# 1 项目简介

通常意义下的家谱是记载着一个家族从祖先到后代各代子女血缘关系的树状结构。本项目是对家谱系统的简单模拟。

# 2 程序说明

## 2.1 数据结构设计

考虑到实际家谱从祖先到后代各子女逐级分支的结构与“树”这一数据结构非常类似，因此，本程序采用二叉树来对实际生活中的家谱进行简单的模拟，其中，二叉树的左分支来表示子女，右分支用来存储兄弟姐妹。这样一来，本项目的实质是基于树结构完成对家谱成员节点信息的建立，查找，插入，修改，删除等功能。

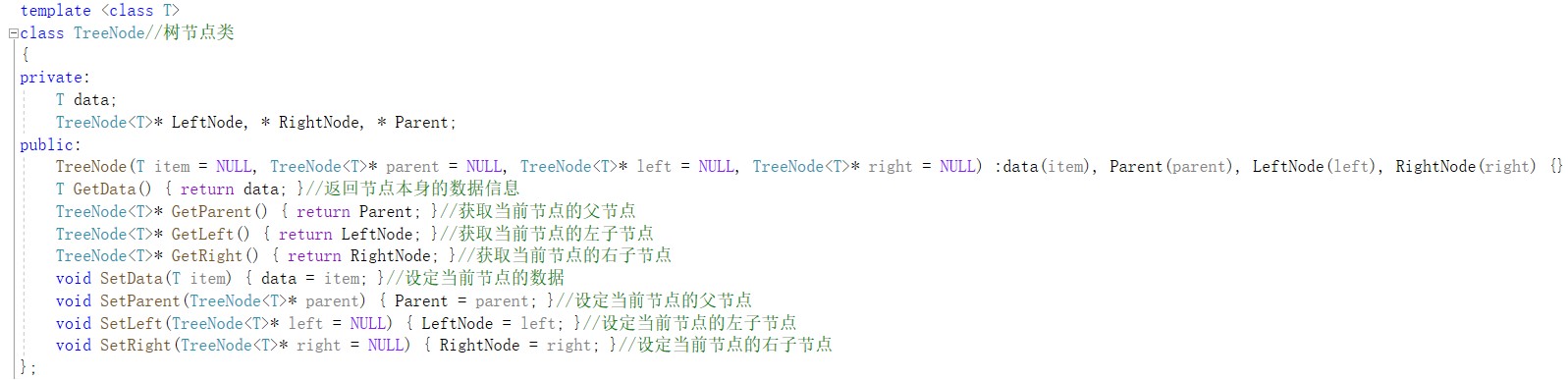
此外，为了防止用户在执行删除操作时将树的祖先节点删除导致形成空树，造成后续程序运行过程出现问题，本程序采用了带根节点的树结构。

此外，相比于题目，本程序额外提供了4个家谱查询功能，详见文档程序实现流程。

## 2.2 类结构设计

①树节点TreeNode类：

此类包含着4个信息：树节点本身的数据（即家族成员姓名）、左节点（即子女节点）、右节点（兄弟姐妹关系）、父节点（父子关系）。此外，还设计了一系列函数以为Tree类实现插入删除等功能提供基础。



