**《学生管理系统》实习报告**

**题目：用树编写一个可维护的学生管理系统**

**班级：19184115**

**专业：计算机科学与技术（人工智能与大数据）创新实验班**

**姓名：洪智豪 学号：19195216**

**完成日期：2020.10.1**

1. **需求分析**
2. 采用菜单的方式显示
   1. 打开程序后默认输出整个学生管理系统的状态
   2. 用横向树来表示目前学生管理系统的状态
   3. 用[编码]名字 的形式来表示节点
   4. 用光标移动选择操作的对象
   5. 选中班级后按回车键显示该班级中的学生 再按一次回车键回到班级
   6. 用菜单提示用户操作
3. 用六位编码表示节点
   1. 前一位数字代表学校编号
   2. 前二位数字代表学院编号
   3. 前三位数字代表专业编号
   4. 前四位数字表示班级编号
   5. 一共六位数字表示学生编号
4. 添加信息
   1. 将光标移动到节点上输入2进入添加子节点模式 输入它的名称完成添加操作

例如：将光标移到 杭州电子科技大学 上 输入2添加计算机学院

* 1. 添加时，会按照编码规则给新节点自动分配编号
  2. 添加学生时需要先输入回车键选中班级后 再输入2添加学生
  3. 当添加的编码重复的节点的时 会提示错误信息：该节点已经存在

1. 删除信息
   1. 将光标移到待删除的节点上，输入3删除该节点
   2. 若该节点存在子节点，则在删除时，会提示用户，是否确定删除
   3. 删除学生需要指定添加学生时需要指定学生的学号
   4. 删除不存在的学生时 会返回错误信息
2. 修改信息
   1. 将光标移到待修改的节点上，输入4修改该节点
   2. 修改学生需要指定添加学生时需要指定学生的学号
   3. 修改不存在的学生时 会返回错误信息
3. 查找符合要求的信息
   1. 输入5进入查找模式
   2. 在目前选定的节点下面查找
   3. 输入1进入精确查找模式 通过输入编码 或者输入姓名 进行精确查找
   4. 输入2进入模糊查找模式 通过输入部分编码或者部分姓名 模糊查找

将所有包含有关键字的信息全部输出

* 1. 若没有找到符合的节点 则显示错误信息 节点不存在

1. 撤回
   1. 可以在进行 增加/删除/修改操作 输入7后执行 撤销操作来还原被修改的信息
   2. 若未进行过增加/删除/修改操作时 进行撤销操作 会提示错误信息
   3. 若进行删除后的撤销操作，撤销可以将原节点返回原位置，保证代码的有序性

例如：111101 111102 111103 中删除了111102 则变为111101 111103

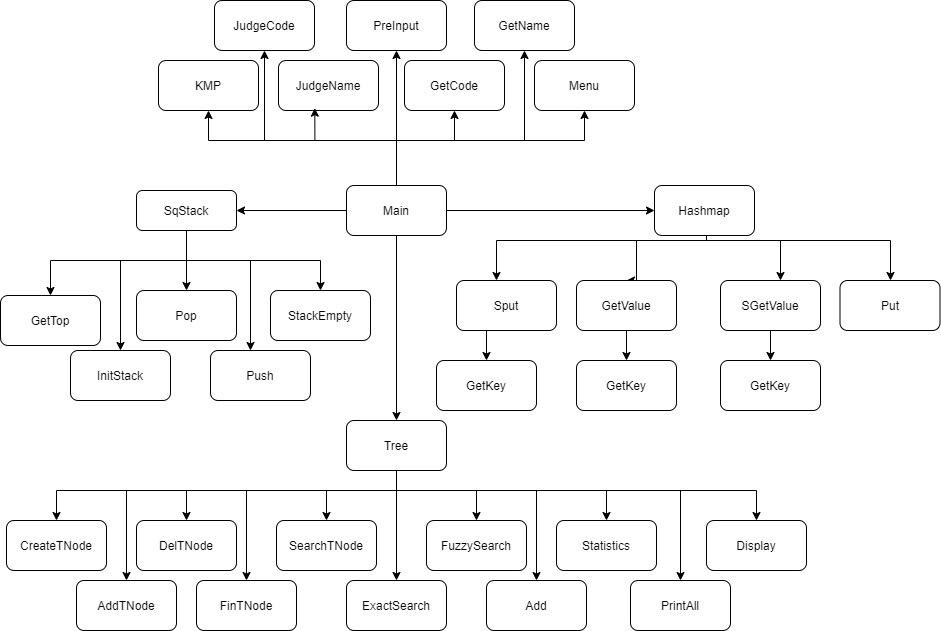
则进行撤销后会恢复111101 111102 111103状态

1. 导入数据
   1. 格式：每行一个[编码 名称]
   2. 可以通过txt文件导入的形式导入默认数据
   3. 导入 数据不符合格式 会报错
2. 统计数据
   1. 在菜单下输入7进入统计模式
   2. 统计并输出 整个系统中的全部信息

各个学校下学院的个数 各个学院下专业的个数

各个专业下班级下的个数 各个班级下学生的个数

1. **概要设计**
2. 模块流程图



1. 数据结构设计
   1. 树
      * 1. Tree CreateTNode();

//创建一个树的节点

* + - 1. Tree AddTNode(Tree t, Tree ele);

//添加树的节点

t:学生管理系统树 ele：待添加的节点

* + - 1. Tree DelTNode(Tree t, Tree ele);

//删除树的节点

t：学生管理系统树 ele：待删除的节点

* + - 1. Tree FinTNode(Tree t, Tree ele);

//找到树的节点

t：学生管理系统树 ele：待查找的节点

* + - 1. Tree SearchTNode(Tree t, char \*code);

//查找树的节点

t：学生管理系统树 code：待查找的节点的编码

* + - 1. Tree ExactSearch(Tree t, char \*code, char \*path);

