执行语句集成在一个函数中，以方便调用输出；设计功能菜单、完善错误处理，以方便用户的使用。

文件读取写入方面，采用可读性较好、更方便程序实现的“一行一条信息”的数据组织；在每次调用可能修改信息的执行语句完毕后，立即将数据写入文件，以避免修改数据丢失、导致错误。

2. 功能模块及系统界面设计



图2-2 系统开始菜单界面

3. 函数功能设计

Genealogy类设计基于课本ChildSiblingTree类模版，因此相似函数再此处不再提及，本段主要说明在Genealogy.h中实现的家谱管理系统功能函数。

|  |  |
| --- | --- |
| Genealogy类主要成员函数（即家谱树主要功能函数） | |
| Genealogy(MemberData persons[], int parents[], int r, int n); | 创建家谱树 |
| int Generation(Member \*r, int gen, void (\*Visit)(const string &)); | 显示第n代所有人的姓名和人数 |
| void FindFather(Member \*r, string n); | 按照姓名查询，输出成员父亲信息 |
| bool FindMember(Member \*r, string n, bool flag = 0); | 按照姓名查询，输出成员本人信息 |
| void FindSon(Member \*r, string n); | 按照姓名查询，输出成员孩子信息 |
| bool FindBirth(Member \*r, string b, bool flag = 0); | 按照出生日期查询成员名单 |
| void MemberInformation(Member \*index) const; | 输出成员信息 |
| void ViewAllInformation(Member \*index) const; | 输出所有成员信息 |
| void Relationship(Member \*r, string a, string b, bool flaga = 0, bool flagb = 0); | 输入两人姓名，确定其关系 |
| Status InsertChild(Member \*cur, int i, const MemberData &e); | 给某人添加孩子 |
| void Delete(Member \*r); | 删除某人（若其有后代，则一并删除） |
| void Modify(Member \*index, MemberData e); | 修改某人信息 |
| Member \*PrevSibling(Member \*r) const; | 返回成员前一个兄弟 |
| Member \*FirstChildParent(Member\* r) const; | 若为某结点第一个孩子，返回该父结点 |
| Member\* Search(string index); | 按照姓名返回成员指针 |
| void ResetGeneration(Member \*r); | 重新计算代数 |
| 文件输入输出函数 | |
| void OutFile(string filename= "Genealogy.txt"); | 输出每条结点信息到文件 |
| void OutAll(ostream &os, Member \*p, int &num, int (&parentnum)[100]); | 依次输出除根结点之外每条信息到文件 |
| template<int N>  void InFile(MemberData (&persons)[N], int (&parents)[N], int &r, int &n, string filename) | 读入文件到树 |

三．主要数据组织

（一）设计性实验：树的孩子表示法的设计与实现

代码3.1

|  |
| --- |
| 1. **struct** ChildTreeNode //孩子树结点设计 2. { 3. //数据成员： 4. ElemType data; //数据域 5. Node<int> \*childLKList; //孩子链表 6. ChildTreeNode(); //无参数构造函数 7. ChildTreeNode(ElemType item,Node<int>\*childlk = NULL); 8. //有参数的构造函数 9. }; 10. **class** ChildTree //孩子树类 11. { 12. protected: 13. ChildTreeNode<ElemType>\*nodes; //树结点 14. int maxSize; //树结点最大个数 15. int root,num; //根位置和结点 16. public: 17. ChildTree(); //无参数构造函数 18. ChildTree(const ElemType&e,int size=DEFAULT\_SIZE);//元素E为根的树 19. ChildTree(ElemType items[],int parents[],int r,int n,int size=DEFAULT\_SIZE); 20. virtual ~ChildTree(); //析构函数 21. ……//其余功能函数此处省略 22. }; |

（二）综合性实验：家谱管理系统

1. 数据文件格式

数据文件默认名称为“Genealogy.txt”，文件中有多条数据。每条数据其中包括了家族成员的姓名、出生日期、婚姻状况、个人地址、是否在世（目前状况）、死亡日期（若其已死亡）、家族代数、父亲结点下标。格式如图3-1所示。信息所表示的家谱树，如图3-2所示。

A close up of text on a white background

Description automatically generated

图3-1 家族成员信息

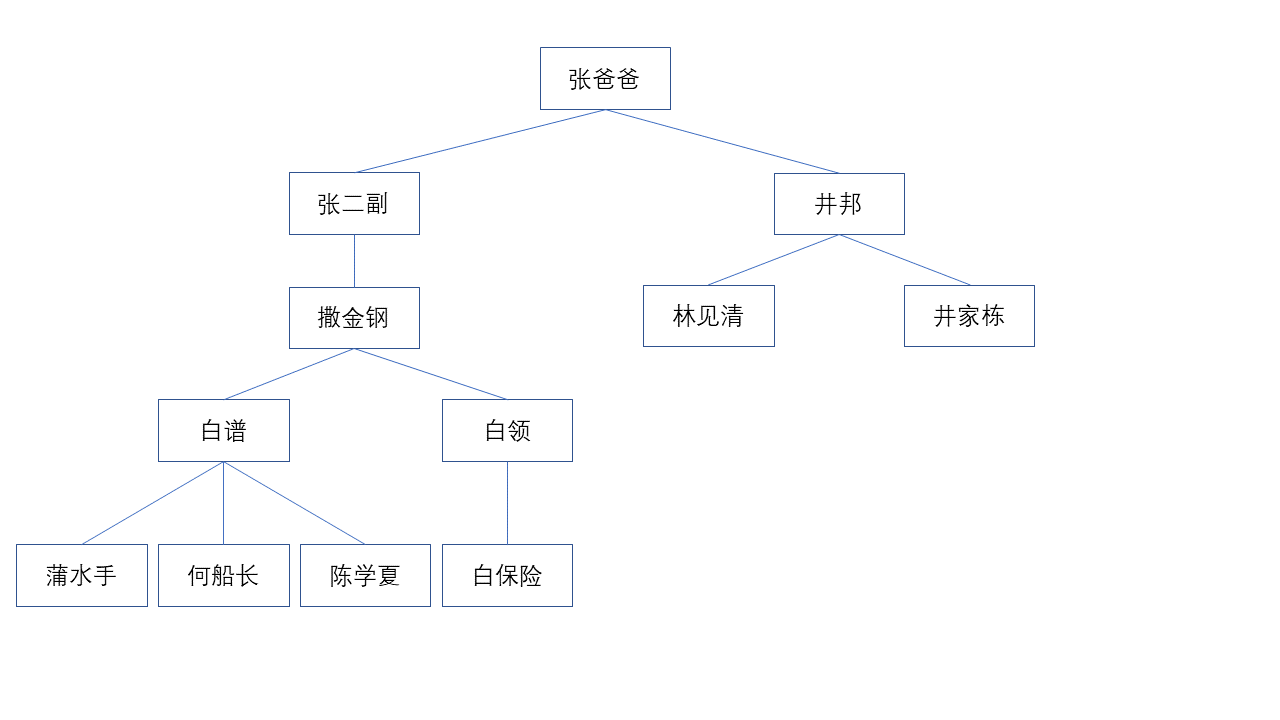


图3-2 家谱树

2. 数据结构设计

实验内容涉及到多条记录的处理，每条记录含有不同类型的数据项，使用结构体存储信息较为方便，再用树的数据结构存储完整家谱树，数据结构设计如下：

代码3.2

|  |
| --- |
| 1. **struct** MemberData //家族成员信息结构体 2. { 3. string name; //姓名 4. string birth; //出生日期 5. **bool** marriage; //婚姻状况 6. string address; //个人地址 7. **bool** alive; //是否在世 8. string death; //死亡日期 9. **int** generation; //家族代数 10. }; 11. **struct** Member //家族成员结构体 12. { 13. MemberData person; //家族成员信息 14. Member \*firstChild; //指向第一个孩子的指针 15. Member \*nextSibling; //指向下一个兄弟的指针 16. Member(); //默认构造函数 17. Member(MemberData pers, Member \*fChild = NULL, Member \*nSibling = NULL); //构造函数 18. }; 19. **class** Genealogy 20. { 21. public: 22. Member \*root; 23. …… //函数成员此处省略 24. }; |

孩子兄弟树结构中，每个树的结点即为家族成员。本例中Member为树的结点，Genealogy为树。

四．测试分析

（一）设计性实验：树的孩子表示法的设计与实现

样例1 普通树

|  |
| --- |
| 1. **char** items[] = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H' }; 2. **int** parents[] = { -1, 0, 0, 0, 1, 1, 3, 3 }; 3. **int** r = 0, n = 8, cur, f = 0; 4. ChildTree<**char**> t(items, parents, r, n); |

样例1测试结果

A screenshot of a cell phone

Description automatically generated

图4-2-1 树的孩子表示法样例1测试

样例2 仅有单个结点的树

|  |
| --- |
| 1. **char** items[]={'A'}; 2. **int** parents[]={-1}; 3. **int** r=0,n=1,cur,f=0; 4. ChildTree<**char**> t(items, parents, r, n); |

样例2测试结果

A screenshot of a computer

Description automatically generated

图4-2-2 树的孩子表示法样例2测试

样例3 除最后一个结点外每个结点仅有一个孩子的树

|  |
| --- |
| 1. **char** items[]={'A','B','C','D','E','F'}; 2. **int** parents[]={-1,0,1,2,3,4}; 3. **int** r=0,n=6,cur,f=0; 4. ChildTree<**char**> t(items, parents, r, n); |

样例3测试结果

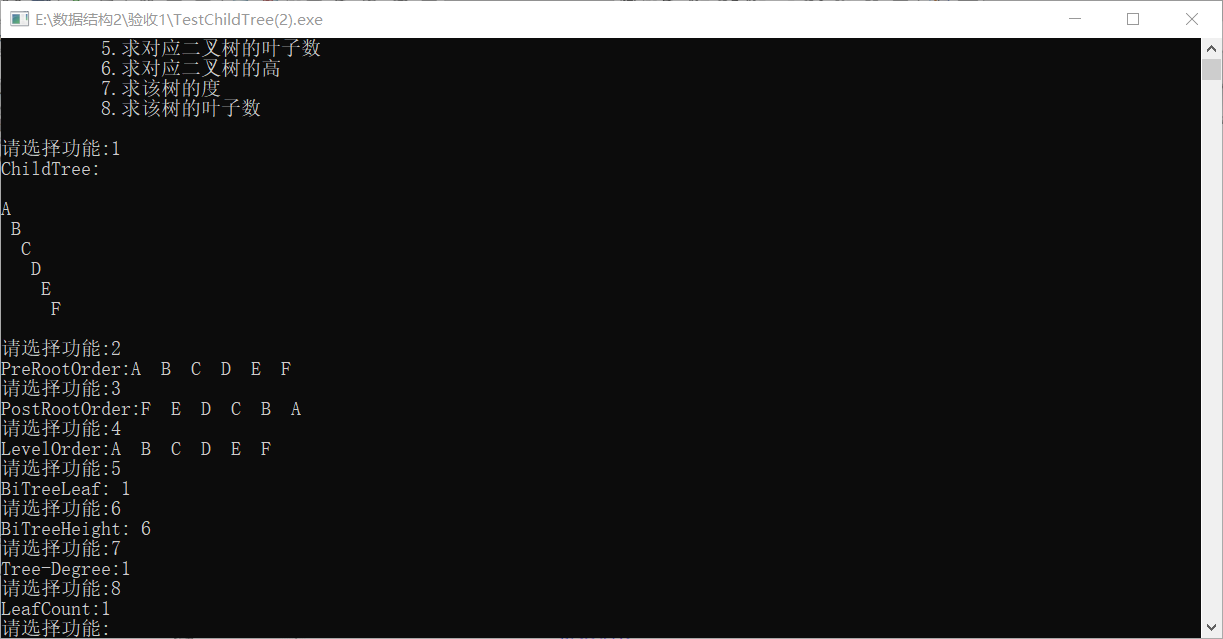


图4-2-3 树的孩子表示法样例3测试

样例4 除根结点外所有结点均为叶子结点的树

|  |
| --- |
| 1. **char** items[]={'A','B','C','D','E','F','G'}; 2. **int** parents[]={-1,0,0,0,0,0,0}; 3. **int** r=0,n=7,cur,f=0; 4. ChildTree<**char**> t(items, parents, r, n); |

样例4测试结果

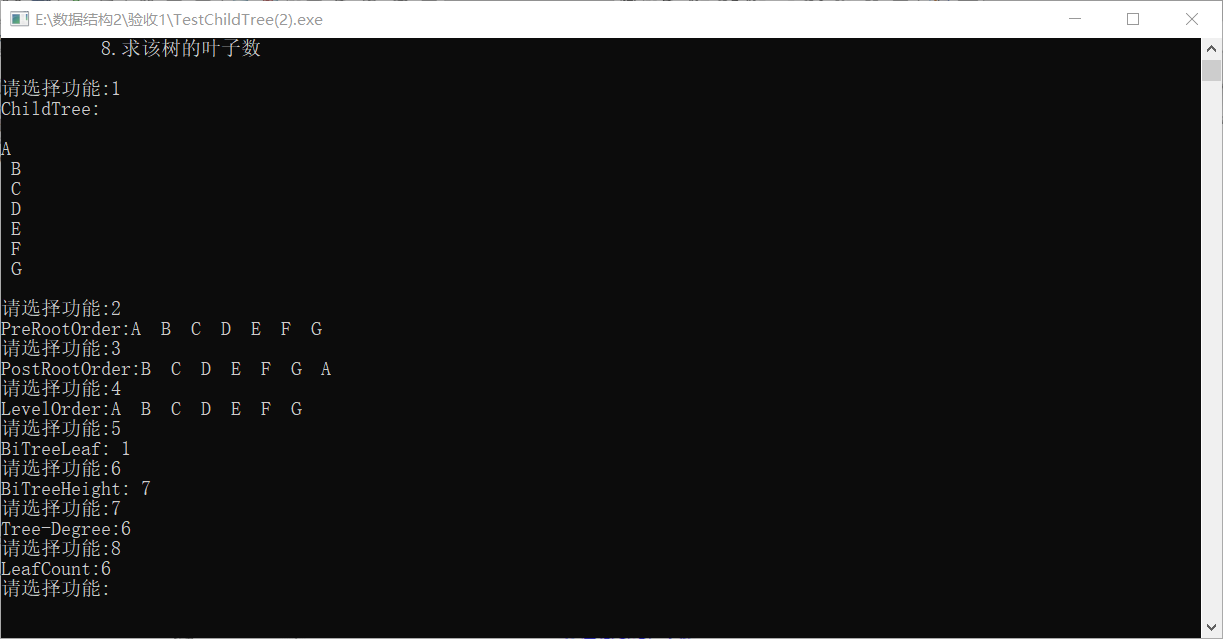


图4-2-4 树的孩子表示法样例4测试

样例5 空树

|  |
| --- |
| 1. **char** items[]={}; 2. **int** parents[]={}; 3. **int** r=0,n=0,cur,f=0; 4. ChildTree<**char**> t(items, parents, r, n); |