## 此外，在家谱管理系统中诸多操作需要直接用到当前节点的父节点，因此，本程序在新建立节点时，直接通过parent指针指向了父节点，而不是通过自根节点开始的遍历寻找特定节点的父节点，以空间换取时间。

## ②二叉树BinaryTree类：



## 家谱管理系统中完善家庭、添加家庭成员、解散家庭等操作的基础功能都是通过调用BinaryTree类函数来实现的，其类函数也是本程序的**核心代码**，接下来会对上述函数进行详细说明：

1. 构造函数：实例化类时自动调用，动态开辟内存存储新节点，需要注意的是：将current指针与root指针一起指向实例化类时建立的第一个新节点，即以该节点作为二叉树的根节点。
2. 析构函数、destroy函数：析构函数对树所占用的节点空间的释放实际上是通过调用destroy函数实现的，而destroy函数还可以把传入参数指出的节点位置作为删除起始位置，删除树的某一部分子树。简单来说，destroy函数是利用递归的思想，在本函数中递归流程与后续遍历的顺序一样，通过遍历树或者子树，完成删除操作。
3. kill函数：功能是：用来删除特定的叶节点或者孤立节点。需要注意的一点是，在删除树的某一个节点时，若采用if先判断该节点是否为NULL，不为NULL则用delete删除，这一思路是有缺陷的。因为在实际调试过程中发现，通过if判断某节点是否为NULL来表示该节点空不空有时会发生这样的问题：本来上其通过指针连接的节点已经被删除，但指针有时不会因为删除其指向的节点的内容而自动指向NULL,因此，无论是kill函数还是destroy函数，本程序在使用delete释放相应节点内存空间之前，都会显式地将与其相关联的其他节点指针指向NULL。

4.RsetRoot函数：功能是重设祖先节点，即树根节点之下的第一个左子节点。此函数在用户执行删除操作将家谱祖先删除之后继续执行其他操作之前会被调用，来重新建立家谱的祖先。以防止程序在一棵空树上运行，出现BUG。

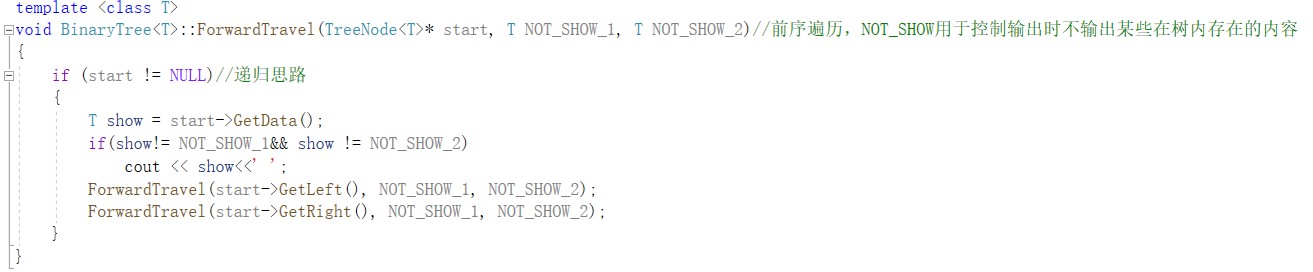
5. Insert 函数：功能是在当前节点的左或右方向插入值为item的新子节点，若传入的方向参数为宏定义KID（0），则插入左子节点，若传入宏定义BROTHER（1），则插入右子节点。也就是说，本程序的二叉树中，左子节点关系表示子女，右子节点关系表示兄弟姐妹。插入的具体流程是：首先判断是否是非法插入位置，如果插入位置合法的话，若是插入子女，先考察其伦理父母是否已经有孩子节点，若无，则直接设定伦理父母节点的左子节点即可，若已经有其他的子节点，则利用循环结构定位到最后一个子女节点，将新子节点插入原来子女的其他兄弟之后。若是插入兄弟节点，同样先判断新插入的节点是不是唯一的兄弟节点，若是，则直接设定右子节点即可，若在这之前已经有其他兄弟节点的存在，则利用循环结构定位到最后一个兄弟节点，将新节点插入到其他兄弟之后。当然，既然利用0与1来判断该插入左子节点还是右子节点，那么就应该额外再加一个判断：如果传入的方向参数是0与1以外的其他数字，则输出提示信息并结束Insert函数的调用。

6.FindTarget函数：功能是从start位置开始寻找数中第一个值为target的节点，并返回其位置。同样利用了递归思路，对每一个节点而言，都先考察其左子节点，若每找到，则考察其右子节点。直到找到该节点并返回其地址，或遍历完整棵树，依旧没有找到，返回NULL。