//精确查找

t：学生管理系统树 code：待查找的节点的编码 path：该节点的父节点

* + - 1. Tree FuzzySearch(Tree t, char \*name, char \*path);

//模糊查找

t：学生管理系统树 name：待查找的节点包含的名字 path：该节点的父节点

* + - 1. void Add(Tree t, char \*code, char \*name);

//预添加

t：学生管理系统树 code：待查找的节点的编码 name:名字

导入数据

* + - 1. void Statistics(Tree t);

//统计各个对象的人数

t：学生管理系统树

* + - 1. void PrintAll(Tree t);

//显示全部数据信息

t：学生管理系统树

* + - 1. void Display(Tree t);

//显示部分数据信息

t：学生管理系统树

* 1. 栈

1. Status GetTop(SqStack \*s, SElemType \*e);

//获取栈顶元素

s：待初始化的栈 e：待初始化的节点

1. Status InitStack(SqStack \*s);

//构造一个空栈

s:生成的空栈

1. Status Pop(SqStack \*s, SElemType \*e);

//出栈

s：撤回栈 e：待出栈的节点

1. Status Push(SqStack(\*s), SElemType e);

//入栈

s：撤回栈 e：待入栈的节点1

1. Status StackEmpty(SqStack s);

//栈是否为空

s：待判断的栈

1. void StackClear(SqStack \*s);

//清空栈里所有元素

s：待清空的栈

* 1. 哈希表

1. void GetKey(char \*code);

//获取键

code:目标的编码

1. int GetValue(char \*code);

//获取值

code:目标的编码

1. void Put(char \*code, int value);

//建立键值对

code:目标的编码 value:编码对应的键值

1. int SGetValue(char \*code);

//统计时用的获取值

code:目标的编码

1. void SPut(char \*code, int value);

//统计时用的键值对

code:目标的编码 value:编码对应的键值

1. 模块设计
   1. Status KMP(char \*text, char \*pattern);

//字符串匹配

text：待匹配的字符串 pattern：

* 1. Status JudgeCode(char \*code);

//检测输入的编号是否合法函数

Code:输入的编号

* 1. Status JudgeName(char \*name);

//检测输入的名称是否合法函数

输入的名字

* 1. void PreInput(Tree t);

//预输入函数

t:学生管理系统树

* 1. void GetCode(char \*code);

//输入编号函数

Code:输入的编号

* 1. void GetName(char \*name);

//输入姓名函数

Name:输入的姓名

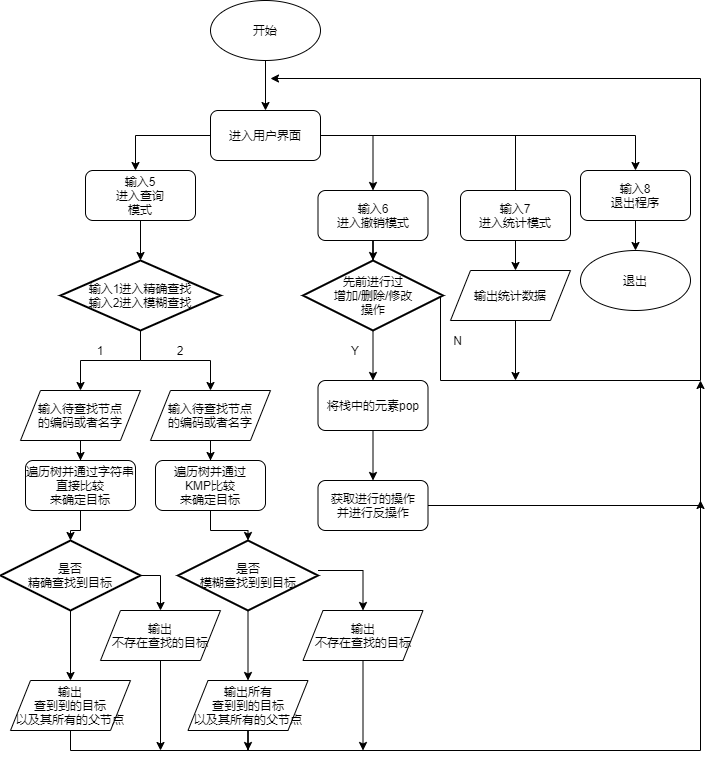
* 1. void Menu(Tree t, SqStack \*s);