样例5测试结果

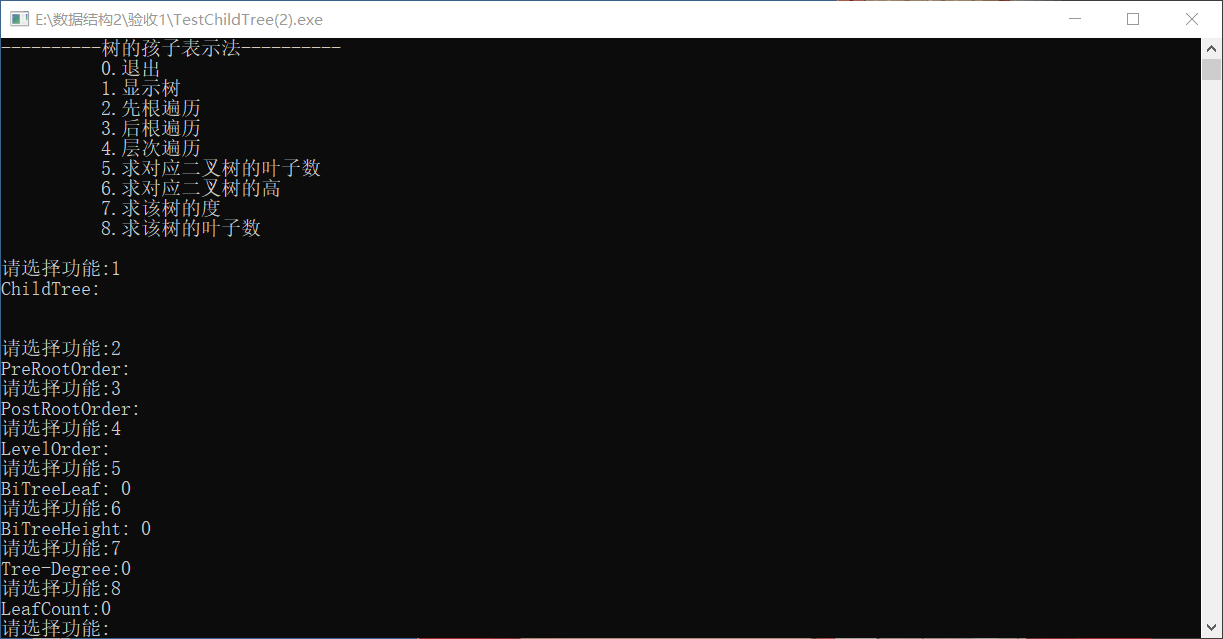


图4-2-5 树的孩子表示法样例5测试

（二）综合性实验：家谱管理系统

1. 从文件输入信息，建立初始家谱树：读入如图3-1信息后，程序显示如图2-2。

2. 用恰当的形式显示家谱树。

读入如图3-1信息后，程序显示如图2-2。选择功能“g 显示完整家谱”，同样输出如图2-2的家谱。

3. 根据代号n，显示第n代所有人的姓名和人数。

选择功能“2 查询家谱第n代”，程序提示“输入查询家谱代数: ”，依法输入数字后，显示如图4-2-3。若输入非法，则显示“查询家谱代数不合法。”，另，如输出超过1位的数字，系统默认选取第一位数字作为查询数字。

