项目说明文档

数据结构课程设计

——家谱管理系统

作 者 姓 名： 杨宇琨

学 号： 2252843

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

# 1 分析

## 1.1 背景分析

家谱作为中国特有的文化遗产，承载了丰富的家族历史、血缘关系和重要事迹，具有重要的历史和文化价值。随着社会的发展和家族文化的传承，对家谱管理的需求日益增加。本项目旨在进行家谱管理的简单模拟，通过程序实现家族成员信息的建立、查找、插入、修改和删除等功能，以满足家谱管理的基本需求。

## 1.2 功能分析

1. 定义家族成员数据结构：

- 需要定义合适的数据结构来存储家族成员的个人信息，包括姓名、性别、出生日期、家庭角色等信息，以便进行管理和操作。

2. 实现家族成员信息的建立：

- 提供功能用于添加新的家族成员信息，包括输入个人信息和与其他成员的关系，将其存储在家谱中。

3. 实现家族成员信息的查找：

- 提供功能用于根据姓名或其他标识查找家族成员的个人信息，以便用户查询特定成员的相关信息。

4. 实现家族成员信息的插入：

- 提供功能用于在家谱中插入新的家族成员信息，包括指定父母、配偶等关系，维护家谱的完整性。

5. 实现家族成员信息的修改：

- 提供功能用于修改已有家族成员的个人信息，例如修改姓名、出生日期等，以及调整与其他成员的关系。

6. 实现家族成员信息的删除：

- 提供功能用于删除指定的家族成员信息，包括个人信息和与其他成员的关系，维护家谱的完整性和准确性。

7. 主函数验证功能：

- 主函数用于验证各个成员函数的功能，包括添加、查找、插入、修改和删除等操作，并展示运行结果，以确保各功能的正确实现和整体的可用性。

通过以上功能，可以实现一个简单的家谱管理系统，满足用户对家族成员信息管理的基本需求，帮助用户更好地管理和传承家族文化。

# 2 设计

## 2.1 数据结构设计

\*\*FamilyMember 结构体设计\*\*：

- 成员姓名 name：用字符数组存储成员姓名，长度为20

- 成员年龄 age：用整型变量存储成员年龄

- 指向第一个孩子节点的指针 firstChild：指向该成员的第一个孩子的指针

- 指向下一个兄弟节点的指针 nextSibling：指向该成员的下一个兄弟成员的指针

- 该结构体表示家谱树中的一个成员，通过指针关系构成树形结构

struct FamilyMember {

char name[20];

int age;

FamilyMember\* firstChild;

FamilyMember\* nextSibling;

FamilyMember() : name(" "), age(0), firstChild(nullptr), nextSibling(nullptr) {}

};

## 2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（LNode）与链表类（LinkList），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统采用struct描述链表结点类（LNode），这样使得链表结点类（LinkList）可以访问链表结点。

