

题目六 家谱管理系统

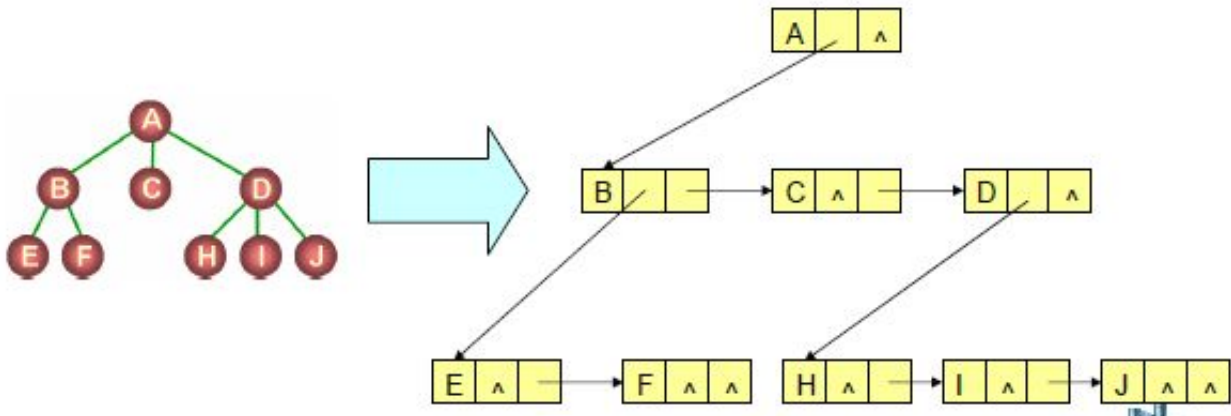
一.设计思路

家谱管理系统要求记录一个家族的亲属血缘关系，包括祖先与后代的关系、同一代人的兄弟关系等。在此基础上，实现查找，插入，修改，删除等功能。

这自然让人联想到用树作为存储的数据结构。然而，每个父辈的子女数量不定，如果采用传统的存储方式，无法确定分配给树中每个节点分配多少空间。分配过多会造成浪费。分配过少则无法解决子女过多的情况。

综合几方面因素，决定采用“孩子——兄弟表示法”来存储多叉树。

其原理可表示如下：



每个节点有两个指针域，一个指向其长子，另一个指向其兄弟。

搜索时，先找到长子，再通过长子找到其同辈的兄弟。

这样即可以用统一的空间存储不同子女数量的家庭，避免了不必要的内存开销。

二.数据结构实现

1.树结点类型（member）

定义了结构体 `struct member` 作为树的结点，即家谱中每个成员。

2.家谱类型（familyTree）

定义了 `member` 的指针 `typedef member* familyTree;` 作为族谱类型。

3.类成员

```
string name;  
member* offspring = NULL;  
member* brother = NULL;
```

`name` 为 C++ STL 容器的 `string` 类，代表成员对象姓名。

`offspring` 和 `brother` 均为 `member *` 类，分别代表改成员对象的长子结点与同辈的兄弟结点。

4.构造函数

```
member(){}    // 默认构造函数
```

默认构造函数，用于一般新建成员结点。内部调用 `string` 类的默认构造函数进行初始化。

```
member(string n):name(n){}    // 用成员名字初始化
```

传入成员姓名进行初始化。在添加结点时使用，免去了后续的赋值过程。

5.析构函数

```
~member(){           // 析构函数
    offspring = NULL;
    brother = NULL;
}
```

析构函数将相应的指针置为空

6.主要功能函数

```
member* findMember(familyTree tree,string m){ // 根据名字查找成员
```

实现根据姓名查找对应的成员对象的功能，返回指向该对象的指针。

是添加，删除，修改等函数都要调用的基础功能函数。

```
member* addOffspring(familyTree tree,string n){ // 为指定成员添加后代
```

实现根据给定的成员对象的指针，添加其子女的功能，子女姓名由参数传入。

将功能A——完善家庭成员与功能B——添加家庭成员合二为一。提高了代码复用率，简化了函数逻辑。

```
member* modify(familyTree tree,string past,string now){ // 修改指定成员的名字
```