//用户界面

t:学生管理系统树 s：撤回栈

1. **详细设计**

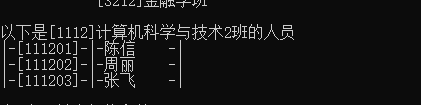
程序流程图



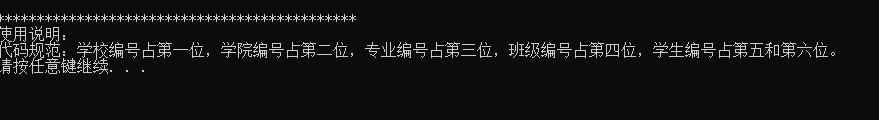
1. **用户手册**
2. 用户界面



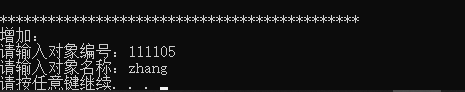
使用wasd以及回车来查看班级中的人

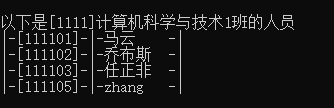


输入1展开编码说明

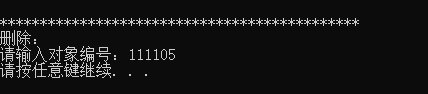


输入2进入增加用户模式

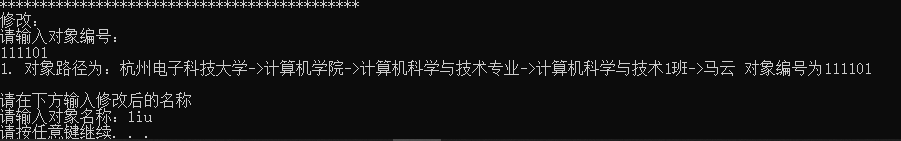




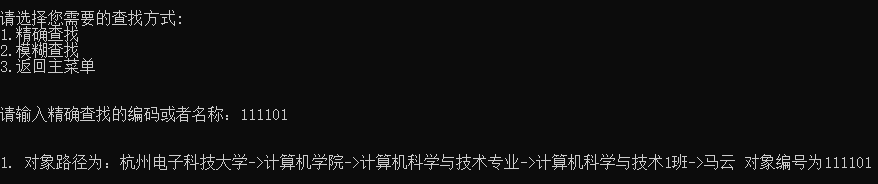
输入3进入删除用户模式



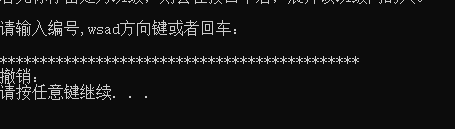
输入4进入修改用户模式



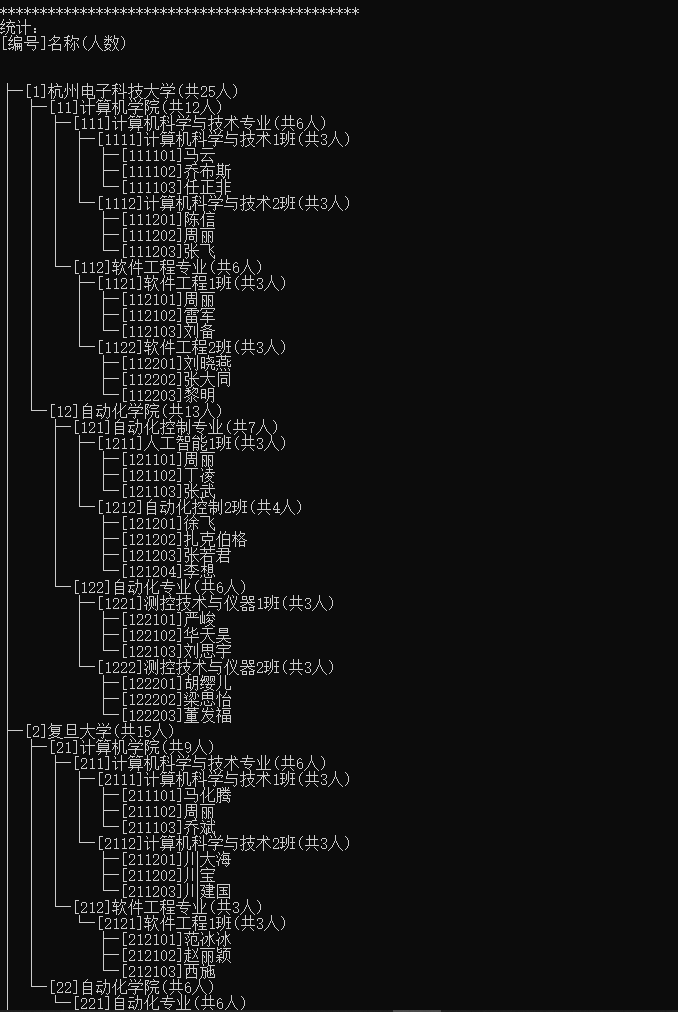
输入5进入查询用户模式



输入6撤销最近的一次修改



输入7输出所有学生并进行统计



输入8退出程序

**五 ．测试结果**

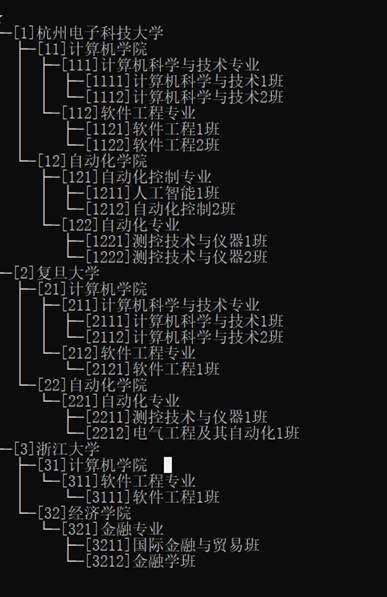
* 1. 通过自动生成或者手动输入生成乘客

1. 通过手动输入来生成乘客并启动电梯
2. 采用菜单的方式显示
   1. 打开程序后默认输出整个学生管理系统的状态

输入用例：无

期望结果：输出整颗树

实测结果：



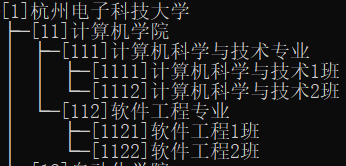
结论：测试通过

* 1. 用[编码]名字 的形式来表示节点

输入用例：无