## 2.3 成员与操作设计

- 公有成员函数：

- 构造函数 FamilyTree(char\* name, int age)：初始化根节点

- addChild(char\* parentName, char\* childName, int childAge)：向指定父节点添加子节点

- findMember(FamilyMember\* current, char\* name)：在家谱树中根据名称查找成员

- findParent(FamilyMember\* start, char\* childName)：在家谱树中根据子节点名称找到父节点

- printFamilyTree(FamilyMember\* current, int level)：递归打印家谱树

- getRoot()：获取家谱树的根节点指针

- modifyMember(char\* name, char\* newName, int newAge)：修改成员信息

- deleteSubtree(FamilyMember\* member)：删除以指定成员为根的子树

- disbandFamily(char\* name)：解散指定成员的家庭

## 2.4 系统设计

1. \*\*家谱管理系统\*\*：

- 通过命令行菜单实现家谱管理功能，包括建立家谱、添加成员、解散局部家庭、修改成员信息等操作

2. \*\*交互式操作\*\*：

- 通过键盘输入进行家谱管理操作的选择和数据输入

3. \*\*数据验证\*\*：

- 对输入的数据进行验证，确保数据的合法性和完整性

4. \*\*家谱树的构建和遍历\*\*：

- 通过递归实现家谱树的构建和遍历，包括添加、修改、删除等操作

5. \*\*用户提示和反馈\*\*：

- 在操作过程中进行错误提示和操作结果的反馈，提高用户体验和系统健壮性

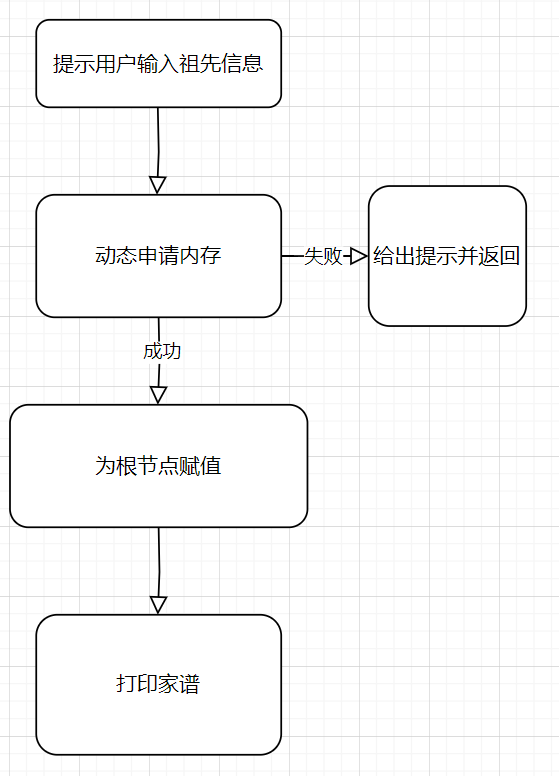
6. \*\*系统交互流程\*\*：

- 通过循环和条件判断实现系统的交互流程，使得用户可以进行多次操作，直到选择退出程序。

# 3 实现

## 3.1 家谱建立功能的实现

### 3.1.1 家谱建立功能流程图



### 3.1.2 家谱建立功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:FamilyTree

Function:To initialize the root of the tree

Input Parameters:Root's name and age

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FamilyTree(char\* name, int age)

{

root = new(std::nothrow) FamilyMember;

if (root == NULL) {

std::cout << "家谱内存申请失败" << std::endl;

return;

}

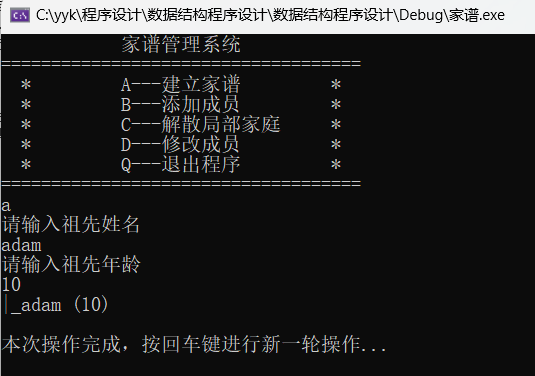
strcpy\_s(root->name, name);

root->age = age

;

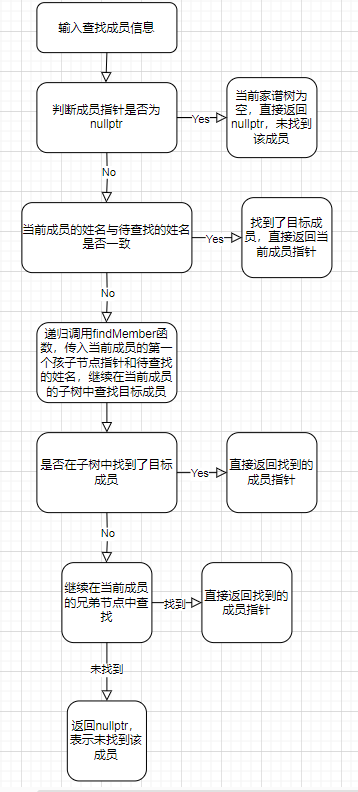
}

### 3.1.3 家谱建立功能截屏示例



## 3.2 查找功能的实现

### 3.2.1 查找功能流程图



### 3.2.2 查找功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:findMember

Function: Find a member by name

Input Parameters:FamilyMember\* current, char\* name

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

FamilyMember\* findMember(FamilyMember\* current, char\* name)

{

if (current == nullptr) {

return nullptr;

}

if (strcmp(current->name, name) == 0) {

return current;

}

FamilyMember\* foundInChildren = findMember(current->firstChild, name);

if (foundInChildren != nullptr) {

return foundInChildren;