样例1

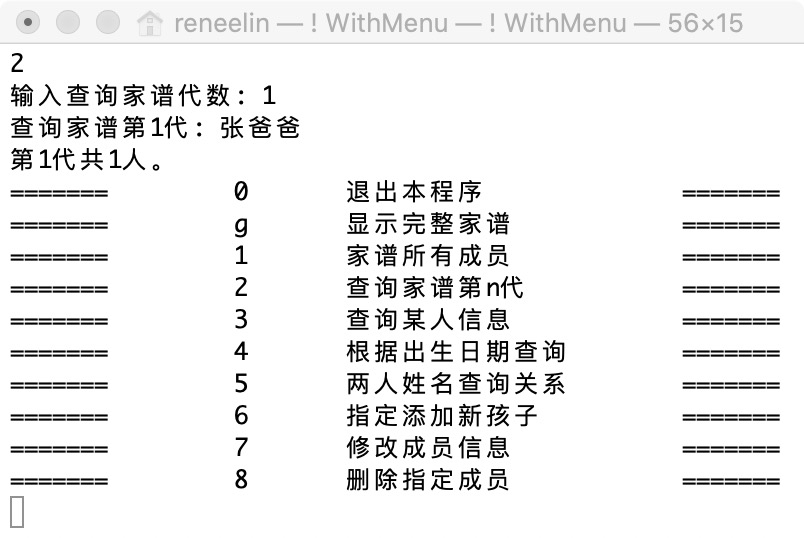


图4-2-3(1) 查询家谱第1代

样例2

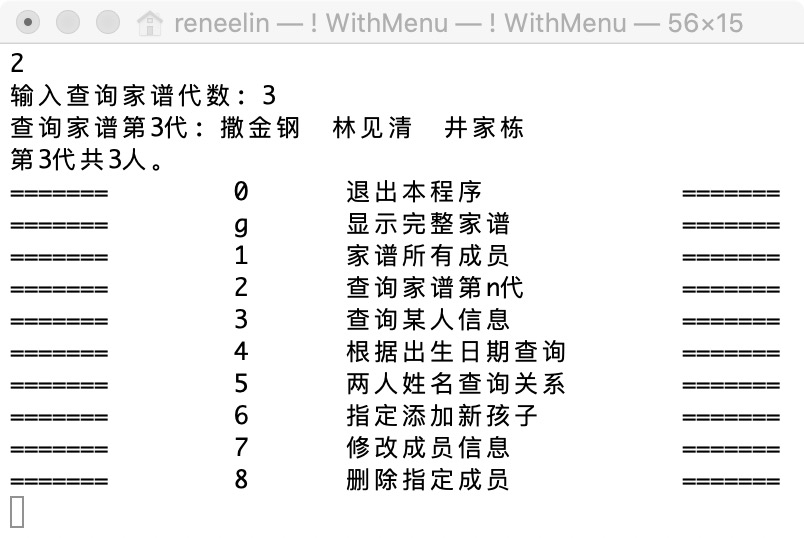


图4-2-3(2) 查询家谱第3代

样例3

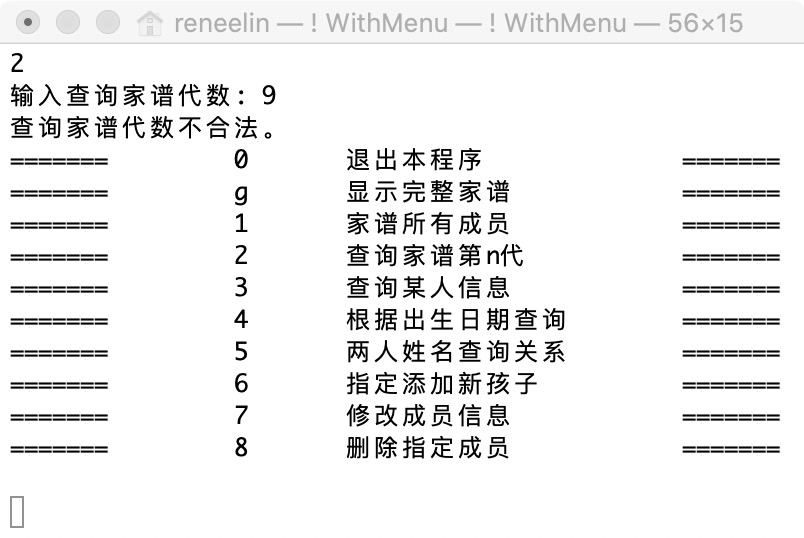


图4-2-3(3) 查询家谱第9代

样例4

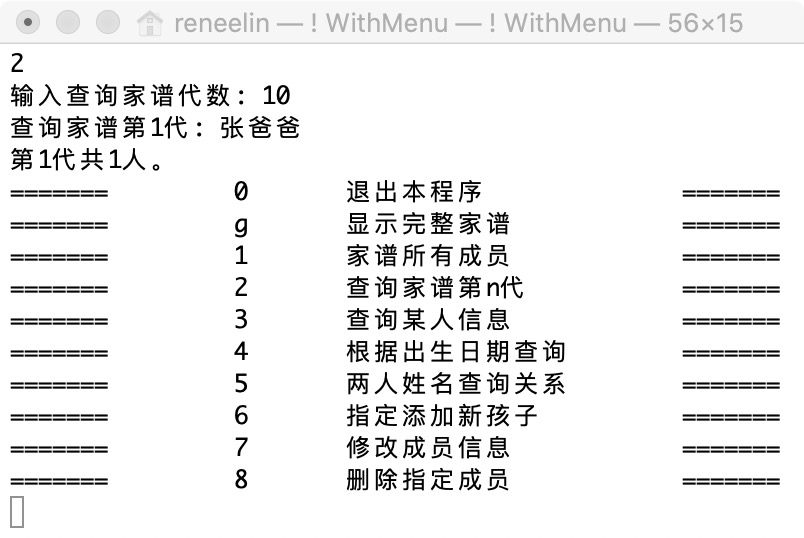


图4-2-3(4) 查询家谱第10代

样例5

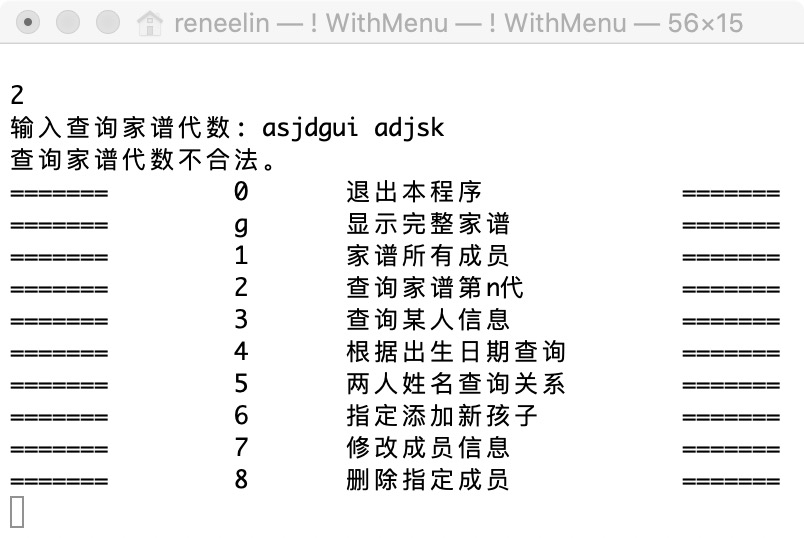


图4-2-3(5) 查询家谱第asjdgui adjsk代

样例6

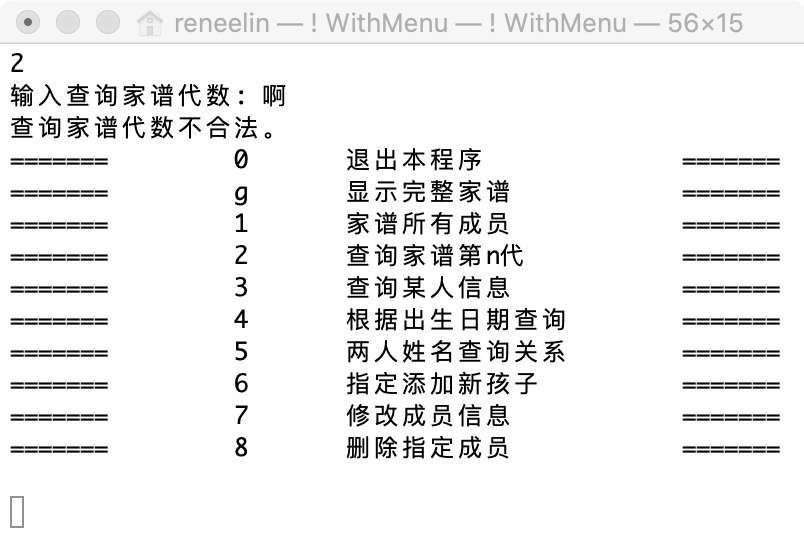


图4-2-3(6) 查询家谱第啊代

4. 按照姓名查询，输出成员信息（包括其本人、父亲、孩子的信息，以及他在家谱中的代数）。

选择功能“3 查询某人信息”，程序提示“输入查询家谱成员姓名: ”，依法输入姓名后，显示如图4-2-4。若家谱中并没有该成员，则显示“查无此人。”。

样例1 没有父亲只有孩子

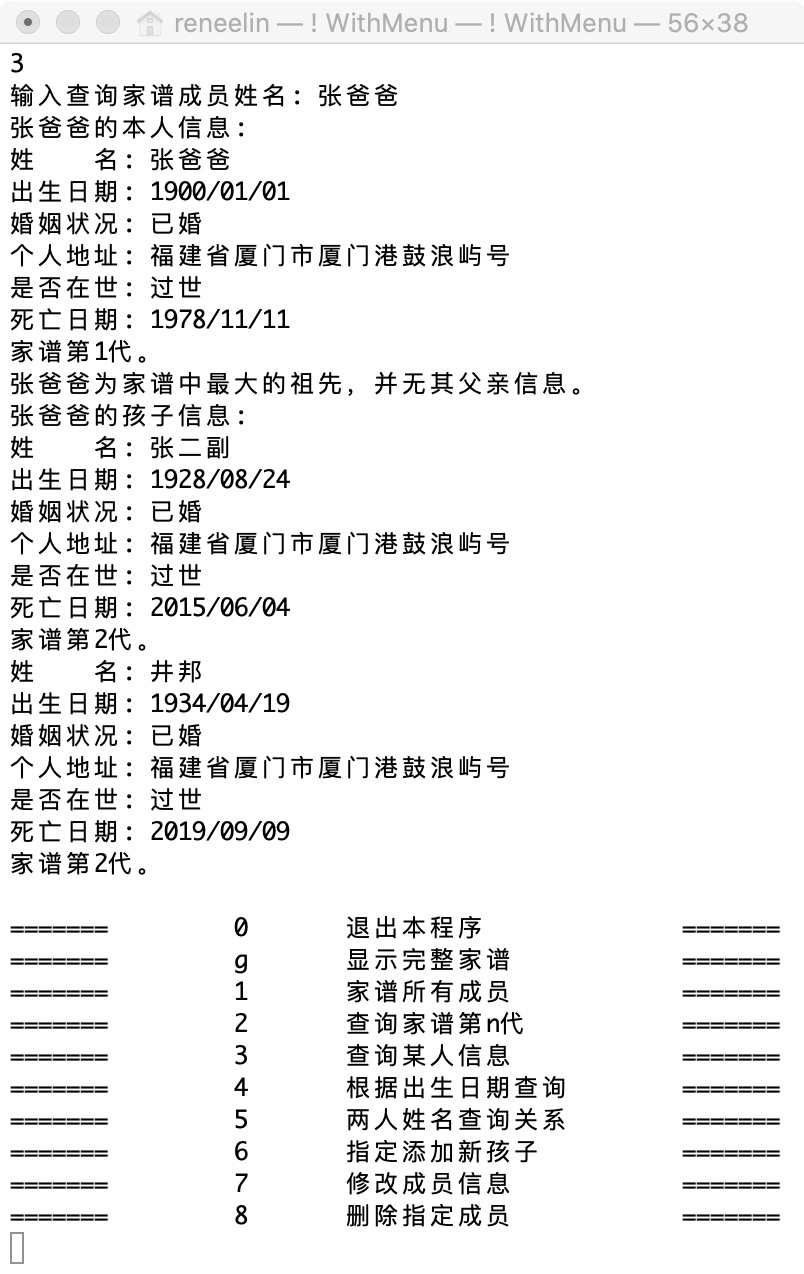


图4-2-4(1) 查询某人信息

样例2 没有孩子只有父亲

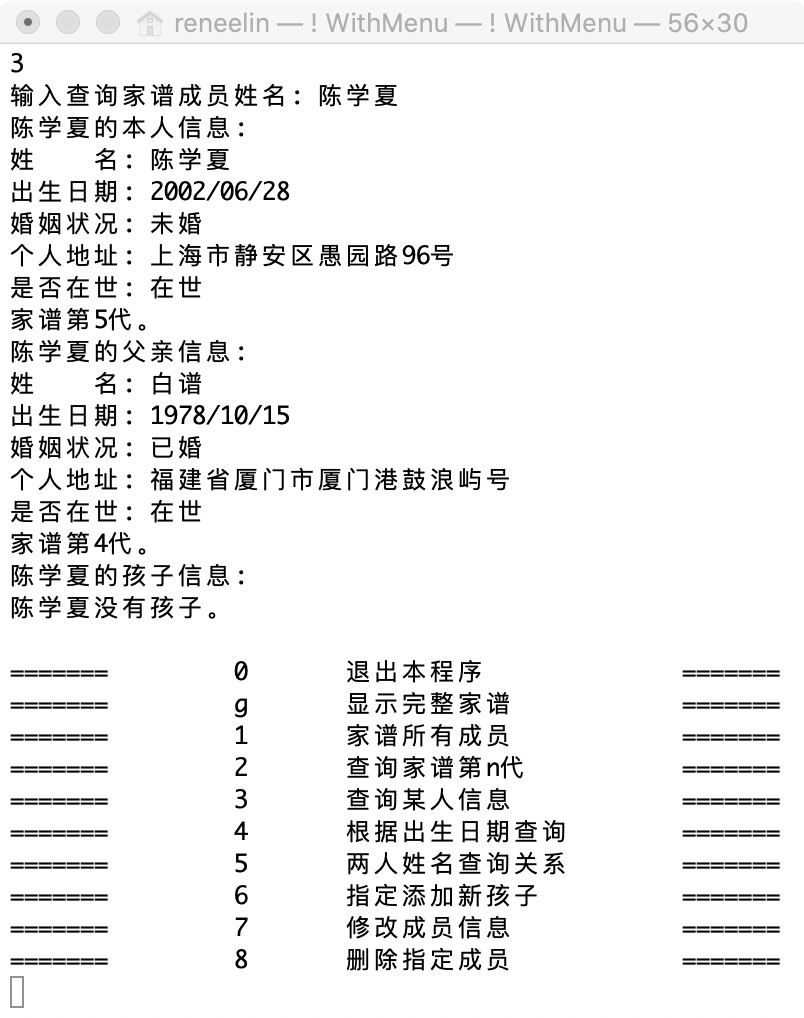


图4-2-4(2) 查询某人信息

样例3 又有孩子又有父亲

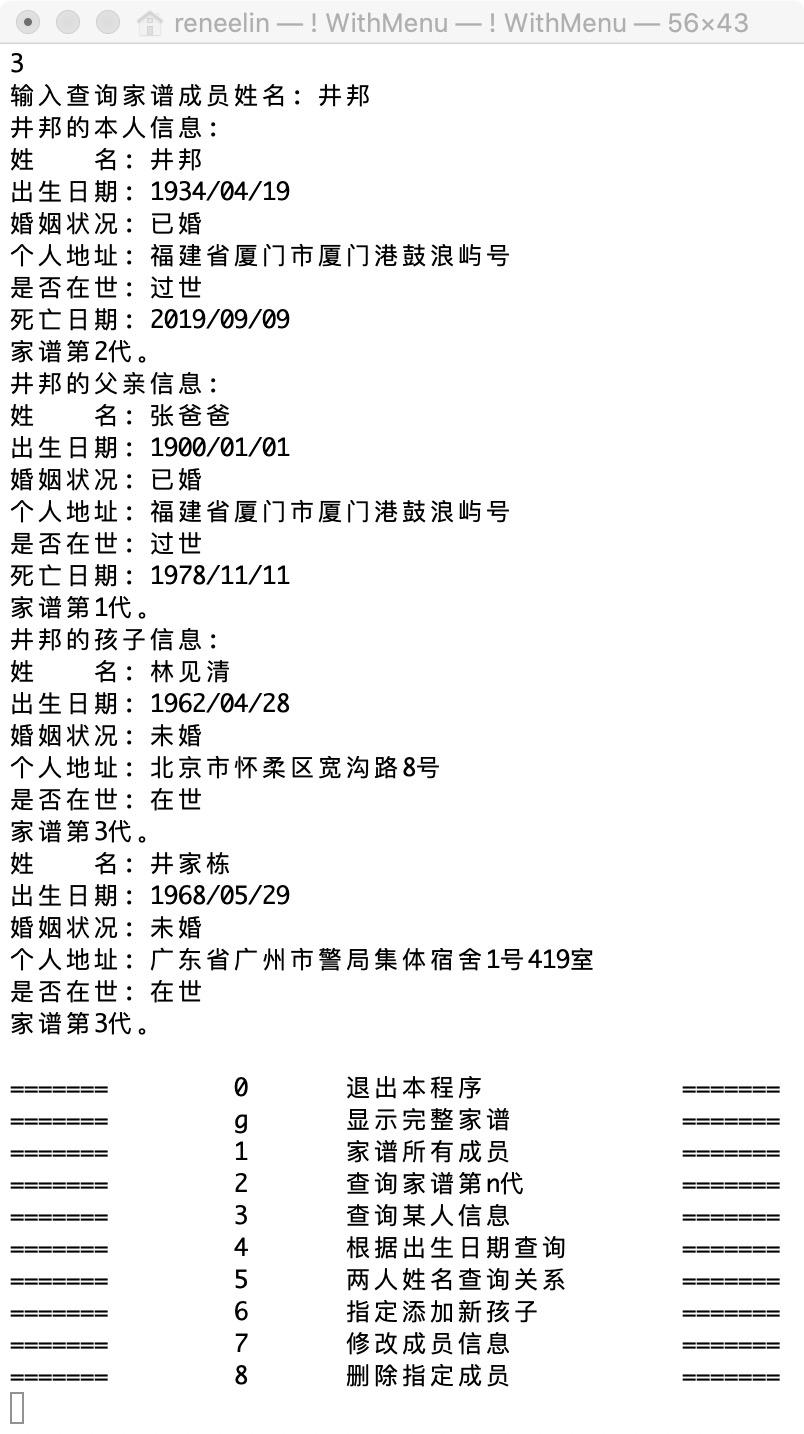


图4-2-4(3) 查询某人信息

样例4 不在家谱中

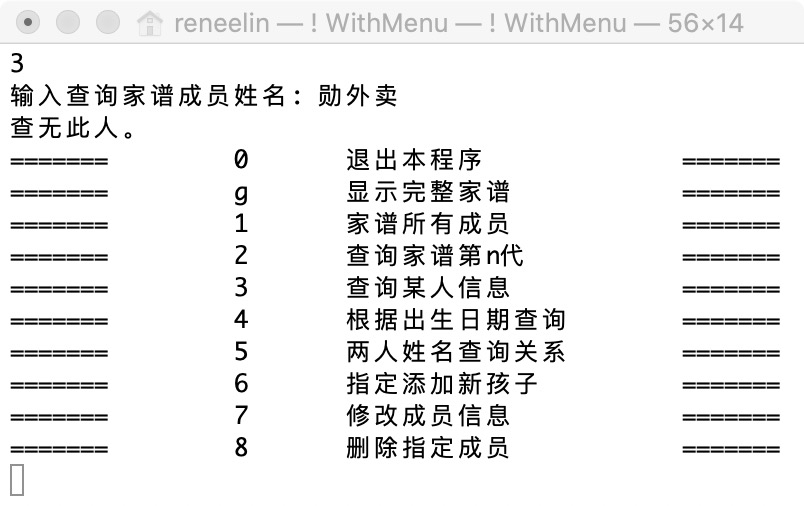


图4-2-4(4) 查询某人信息

5. 按照出生日期查询成员名单。

选择功能“4 根据出生日期查询”，程序提示“输入查询出生日期: (格式 -> yyyy/mm/dd) ”，依法输入出生日期后，显示如图4-2-5。若输入非法，或家谱中并没有成员出生日期为输入值，则显示“出生日期为*输入值*的成员: 无。”。

样例1 家谱中只有一人为该出生日期



图4-2-5(1) 根据出生日期查询

样例2 家谱中有多人为该出生日期



图4-2-5(2) 根据出生日期查询

样例3 家谱中无人为该出生日期



图4-2-5(3) 根据出生日期查询

样例4 输入出生日期为乱码



图4-2-5(4) 根据出生日期查询

6. 输入两人姓名，确定其关系。

选择功能“5 两人姓名查询关系”，程序提示“输入两人姓名查询关系: (格式以空格分隔 -> 某某某 某某某)”，依法输入两人名字后，分别显示两人信息及其关系，显示如图4-2-6。

