**公司员工管理**

**一 目的**

**设计一个虚基类Staff（员工），包含编号、姓名和年龄保护数据成员以及相关的成员函数；由Staff派生出工程师类Engineer，包含专业和职称保护数据成员以及相关的成员函数；由Staff派生出领导类Leader，包含职务和部门保护数据成员以及相关成员函数；然后由Engineer和Leader类派生出主任工程师类Chairman。利用文件处理方式实现对公司人员（包括工程师、领导和主任工程师）进行管理，具有增加数据、更新数据、查询数据、删除数据以及重组文件的功能。**

**二 需求分析**

**1、功能分析**

1. 显示员工数据信息
2. 添加员工数据信息
3. 查找员工数据信息
4. 修改员工数据信息
5. 删除员工数据信息
6. **显示功能**

函数原型：showStaff()

说明：输出员工的信息，包括编号、姓名、年龄、性别等。

1. **添加功能**

函数原型：addStaff()

说明：任意添加职工信息、工程师信息、领导信息和主任工程师信息。

1. **查找功能**

函数原型：searchStaff()

说明：可根据不同种类员工的编号或姓名查询所有的信息，如未找到提示相应信息，如果找到，则显示所有信息。

1. **修改功能**

函数原型：updateStaff()

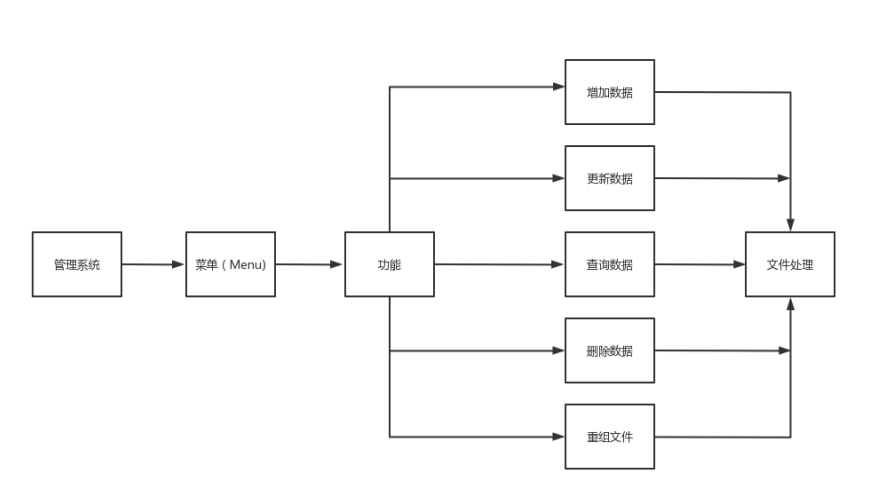
说明：对已经添加的记录（职工信息、工程师信息、领导信息和主任工程师信息）进行修改。

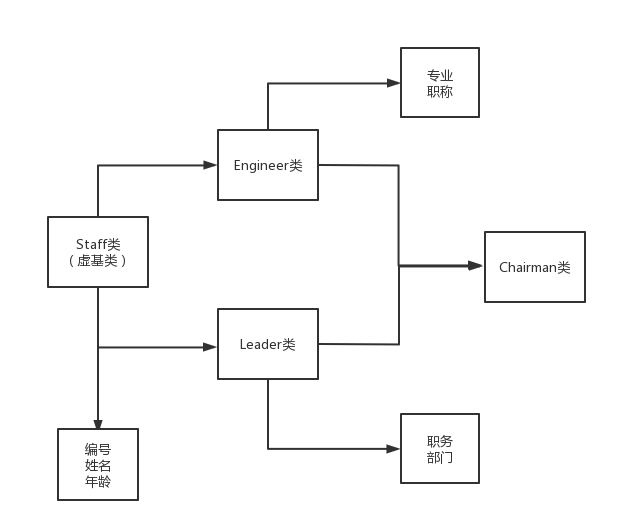
1. **删除功能**

函数原型：deleteStaff()