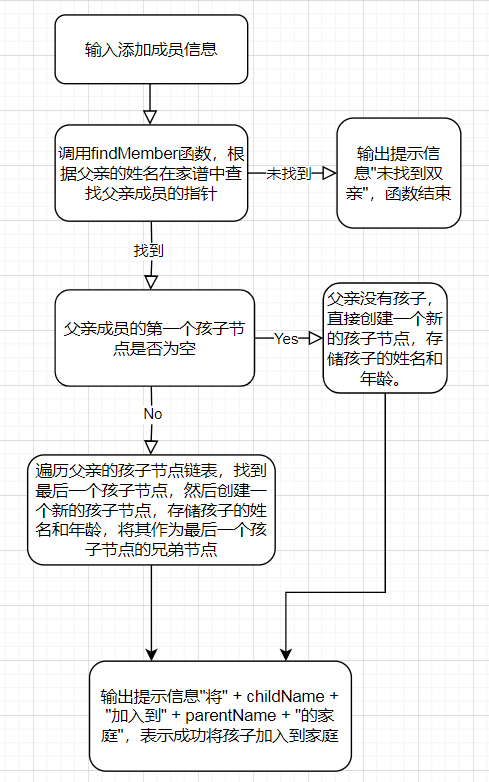
}

return findMember(current->nextSibling, name);

}

## 3.3 插入功能的实现

### 3.3.1 插入功能流程图



### 3.3.2 插入功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:addChild

Function:Add a child to a member

Input Parameters:char\* parentName, char\* childName, int childAge

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void addChild(char\* parentName, char\* childName, int childAge)

{

FamilyMember\* parent = findMember(root, parentName);

if (parent == nullptr) {

std::cout << "未找到双亲" << std::endl;

return;

}

if (parent->firstChild == nullptr) {

parent->firstChild = new(std::nothrow) FamilyMember;

if (parent->firstChild == NULL) {

std::cout << "家谱内存申请失败" << std::endl;

return;

}

parent->firstChild->age = childAge;

strcpy\_s(parent->firstChild->name, childName);

}

else {

FamilyMember\* sibling = parent->firstChild;

while (sibling->nextSibling != nullptr) {

sibling = sibling->nextSibling;

}

sibling->nextSibling = new(std::nothrow) FamilyMember;

if (sibling->nextSibling == NULL) {

std::cout << "家谱内存申请失败" << std::endl;

return;

}

sibling->nextSibling->age = childAge;

strcpy\_s(sibling->nextSibling->name, childName);

sibling->nextSibling->nextSibling = nullptr;

}

std::cout << "将" << childName << "加入到" << parentName << "的家庭" << std::endl;

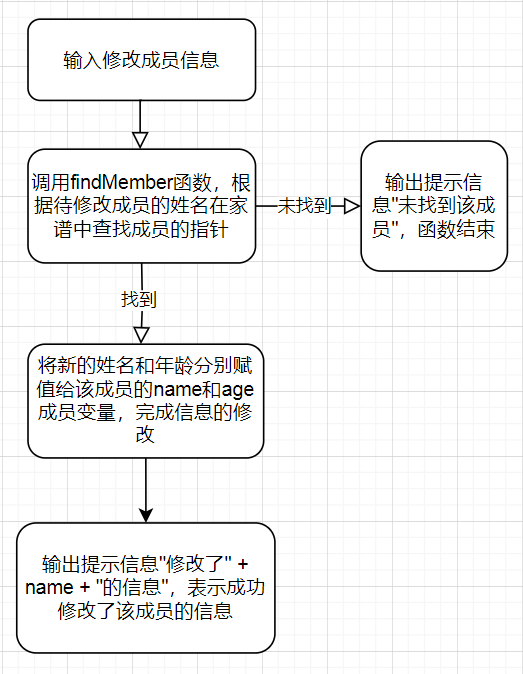
### }

### 3.3.3 插入功能截图示例



## 3.4 修改功能的实现

### 3.4.1 修改功能流程图



### 3.4.2 修改功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:modifyMember

Function:To modify the information of a member

Input Parameters:char\* name,char\* newName, int newAge

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void modifyMember(char\* name, char\* newName, int newAge)

{

FamilyMember\* member = findMember(root, name);

if (member != nullptr) {

strcpy\_s(member->name, newName);

member->age = newAge;

std::cout << "修改了" << name << "的信息" << std::endl;

}

else {

std::cout << "未找到该成员" << std::endl;

}

### }

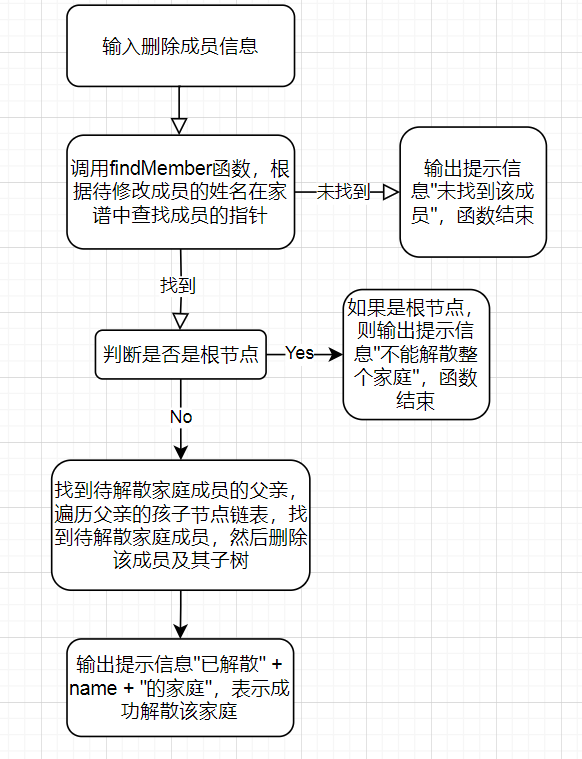
### 3.4.3 修改功能截屏示例

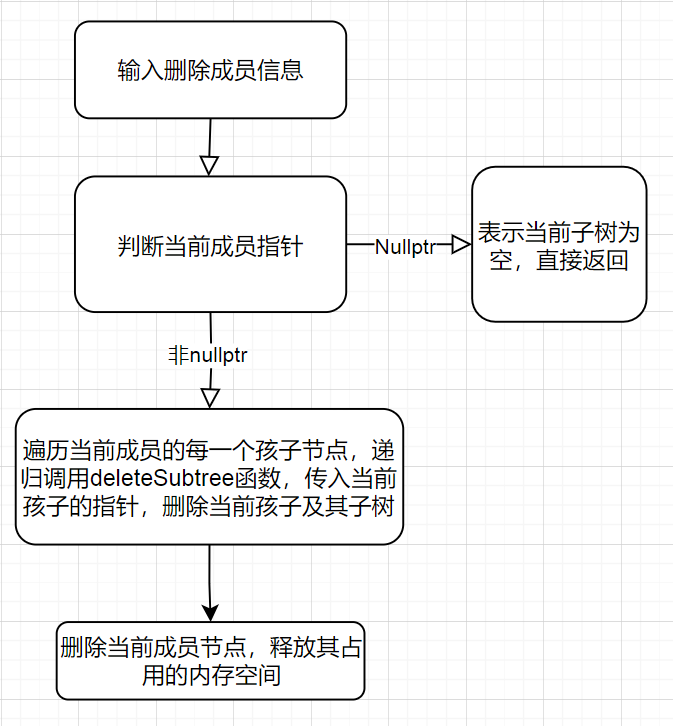


## 3.5 删除功能的实现

### 

### 3.5.1 删除功能流程图





### 3.5.2 删除功能核心代码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:deleteSubtree

Function:To delete a member and its subtree

Input Parameters:FamilyMember\* member

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void deleteSubtree(FamilyMember\* member)

{

if (member == nullptr) {

return;

}

FamilyMember\* child;

child = member->firstChild;

while (child != NULL) {

FamilyMember\* temp;

temp = child->nextSibling;

deleteSubtree(child);

child = temp;

}

delete member;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Function Name:disbandFamily

Function:To disband a family

Input Parameters:char\* name

Return Value:void

Documentation：

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void disbandFamily(char\* name)

{

FamilyMember\* member = findMember(root, name);

if (member != nullptr) {

if (member == root) {

std::cout << "不能解散整个家庭" << std::endl;

return;

}

FamilyMember\* parent = findParent(root, member->name);

if (parent->firstChild == member)

parent->firstChild = parent->firstChild->nextSibling;

else {

FamilyMember\* child = parent->firstChild;

while (child->nextSibling != member) {

child = child->nextSibling;

}

child->nextSibling = child->nextSibling->nextSibling;

}

deleteSubtree(member);

std::cout << "已解散" << name << "的家庭" << std::endl;

}

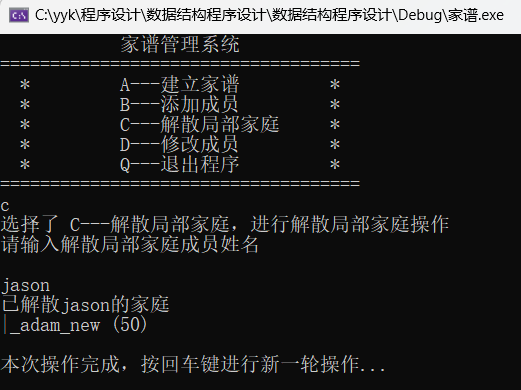
else {

std::cout << "未找到该成员" << std::endl;

}

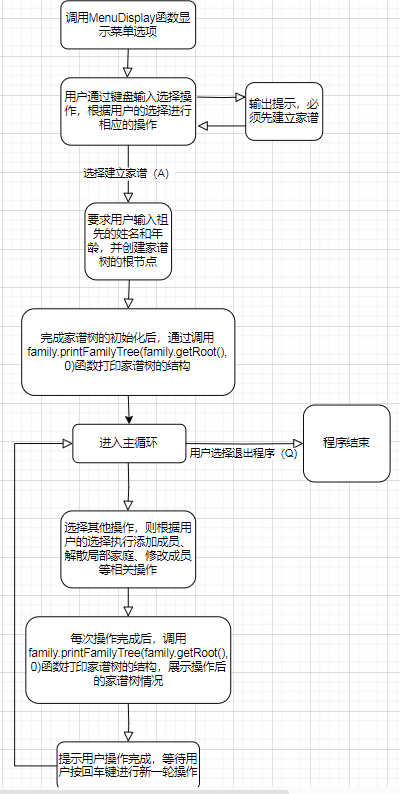
### }

### 3.5.3 删除功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统核心代码

//Display our list

MenuDisplay();

while (true) {

char controller = \_getch();//\_getch can avoid displaying undesired input on the screen

if ((controller >= 'A' && controller <= 'D') || controller == 'Q' || controller == 'q' || (controller >= 'a' && controller <= 'd')) {

std::cout << controller << std::endl;//Display the choice on the screen

switch (controller)

{

case 'q':

case 'Q':

std::cout << "程序结束！" << std::endl;

return 0;

case 'a':

case 'A':

break;

default:

std::cout << "请先建立家谱！" << std::endl;

continue;

}

break;

}//end of if

}//end of while

int Ancestorsage;

char AncestorsName[max\_input\_name];

std::cout << "请输入祖先姓名" << std::endl;

inputname(AncestorsName);

std::cout << "请输入祖先年龄" << std::endl;

Ancestorsage = input<int>(0, 100000000);

FamilyTree family(AncestorsName, Ancestorsage);

family.printFamilyTree(family.getRoot(), 0);

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

while (true) {

//Clear our screen

system("cls");

//Display our prompt

MenuDisplay();

char controller;//Create a variable to hold the choice we input

while (true) {

controller = \_getch();//\_getch can avoid displaying undesired input on the screen

if ((controller >= 'A' && controller <= 'D') || controller == 'Q' || controller == 'q' || (controller >= 'a' && controller <= 'd')) {

std::cout << controller << std::endl;//Display the choice on the screen

switch (controller)

{

case 'q':

case 'Q':

std::cout << "程序结束！" << std::endl;

return 0;

case 'a':

case 'A':

std::cout << "不可重复建立家谱！" << std::endl;

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

break;

case 'b':

case 'B':

std::cout << "选择了 B---添加成员，进行添加成员操作" << std::endl;

{

int age;

char Pname[max\_input\_name];

char Nname[max\_input\_name];

std::cout << "请输入双亲姓名" << std::endl;

inputname(Pname);

std::cout << "请输入新添加成员姓名" << std::endl;

inputname(Nname);

std::cout << "请输入新添加成员年龄" << std::endl;

age = input<int>(0, 100000000);

family.addChild(Pname, Nname, age);

}

family.printFamilyTree(family.getRoot(), 0);

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

break;

case 'c':

case 'C':

std::cout << "选择了 C---解散局部家庭，进行解散局部家庭操作" << std::endl;

{

char name[max\_input\_name];

std::cout << "请输入解散局部家庭成员姓名" << std::endl;

inputname(name);

family.disbandFamily(name);

family.printFamilyTree(family.getRoot(), 0);

}

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

break;

case 'D':

case 'd':

std::cout << "选择了 D---修改成员，进行修改成员操作" << std::endl;

{

int age;

char name[max\_input\_name];

char Nname[max\_input\_name];

std::cout << "请输入修改成员原姓名" << std::endl;

inputname(name);

std::cout << "请输入修改成员现姓名" << std::endl;

inputname(Nname);

std::cout << "请输入修改成员现年龄" << std::endl;

age = input<int>(0, 100000000);

family.modifyMember(name, Nname, age);

family.printFamilyTree(family.getRoot(), 0);

}

std::cout << std::endl;

std::cout << "本次操作完成，按回车键进行新一轮操作...";

pause();

break;

default:

continue;

}

break;

}//end of if

}//end of while

}

### return 0;

### 3.6.3 总体系统截屏示例