7.FindParent函数：功能：寻找某节点的伦理父母所在的位置

首先需要区分两个概念：树中的父节点是指某二叉结构中左右分叉的节点是左节点或者右节点的父节点；但是在本程序左边分支存子女，右边分支存兄弟姐妹的树结构中体现伦理上的父亲的节点不一定“紧挨着”子女节点，构成兄弟姐妹关系的那以分支的最左一个节点的父节点才是伦理上的父亲所在位置。因此，在FindParentt函数中，在循环结构下，调用节点的GetParent函数获取节点的父节点，只要node->GetParent()->GetLeft() != node，则一直向上寻找。

1. 前序遍历ForwardTravel、中序遍历MiddleTravel、后序遍历BackwardTravel函数：这三种遍历方式作为遍历树的基本方式，本程序中都是利用递归思路实现，而他们的递归思路大同小异，只是顺序不同。因此，这里以前序遍历为例进行说明：



按照先中间节点再左子节点最后右子节点的顺序应用于每一个节点，即可完成对整棵树的遍历。

1. Kid函数：功能：输出某节点的所有伦理子女

Brother函数：功能：输出某节点的所有伦理兄弟

上述两个函数的逻辑流程相似点较高，基于本程序树中，左分支存子女，右分支存兄弟姐妹的树状结构，先判断该节点是否有左子节点，若无，则表明该节点无伦理子女，若有，则输出某节点的伦理子女，只需要先指向其左子节点，接着如果该左子节点有右分支的话，输出该左子节点的右分支即可。而输出某节点的伦理兄弟姐妹，首先与Kid函数一样，先判断该节点是否存在其他的兄弟姐妹，若存在，则定位到该节点所属右分支的第一个节点，输出该右分支中除该节点以外的其他节点即可。当然，这里需要注意的一点是：在输出过程中，可以通过参数not\_show来控制输出，从而当循环遍历到该节点自身时，与not\_show比较，不将该节点本身输出即可。

**3.程序实现流程**

# 在原有题目的基础上，本程序额外提供了E-H这4个查询兄弟、子女、父母、祖先的功能，以增强程序的实用性。接下来对本程序的执行流程进行说明：

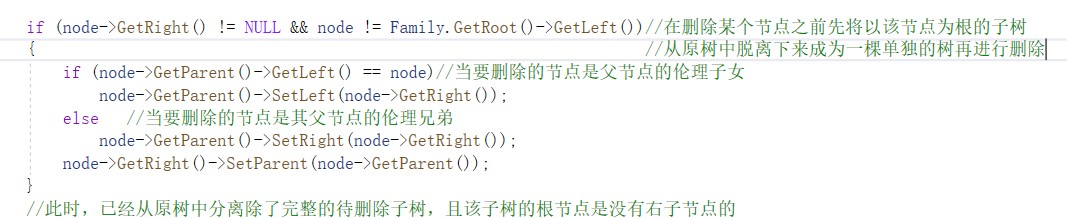
①首先实例化家谱类，即带根节点的树结构，紧接着建立家谱的第一个祖先。

②依靠分支结构，依据用户选择的不同的操作选项进入不同的程序分支，但是，在具体执行分支之前，先判断家谱祖先是否依旧存在，即树根节点下的第一个左子节点是否依旧存在，若存在，则直接进入分支，若不存在，则调用RsetRoot函数重新建立一个家谱祖先，再进入下面的分支程序：

1.完善家谱：输入要建立家庭的人的姓名name，调用FindTarget函数查找name是否在当前家谱存在，若不存在则结束此操作，若存在，则调用Inset函数向家谱树中插入新建的子女节点，之后调用Kid函数输出name的子女。

2.添加家庭成员：流程与“完善家谱”操作一致，只不过完善家谱是为某节点添加一个子女，而完善家谱操作是为某节点添加n(n>=1)个子女。

3.解散局部家庭：输入要解散家庭的人的姓名name，由于一旦执行删除操作，原来数据无法恢复，因此这里为用户提供了取消当前删除操作的选项，降低用户误触的概率。若用户确定执行删除操作，则首先调用FindTarget函数查找name是否在当前家谱存在，若不存在则结束此操作，若存在，则先调用Kid函数输出name的伦理子女，接着通过下面的程序将以name节点为根节点的子树从原家谱树中剥离下来：