说明：对已经添加的记录（工程师信息、领导信息和主任工程师信息）进行删除。

**三 概要设计**

**结构框图**



**四 详细设计**

**1、类的设计**

//员工类

class Staff {

protected:

string id\_; //编号

string name\_; //姓名

int age\_; //年龄

Sex sex\_; //性别

public:

Staff() :age\_(0) {

}

virtual ~Staff() {

}

//编辑员工信息

virtual void edit() = 0;

//打印员工信息

virtual void print() const = 0;

//将员工信息从文件流输入

virtual void load(ifstream& input) = 0;

//将员工信息输出到文件流

virtual void save(ofstream& output) const = 0;

//获取员工类型

virtual StaffType type() const = 0;

//获取员工编号

const string& getID() const {

return id\_;

}

//设置员工编号

void setID(const string& id) {

id\_ = id;

}

//获取员工姓名

const string& getName() const {

return name\_;

}

void editStaff() {

cout << "编号：" << id\_ << endl;

cout << "类别：" << getTypeName() << endl;

cout << "姓名：";

cin >> name\_;

cout << "性别：（输入提示，1代表男性，2代表女性）";

cin >> sex\_;

cout << "年龄：";

cin >> age\_;

}

void printStaff() const {

cout << "编号：" << id\_ << " ";

cout << "类别：" << getTypeName() << " ";

cout << "姓名：" << name\_ << " ";

cout << "性别：" << sex\_ << " ";

cout << "年龄：" << age\_ << " ";

}

void loadStaff(ifstream& input) {

input >> id\_;

input >> name\_;

sex\_.load(input);

input >> age\_;

}

void saveStaff(ofstream& output) const {

output << id\_ << " ";

output << name\_ << " ";

sex\_.save(output);

output << age\_ << " ";

}

string getTypeName() const {

switch (type()) {

case ENGINEER:

return "工程师";

case LEADER:

return "领导";

case CHAIRMAN:

return "主任工程师";

default:

return "未知";

}

}

};

//工程师类，虚继承

class Engineer : public virtual Staff {

protected:

string major\_; //专业

string title\_; //职称

public:

Engineer() {

}

virtual ~Engineer() {

}

virtual void edit() {

editStaff();

editEngineer();

}

virtual void print() const {

printStaff();

printEngineer();

cout << endl;

}

virtual void load(ifstream& input) {

loadStaff(input);

loadEngineer(input);

}

virtual void save(ofstream& output) const {

saveStaff(output);

saveEngineer(output);

}

virtual StaffType type() const {

return ENGINEER;

}

void editEngineer() {

cout << "专业：";

cin >> major\_;

cout << "职称：";

cin >> title\_;

}

void printEngineer() const {

cout << "专业：" << major\_ << " ";

cout << "职称：" << title\_ << " ";

}

void loadEngineer(ifstream& input) {

input >> major\_;

input >> title\_;

}

void saveEngineer(ofstream& output) const {

output << major\_ << " ";

output << title\_ << " ";

}

};

//领导类，虚继承

class Leader : public virtual Staff {

protected:

string job\_; //职务

string department\_; //部门

public:

Leader() {

}

virtual ~Leader() {

}

virtual void edit() {

editStaff();

editLeader();

}

virtual void print() const {

printStaff();

printLeader();

cout << endl;

}

virtual void load(ifstream& input) {

loadStaff(input);

loadLeader(input);

}

virtual void save(ofstream& output) const {

saveStaff(output);

saveLeader(output);

}

virtual StaffType type() const {

return LEADER;

}

void editLeader() {

cout << "职务：";

cin >> job\_;

cout << "部门：";

cin >> department\_;

}

void printLeader() const {

cout << "职务：" << job\_ << " ";

cout << "部门：" << department\_ << " ";

}

void loadLeader(ifstream& input) {

input >> job\_;

input >> department\_;

}

void saveLeader(ofstream& output) const {

output << job\_ << " ";

output << department\_ << " ";

}

};