期望结果：输出[编码]名字

实测结果：



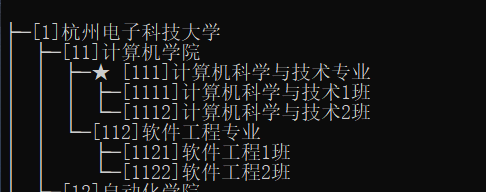
结论：测试通过

* 1. 用光标移动选择操作的对象

输入用例：无

期望结果： 光标★移动

实测结果：



结论：测试通过

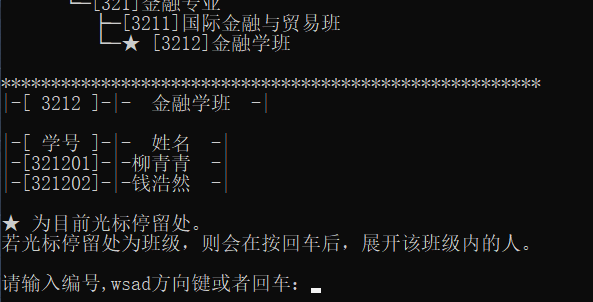
* 1. 选中班级后按回车键显示该班级中的学生 再按一次回车键回到班级

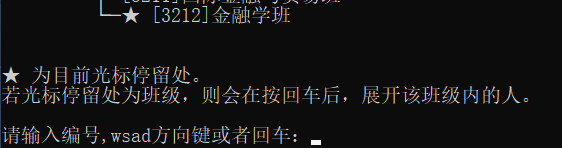
输入用例：无

* 1. 期望结果： 选中班级后按回车键显示该班级中的学生

再按一次回车键回到班级

实测结果：





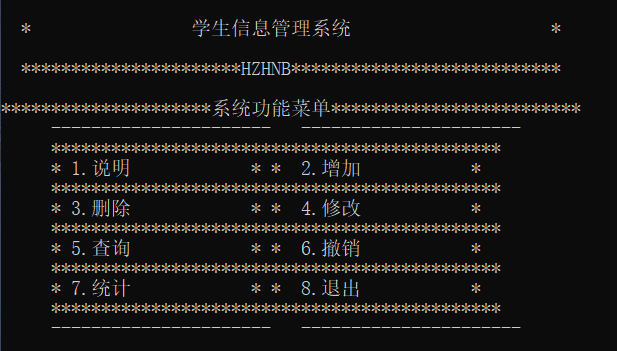
结论：测试通过

* 1. 用菜单提示用户操作

输入用例：无

期望结果： 输出菜单

实测结果：



结论：测试通过

1. 用六位编码表示节点
   1. 前一位数字代表学校编号

输入用例：无

期望结果： 用一位数字表示学校

实测结果：



结论：测试通过

* 1. 前二位数字代表学院编号

输入用例：无

期望结果： 用两位数字表示学院

实测结果：



结论：测试通过

* 1. 前三位数字代表专业编号

输入用例：无

期望结果： 三位数字代表专业编号

实测结果：



结论：测试通过

* 1. 前四位数字表示班级编号

输入用例：无

期望结果： 四位数字表示班级编号

实测结果：



结论：测试通过

* 1. 一共六位数字表示学生编号

输入用例：无

期望结果： 六位数字表示学生编号

实测结果：



结论：测试通过

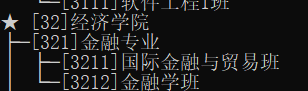
1. 添加信息
   1. 将光标移动到节点上输入2进入添加子节点模式 输入它的名称完成添加操作