根据给定的成员对象指针，修改其姓名。新的姓名由参数传入。

内部调用 `findMember(familyTree tree,string m)` 函数。

```
void deleteFamily(familyTree tree){ // 删除指定成员
```

删除给定成员对象的全部子女结点。这些子女成员的子女也被递归地删除。
该节点本身仍然保留下来。

```
void displayOffspring(familyTree p){ //遍历输出子女信息
```

按照添加的先后顺序，循环输出所有子女节点的信息。

7.辅助函数

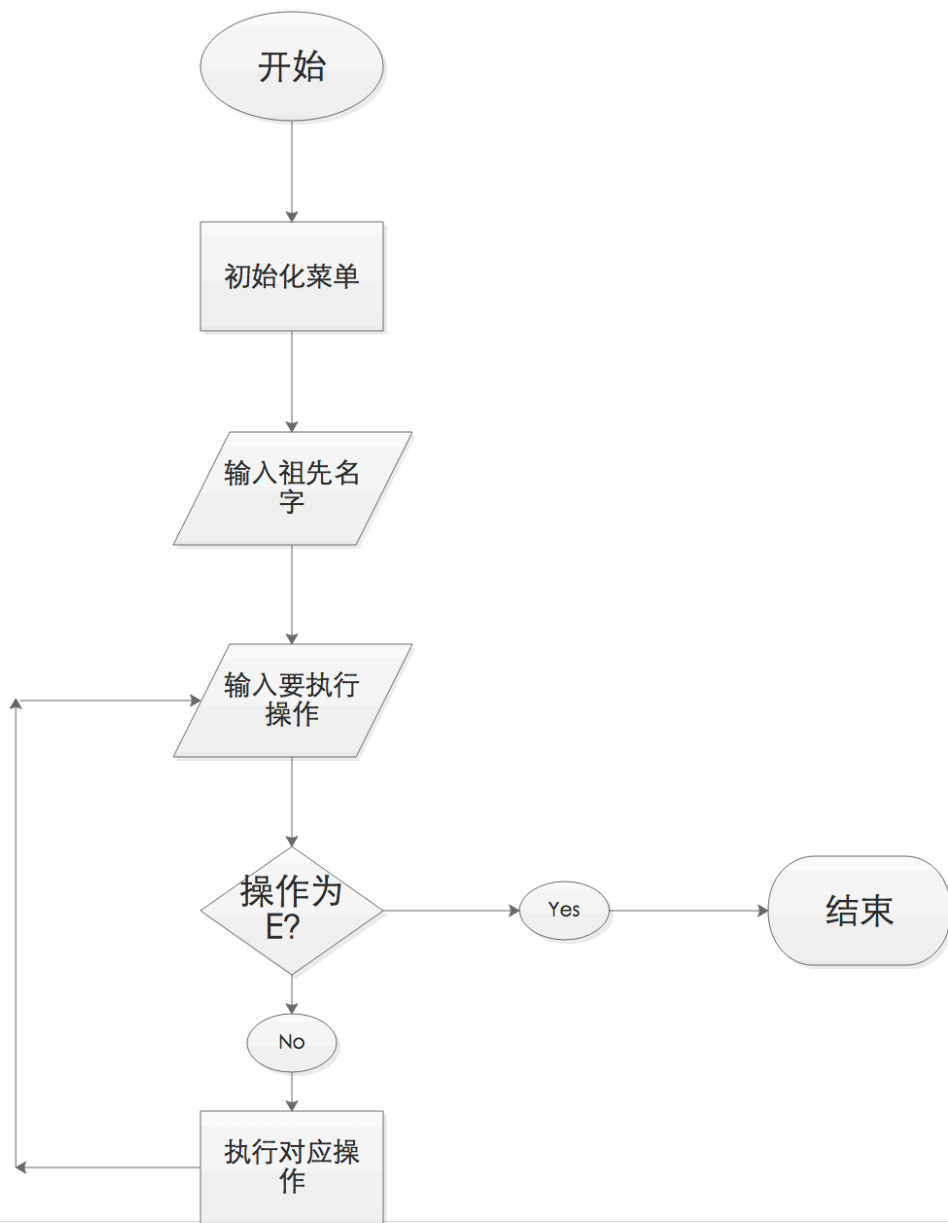
```
void menu(){ //打印菜单

    cout<<"**          家谱管理系统          **\n"
    <<"===== \n"
    <<"**    请选择要执行的操作:    **\n"
    <<"**    A --- 完善家庭          **\n"
    <<"**    B --- 添加家庭成员      **\n"
    <<"**    C --- 解散局部家庭      **\n"
    <<"**    D --- 更改家庭成员姓名  **\n"
    <<"**    E --- 退出程序          **\n"
    <<"===== \n"
    <<"首先建立一个家谱! \n"
    <<"请输入祖先的姓名: ";
}
```