在图3-1基础初始信息上，新增孩子信息：勋外卖 2019/12/12 0 浪迹天涯 1，父亲为白保险。

两人关系共存在8种输出表达：*输入值1*是*输入值2*的父亲，*输入值2*是*输入值1*的孩子；

*输入值1*是*输入值2*的叔伯，*输入值2*是*输入值1*的侄子；

*输入值1*是*输入值2*的祖父，*输入值2*是*输入值1*的孙子；

*输入值1*是*输入值2*的祖叔伯，*输入值2*是*输入值1*的孙侄；

*输入值1*是*输入值2*的曾祖父，*输入值2*是*输入值1*的曾孙子；

*输入值1*是*输入值2*的曾祖叔伯，*输入值2*是*输入值1*的曾孙侄；

*输入值1*是*输入值2*的直系祖先，*输入值2*是*输入值1*的直系祖先；

*输入值1*是*输入值2*的旁系祖先，*输入值2*是*输入值1*的旁系祖先。

样例1 井邦 林见清

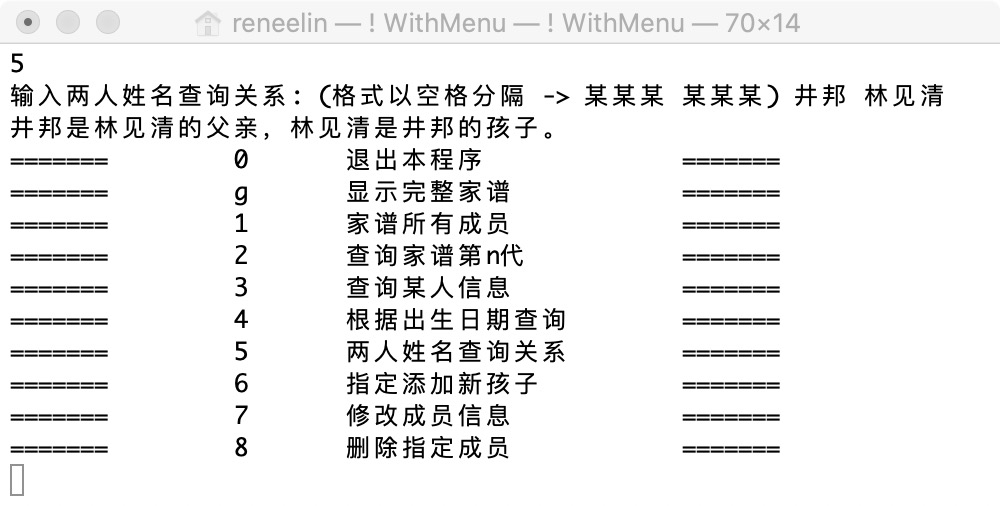


图4-2-6(1) 两人姓名查询关系

样例2 林见清 白谱

A screenshot of a cell phone

Description automatically generated

图4-2-6(2) 两人姓名查询关系

样例3 张爸爸 林见清

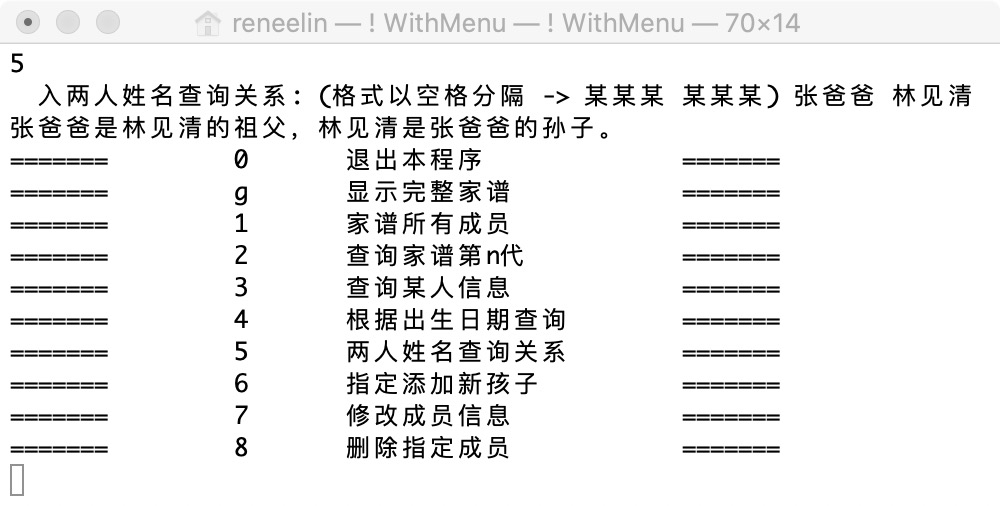


图4-2-6(3) 两人姓名查询关系

样例4 井邦 白领

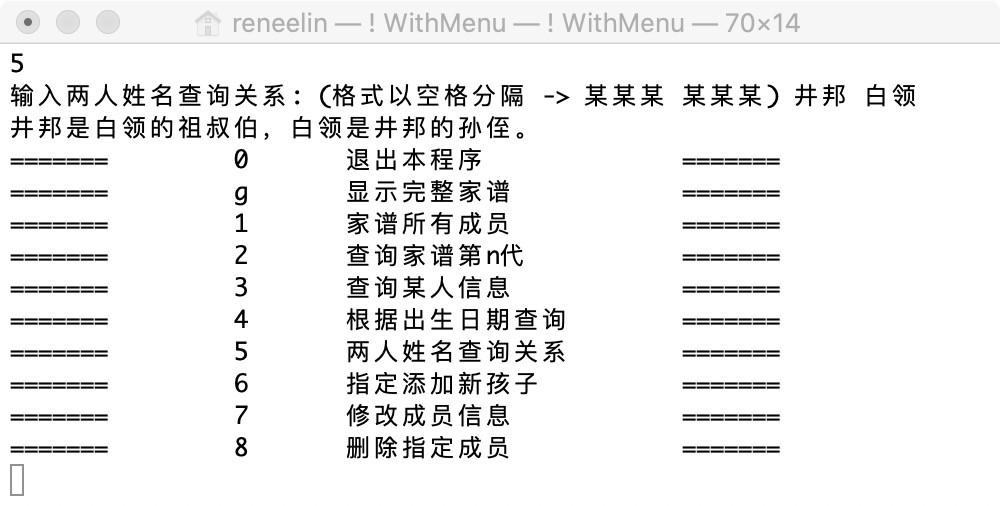


图4-2-6(4) 两人姓名查询关系

样例5 张二副 何船长

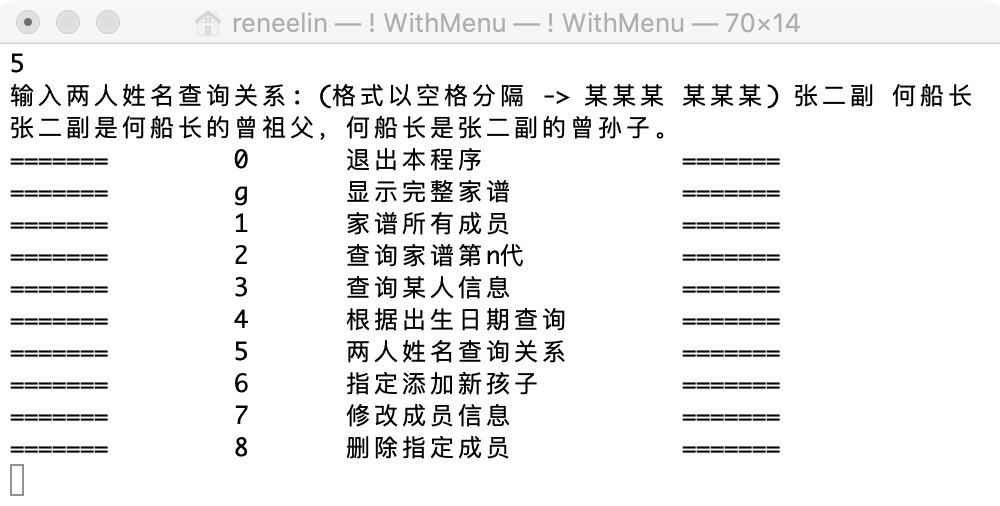


图4-2-6(5) 两人姓名查询关系

样例6 井邦 陈学夏

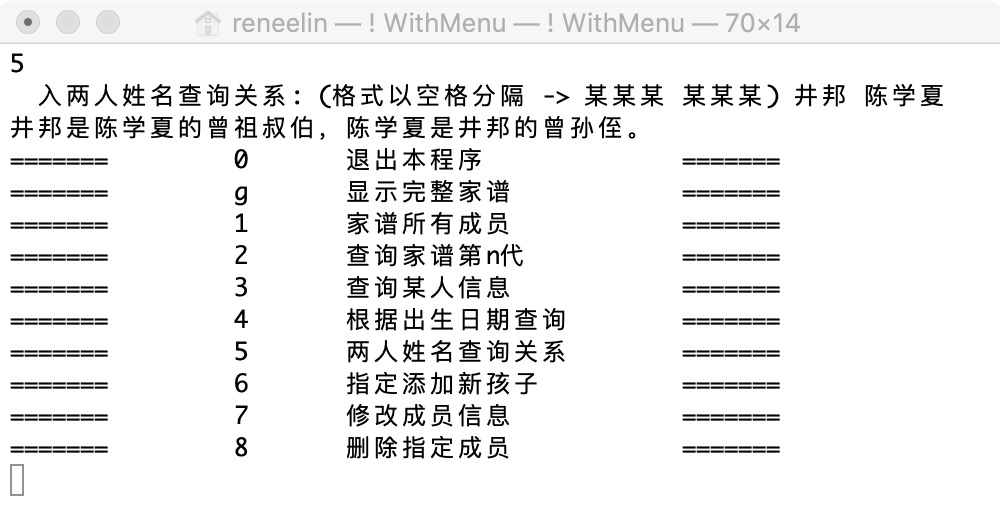


图4-2-6(6) 两人姓名查询关系

样例7 张爸爸 蒲水手

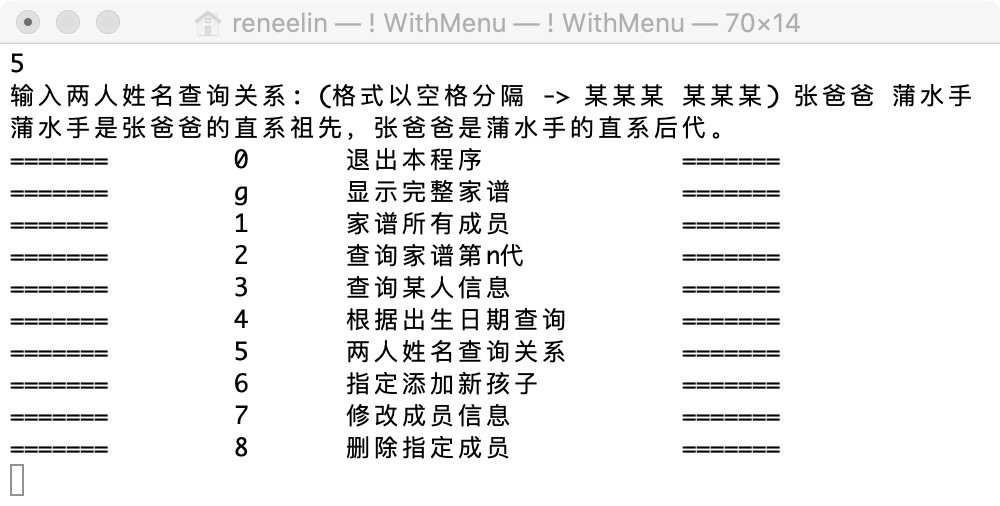


图4-2-6(7) 两人姓名查询关系

样例8 井邦 勋外卖

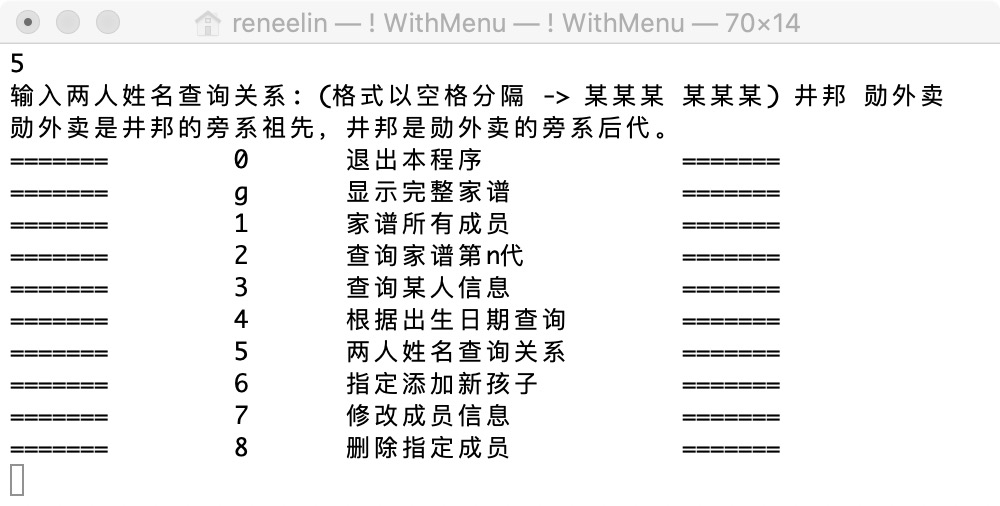


图4-2-6(8) 两人姓名查询关系

7. 给某人添加孩子。

选择功能“6 指定添加新孩子”，程序提示“输入新增孩子信息: ”，并根据提示依次输入所需信息，依法输入信息后，显示更新后的家谱以及添加的新孩子的本人、父亲、孩子信息，如图4-2-7。