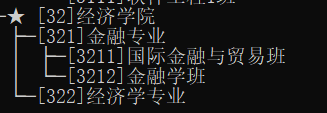
例如：将光标移到 杭州电子科技大学 上 输入2添加计算机学院

输入用例：选中[32]经济学院 在下面添加一个经济学专业

期望结果：操作后 [32]经济学院 在下面多了一个经济学专业

实测结果：





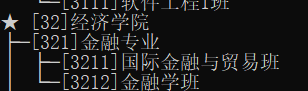
结论：测试通过

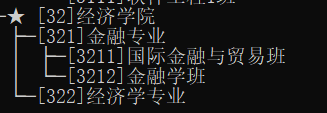
* 1. 添加时，会给新节点自动分配编号

输入用例：无

期望结果： 选中[32]经济学院 在下面添加一个经济学专业 会自动分配322号

实测结果：操作后 [32]经济学院 在下面多了一个[322]经济学专业





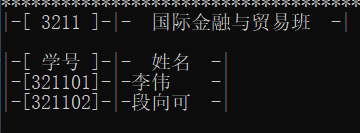
结论：测试通过

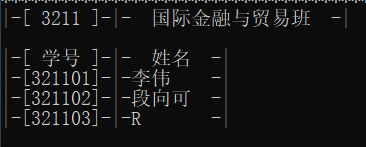
* 1. 添加学生时需要先输入回车键选中班级后 再输入2添加学生

输入用例：在 [3211]国际金融与贸易班下添加一位名字为R的学生

期望结果： [3211]国际金融与贸易班下多一个321103 R的学生

实测结果：





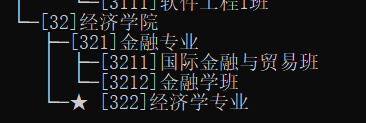
结论：测试通过

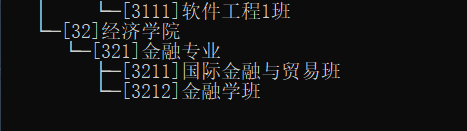
1. 删除信息
   1. 将光标移到待删除的节点上，输入3删除该节点

输入用例：删除[322]经济学专业

期望结果： [322]经济学专业被删除

实测结果：





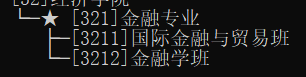
结论：测试通过

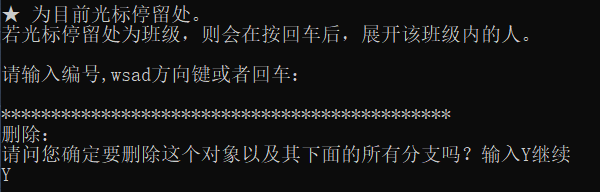
* 1. 若该节点存在子节点，则在删除时，会提示用户，是否确定删除

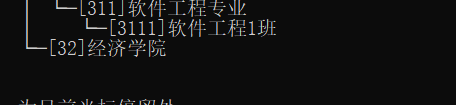
输入用例：无

期望结果： 删除[321]经济学专业

实测结果：有提示 并成功删除







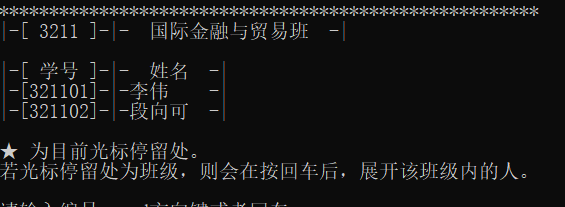
结论：测试通过

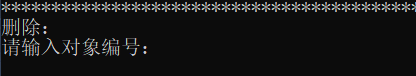
* 1. 删除学生需要指定添加学生时需要指定学生的学号

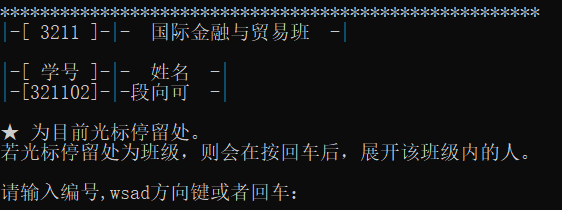
输入用例：删除 321101 李伟

期望结果： 3211班中321101 李伟学生被删除

实测结果：







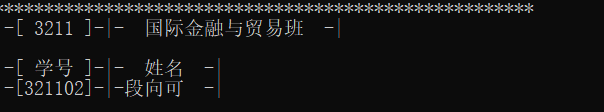
结论：测试通过

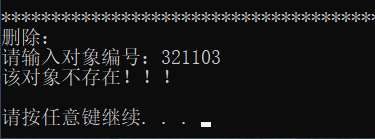
* 1. 删除不存在的学生时 会返回错误信息

输入用例：删除不存在的学生 321103

期望结果： 返回错误信息

实测结果：





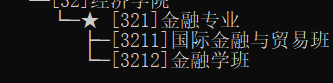
结论：测试通过