打印主菜单。

三.系统实现

1.系统执行框架



首先调用 `void menu()` 函数初始化菜单界面。
随后要求用户输入该家谱的初代祖先节点。

核心代码如下：

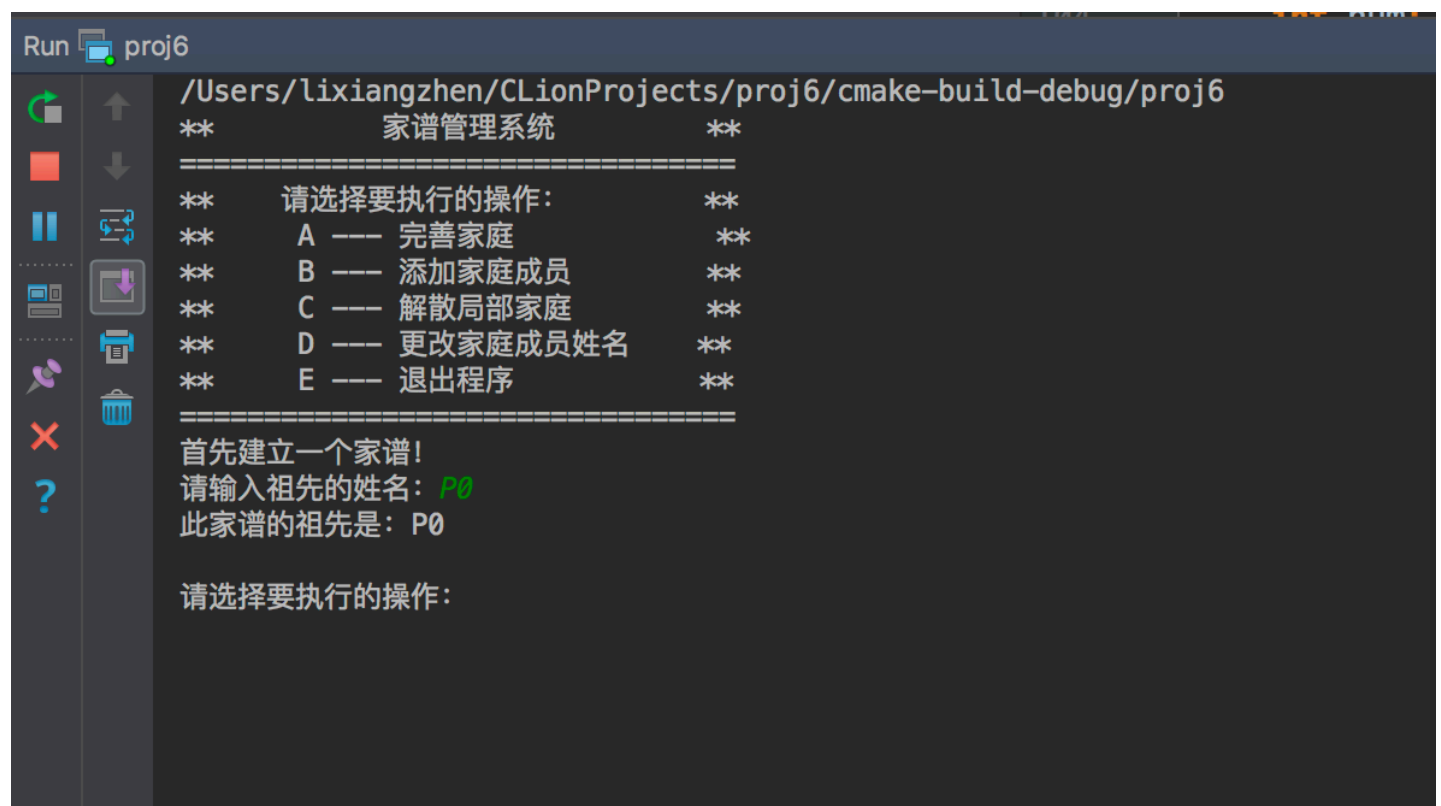
```

string n;
familyTree myTree;
menu(); //初始化输出菜单
cin>>n;
cout<<"此家谱的祖先是: "<<n<<endl;
myTree = new member(n); //新建一颗家谱树