//主任工程师类，多重继承

class Chairman : public Leader, public Engineer {

public:

Chairman() {

}

virtual ~Chairman() {

}

virtual void edit() {

editStaff();

editLeader();

editEngineer();

}

virtual void print() const {

printStaff();

printLeader();

printEngineer();

cout << endl;

}

virtual void load(ifstream& input) {

loadStaff(input);

loadLeader(input);

loadEngineer(input);

}

virtual void save(ofstream& output) const {

saveStaff(output);

saveLeader(output);

saveEngineer(output);

}

virtual StaffType type() const {

return CHAIRMAN;

}

};

1. **功能模块设计**

//展示员工信息

void showStaffs() const {

cout << "#显示员工信息#" << endl;

for ( int i = 0; i < staffs\_.size(); ++i) {

Staff\* employee = staffs\_[i];

employee->print();

}

}

//添加员工信息

void addStaff() {

try {

string id = generateStaffID();

Staff\* employee = generateStaff();

if (employee) {

employee->setID(id);

cout << "#添加员工信息#" << endl;

employee->edit();

staffs\_.push\_back(employee);

save();

cout << "#新添员工信息#" << endl;

employee->print();

}

} catch (StaffException& exception) {

cout << exception.show() << endl;

}

}

//寻找员工

void searchStaff() {

while (1) {

cout << "#查找员工信息#" << endl;

cout << " 1 按编号查找" << endl;

cout << " 2 按姓名查找" << endl;

cout << " 0 返回" << endl;

int option;

cin >> option;

switch (option) {

case 1:

searchStaffByID();

break;

case 2:

searchStaffByName();

break;

case 0:

return;

}

}

}

//更新员工信息

void updateStaff() {

cout << "#修改员工信息#" << endl;

cout << "输入员工编号：";

string id;

cin >> id;

Staff\* employee = findStaffByID(id);

if (employee) {

cout << "#当前员工信息#" << endl;

employee->print();

cout << endl;

cout << "重新编辑员工信息：" << endl;

employee->edit();

save();

cout << "修改成功！" << endl;

} else {

cout << "未找到相关记录！" << endl;

}

}

//删除员工

void deleteStaff() {

cout << "#删除员工信息#" << endl;

cout << "输入员工编号：";

string id;

cin >> id;

if (removeStaffByID(id)) {

save();

cout << "删除成功！" << endl;

} else {

cout << "未找到相关记录！" << endl;

}

}

//按编号寻找员工信息

void searchStaffByID() {

cout << "#按编号查找#" << endl;

cout << "输入员工编号：";

string id;

cin >> id;

Staff\* employee = findStaffByID(id);

if (employee) {

employee->print();

} else {

cout << "未找到相关记录！" << endl;

}

}

//按姓名寻找员工信息

void searchStaffByName() {

cout << "#按姓名查找#" << endl;

cout << "输入员工姓名：";

string name;

cin >> name;

int position = findStaffByName(name);

if (position != -1) {

do {

staffs\_[position]->print();

position = findStaffByName(name, position + 1);

} while (position != -1);

} else {

cout << "未找到相关记录！" << endl;

}

}

//得到员工编号

Staff\* findStaffByID(string id) const {

for (int i = 0; i < staffs\_.size(); i++) {

Staff\* employee = staffs\_[i];

if (employee->getID() == id) {

return employee;

}

}

return NULL;

}

//得到员工姓名

int findStaffByName(string name, int position = 0) const {

for (int i = position; i < staffs\_.size(); i++) {

Staff\* employee = staffs\_[i];

if (employee->getName() == name) {

return (int)i;

}

}

return -1;

}

//通过编号删除员工信息

bool removeStaffByID(string id) {

for (StaffArray::iterator iter = staffs\_.begin(); iter != staffs\_.end(); ++iter) {

Staff\* employee = \*iter;//对迭代器解引用，并访问实际对象

if (employee->getID() == id) {

delete employee;

staffs\_.erase(iter);//删除

return true;

}

}

return false;

}

//添加员工编号

string generateStaffID() {

cout << "#创建员工编号#" << endl;

cout << "编号：";

string id;

cin >> id;

if (findStaffByID(id)) {//判断编号是否存在

throw StaffException("该编号已存在，创建失败！");

}

return id;

}

//添加员工类型

Staff\* generateStaff() {

cout << "#添加员工类型#" << endl;

cout << " 1 工程师" << endl;

cout << " 2 领导" << endl;

cout << " 3 主任工程师" << endl;

cout << " 0 返回" << endl;

int option;

cin >> option;

Staff\* employee = NULL;

switch (option) {

case 1:

employee = createStaff(ENGINEER);

break;

case 2:

employee = createStaff(LEADER);

break;

case 3:

employee = createStaff(CHAIRMAN);

break;

}

return employee;

}

//将文件数据加载到内存

void load() {

ifstream input;

input.open("staffs.txt", ios::in);

if (input.is\_open()) {

int count;

input >> count;

for (int i = 0; i < count; i++) {

int type;

input >> type;

Staff\* employee = createStaff((StaffType)type);

if (employee) {

employee->load(input);

staffs\_.push\_back(employee);

} else {

throw StaffException("格式错误！");

}

}

}

}

//将内存数据存储到文件

void save() const {

ofstream output;

output.open("staffs.txt", ios::out);

if (output.is\_open()) {

output << staffs\_.size() << endl;

for (int i = 0; i < staffs\_.size(); i++) {

Staff\* employee = staffs\_[i];

output << employee->type() << " ";

employee->save(output);

output << endl;

}

}

}

Staff\* createStaff(StaffType type) {

switch (type) {

case ENGINEER:

return new Engineer();

case LEADER:

return new Leader();

case CHAIRMAN:

return new Chairman();

}

return NULL;

}

void free() {

while (!staffs\_.empty()) {

delete staffs\_.back();

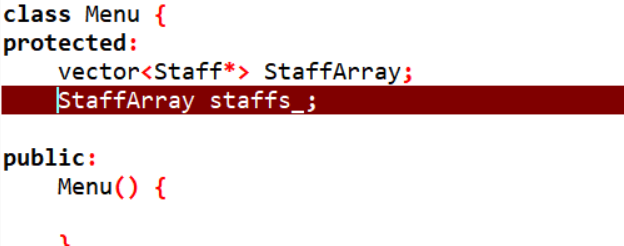
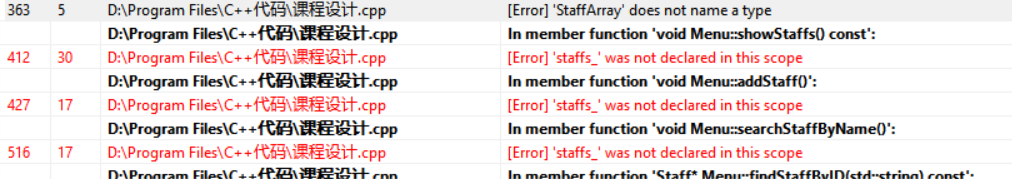
staffs\_.pop\_back();//将数据存进容器

}

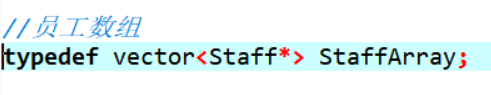
}

**五 调试分析**

1. **我一开始将员工动态数组放在menu类中**

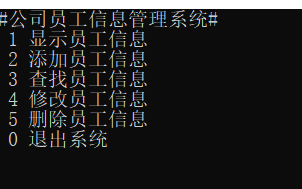


因此，我用typedef在类外定义员工数组

在直接用创建staffs\_对象。

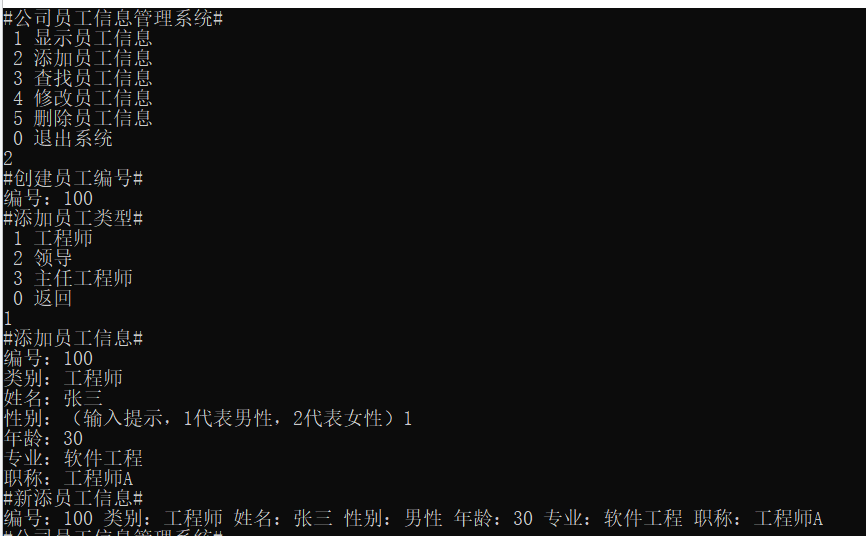
**六 测试结果**

**1、菜单页面**

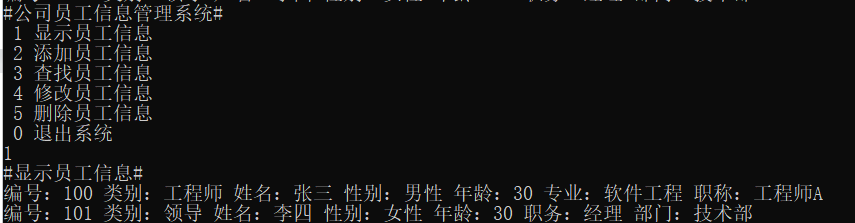


**2、添加员工信息**

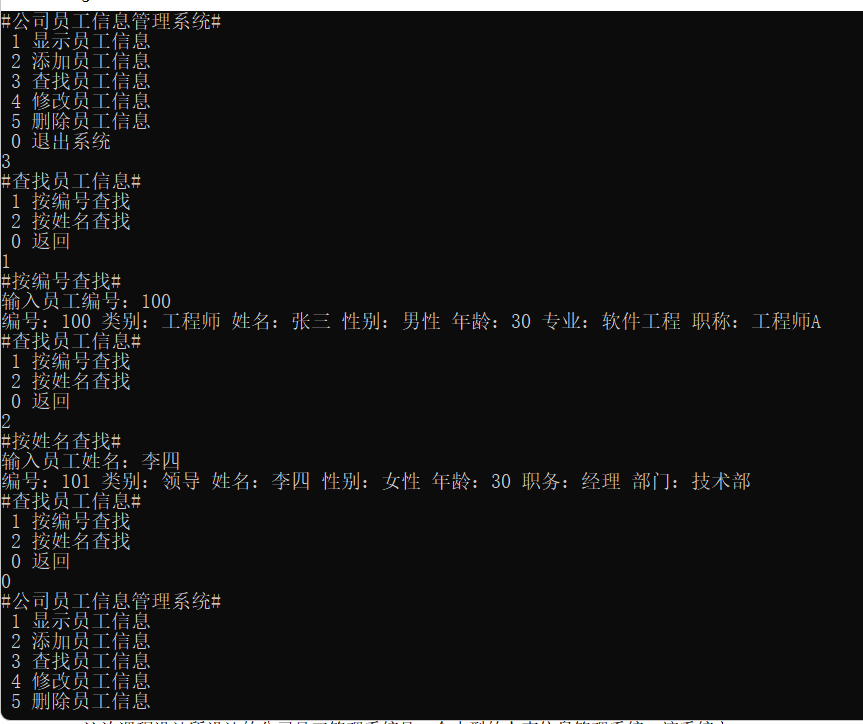
依次输入需要的信息，添加完成后显示新添加的员工信息。



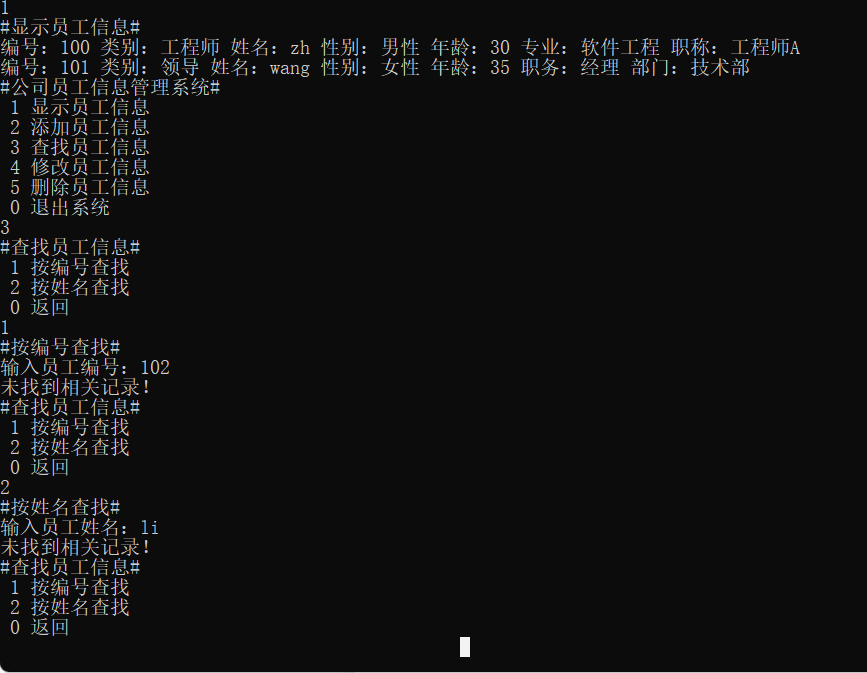
1. **显示员工信息**



1. **查找员工**

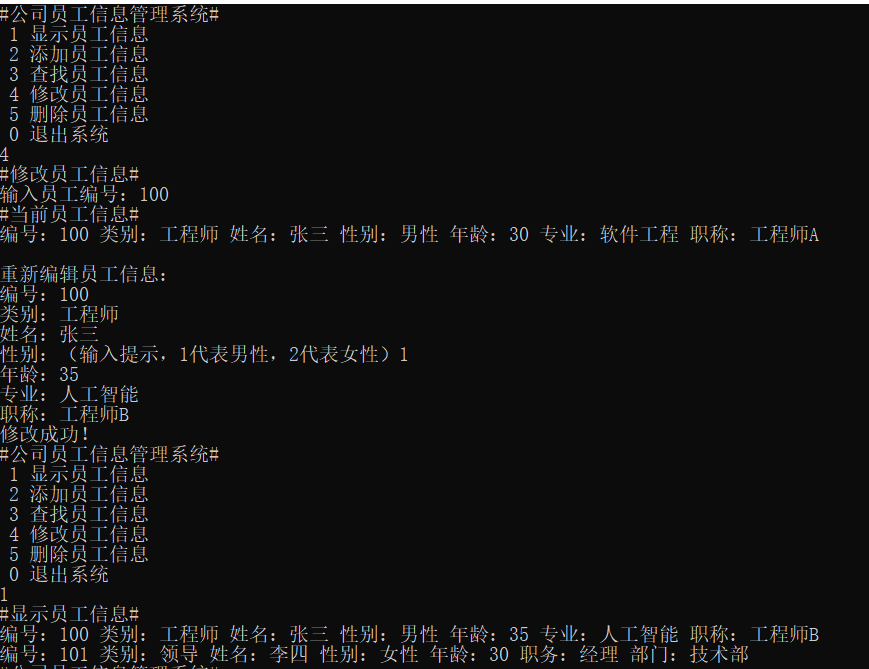


员工不存在时：

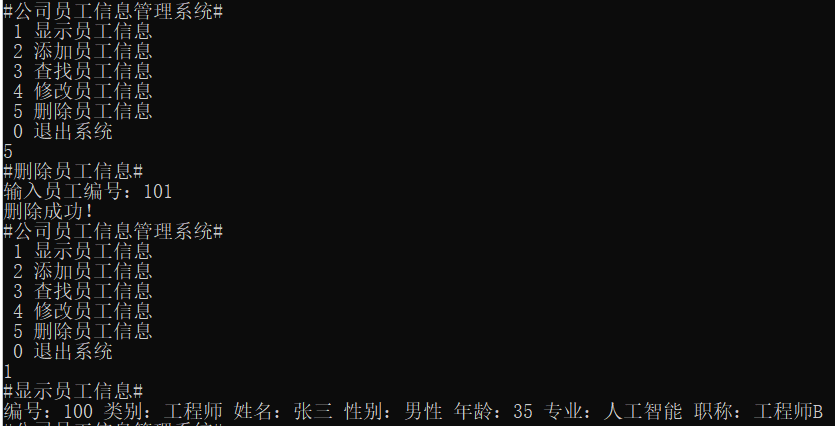


1. **修改员工信息**

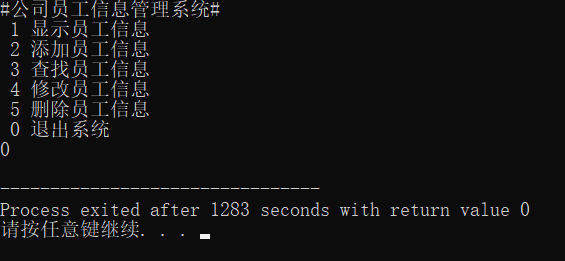
通过员工编号修改员工信息，修改前显示当前信息，重新编辑信息。



**6、删除员工信息**



**7、退出系统**



**七 用户使用说明**

1. 在使用程序时根据菜单选择不同功能。
2. 不能出现重复编号，在输入相同编号时，系统会提示。
3. 用户应依照程序提示进行数据的输入，否则可能会导致程序死循环。

**八 课程设计总结**

这次课程设计所设计的公司员工管理系统是一个小型的人事信息管理系统，该系统实现了职工信息的添加、删除、修改、显示、查询等功能，并且运用了文件操作技术，可以实现数据的永久性保存。

通过这次课程设计，我拓宽了知识面，锻炼了能力，综合素质得到较大提高。安排课程设计的基本目的，在于通过理论与实际的结合、人与人的沟通，进一步提高思想觉悟。尤其是观察、分析和解决问题的实际工作能力，以便培养成为能够主动适应社会主义现代化建设需要的高素质的复合型人才。作为整个学习体系的有机组成部分，课程设计的一个重要功能，在于运用学习成果，检验学习成果。运用学习成果，把课堂上学到的系统化的理论知识，尝试性地应用于实际设计工作，并从理论的高度对设计工作的现代化提出一些有针对性的建议和设想。检验学习成果，看一看课堂学习与实际工作到底有多大距离，并通过综合分析，找出学习中存在的不足，以便为完善学习计划，改变学习内容与方法提供实践依据。对我们计算机专业的本科生来说，实际能力的培养至关重要，而这种实际能力的培养单靠课堂教学是远远不够的，必须从课堂走向实践。这也是一次预演和准备毕业设计工作。通过课程设计，让我们找出自身状况与实际需要的差距，并在以后的学习期间及时补充相关知识，为求职与正式工作做好充分的知识、能力准备，从而缩短从校园走向社会的心理转型期。课程设计促进了我系人才培养计划的完善和课程设置的调整。课程设计达到了专业学习的预期目的。在完成课程设计之后，我感到不仅实际动手能力有所提高，更重要的是通过对软件开发流程的了解，进一步激发了我对专业知识的兴趣，并能够结合实际存在的问题在专业领域内进行更深入的学习。