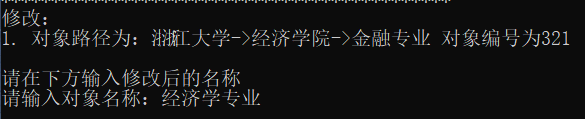
1. 修改信息
   1. 将光标移到待修改的节点上，输入4修改该节点

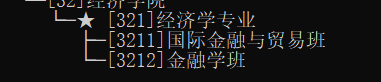
输入用例：修改[321]金融专业为[321]经济学专业

期望结果： [321]金融专业被修改为[321]经济学专业

实测结果：







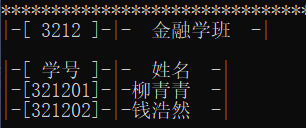
结论：测试通过

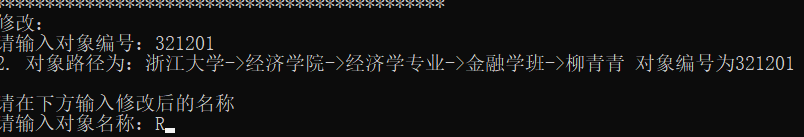
* 1. 修改学生需要指定添加学生时需要指定学生的学号

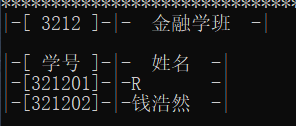
输入用例：修改[321201]柳青青 为 [321201]R

期望结果： [321201]学生名字变为 R

实测结果：







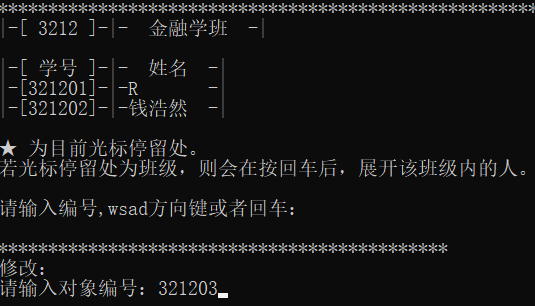
结论：测试通过

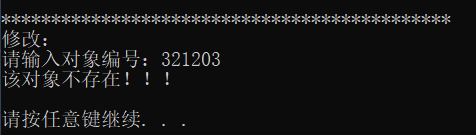
* 1. 修改不存在的学生时 会返回错误信息

输入用例：修改不存在的学生321203

期望结果： 返回错误信息

实测结果：





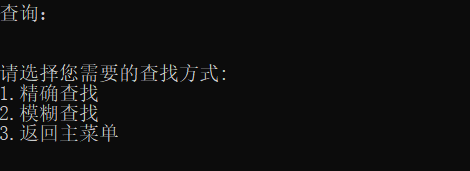
结论：测试通过

1. 查找符合要求的信息
   1. 输入5进入查找模式

输入用例：输入5

期望结果： 进入查询模式

实测结果：



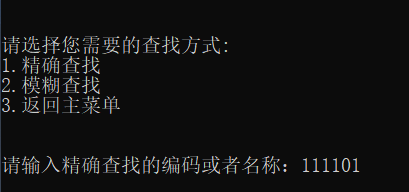
结论：测试通过

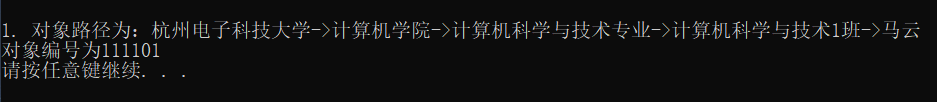
* 1. 输入1进入精确查找模式 通过输入编码 或者输入姓名 进行精确查找

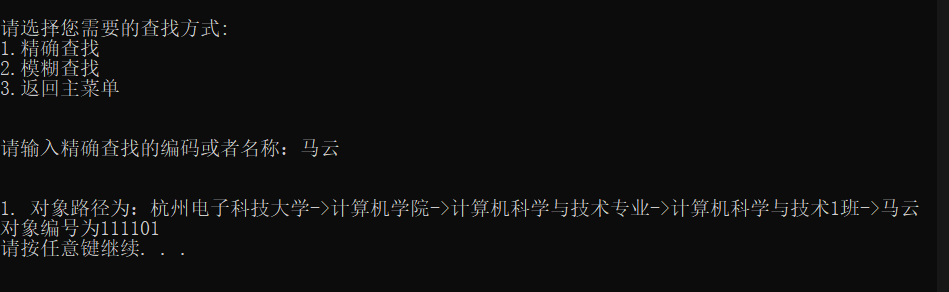
输入用例：查找 111101 以及查找马云

期望结果： 返回111101 查找的结果 以及 马云的查找结果

实测结果：







结论：测试通过

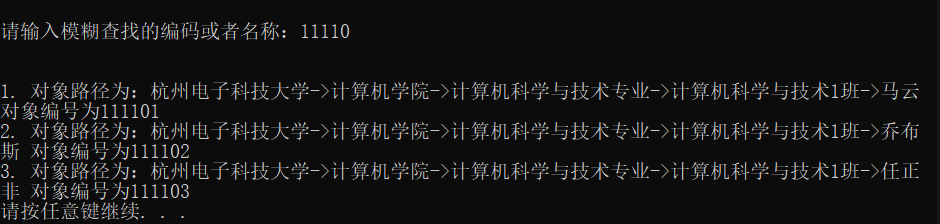
* 1. 输入2进入模糊查找模式 通过输入部分编码或者部分姓名 模糊查找

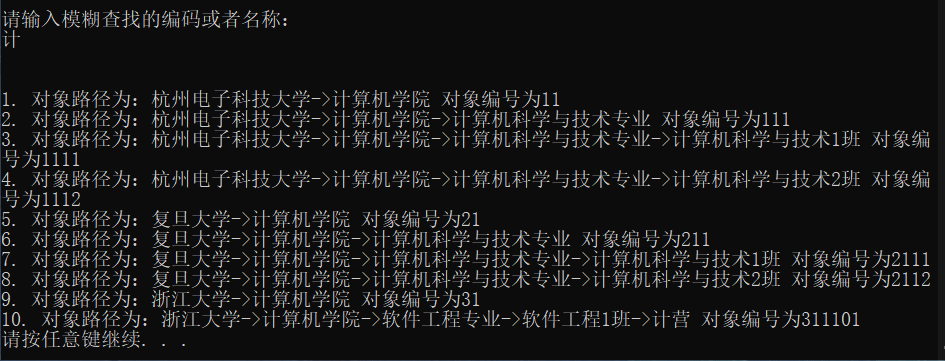
将所有包含有关键字的信息全部输出

输入用例：11110 查找的结果 以及 计 的查找结果

期望结果： 返回所有含有11110的编码的节点 返回所有名字包含 计 的节点

实测结果：





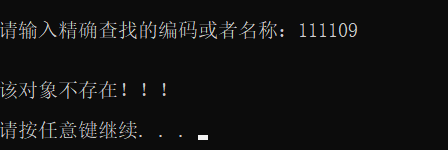
结论：测试通过

* 1. 若没有找到符合的节点 则显示错误信息 节点不存在

输入用例：查询不存在的节点111109

期望结果： 返回节点不存在的信息

实测结果：



结论：测试通过

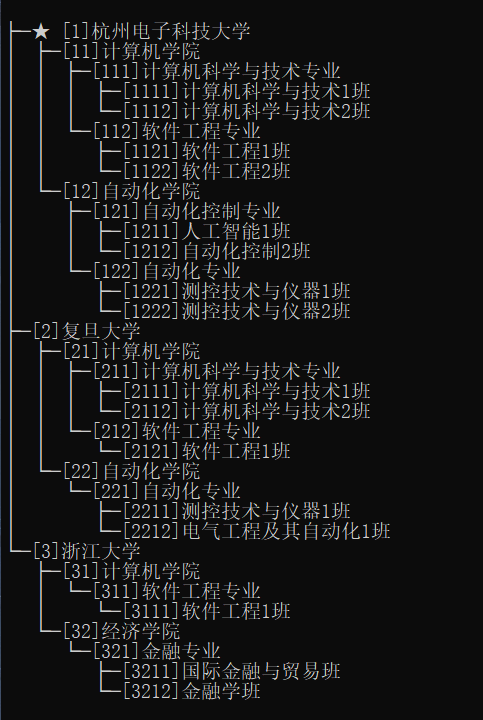
1. 撤回
   1. 可以在进行 增加/删除/修改操作 输入7后执行 撤销操作来还原被修改的信息