```

- 初始化输出菜单
- 新建一颗家谱树

程序执行情况如下：



The screenshot shows the CLion IDE with a project named 'proj6'. The terminal output displays the following text:

```

/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**          家谱管理系统          **
=====
**  请选择要执行的操作:  **
**  A --- 完善家庭      **
**  B --- 添加家庭成员  **
**  C --- 解散局部家庭  **
**  D --- 更改家庭成员姓名 **
**  E --- 退出程序      **
=====
首先建立一个家谱!
请输入祖先的姓名: P0
此家谱的祖先是: P0

请选择要执行的操作:

```

随后输入所选操作，利用 `switch--case` 语句判定不同操作，并跳转到相应位置。

核心代码如下：

```
char op;
int num;
member* p;
cout<<"\n请选择要执行的操作： ";
/*
 * 依据不同的选择执行操作
 */
while(cin>>op && op != 'E'){
    switch (op){
        case 'A':
            cout<<"请输入要建立家庭的人的姓名： ";
            cin>>n;
            p = findMember(myTree,n);
            if(p == NULL){          //处理非法输入
                cout<<"查无此人! \n";
                break;
            }
            cout<<"请输入" <<n<<"的儿女数： ";
            cin>>num;
            cout<<"请依此输入"<<n<<"的儿女的姓名： ";
            for(int i = 0;i < num;++i){    //循环添加子女
                cin>>n;
                addOffspring(p,n);
            }
            displayOffspring(p);
            break;
        case 'B':
            cout<<"请输入要添加儿子(或女儿)的人的姓名： ";
            cin>>n;
            p = findMember(myTree,n);
            if(p == NULL){          //处理非法输入
                cout<<"查无此人! \n";
                break;
            }
            cout<<"请输入"<<n<<"新添加的儿子(或女儿)的姓名： ";
            cin>>n;
            addOffspring(p,n);
            displayOffspring(p);
    }
}
```

```

        break;
    case 'C':
        cout<<"请输入要解散的人的姓名: ";
        cin>>n;
        cout<<"要解散家庭的人是: "<<n<<endl;
        p = findMember(myTree,n);
        if(p == NULL){           //处理非法输入
            cout<<"查无此人! \n";
            break;
        }
        displayOffspring(p);
        deleteFamily(p->offspring); //删除该节点所有子女
        p->offspring = NULL;         //将相应指针置空, 以免非法访问
        break;
    case 'D':
        cout<<"请输入要更改的人的目前姓名: ";
        cin>>n;
        string now;
        cout<<"请输入更改后的姓名: ";
        cin>>now;
        modify(myTree,n,now);      //修改该节点
        cout<<n<<"已更名为"<<now<<endl;
        break;
    }
    cout<<"\n请选择要执行的操作: ";
}

```

程序执行情况如下:


```
Run proj6
/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**          家谱管理系统          **
=====
**      请选择要执行的操作:      **
**      A --- 完善家庭          **
**      B --- 添加家庭成员      **
**      C --- 解散局部家庭      **
**      D --- 更改家庭成员姓名  **
**      E --- 退出程序          **
**                               **
=====
首先建立一个家谱!
请输入祖先的姓名: p0
此家谱的祖先是: p0

请选择要执行的操作: A
请输入要建立家庭的人的姓名:
```

2.查找功能

核心代码如下:

```

member* findMember(familyTree tree,string m){ // 根据名字查找成员
    if(tree == NULL)    //查找失败
        return NULL;
    if(tree->name == m) //查找成功
        return tree;

    member* temp = findMember(tree->brother,m);
    if(temp != NULL)    //先查找长子
        return temp;
    else
        return findMember(tree->offspring,m);    //递归查找同辈成员
}