将子树剥离下来以免让删除操作影响其他的节点，接着调用destroy函数与kill函数将剥离下的子树删除，即完成解散家庭操作。

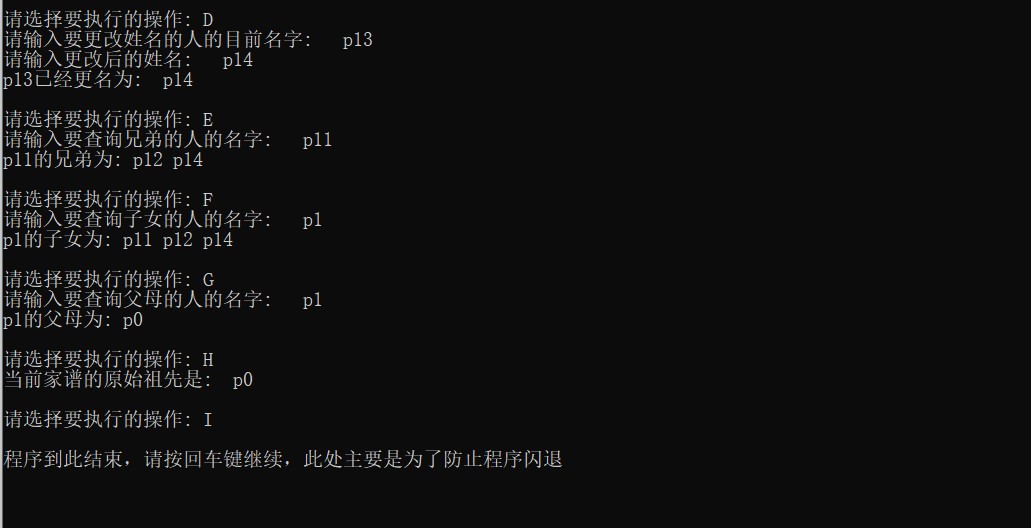
\*\*说明：本程序对“解散家庭”的理解是：从树中删除该节点和该节点所有的伦理后代节点

1. 更改家庭成员姓名：输入要更改姓名的人的原名name，调用FindTarget函数查找name是否在当前家谱存在，若不存在则结束此操作，若存在，则调用SetData函数重新设定并输出姓名即可。
2. E-H这4个查询操作均是先输入待操作对象姓名name，接着调用FindTarget函数查找name是否在当前家谱存在，若不存在则结束此操作，若存在，则调用FindParent函数、Kid函数、GetRoot函数、GetLeft函数、GetNode函数进行组合完成先关输出。(例如：查询兄弟姐妹：先调用FindParent，再调用Kid函数)