新增孩子信息：勋外卖 2019/12/12 0 浪迹天涯 1。

样例1 最早的祖先

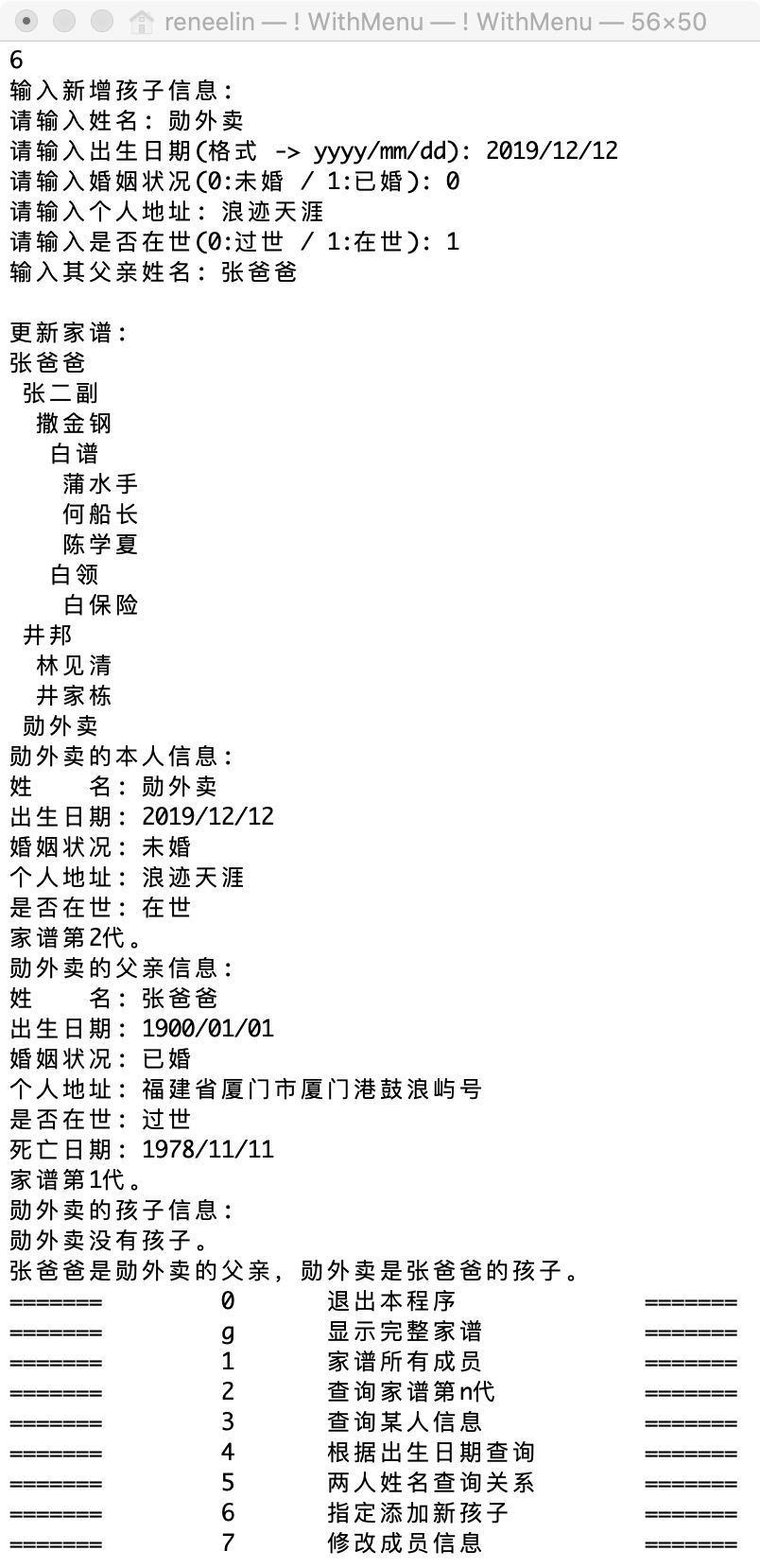


图4-2-7(1) 指定添加新孩子

样例2 有孩子的结点

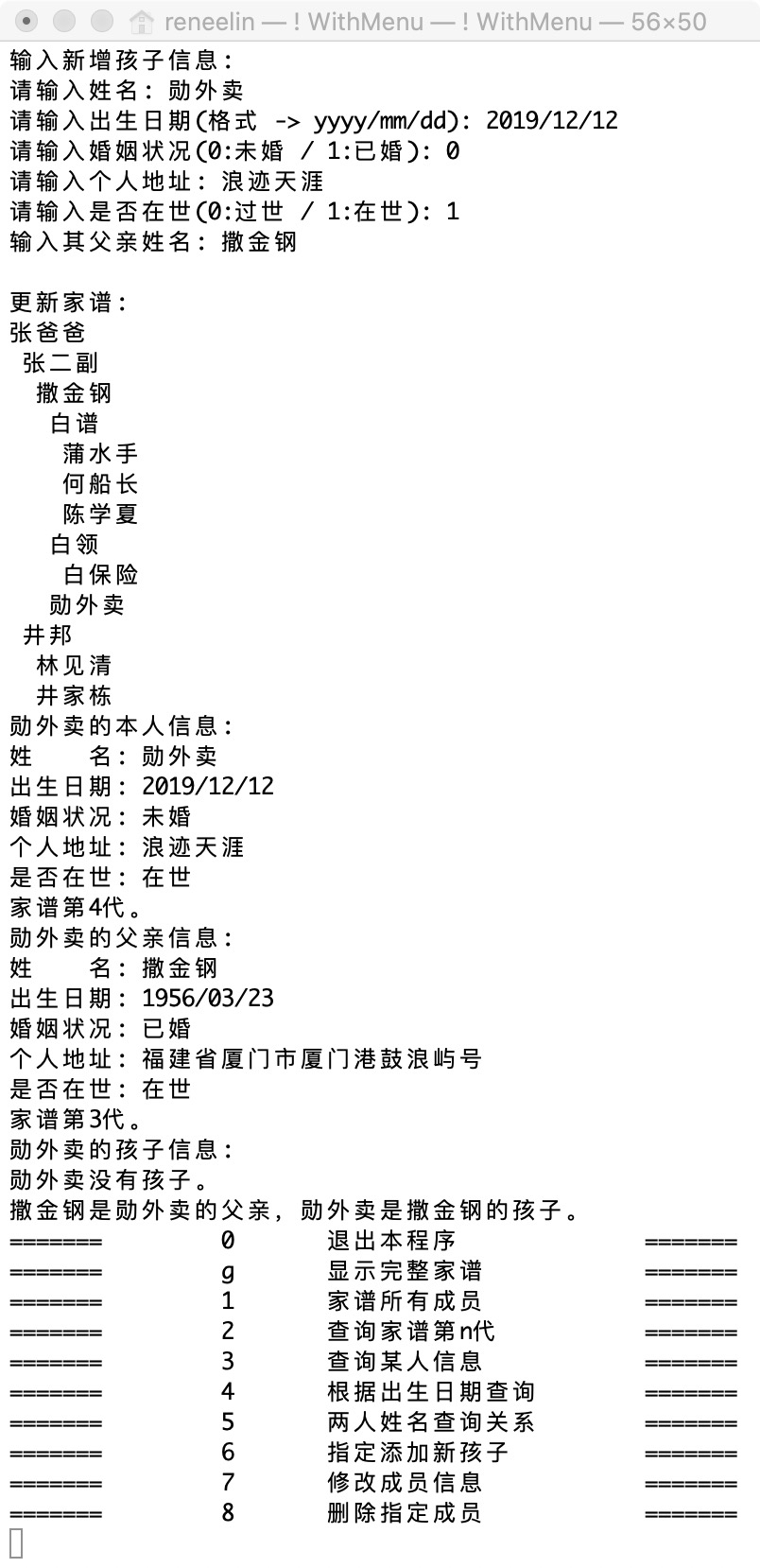


图4-2-7(2) 指定添加新孩子

样例3 没孩子的结点

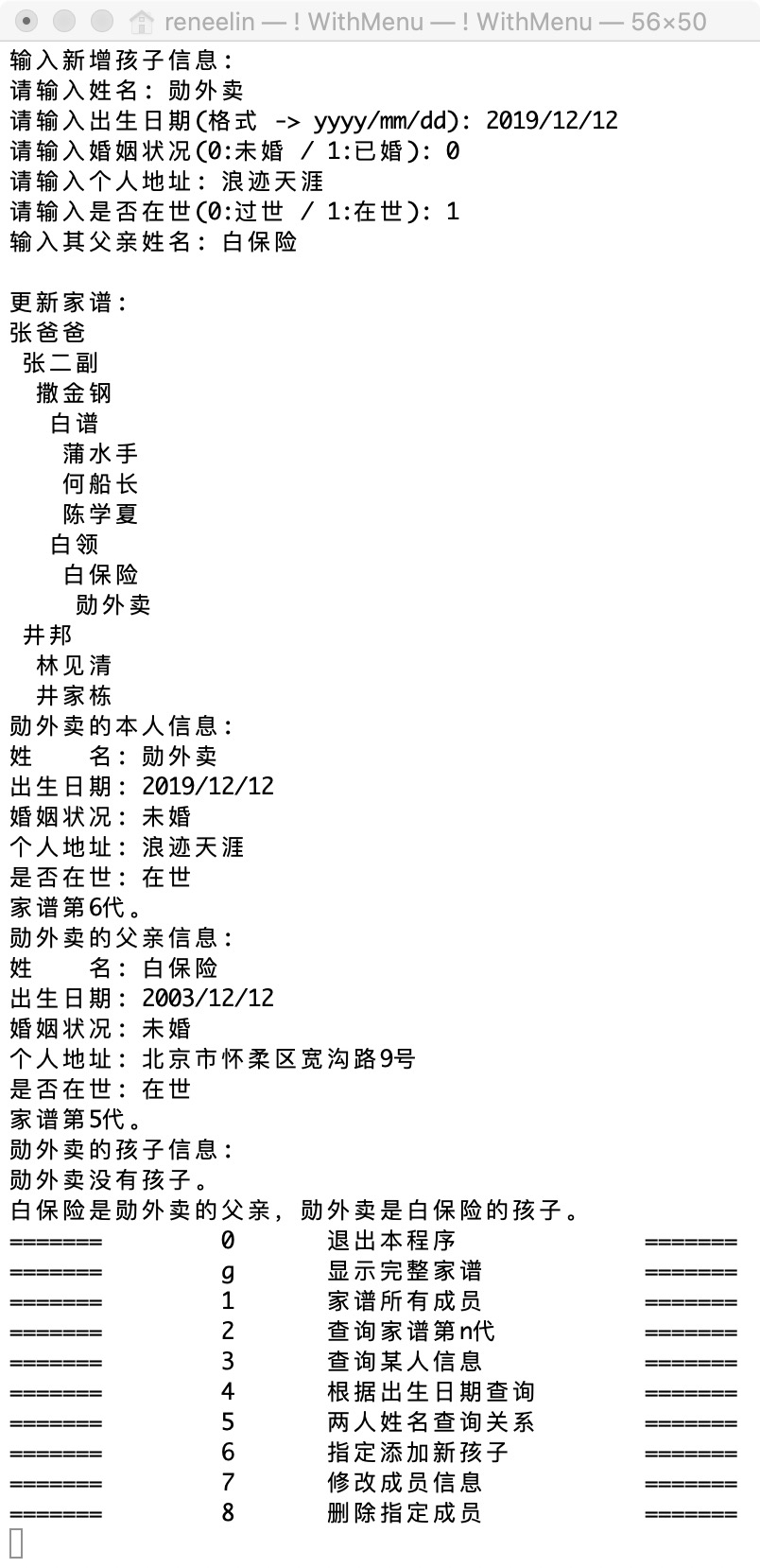


图4-2-7(3) 指定添加新孩子

8. 删除某人（若其还有后代，则一并删除）。

选择功能“8 删除置顶成员”，程序提示“输入删除家谱成员姓名: ”，若家谱中不存在此人，显示“查无此人”；若存在，显示“删除成功”，如图4-2-8。

样例1 没有父亲只有孩子

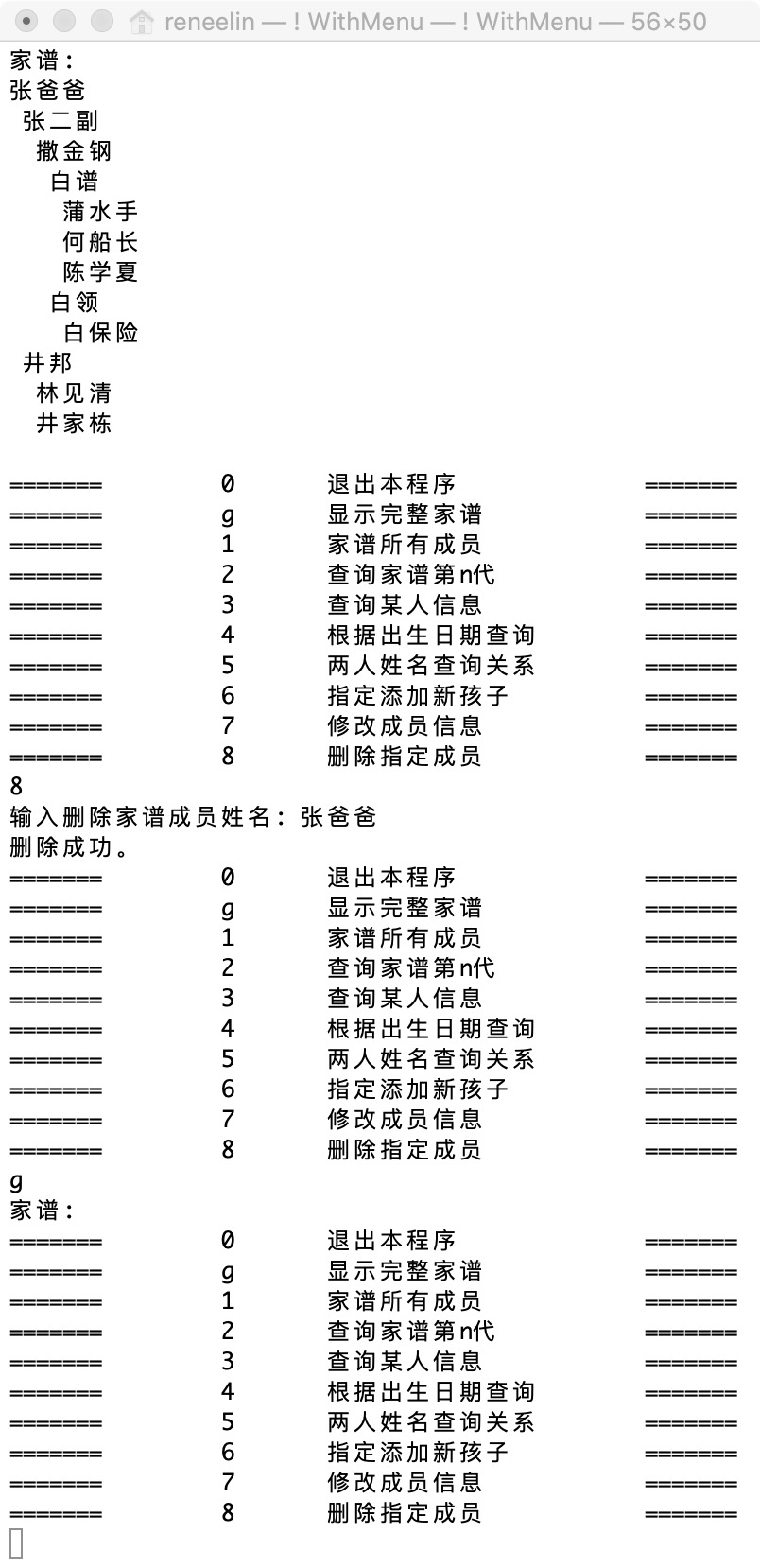
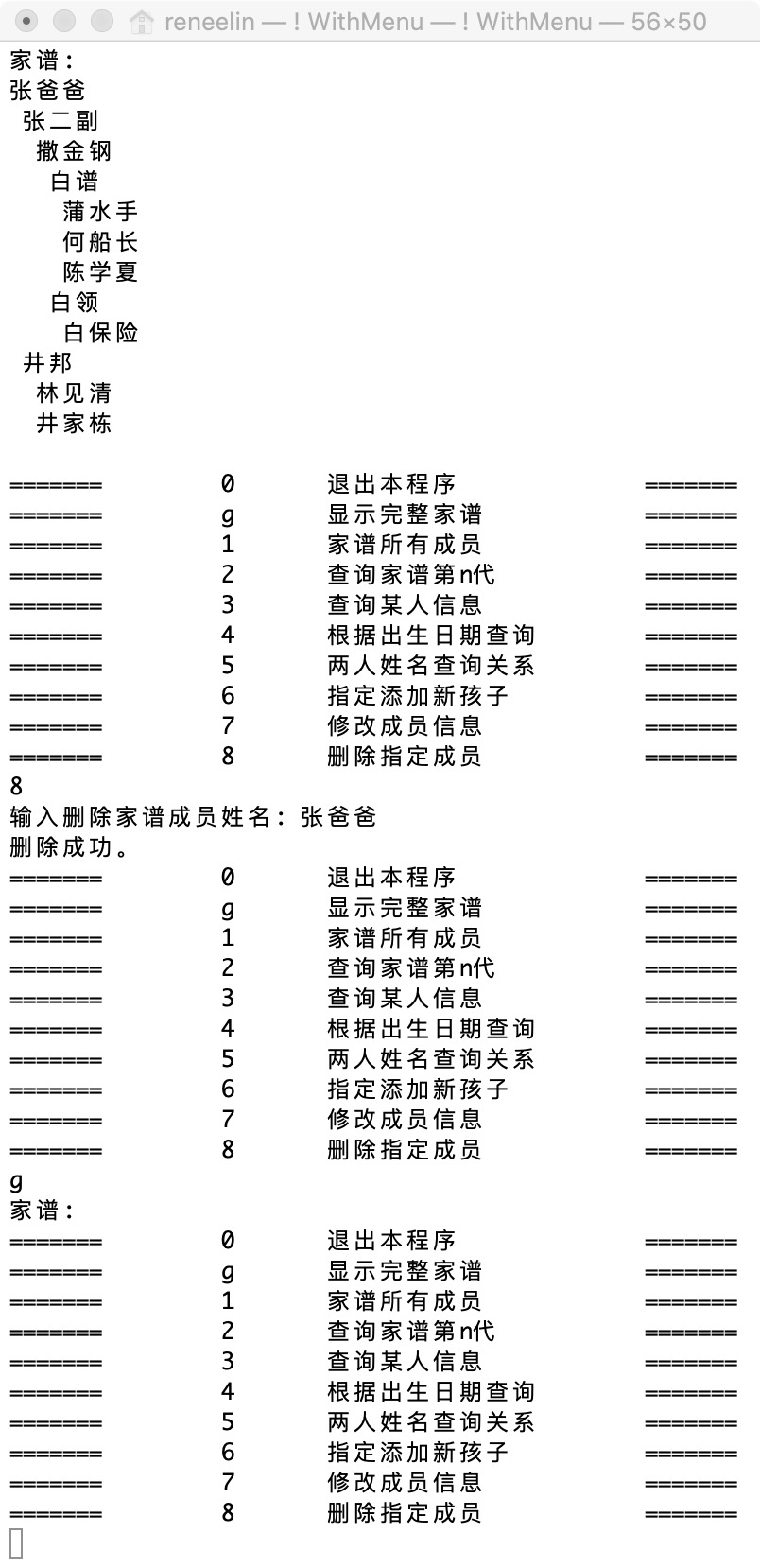