```

- 根据名字查找成员
- 处理空树
- 递归查找子女。

关联调用情况如下：

```

member* modify(familyTree tree,string past,string m){
    member* temp = findMember(tree,past);
    if(temp == NULL)    //判断操作合法性
        cout<<"查无此人! \n";
}

```

```

cout<<"请输入要建立家庭的人的姓名";
cin>>n;
p = findMember(myTree,n);
if(p == NULL){           //处理非法输入
    cout<<"查无此人! \n";
    break;
}

```

3.添加功能

核心代码如下：

```

member* addOffspring(familyTree tree,string n){ // 为指定成员添加后代
    if(tree->offspring == NULL)    //处理空树情况
        tree->offspring = new member(n);
    else{
        member* temp = tree->offspring;
        while(temp->brother != NULL)    //找到同辈最后一个子女
            temp = temp->brother;
        temp->brother = new member(n);
    }
    return tree;
}

```

- 处理空树情况
- 找到同辈最后一个子女
- 将新节点加到后面

程序执行情况如下：

```
Run proj6
/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**      家谱管理系统      **
=====
**      请选择要执行的操作：      **
**      A --- 完善家庭      **
**      B --- 添加家庭成员      **
**      C --- 解散局部家庭      **
**      D --- 更改家庭成员姓名      **
**      E --- 退出程序      **
=====
首先建立一个家谱！
请输入祖先的姓名： p0
此家谱的祖先是： p0

请选择要执行的操作： A
请输入要建立家庭的人的姓名： p0
请输入p0的儿女数： 3
请依此输入p0的儿女的姓名： p01 p02 p03
p0的第一代子孙是： p01 p02 p03

请选择要执行的操作： B
请输入要添加儿子(或女儿)的人的姓名： p0
请输入p0新添加的儿子(或女儿)的姓名： p04
p0的第一代子孙是： p01 p02 p03 p04

请选择要执行的操作：
```

4.修改功能

核心代码如下：

```
member* modify(familyTree tree,string past,string now){ // 修改指定成员的名字
    member* temp = findMember(tree,past);
    if(temp == NULL)    //判断操作合法性
        cout<<"查无此人! \n";
    else
        temp->name = now;

    return temp;
}
```

- 判断成员是否存在
- 存在则执行修改。

程序执行情况如下：

```
Run proj6
/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**          家谱管理系统          **
=====
**  请选择要执行的操作:  **
**  A  ---  完善家庭      **
**  B  ---  添加家庭成员  **
**  C  ---  解散局部家庭  **
**  D  ---  更改家庭成员姓名 **
**  E  ---  退出程序      **
=====
首先建立一个家谱!
请输入祖先的姓名: p0
此家谱的祖先是: p0

请选择要执行的操作: A
请输入要建立家庭的人的姓名: p0
请输入p0的儿女数: 3
请依次输入p0的儿女的姓名: p01 p02 p03
p0的第一代子孙是: p01 p02 p03

请选择要执行的操作: B
请输入要添加儿子(或女儿)的人的姓名: p0
请输入p0新添加的儿子(或女儿)的姓名: p04
p0的第一代子孙是: p01 p02 p03 p04

请选择要执行的操作: D
请输入要更改的人的目前姓名: p02
请输入更改后的姓名: jack
p02已更名为jack

请选择要执行的操作:
```

5.删除功能

核心代码如下:

```
void deleteFamily(familyTree tree){ // 删除指定成员
    if(tree != NULL){
        deleteFamily(tree->brother); //递归的删除该节点所有兄弟及子女节点
        deleteFamily(tree->offspring);
        delete tree;
    }
}
```

- 递归删除该节点一下所有节点。
- 该节点本身不变

程序执行情况如下：

```
请选择要执行的操作: A
请输入要建立家庭的人的姓名: p02
请输入p02的儿女数: 2
请依此输入p02的儿女的姓名: jack smith
p02的第一代子孙是: jack smith

请选择要执行的操作: C
请输入要解散的人的姓名: p02
要解散家庭的人是: p02
p02的第一代子孙是: jack smith

请选择要执行的操作: B
请输入要添加儿子(或女儿)的人的姓名: p02
请输入p02新添加的儿子(或女儿)的姓名: bob
p02的第一代子孙是: bob

请选择要执行的操作:
```