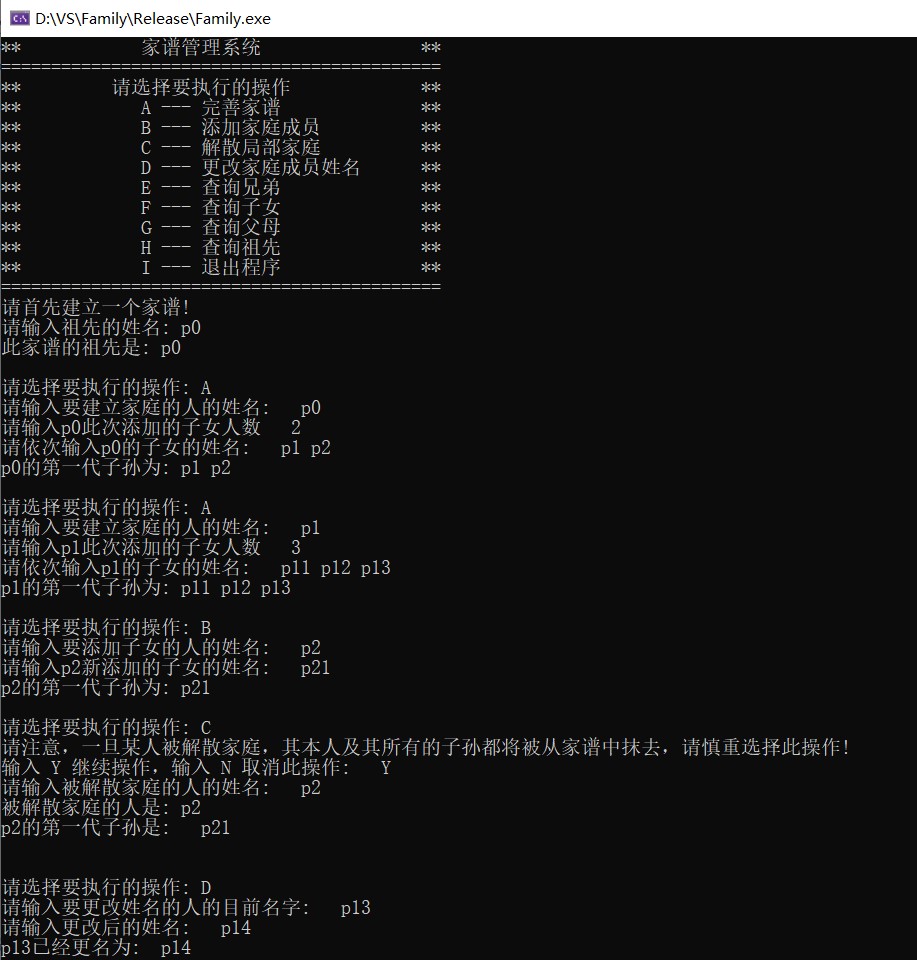
# 4.测试

## 4.1 功能测试

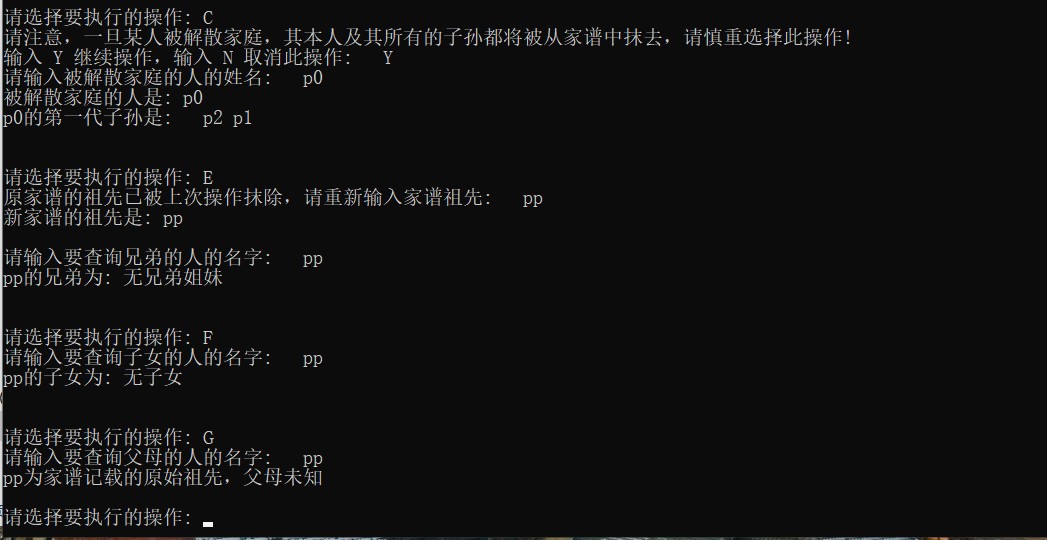
# ①D-I功能测试（D:更改家庭成员姓名，E:查询兄弟，F:查询子女，G:查询父母，H:查询祖先, I:退出程序）：（限于排版，先展示这里的功能）