图4-2-8(1) 删除指定成员

样例2 没有孩子只有父亲

在图3-1基础初始信息上，新增孩子信息：勋外卖 2019/12/12 0 浪迹天涯 1，父亲为撒金钢。

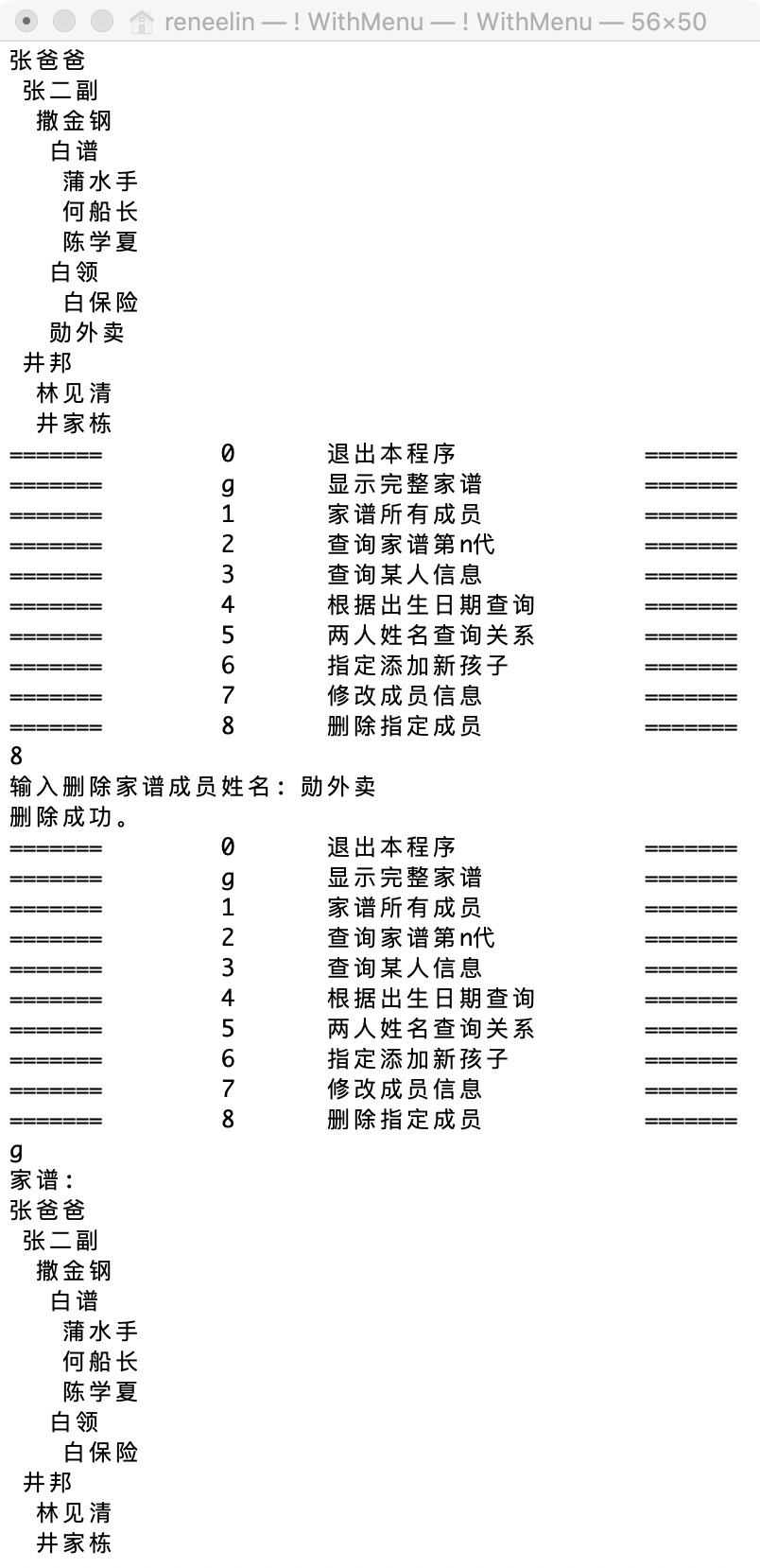
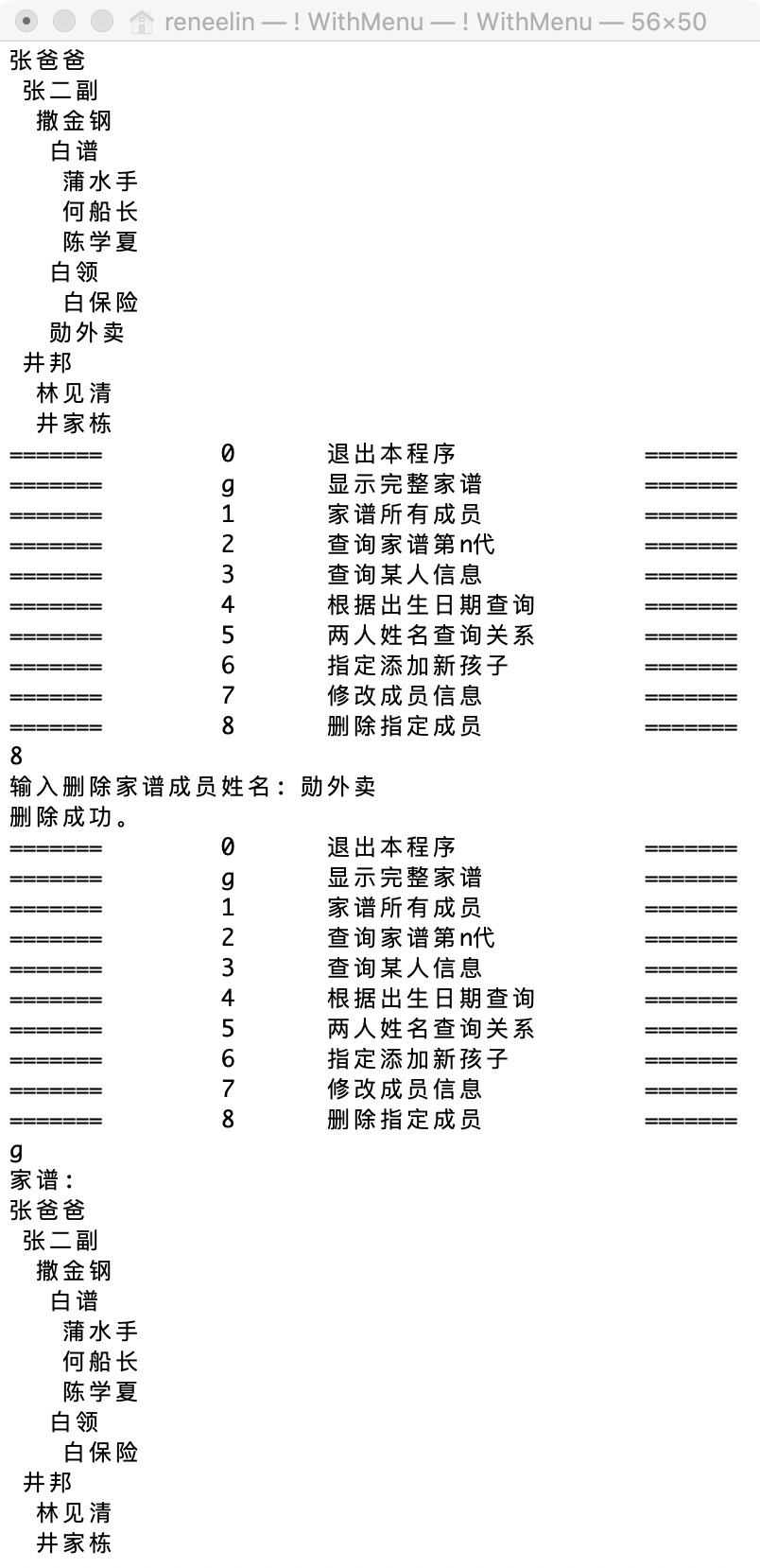


图4-2-8(2) 删除指定成员

样例3 又有孩子又有父亲

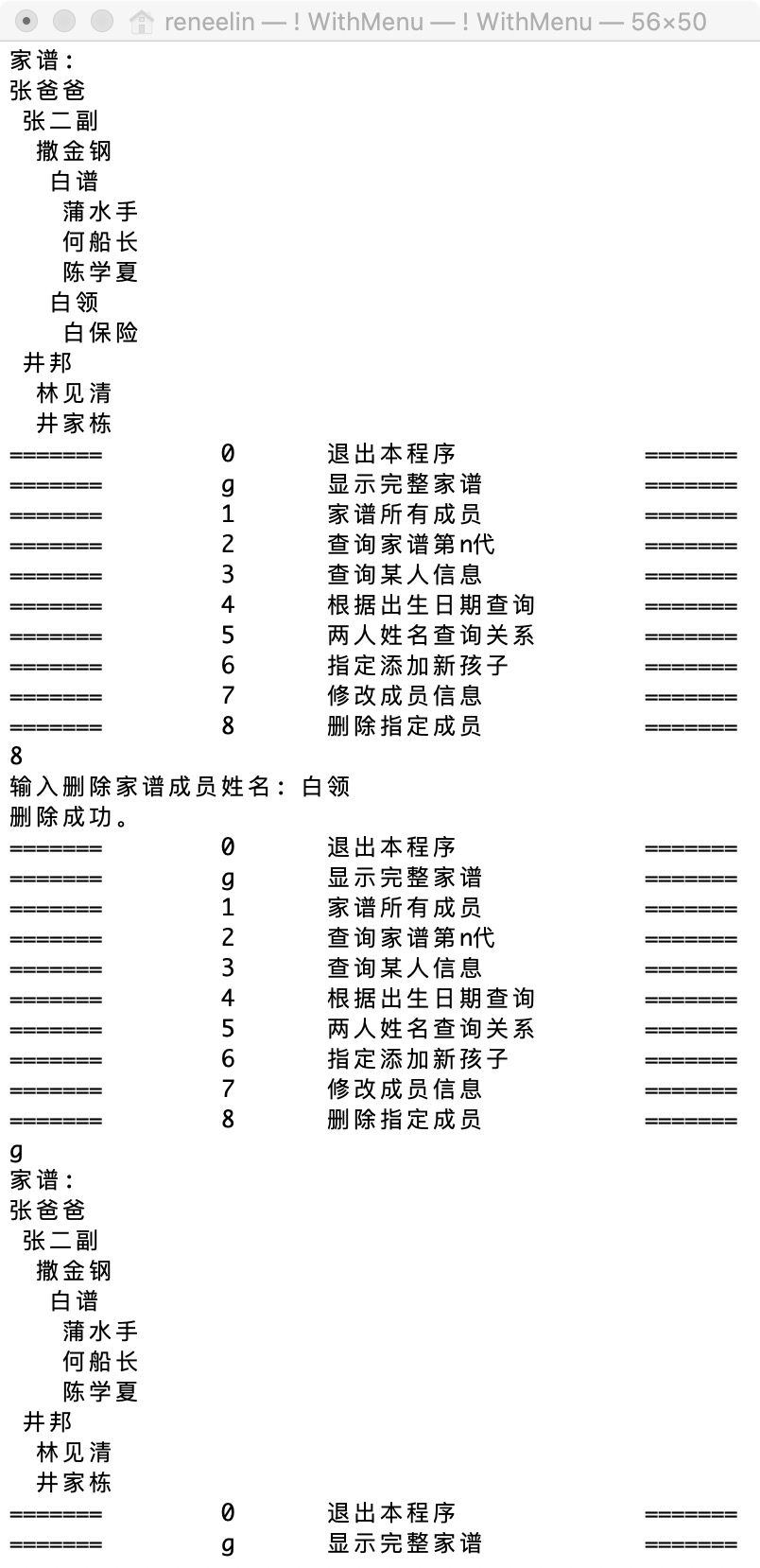
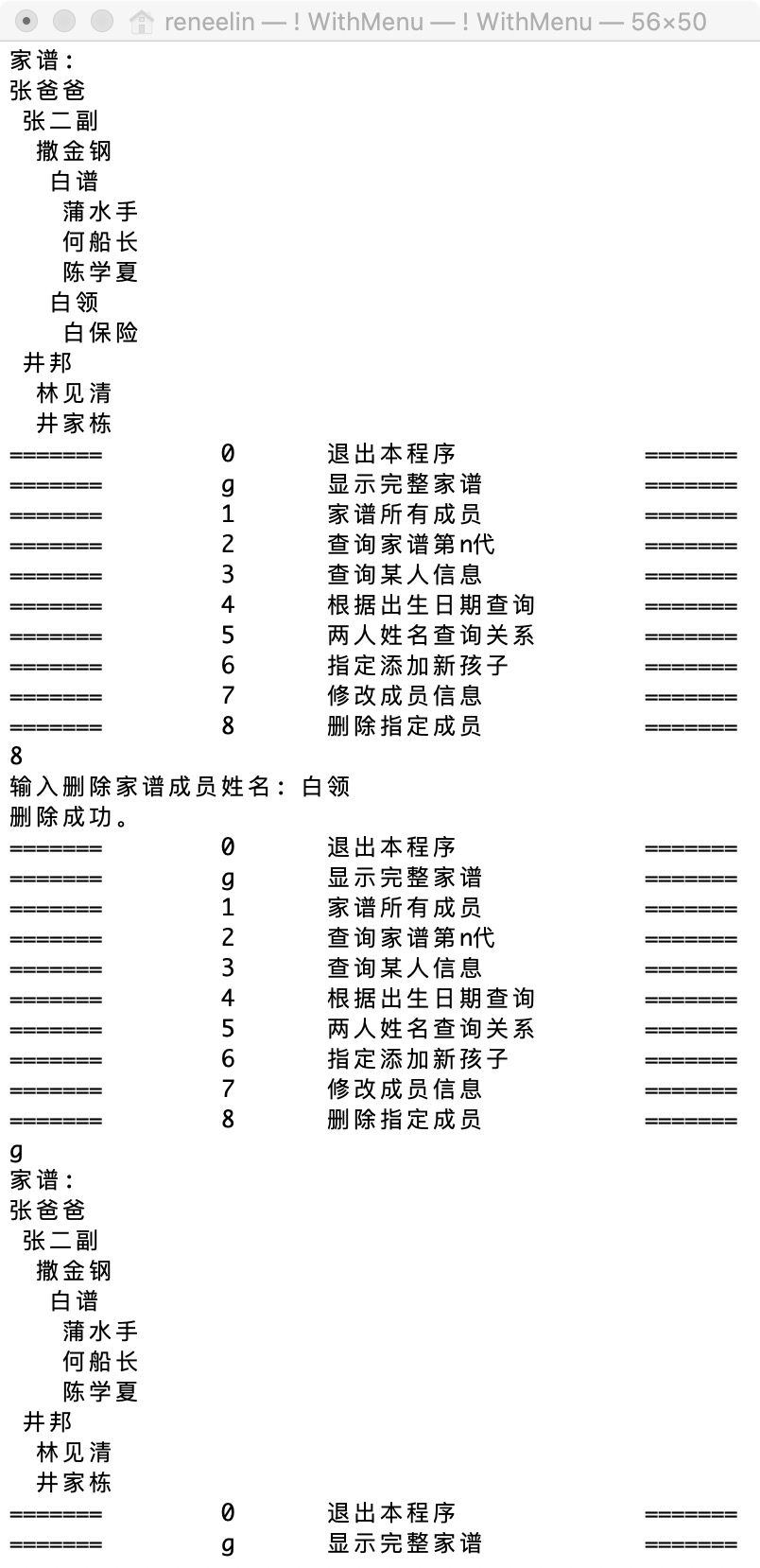