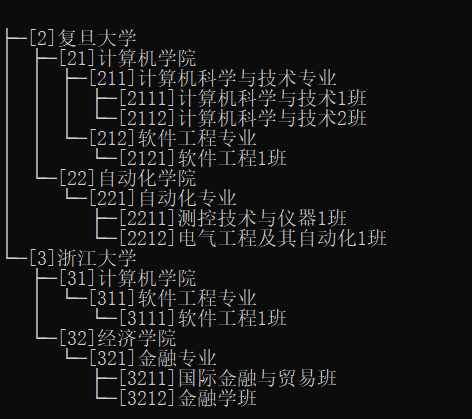
输入用例：删除[1]杭州电子科技大学后输入7恢复[1]杭州电子科技大学以及

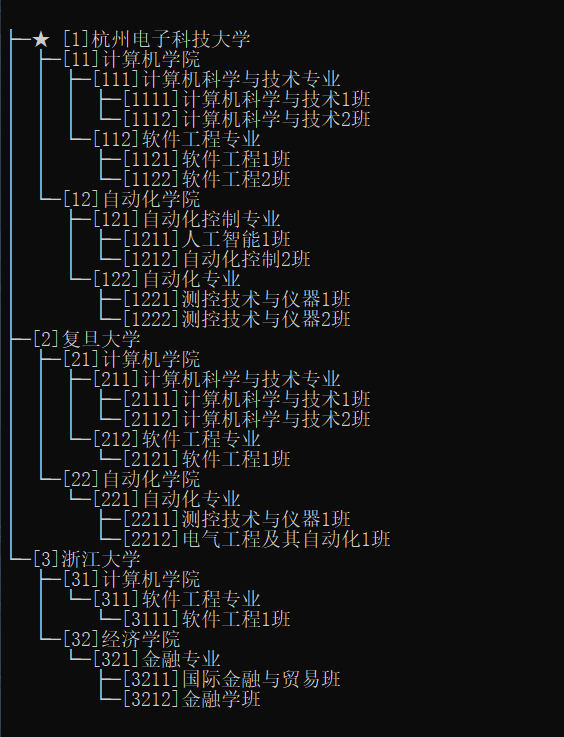
它的子节点

期望结果： 删除[1] 杭州电子科技大学后撤销恢复

实测结果：







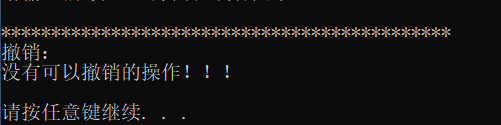
结论：测试通过

* 1. 若未进行过增加/删除/修改操作时 进行撤销操作 会提示错误信息

输入用例：打开程序后输入6

期望结果： 返回错误信息： 没有可以撤销的操作 ！

实测结果：



结论：测试通过

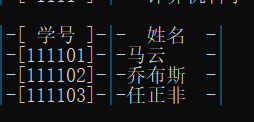
* 1. 若进行删除后的撤销操作，撤销可以将原节点返回原位置，保证代码的有序性

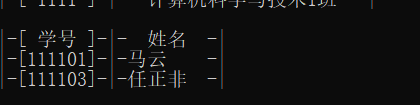
输入用例：删除111102后撤销恢复

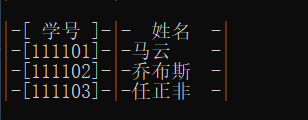
期望结果： 111101 111102 111103 中删除了111102 则变为111101 111103