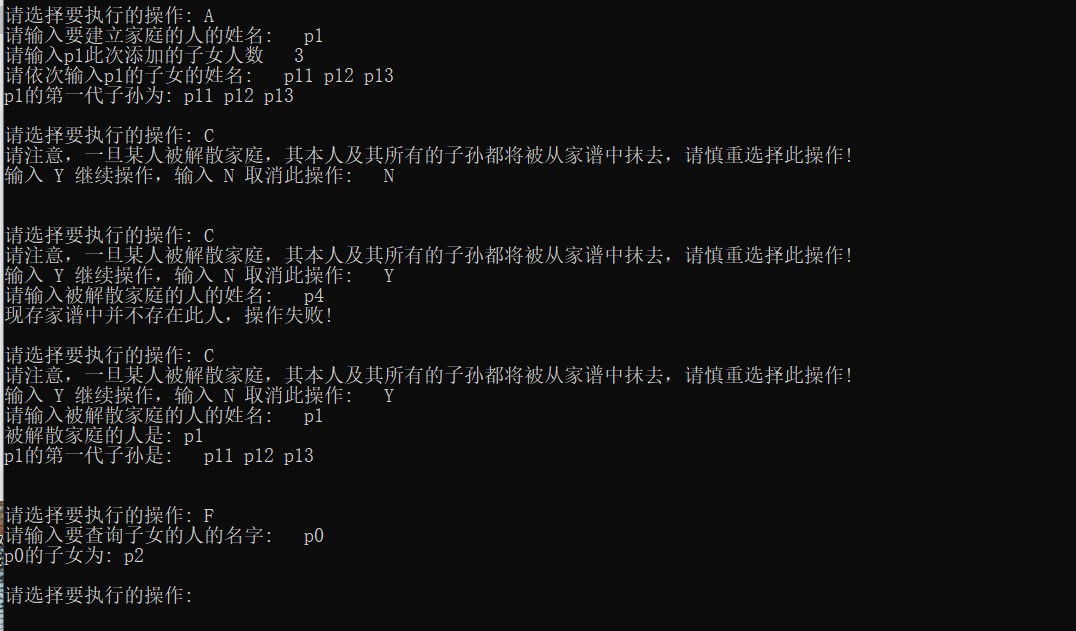
②A-C功能测试（先建立家谱， A: 完善家庭， B: 添加家庭成员 C:解散局部家庭）：



③边界条件测试：当删除家谱组先后，程序能够提示用户新建祖先并继续正常运行

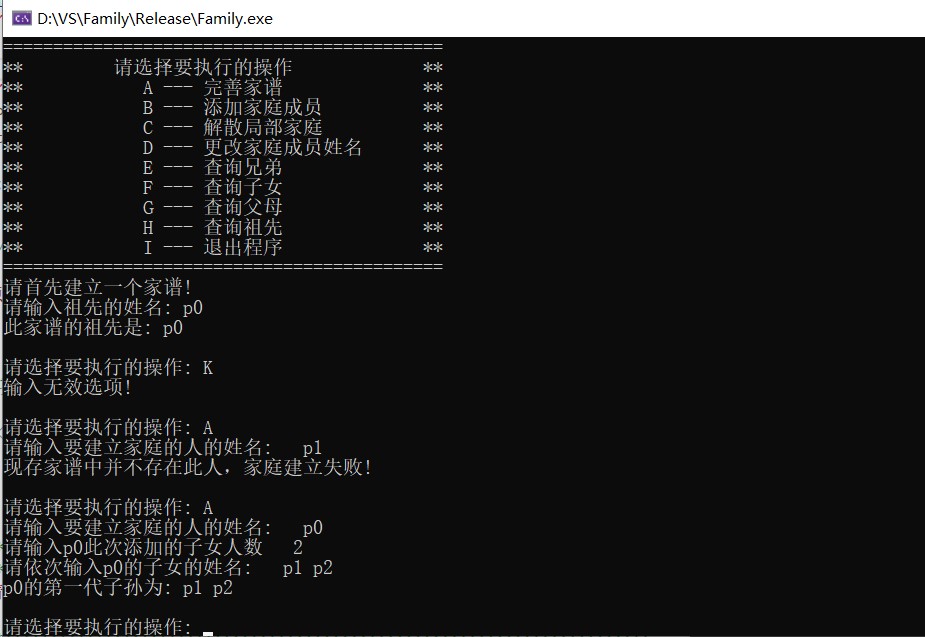


④删除操作详细测试（退出删除操作、删除目标不存在、删除操作验证)：

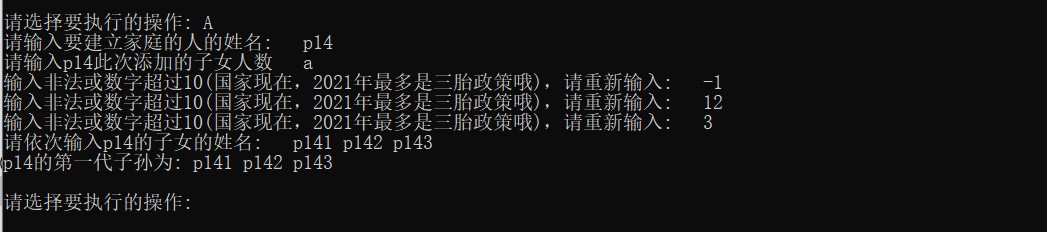


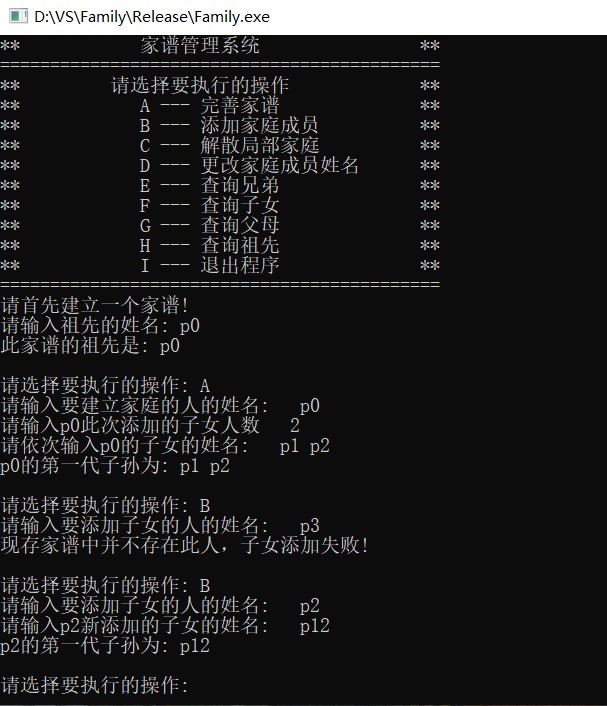
⑤非法输入处理：

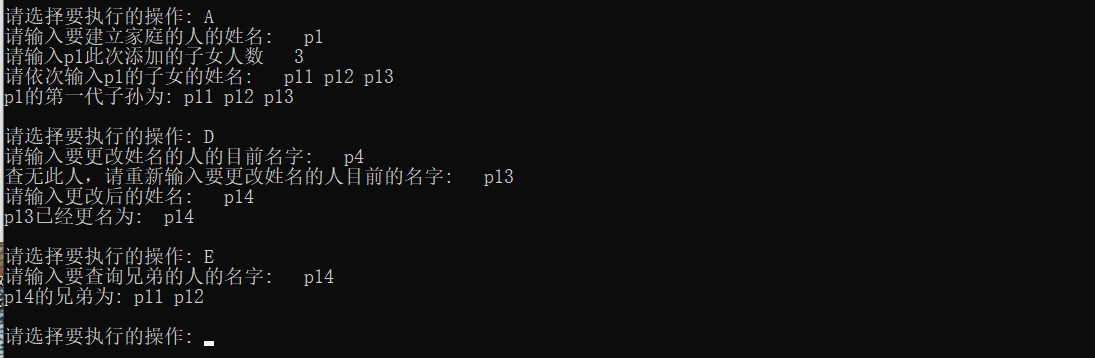
1. 输入无效操作选项、待完善家庭的目标不存在：



1. 插入的子女数量为非数字字符或超范围：



3.待添加子女的目标对象不存在：

4.待更改姓名的原名者不存在：