图4-2-8(3) 删除指定成员

样例4 不在家谱中

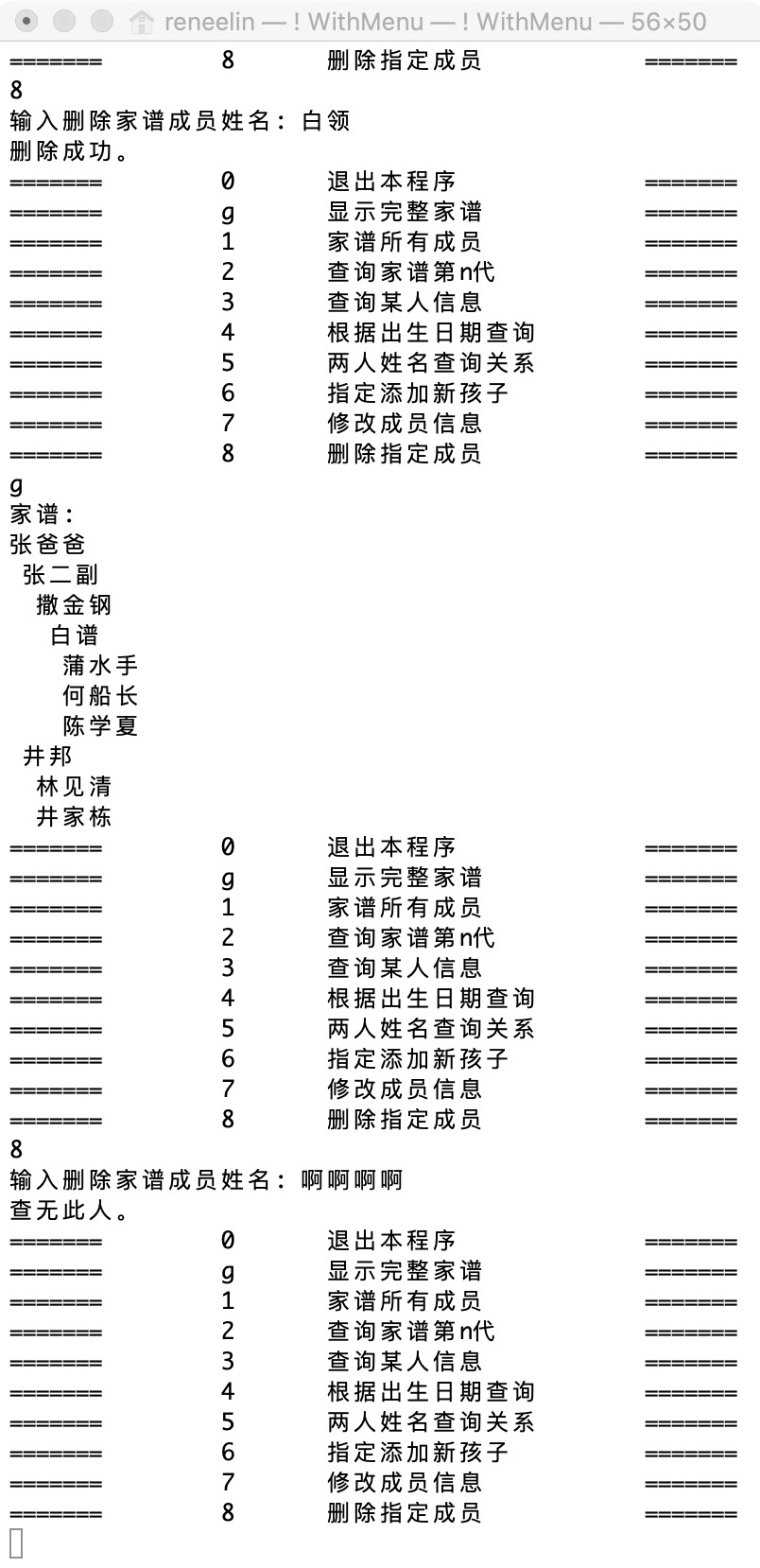


图4-2-8(4) 删除指定成员

9. 修改某人信息。

选择功能“7 修改成员信息”，程序提示“输入修改成员姓名: ”，依法输入姓名后并根据提示依次输入所需信息，依法输入信息后，显示更新后的家谱以及添加的新孩子的本人、父亲、孩子信息，如图4-2-7。