则进行撤销后恢复111101 111102 111103状态

实测结果：







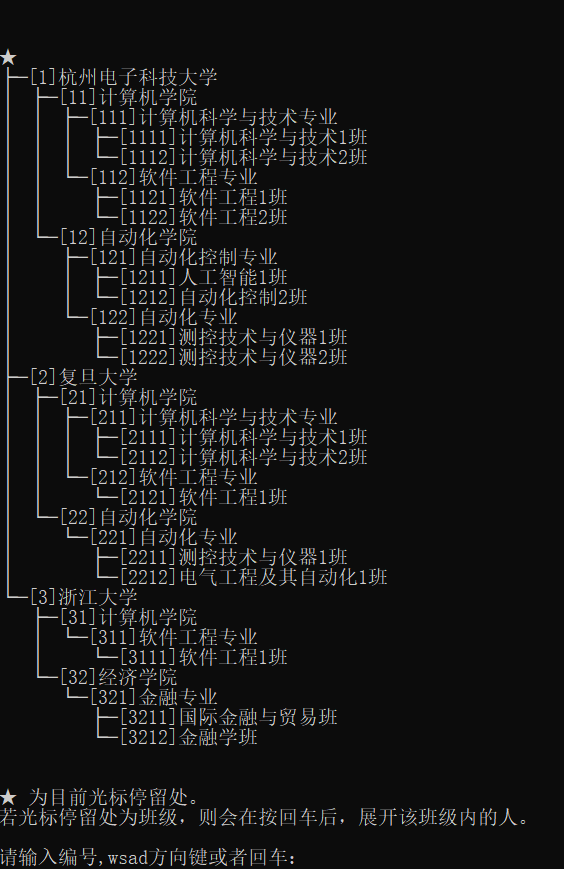
结论：测试通过

1. 导入数据
   1. 可以通过txt文件导入的形式导入默认数据

输入用例：input.txt

期望结果： 打开程序后显示默认导入的数据

实测结果：



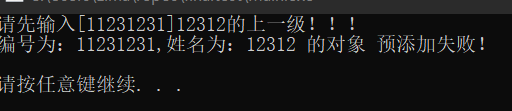
结论：测试通过

* 1. 导入 数据不符合格式 会报错

输入用例：11231231 12312

期望结果： 显示导入错误

实测结果：



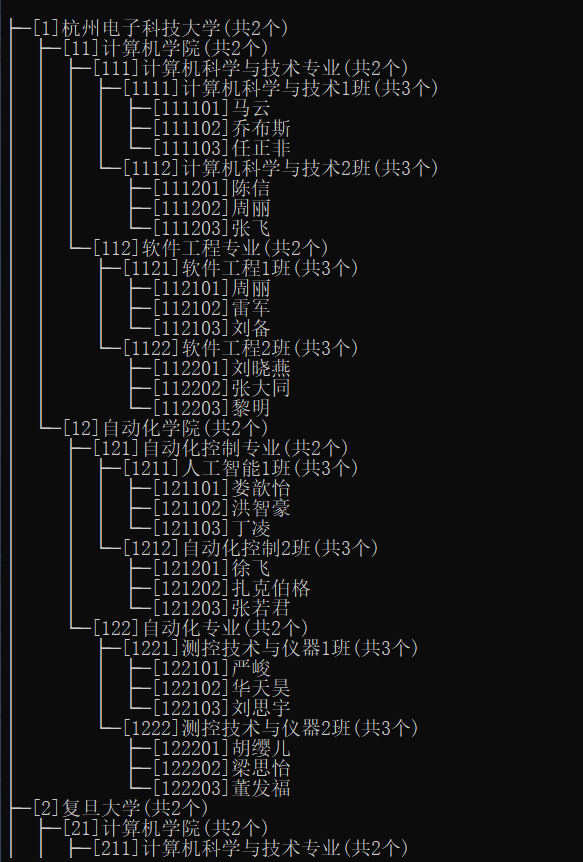
结论：测试通过

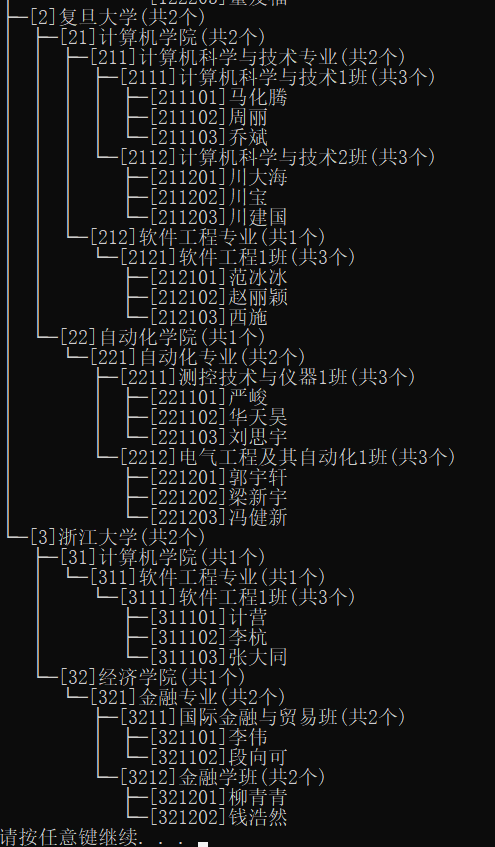
1. 统计数据
   1. 在菜单下输入7进入统计模式

输入用例：7

期望结果： 输出学生管理系统中的全部数据并统计

实测结果：





结论：测试通过

* 1. 统计并输出 整个系统中的全部信息

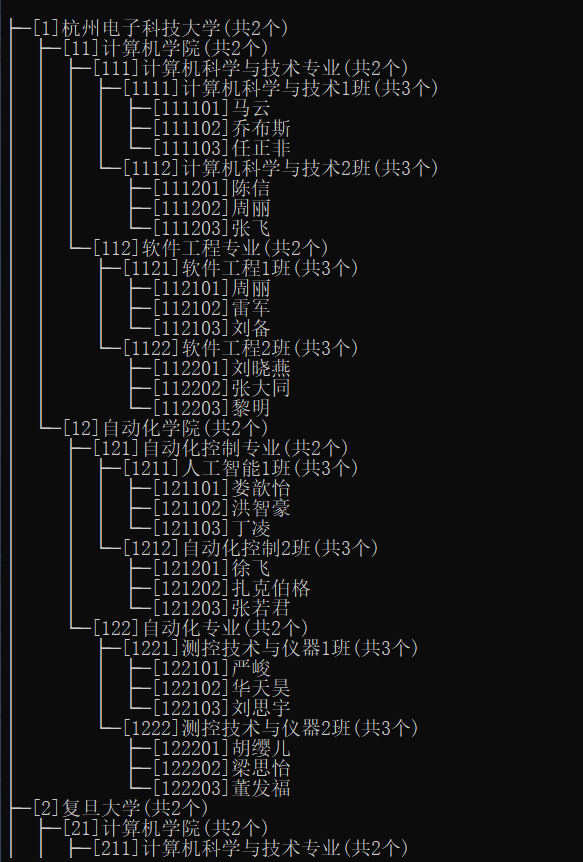
各个学校下学院的个数 各个学院下专业的个数

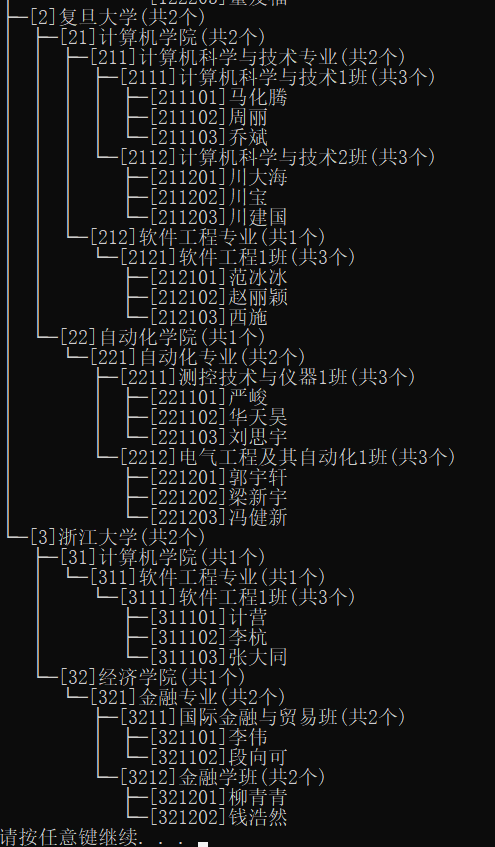
各个专业下班级下的个数 各个班级下学生的个数

输入用例：7

期望结果： 输出统计数据

实测结果：





结论：测试通过

**六 ． 附录**

HashMap.h

保存哈希表的数据结构以及函数

HeadFile.h

头文件与全局常量和全局变量定义以及初始化

input.txt

导入数据

Main.c

主函数

Menu.h

控制前端界面与用户交互输出

SqStack.h

保存栈的数据结构以及其函数

Tools.h

工具函数类

Tree.h

保存树的数据结构以及其函数