6.遍历功能

核心代码如下：

```

void displayOffspring(familyTree p){ //遍历输出子女信息
    cout<<p->name<<"的第一代子孙是： ";
    p = p->offspring;          //先输出长子
    while(p != NULL){          //循环输出兄弟节点
        cout<<p->name<<" ";
        p = p->brother;
    }
    cout<<endl;
}

```

- 先输出长子
- 循环输出兄弟节点

关联调用情况如下：

```

        cout<<"请输入"<<n<<"新添加的儿子(或女儿)的姓名： ";
        cin>>n;
        addOffspring(p,n);
        displayOffspring(p);
        break;
    case 'C':

```

```

        cout<<"要解散家庭的人是："<<n<<endl;
        p = findMember(myTree,n);
        if(p == NULL){ //处理非法输入
            cout<<"查无此人！ \n";
            break;
        }
        displayOffspring(p);
        deleteFamily(p->offspring); //删除该节点所有子女
        p->offspring = NULL;         //将相应指针置空，以免非法访问
        break;
    case 'D':
        cout<<"请输入要删除的个体成员姓名："<<n<<endl;

```


四.测试

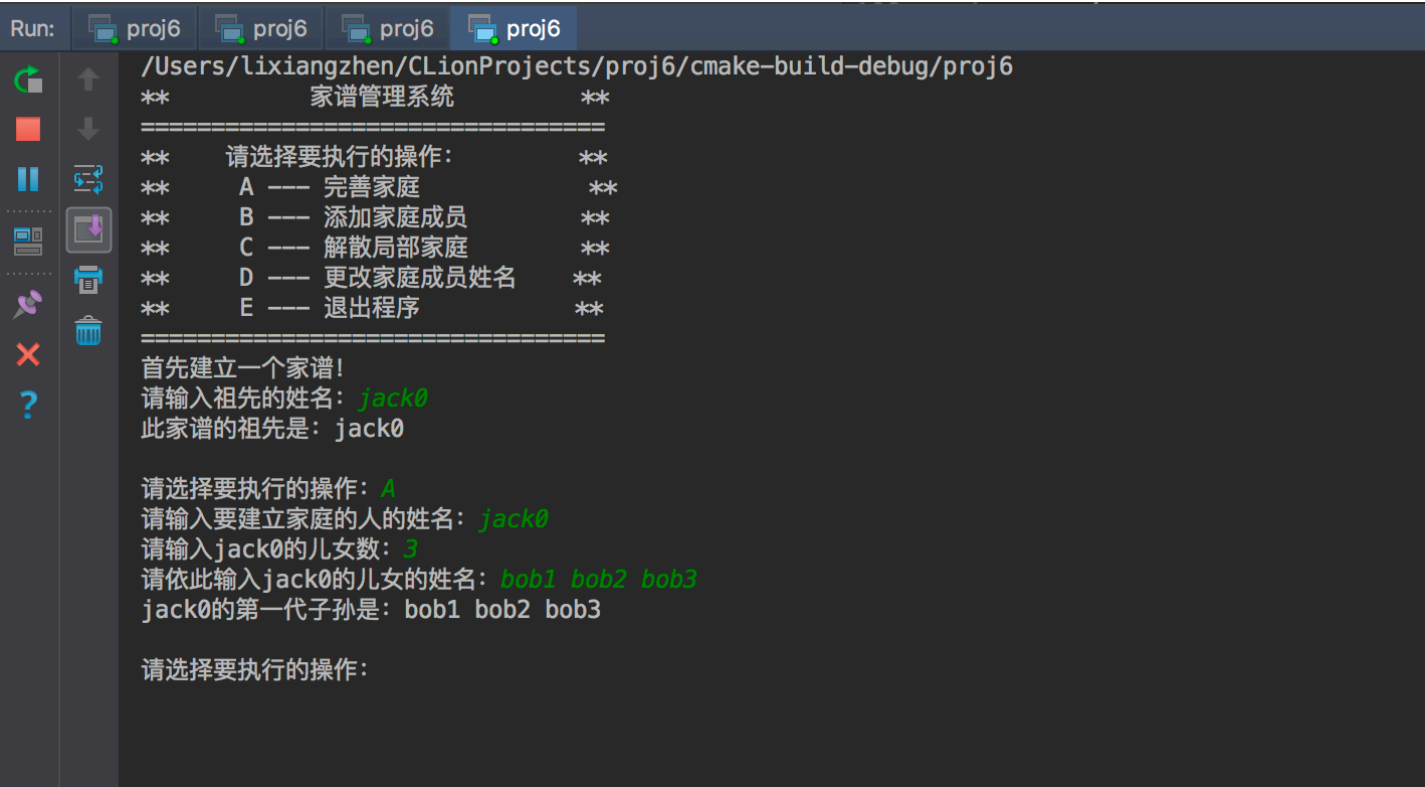
1.基本功能测试

测试用例

初代祖先: jack0
第一代子孙: bob1 , bob2 , bob3 .
bob2的子孙: smith1 , smith2

建立并完善初代族谱

程序执行情况如下:



添加第二代家庭成员

程序执行情况如下：

```
Run: proj6 proj6 proj6 proj6
/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**      家谱管理系统      **
=====
**      请选择要执行的操作:      **
**      A --- 完善家庭          **
**      B --- 添加家庭成员      **
**      C --- 解散局部家庭      **
**      D --- 更改家庭成员姓名  **
**      E --- 退出程序          **
**      **
=====
首先建立一个家谱!
请输入祖先的姓名: jack0
此家谱的祖先是: jack0

请选择要执行的操作: A
请输入要建立家庭的人的姓名: jack0
请输入jack0的儿女数: 3
请依此输入jack0的儿女的姓名: bob1 bob2 bob3