样例1 修改姓名

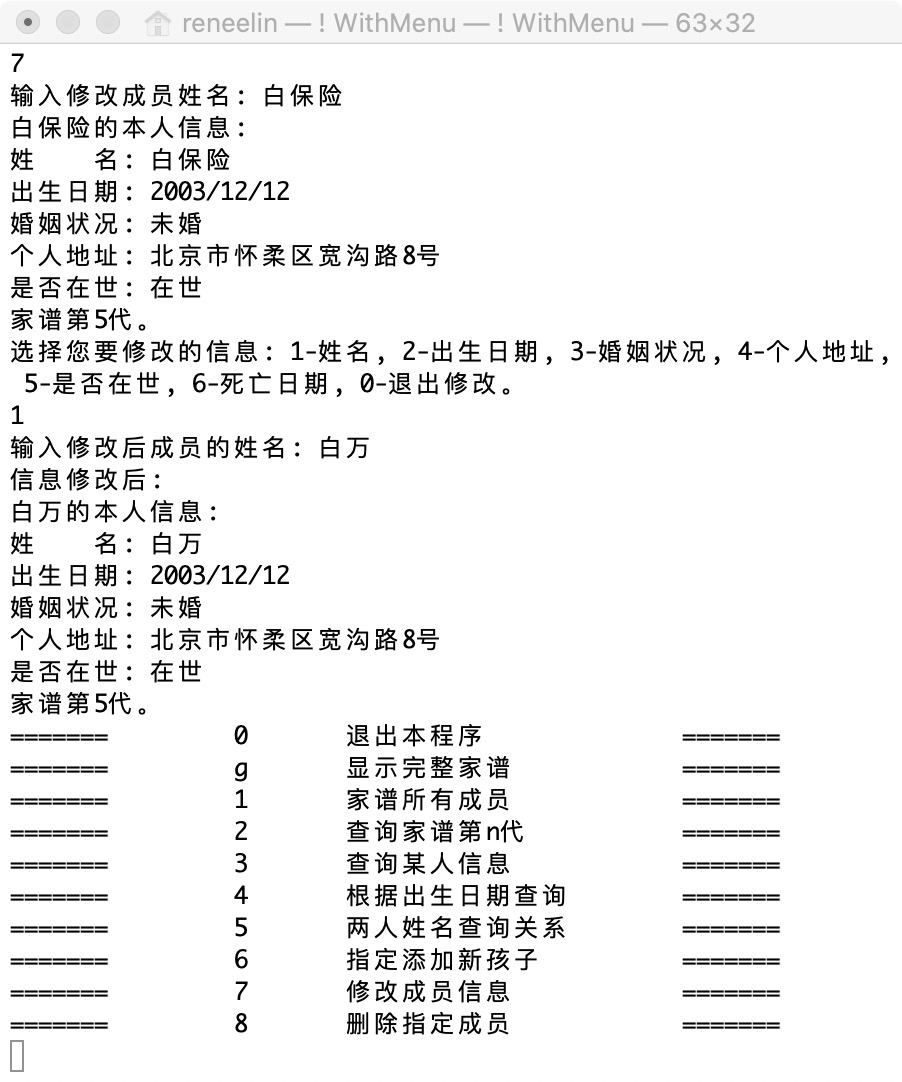


图4-2-9(1) 修改成员信息

样例2 修改出生日期

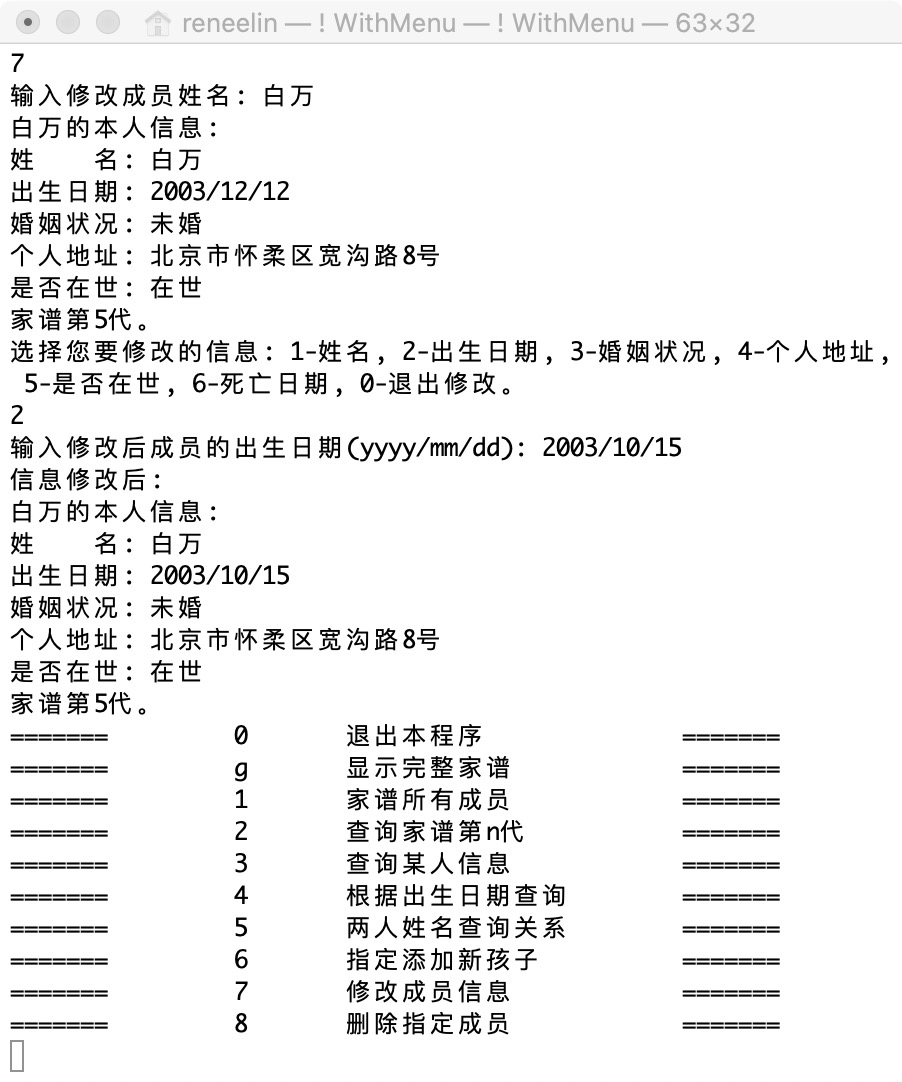


图4-2-9(2) 修改成员信息

样例3 修改婚姻状况

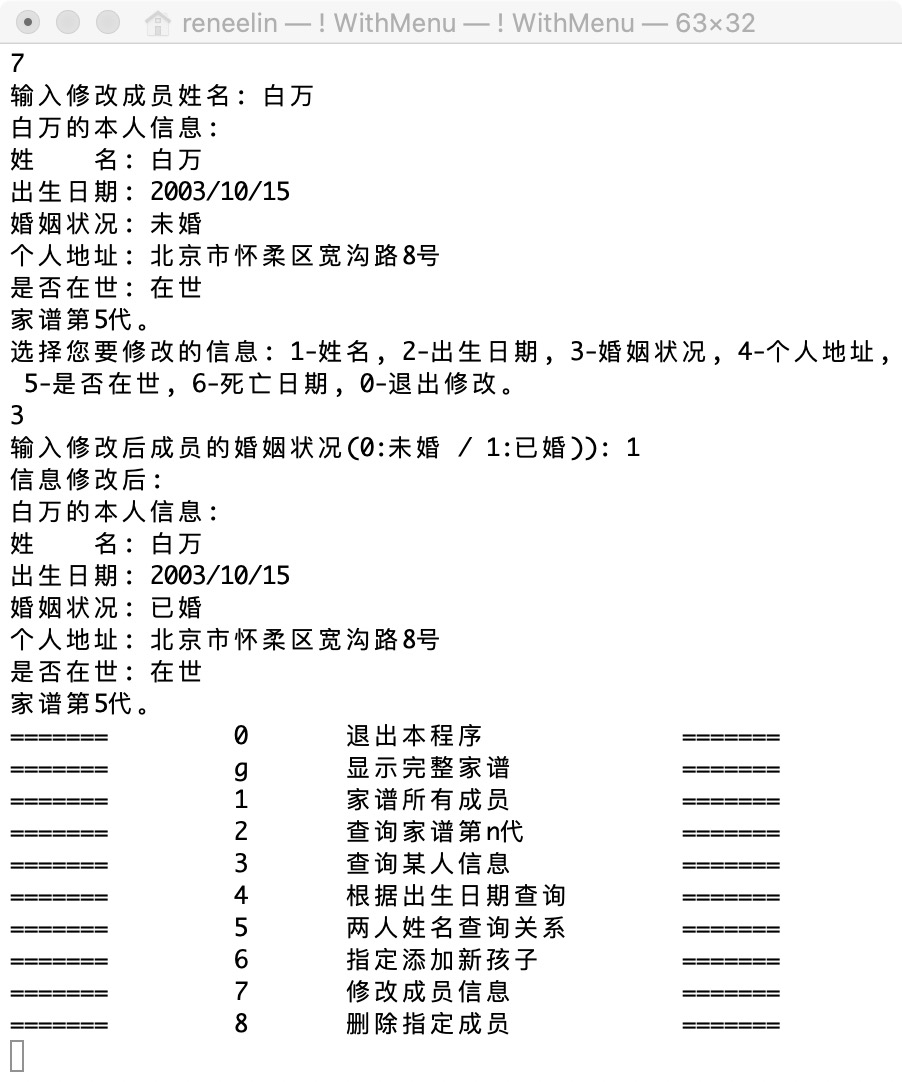


图4-2-9(3) 修改成员信息

样例4 修改个人地址

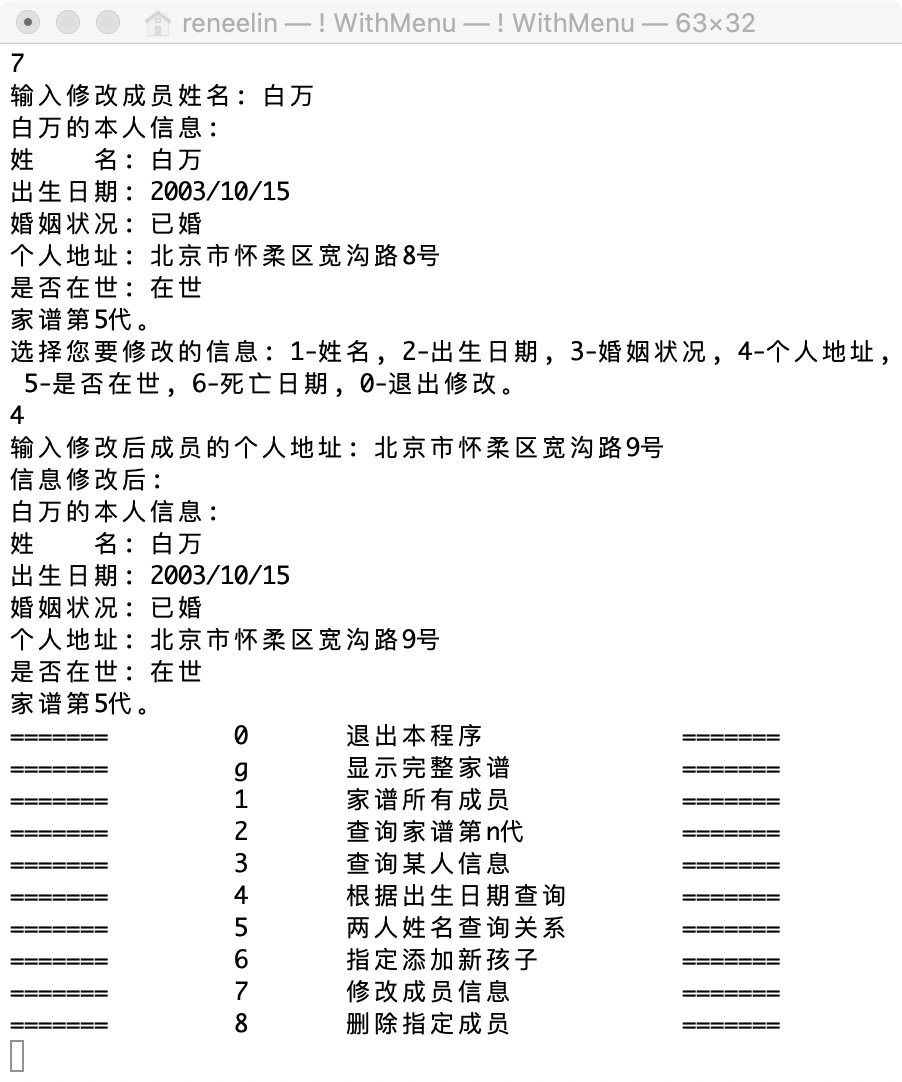


图4-2-9(4) 修改成员信息

样例5 修改在世人的在世情况



图4-2-9(5) 修改成员信息

样例6 修改过世人的在世情况

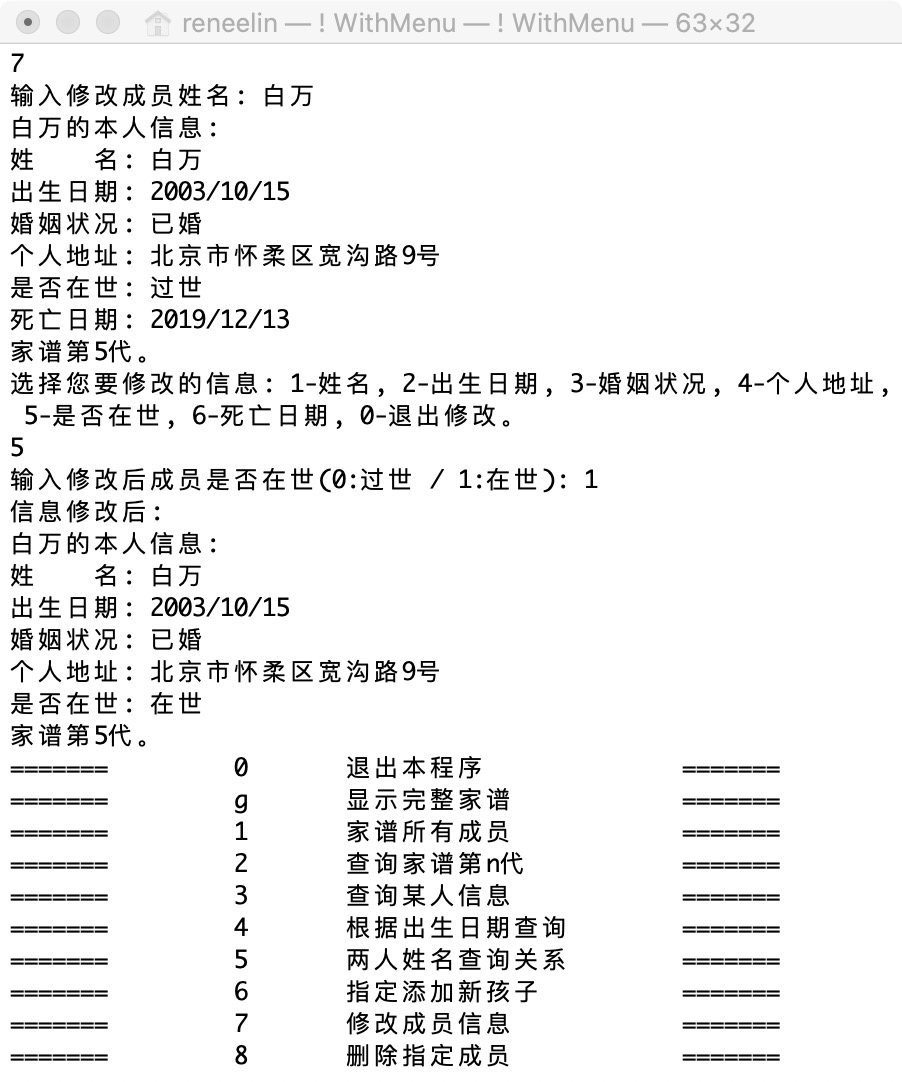


图4-2-9(6) 修改成员信息

样例7 修改在世人的死亡日期

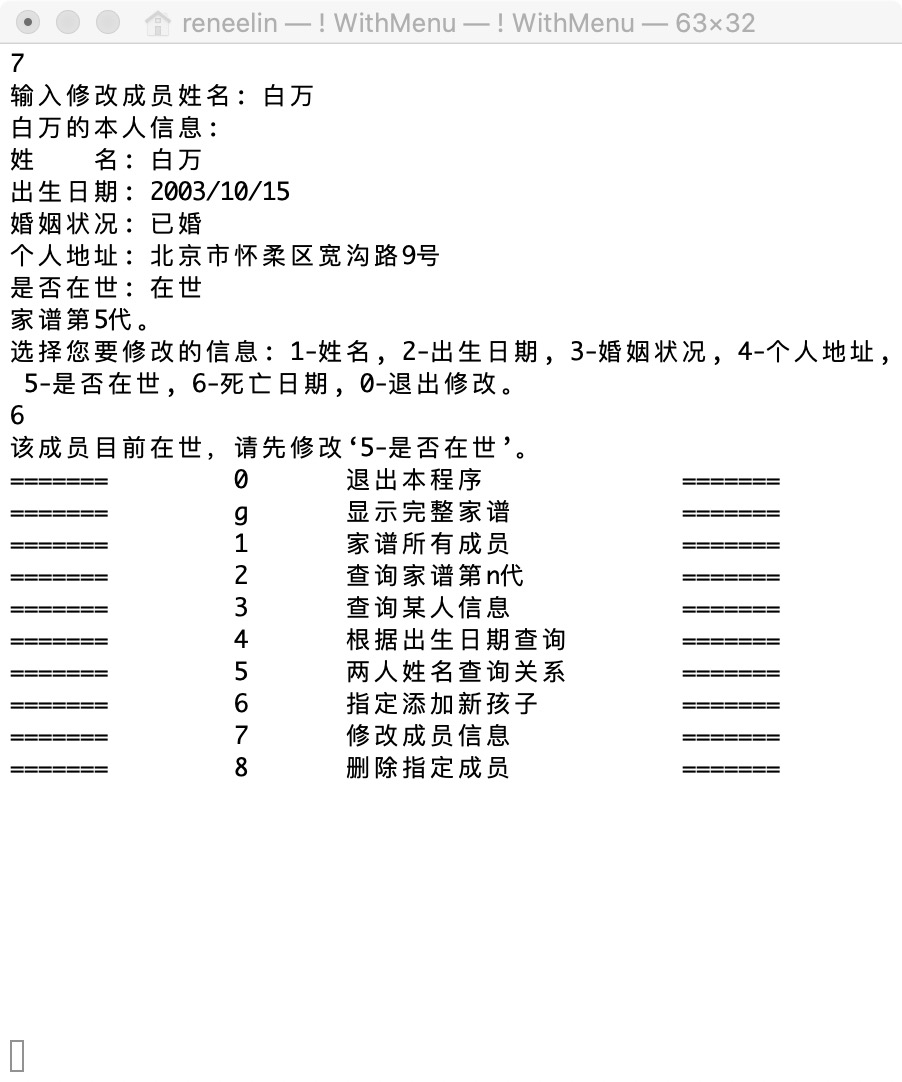


图4-2-9(7) 修改成员信息

样例8 修改过世人的死亡日期

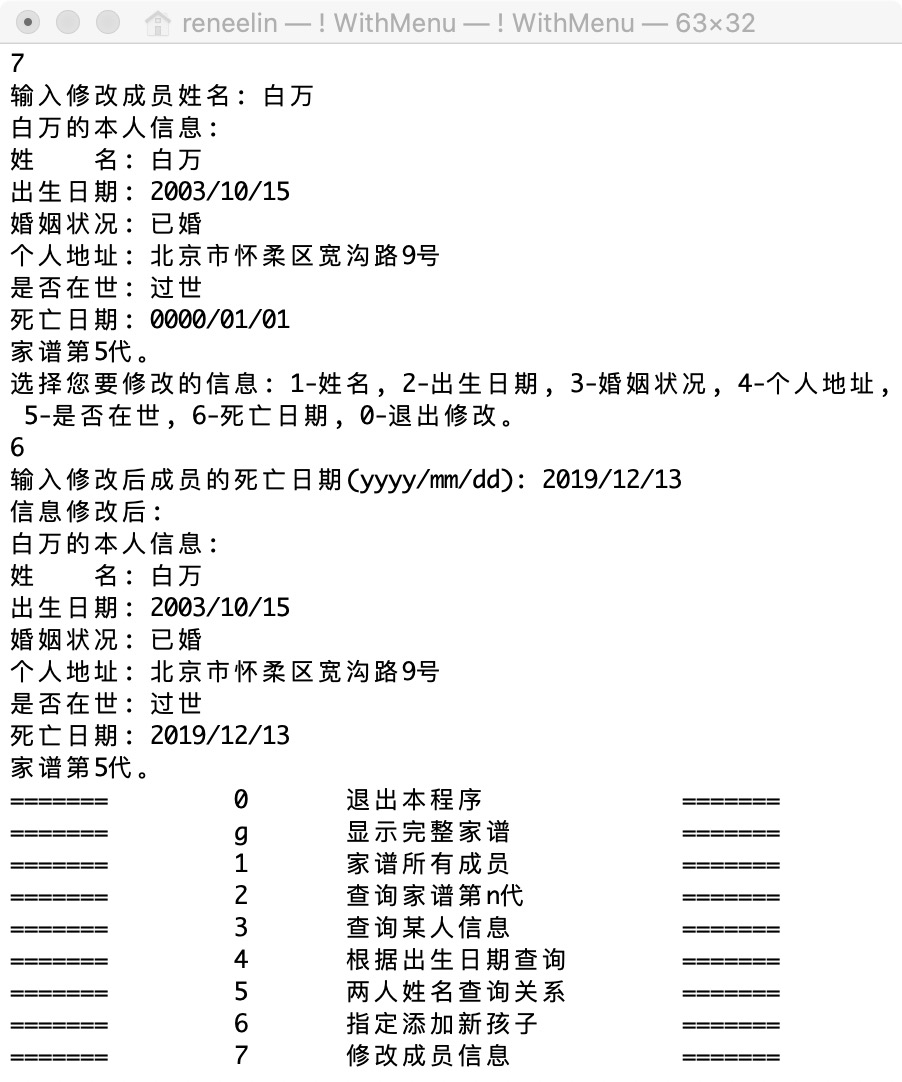


图4-2-9(8) 修改成员信息

五．课程设计中遇到的问题和解决方法

（一）设计性实验：树的孩子表示法的设计与实现

主要的问题时树的孩子表示法虽然容易找到孩子结点，但是比较难找到该结点的兄弟结点。解决办法为：遍历整个数组，当遇到非空且值不为当前结点的叶子结点时，遍历该结点对应的链表。如果在此叶子结点没有发现右兄弟，继续循环进行查找；否则跳出循环，返回右兄弟。在测试空树的样例时，发现空树的叶子结点个数为1，产生了错误。于是进行改进：增加了当为空树时，返回值为0的情况。

（二）综合性实验：家谱管理系统

调试时的主要问题是Delete函数。函数使用递归删除，从要删除的成员开始，释放他的后代指针。但是调用Delete函数以后，如果再遍历家谱，编译器就会提示0xDDDDDDDD内存错误。解决办法是：当要删除的成员为第一个孩子时，找到他的父母指针，让父母结点的第一个孩子指针指向要删除成员的右兄弟；如果要删除的成员不是第一个孩子，让前一个兄弟的右兄弟指针指向要删除成员的右兄弟。

这样就可以让要删除的成员以及他的后代从家谱上分离出来，并且避免了释放指针出现的异常情况。

六．实验总结

17120193 张赛

树的孩子表示法不同于常见的两种树的表示方法，只用单一的孩子标志位来确定树中各元素的位置。在对构造算法设计时，也曾经困惑与初始数组元素之间的关系如何确定，不过后来在组员的帮助下，该问题得以解决。

在后续函数的设计与实现中，发现树所对应的二叉树的高或者叶子结点数量的求法其实可以参考一定二叉树中对应函数的设计思路，只是两者的标志位有所不同，所以在设计和测试时较为容易就完成了。

在总体来说，本次的设计性实验中，我也更加深入地了解了二叉树的相关性质和递归算法的具体运用。

18120189 林艺珺

管理系统是计算机专业最常见的、也是最简单的一个应用。在设计系统的时候，首先对于数据存储的形式进行思考组织，选择合适的数据结构和其表示法，此题显然应该使用树进行存储，并使用孩子兄弟表示法以方便实现功能；其次，对于题目中所要求的功能实现进行分析，对于每个函数涉及到的对树的使用进行分析，并将相似的流程集成为一个函数；再次，根据各个独立的流程进行函数设计；最后，编写菜单并完成错误处理。

在验收之前，我没有完全意识到有规划地进行测试这项工作的重要性，我的习惯是在程序运行过后，按照自己即兴能想到的数据进行测试，从未想过按照数据的特点进行归类，以最大覆盖程度进行测试，最高效率检测有可能出现的问题和程序错误，通过测试后的改进使之尽量变得更健壮。

一个成熟系统，功能实现是其最基本的标准，而其易用性和高效性是评断好坏的重要标准，而用户界面设计的合理性和数据结构的选择，都影响到系统所运用的空间和时间。所以数据结构在计算机应用中有极大的作用、占有极大的地位，因此数据结构的学习是极其重要的。

18120206 徐奕婷

设计系统最重要的一步是：理清思路。先对需要哪些数据类、需要完成哪些功能有一定了解，再写好系统的大概框架，然后一步步完善它的功能。

考量过选题要求后，我们选择了树的孩子兄弟表示法来构造家谱树。

在逐渐完善系统的功能时，我深刻地认识到函数之间的相互联系。比如说，调用Search函数可以返回你所查找的成员指针，而在系统的修改成员信息和删除指定成员功能中，都是基于Search函数的定位功能来实现后续操作的。

测试数据时要注意多试几组，尽量考虑到不同的情况。这样才较可能发现代码中的问题。在测试Delete函数时就发生过这种情况：因为测试样例较小，删除指定成员的功能看似实现了，但一放到数据较多的样例里去，就会出现明显的bug。此外，测试非法输入也很重要。一个强健的程序应该有判断非法输入并处理它们的办法，不能因为非法输入就出现系统异常。

总之，本次实验让我重温了设计系统的方法，之前学习的递归函数也有了一定应用，并且更扎实地掌握了二叉树、树的相关知识。

18120212 叶菁

家谱数据结构的设计要考虑数据项之间的关系，以及后续处理数据的复杂程度。在孩子兄弟表示法的结构基础上，增加存储数据的结构体，使程序结构层次更清晰。在文件输入和输出过程中，txt文件的文字编码必须调整成和C语言的汉字编码一致，才能正常显示，否则会产生乱码。在输出的文件中，除了输出储存的数据项，还需输出孩子父母在文件中的编号，这需要分析存储空间暂时储存的编号与双亲的孩子编号之间的关系。对于菜单，能根据输入的信息选择功能，对于用户输入信息有容错机制，这需要读取程序需要的信息，把多余的输入清除掉，以防对后续操作产生影响。通过这次实验，特别是家谱管理系统，深入学习了树的数据结构和基本处理，复习了C++的知识，增强了写码能力。

18120221 陈思文

树的孩子表示法用一维数组顺序存储树中的各结点的信息，并将各结点的孩子信息组成一个单链表。虽然孩子容易找，但是在寻找双亲或者兄弟时比较困难，需要进行遍历。测试用例是用来发现错误而不是证明程序能够运行。在进行测试时，需要考虑各个极端情况。