jack0的第一代子孙是: bob1 bob2 bob3

请选择要执行的操作: B
请输入要添加儿子(或女儿)的人的姓名: bob2
请输入bob2新添加的儿子(或女儿)的姓名: smith1
bob2的第一代子孙是: smith1

请选择要执行的操作: B
请输入要添加儿子(或女儿)的人的姓名: bob2
请输入bob2新添加的儿子(或女儿)的姓名: smith2
bob2的第一代子孙是: smith1 smith2

请选择要执行的操作:
```

更改成员姓名

程序执行情况如下：

bob2的第一代子孙是: smith1 smith2

请选择要执行的操作: *D*
请输入要更改的人的目前姓名: *bob3*
请输入更改后的姓名: *bob233*
bob3已更名为bob233

请选择要执行的操作:

解散局部家庭

程序执行情况如下:

bob3已更名为bob233

请选择要执行的操作: *C*
请输入要解散的人的姓名: *bob2*
要解散家庭的人是: bob2
bob2的第一代子孙是: smith1 smith2

请选择要执行的操作:

程序退出

程序执行情况如下:

```
请输入要解散的人的姓名: bob2
要解散家庭的人是: bob2
bob2的第一代子孙是: smith1 smith2

请选择要执行的操作: E

Process finished with exit code 0
|
```

2.边界测试

操作对象不存在

程序执行情况如下:

```
Run proj6
/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6
**          家谱管理系统          **
=====
**  请选择要执行的操作:  **
**  A --- 完善家庭      **
**  B --- 添加家庭成员  **
**  C --- 解散局部家庭  **
**  D --- 更改家庭成员姓名 **
**  E --- 退出程序      **
=====
首先建立一个家谱!
请输入祖先的姓名: P0
此家谱的祖先是: P0

请选择要执行的操作: A
请输入要建立家庭的人的姓名: P0
请输入P0的儿女数: 2
请依此输入P0的儿女的姓名: JACK ALICE
P0的第一代子孙是: JACK ALICE

请选择要执行的操作: C
请输入要解散的人的姓名: BOB
要解散家庭的人是: BOB
查无此人!

请选择要执行的操作:
```

所选操作非法则操作被忽视

程序执行情况如下：

Run:

proj6

proj6

/Users/lixiangzhen/CLionProjects/proj6/cmake-build-debug/proj6

** 家谱管理系统 **

=====

** 请选择要执行的操作: **

** A --- 完善家庭 **

** B --- 添加家庭成员 **

** C --- 解散局部家庭 **

** D --- 更改家庭成员姓名 **

** E --- 退出程序 **

=====

首先建立一个家谱!

请输入祖先的姓名: **ALICE**

此家谱的祖先是: ALICE

请选择要执行的操作: **W**

请选择要执行的操作: