|  |  |  |  |
| --- | --- | --- | --- |
| **学号** | **2106410224** | **成绩** |  |





**课程设计说明书**

**设计名称 C++课程设计**

**设计题目 职工工资管理系统设计**

**设计时间 2022.6.6—2022.6.10**

**学 院 信息与控制工程学院**

**专 业 计算机科学与技术**

**班 级 计算机2102**

**姓 名 洪利达**

**指导教师 宋晓宇**

**2022 年 6 月 6 日**

C++课程设计说明书

**目录**

1. 设计任务说明

2. 开发工具

3. 设计思路

4. 系统效果图（每个图要有文字说明，可附多图。）

5. 源代码

6. 参考文献

7. 设计体会

1. **设计任务说明：**

**题目 5：职工工资管理系统设计**

功能：职工工资管理系统应包含职工的全部信息。

每个职工是一条记录，包括职工号、 姓名、月工资、岗位、年龄、性别等信息。公司主要有 4 类人员：经理、技术员、销售员、 销售经理。

**不同人员月工资的计算办法为：**

1. 经理：固定月薪为 8000；
2. 技术员：工作时间×小时工资（100 元每小时）；
3. 销售员：销售额×4%提成；

4）销售经理：底薪（5000）＋所辖部门销售额总额×0.5%。

**系统要求实现以下功能：**

1. 数据录入：输入各种数据；
2. 数据统计：各销售经理的工资计算及最终按工资进行的冒泡排序；
3. 查找功能：完成按职工号或姓名查找职工的相关记录，并显示；
4. 信息删除与修改——输入职工号，删除该职工的信息；
5. 数据备份：把相关数据写入文件。
6. **开发工具：**

Microsoft Visual Studio 2022

1. **设计思路：**

（1）人机交互界面

使用cout和cin输入输出函数设计出主界面交互菜单，通过switch case语句判断输入的选项值，执行相对应得子函数。

用户推出菜单选项之后，需要使用C++语言文件操作，保存已经录入的全部员工信息和工资。程序下次启动时可以直接查询上次保存的数据。

（2）添加员工信息

设计为单独的子函数供主函数调用。

定义一个结构体类型，包括多个成员变量（编号，姓名，性别，年龄，工资等），并用该类型定义一个全局的结构体数组，供外部录入员工信息时存储。

定义一个全局变量累存储当前员工的个数。

添加员工信息时，需要从外部读入员工的编号，姓名，性别，年龄和工作时间等，在获取编号时需要判断是否输入重复，如果编号相同则重新输入，员工数量加一，并根据当前员工个数将其保存在结构体数组对应序号的元素中。

在录入员工信息时可以选择录入人数来实现一次录入多名人员信息。

（3）显示全部员工的信息

通过对当前结构体数组进行遍历，并显示输出

（4）排序功能

利用选择排序或者冒泡排序法，将当前结构体数组根据元素中计算实发工资变量，从小到大排序或者从大到小。然后对当排序完成的结构体数组进行遍历显示输出，从而达到按实发工资排序的效果。

也可以根据编号的大小来进行升序或降序的排列。

（5）查找单个员工

根据输入需要查找员工的编号或姓名，遍历当前结构体数组，判断是否含有一致编号的元素，若有则显示输出对应的员工信息，否则显示无此人。

（6）删除单个员工

输入需要删除员工的编号，遍历当前结构体数组，判断是否含有一致编号的元素，若有则从该元素之后依次将下一个元素赋值到上一个元素，直至最后一个赋值完成，最后当前员工个数变量减1。

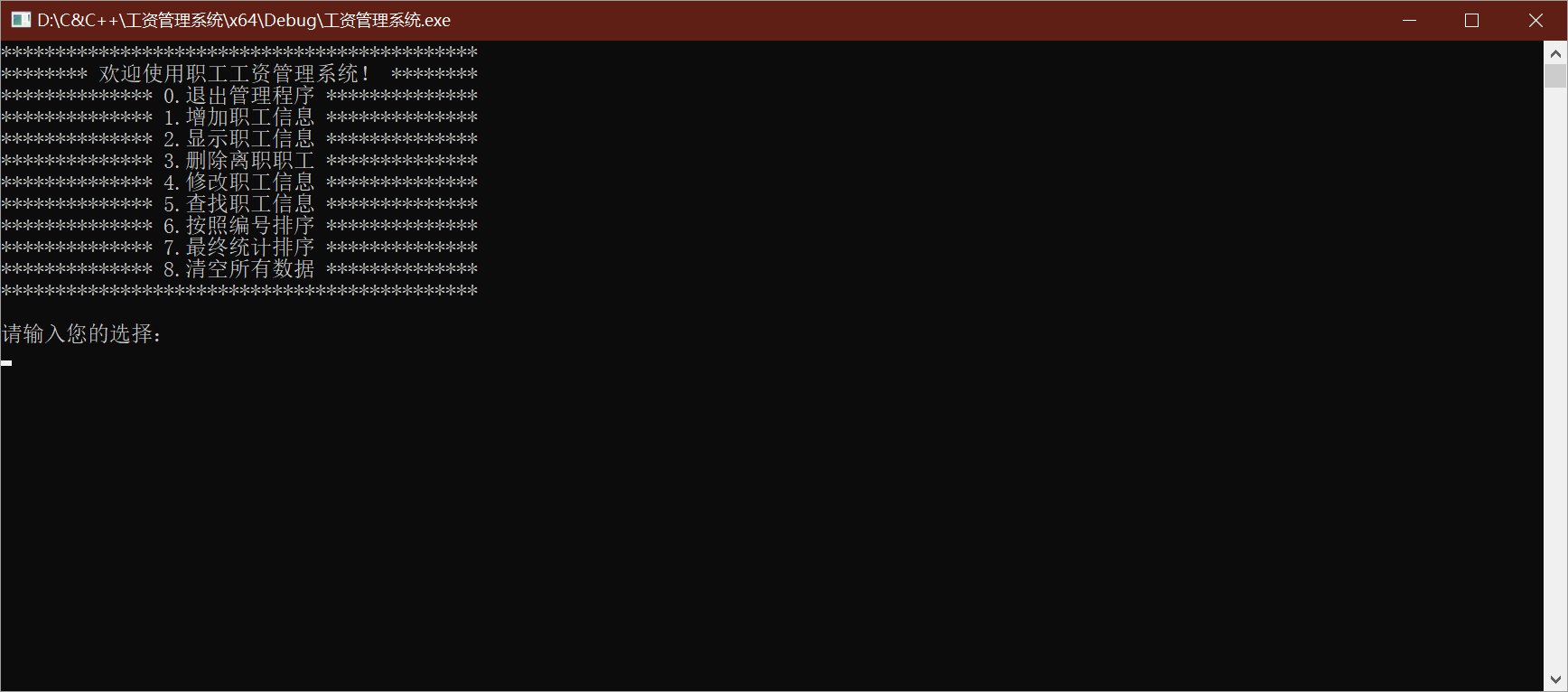
（7）修改员工信息

输入需要修改学员工的编号，遍历当前结构体数组，判断是否含有一致编号的元素，若有则依次从外部读入用户所修改的值，保存到对应成员变量中。

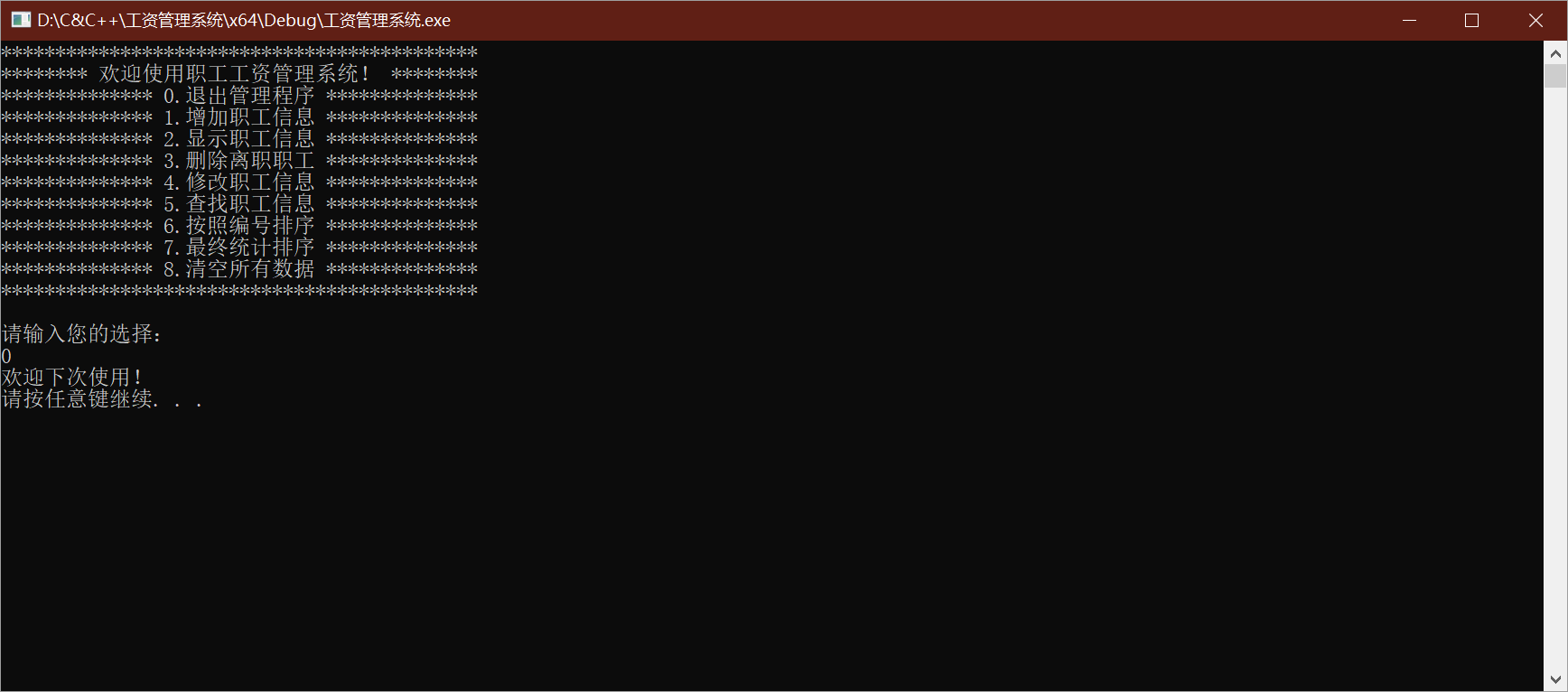
1. 清空所有数据

通过对文件的操作来实现所存储文件的内容清除，在选择清除之前，我们可以通过再次确认来避免误操作。

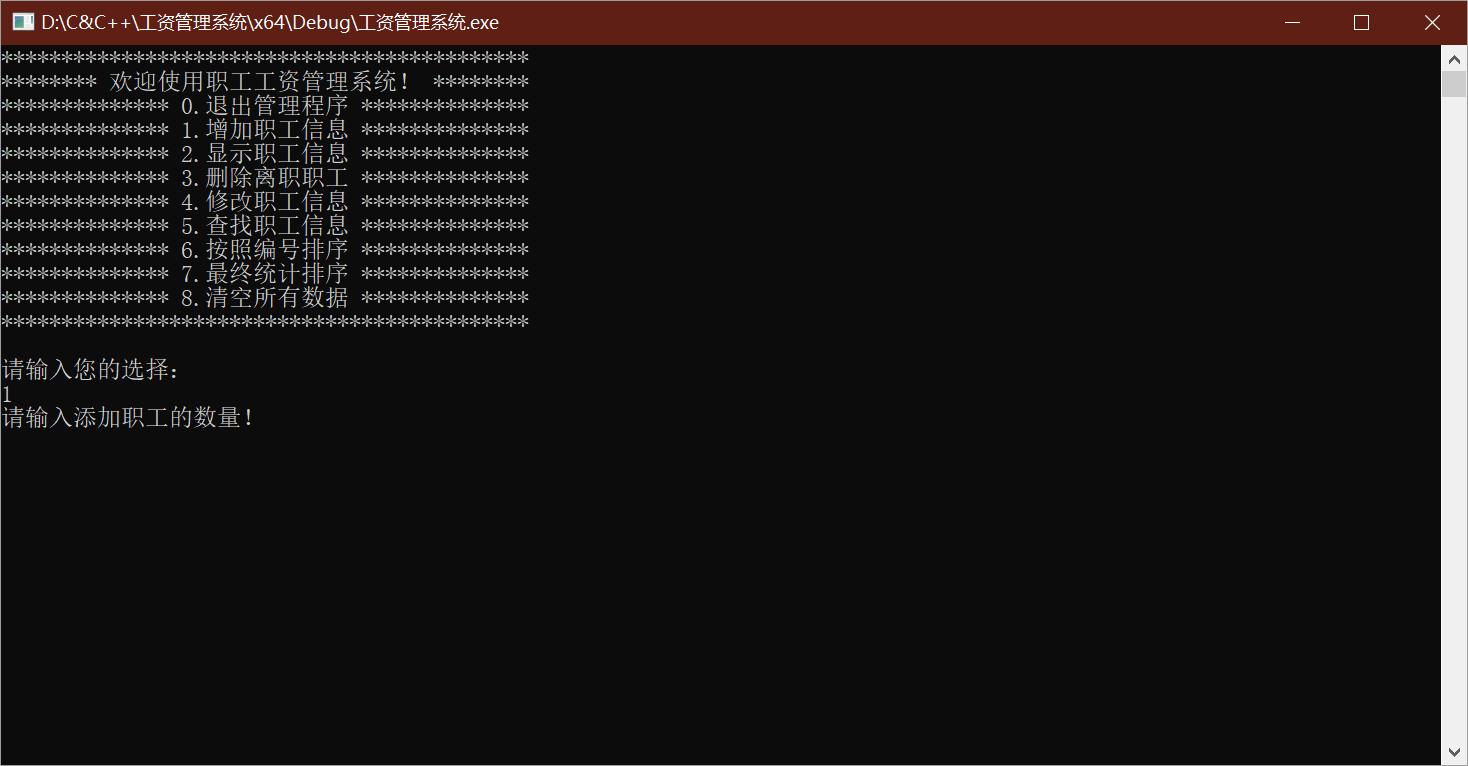
1. **系统效果图**
2. **运行：**



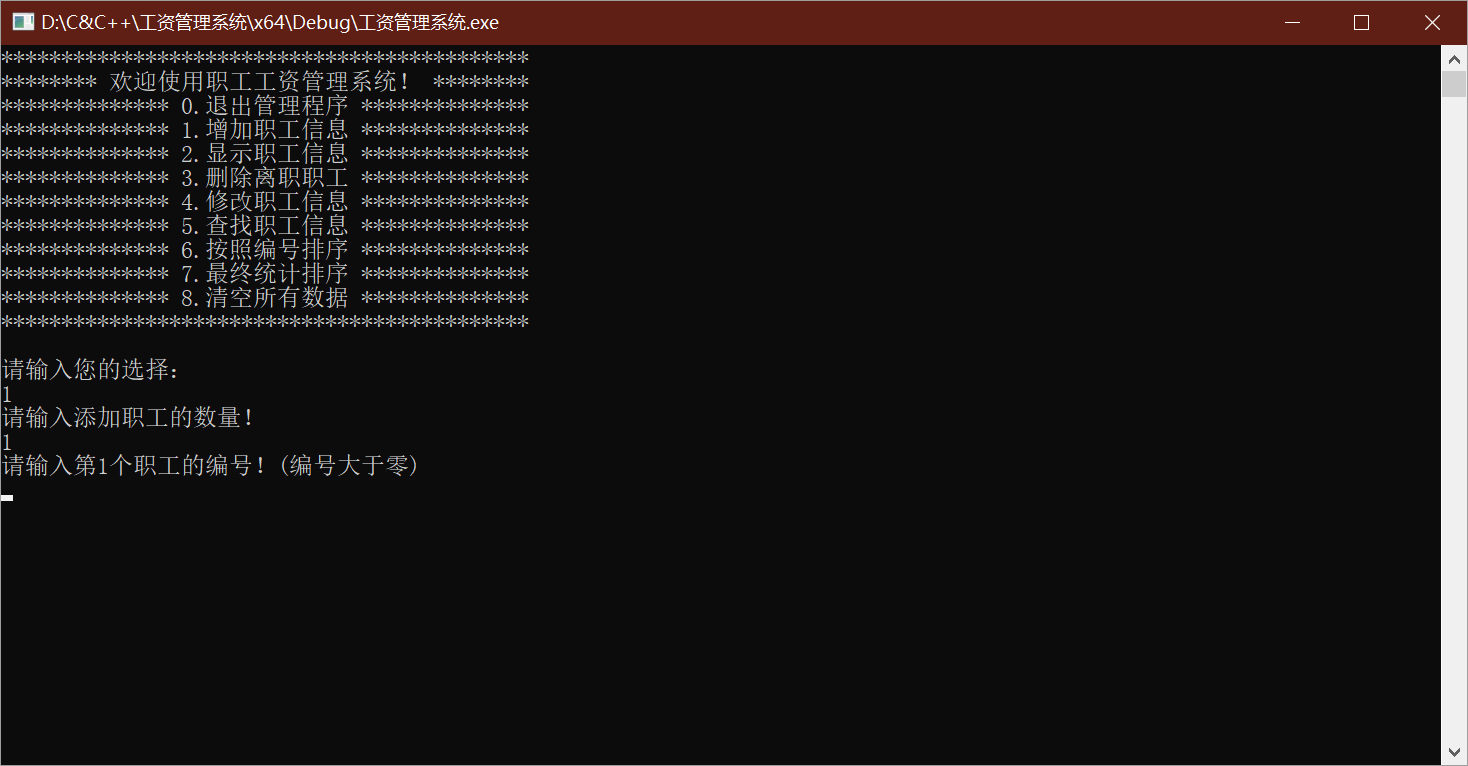
1. **输入“0”即可退出程序：**



1. **输入“1”即可批量添加职工信息：**

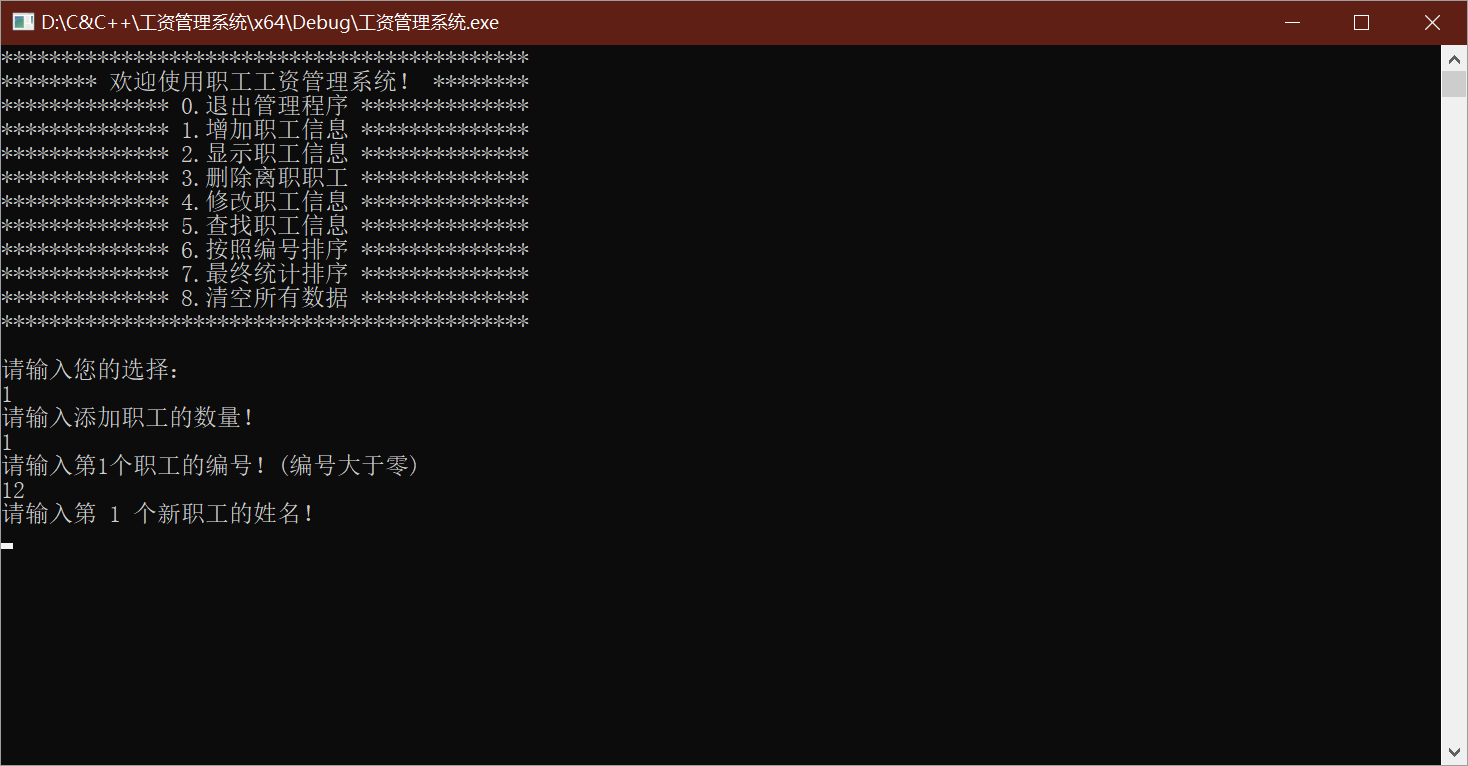
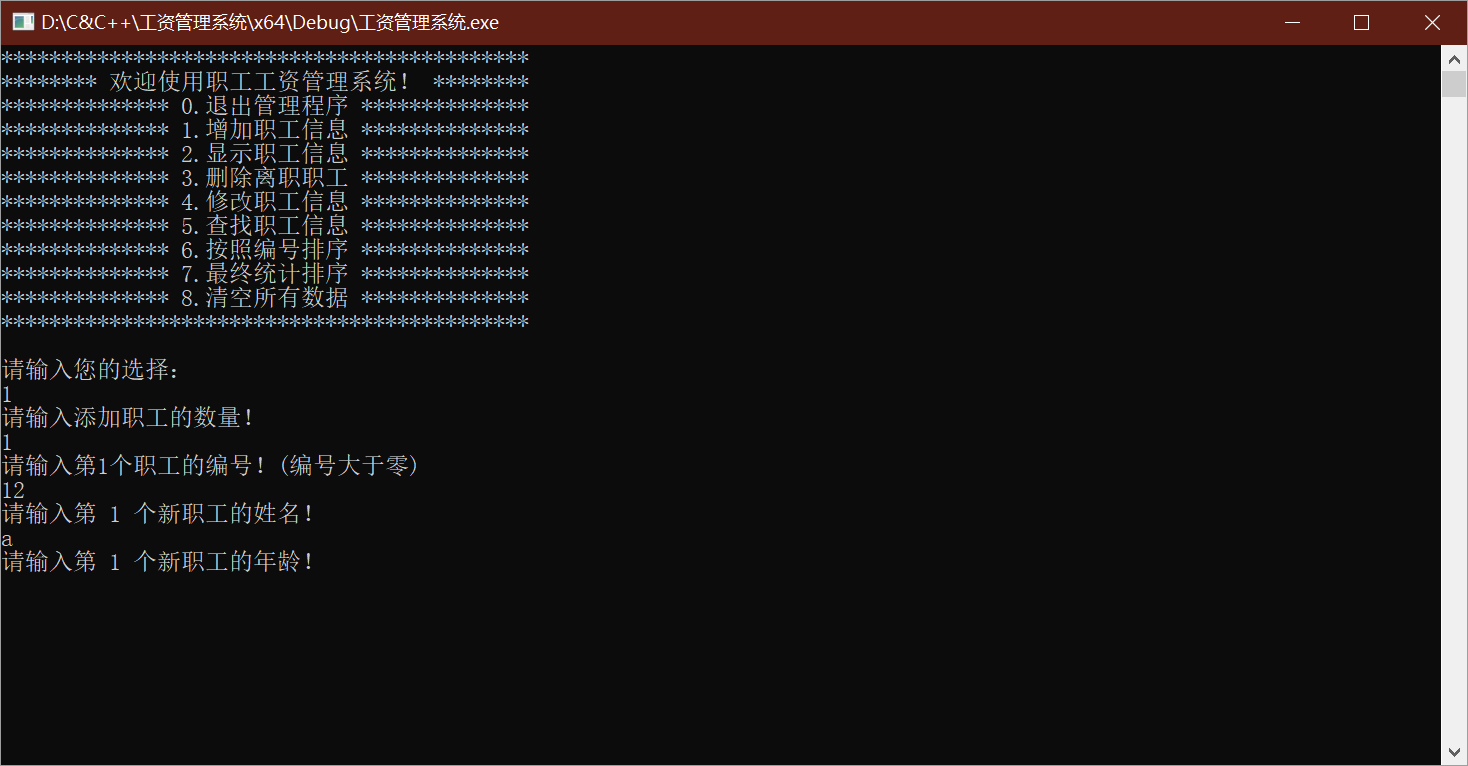


**选择数量之后：**

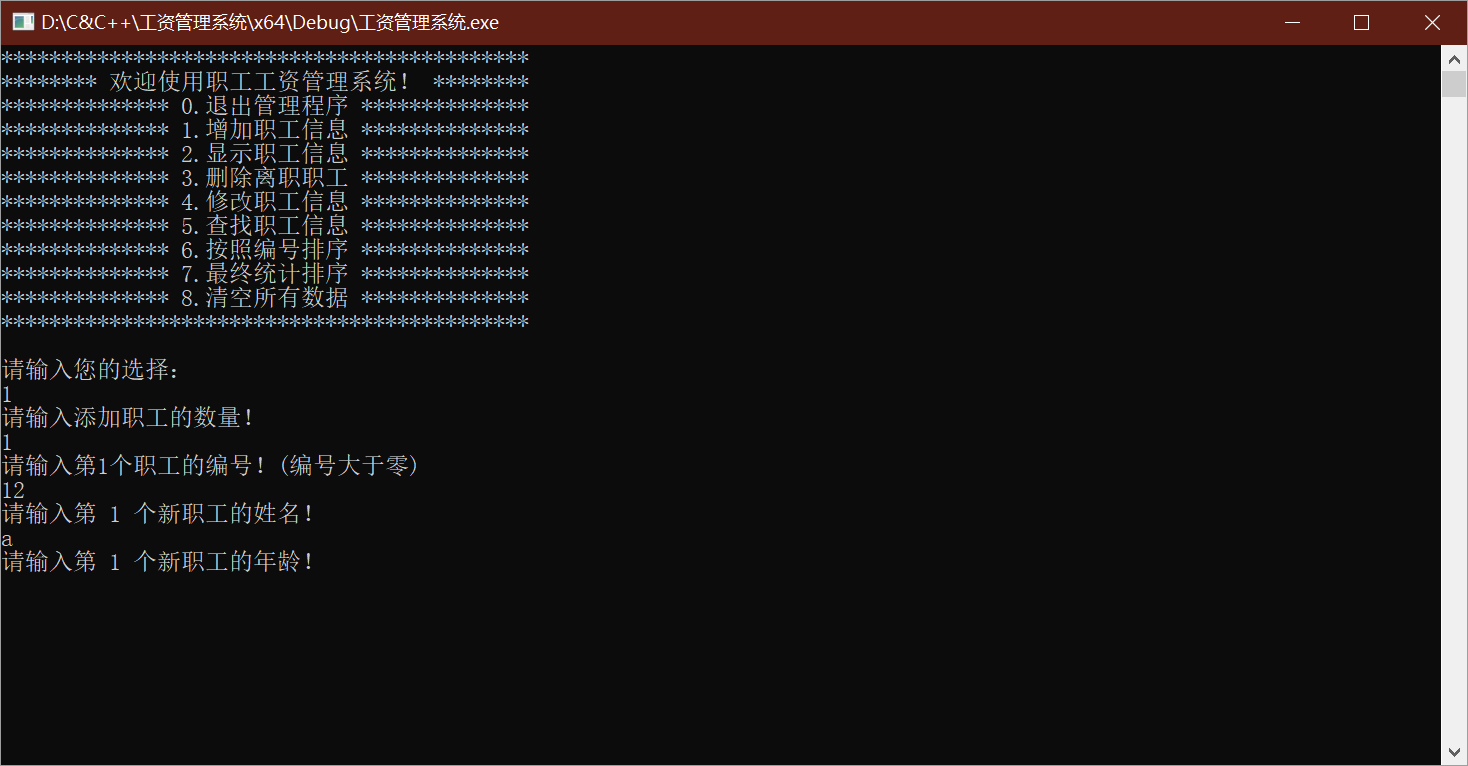


**输入编号之后（编号不可重复）：**

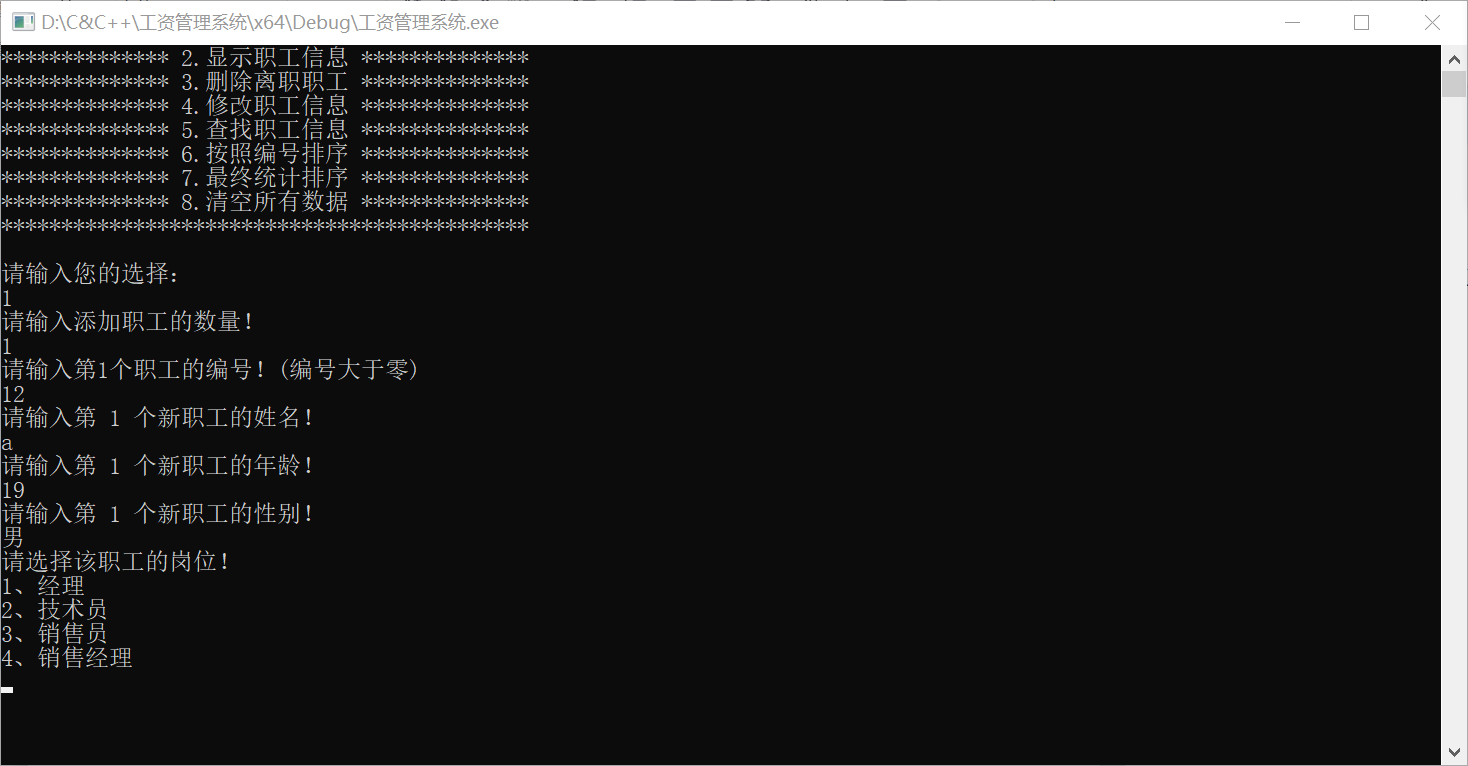
**姓名： 年龄：**



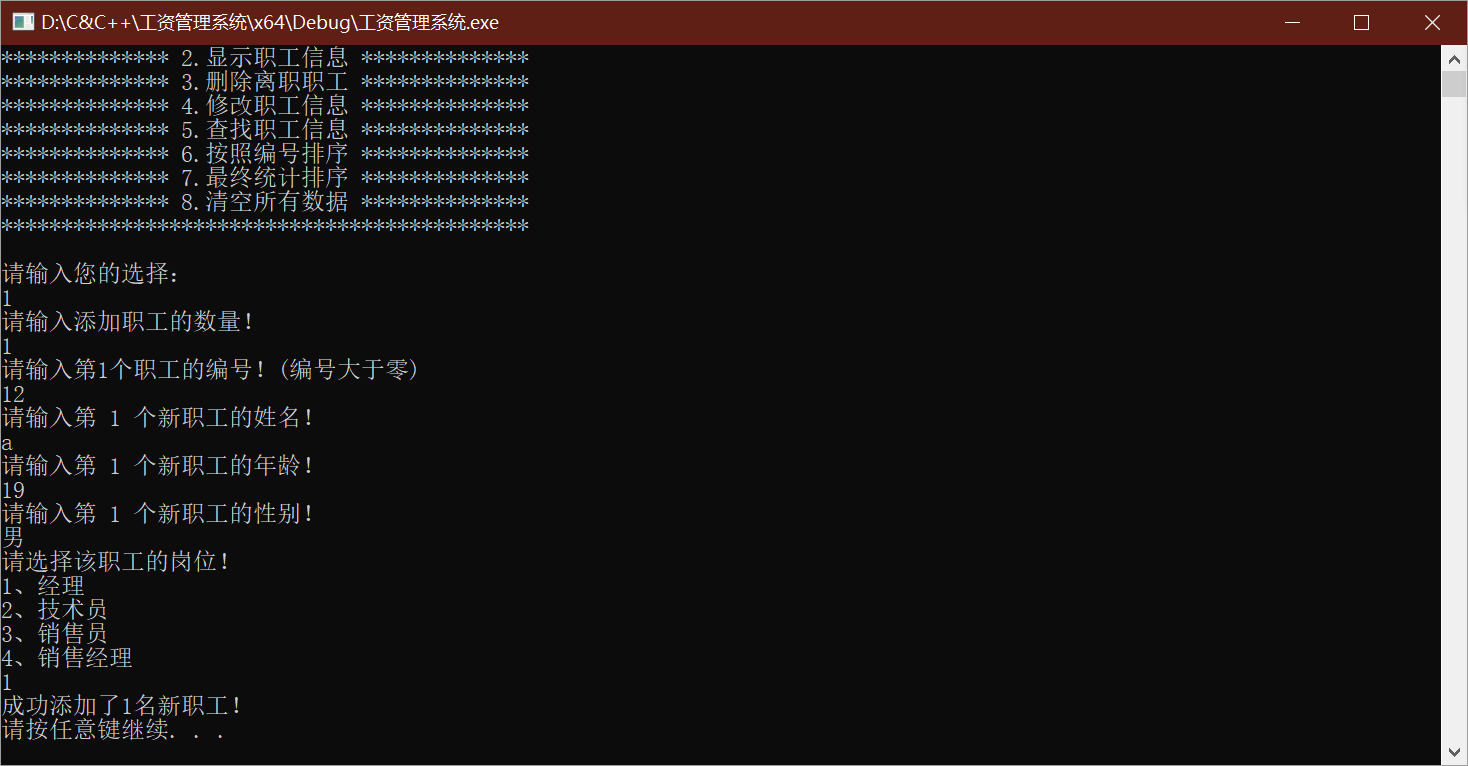
**性别：**



**此时选择职工岗位（依据岗位不同，需要录入不同的信息）：**



1. **经理**



1. **技术员**

**工作时长：**



1. **销售员**

**销售额：**



**工作部门：**

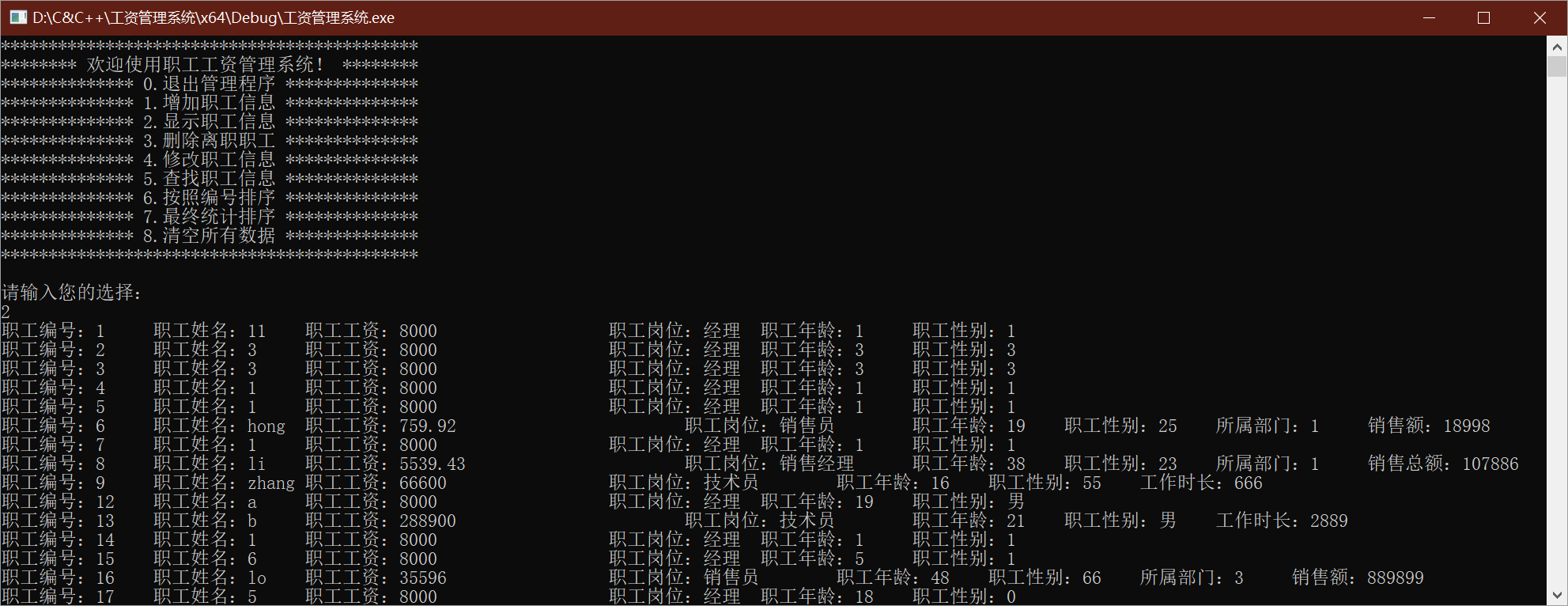


1. **销售经理**

**工作部门：**



1. **输入“2”即可显示职工信息：**



1. **输入“3”即可删除职工信息：**

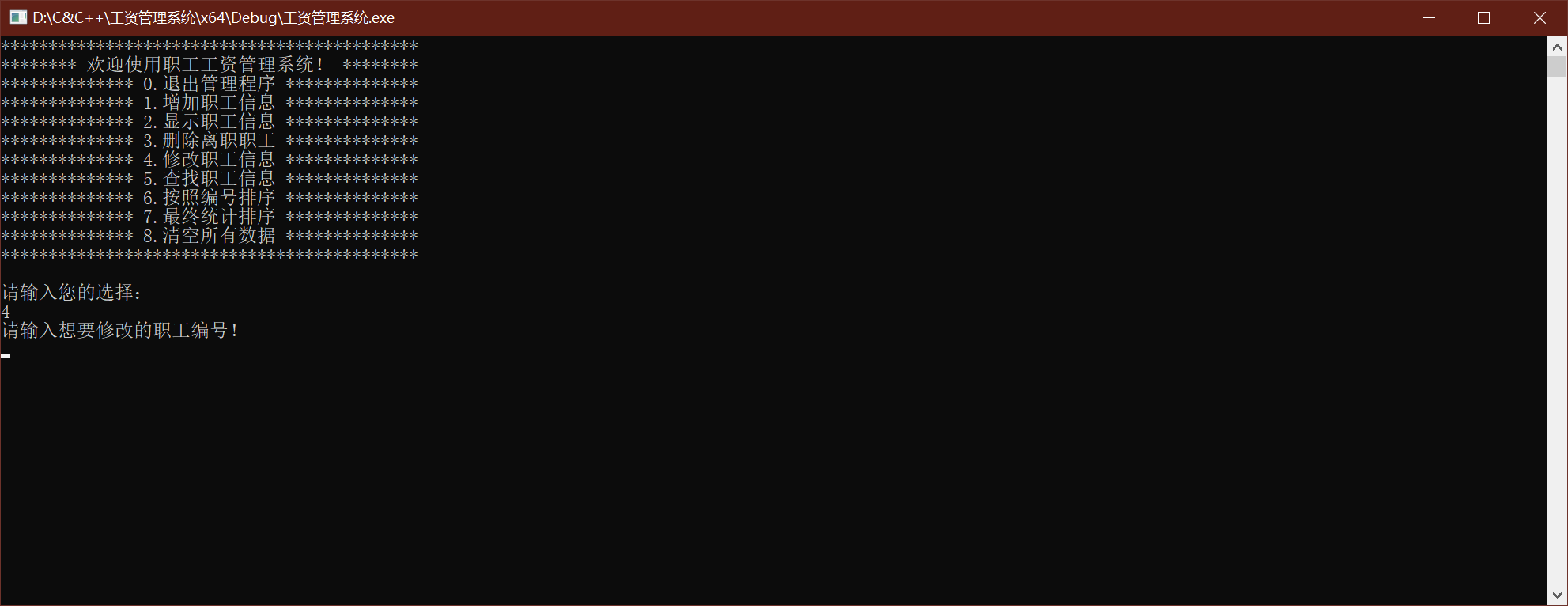
**输入编号：**



**删除成功：**



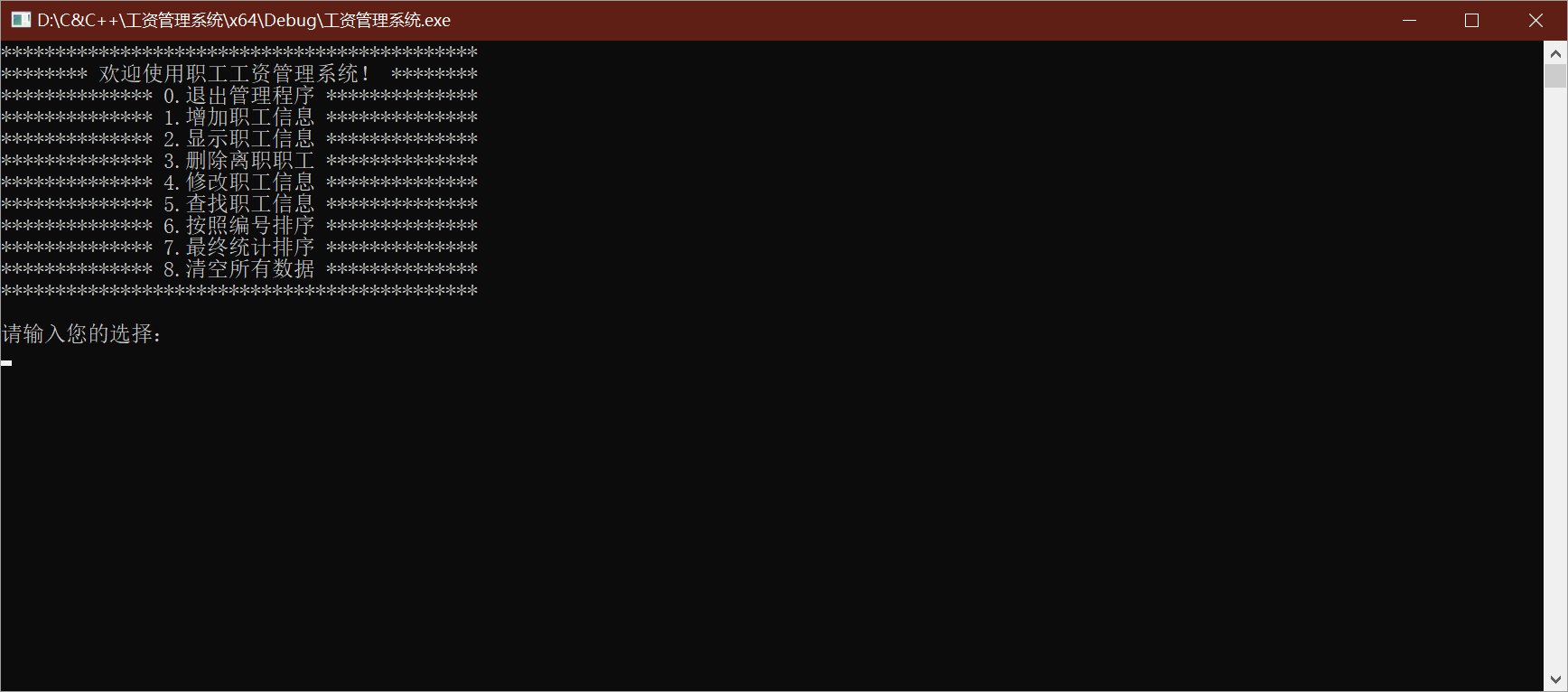
1. **输入“4”即可修改职工信息：**



**输入编号：**



**选择“否”：**



**选择“是”：**



**输入新编号：**



**输入新姓名：**



**输入新年龄：**



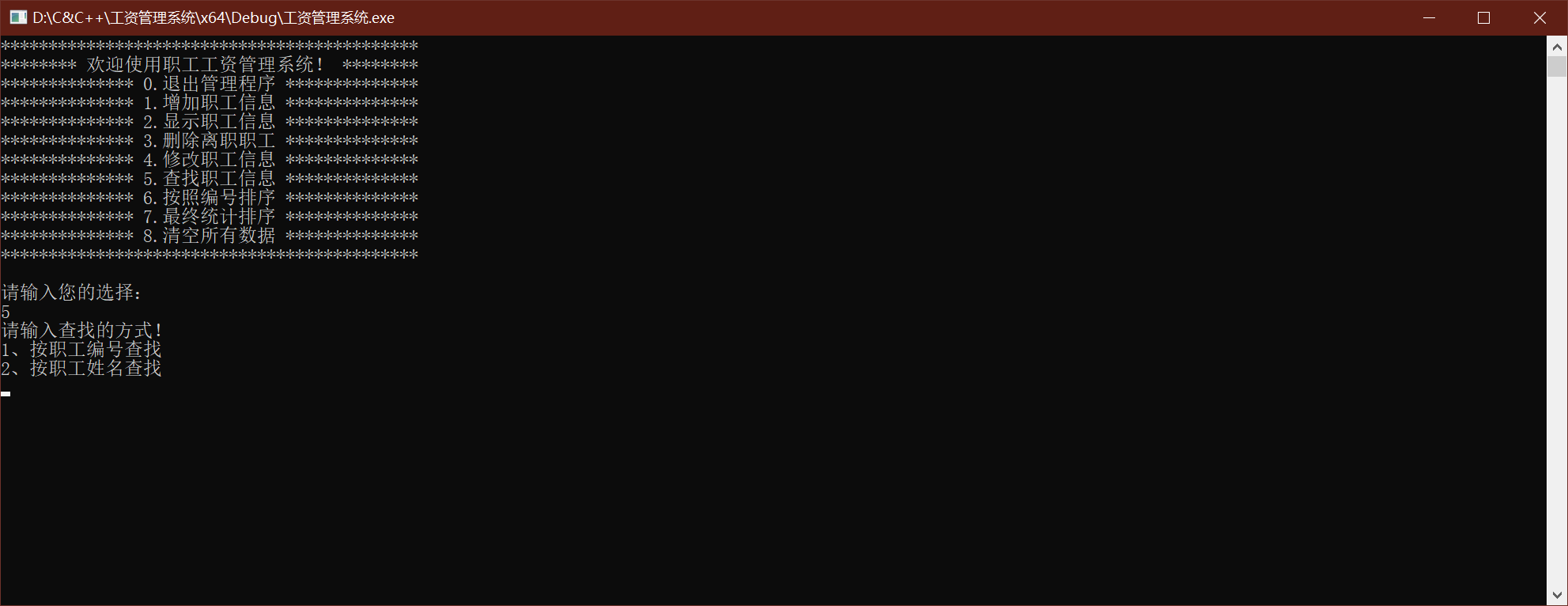
**输入新岗位：**



**修改成功：**



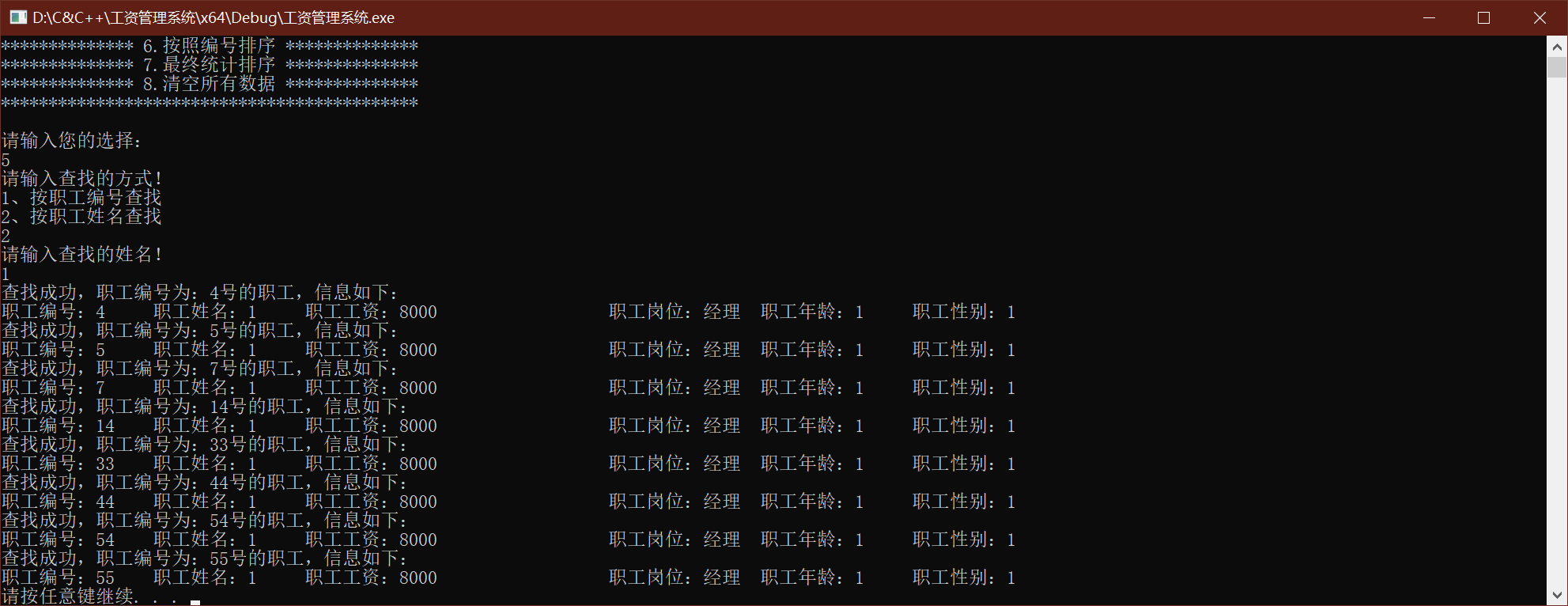
1. **输入“5”即可查找职工信息：**



**按编号查找：**

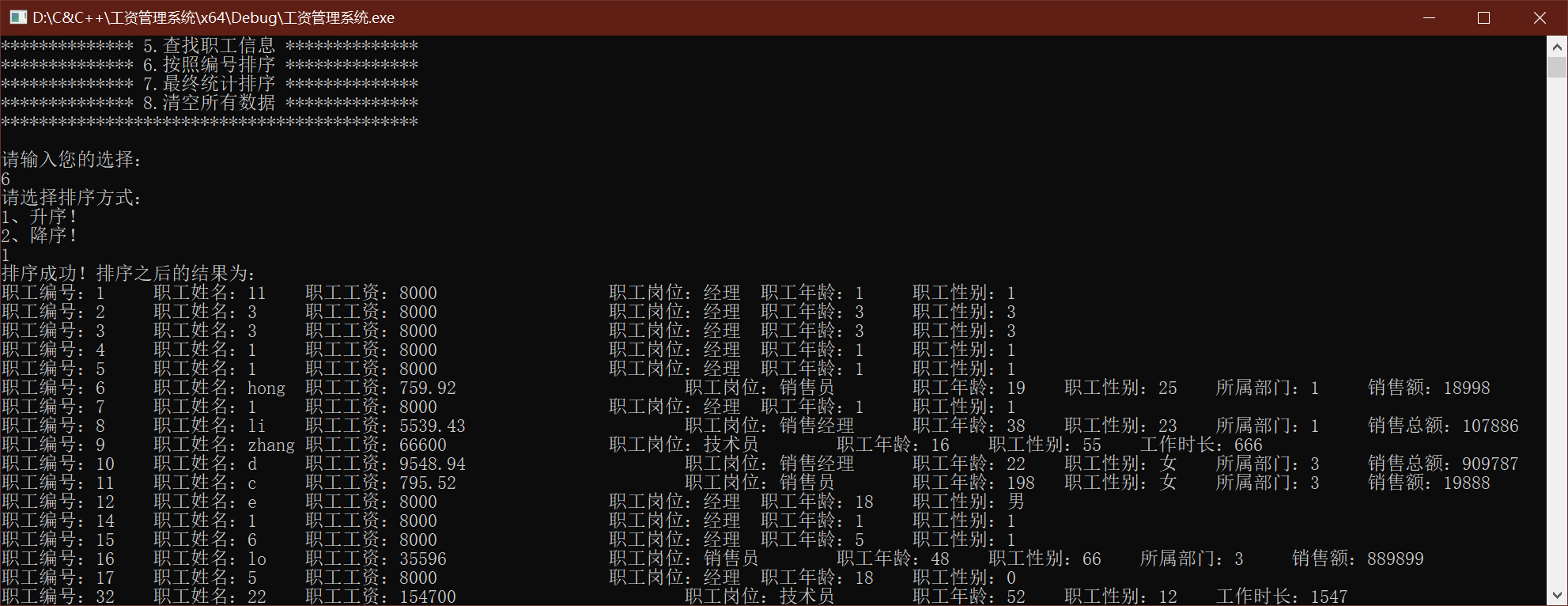


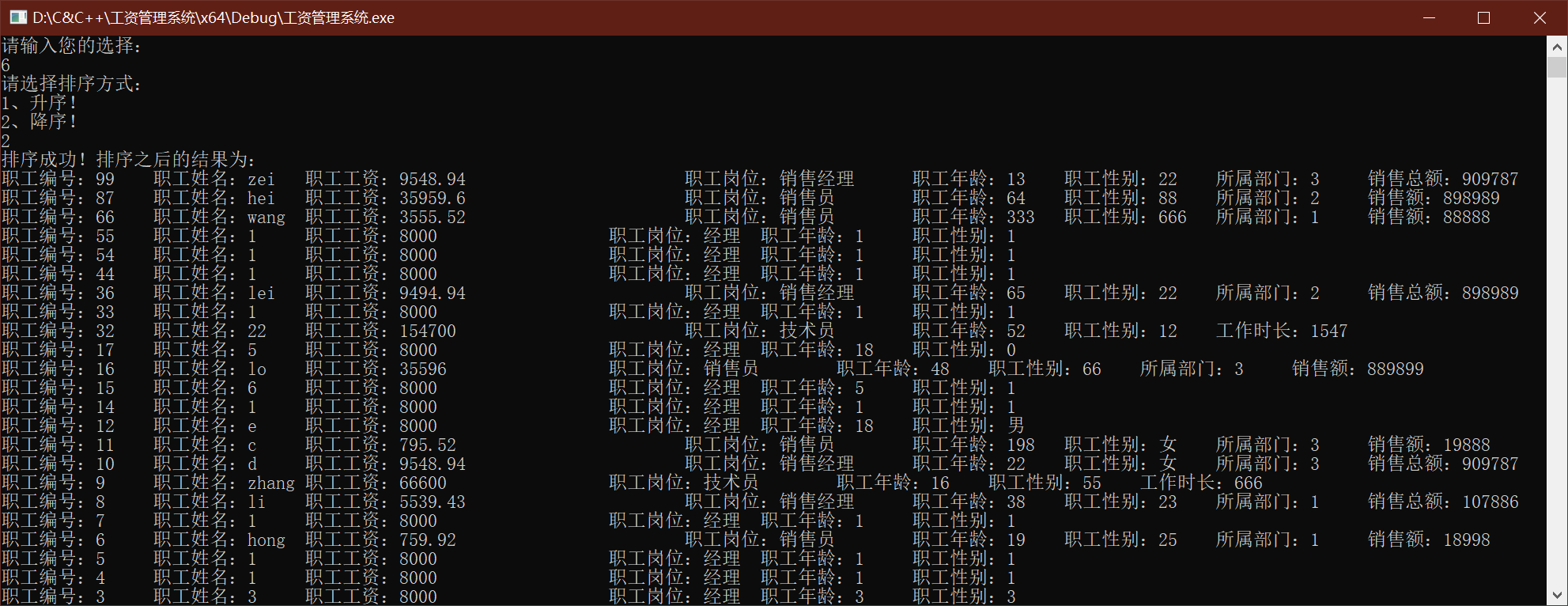
**按姓名查找：**



1. **输入“6”即可按照编号升序（降序）排序：**

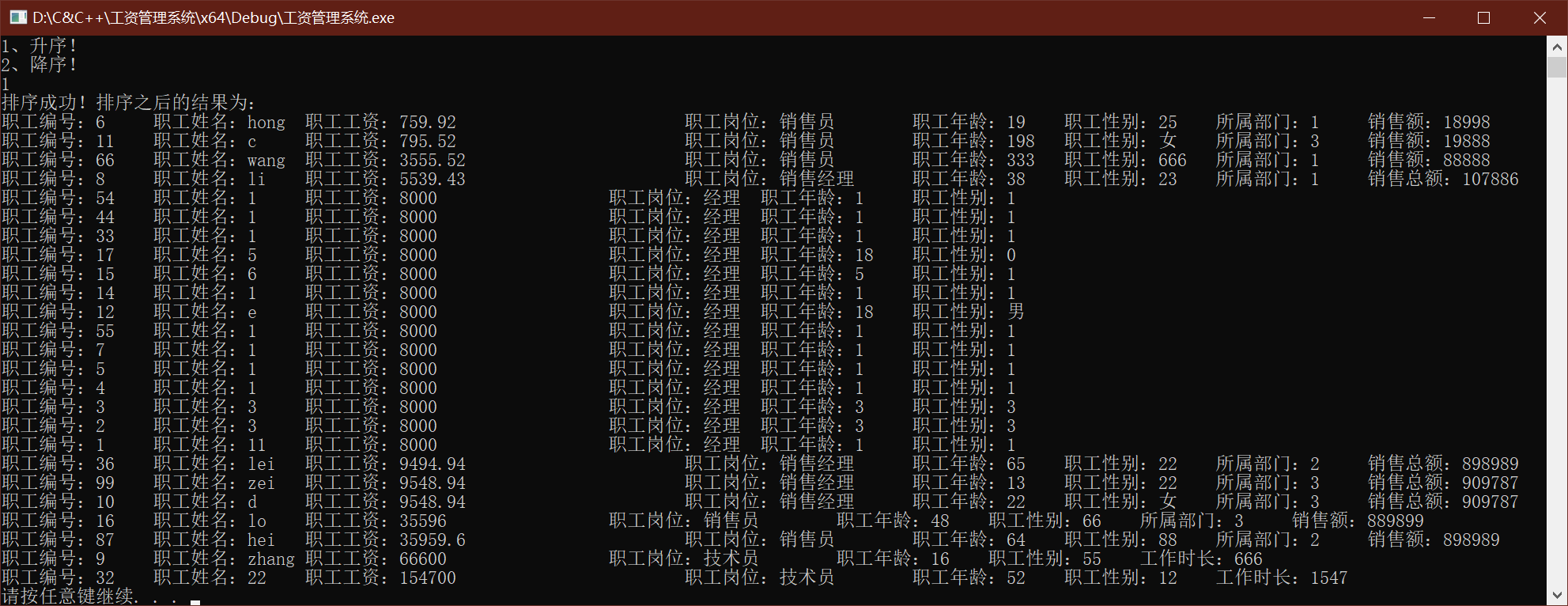
**升序：**

**降序：**

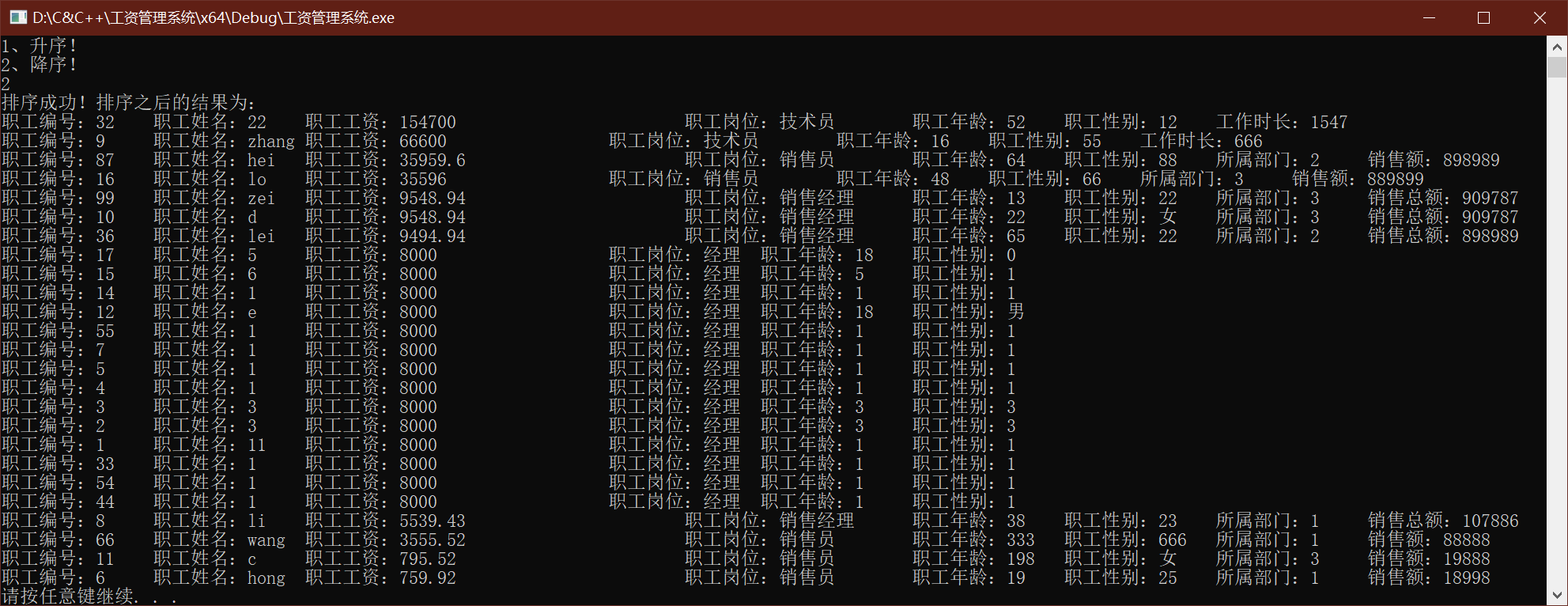


1. **输入“7”即可按照工资升序（降序）排序：**

**升序：**

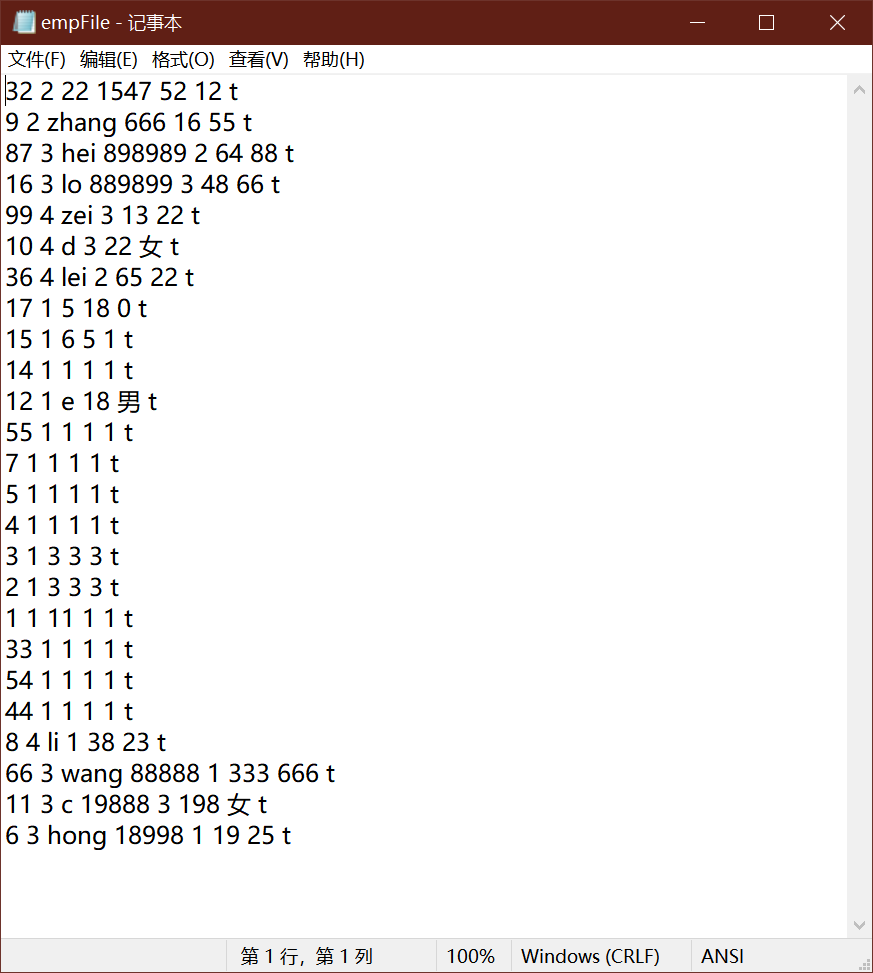
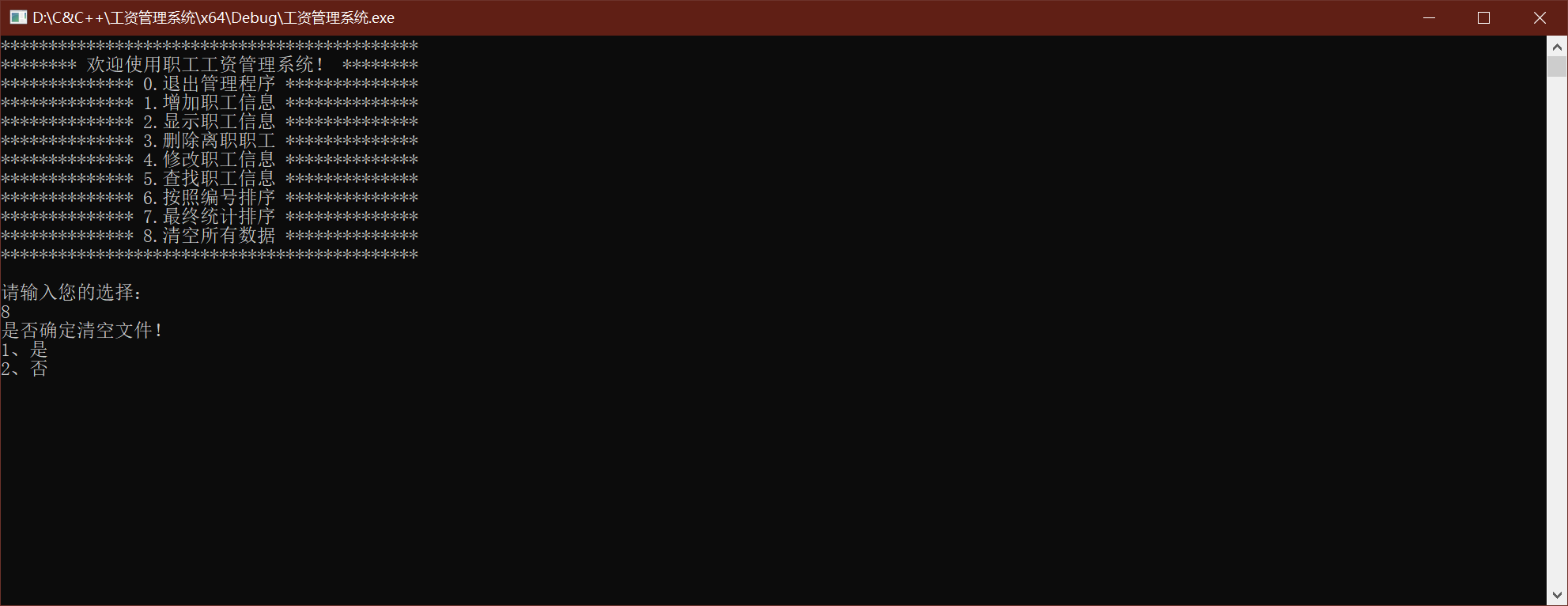


**降序：**



1. **输入“8”即可清除文件内所有内容：**

**清除前：**



**清除后：**



1. **源代码：**

**manager.h：**

//经理文件

#pragma once

#include <iostream>

using namespace std;

#include "worker.h"

class Manager :public Worker

{

public:

//构造函数

Manager(int id, int dId, string name, int age, string sex);

//显示个人信息

virtual void showInfo();

//获取岗位名称

virtual string getDeptName();

};

**salesManager.h：**

//销售经理文件

#pragma once

#include <iostream>

using namespace std;

#include "worker.h"

class SalesManager :public Worker

{

public:

//构造函数

SalesManager(int id, int dId, string name, int department\_M, int age, string sex);

//显示个人信息

virtual void showInfo();

//获取岗位名称

virtual string getDeptName();

};

**salesperson.h：**

//销售员文件

#pragma once

#include <iostream>

using namespace std;

#include "worker.h"

class Salesperson :public Worker

{

public:

//构造函数

Salesperson(int id, int dId, string name, double salesVolume, int department\_S, int age, string sex);

//显示个人信息

virtual void showInfo();

//获取岗位名称

virtual string getDeptName();

};

**technician.h：**

//技术员文件

#pragma once

#include <iostream>

using namespace std;

#include "worker.h"

class Technician :public Worker

{

public:

//构造函数

Technician(int id, int dId, string name, double time, int age, string sex);

//显示个人信息

virtual void showInfo();

//获取岗位名称

virtual string getDeptName();

};

**worker.h：**

#pragma once

#include <iostream>

using namespace std;

#include <string>

//职工类

class Worker

{

public:

//显示个人信息

virtual void showInfo() = 0;

//获取岗位名称

virtual string getDeptName() = 0;

//职工编号

int m\_Id ;

//职工姓名

string m\_Name;

//月工资

double m\_Wages;

//月工资\_销售经理

double m\_Wages\_M;

//部门编号

int m\_DeptId;

//年龄

int m\_Age;

//性别

string m\_Sex;

//工作时间

double m\_Time;

//部门\_经理

int m\_Department\_M;

//部门\_销售员

int m\_Department\_S;

//销售额

double m\_SalesVolume;

//销售额总额

double m\_AllSalesVolume;

};

**workerManager.h：**

#pragma once

#include <iostream>

using namespace std;

#include "worker.h"

#include "manager.h"

#include "technician.h"

#include "salesperson.h"

#include "salesManager.h"

#include <fstream>

#define FILENAME "empFile.txt"

class WorkerManager

{

public:

//构造函数

WorkerManager();

//菜单

void Show\_Menu();

//退出功能

void ExitSystem();

//记录职工人数

int m\_EmpNum;

//职工数组指针

Worker\*\* m\_EmpArray;

//判断文件是否为空

bool m\_FileIsEmpty;

//统计文件中人数

int get\_EmpNum();

//初始化员工

void init\_Emp();

//显示员工

void Show\_Emp();

//添加职工

void Add\_Emp();

//删除职工

void Del\_Emo();

//修改职工

void Mod\_Emp();

//查找职工

void Find\_Emp();

//编号排序

void Sort\_1\_Emp();

//工资排序

void Sort\_2\_Emp();

//销售经理工资计算

void Sm\_Wages();

//判断职工是否存在

int IsExist(int id);

//清空文件

void Clean\_File();

//保存

void save();

//析构函数

~WorkerManager();

};

**manager.cpp：**

#include "manager.h"

//构造函数

Manager::Manager(int id, int dId, string name, int age, string sex)

{

this->m\_Id = id;

this->m\_Name = name;

this->m\_Wages = 8000;

this->m\_DeptId = dId;

this->m\_Age = age;

this->m\_Sex = sex;

}

//显示个人信息

void Manager::showInfo()

{

cout << "职工编号：" << this->m\_Id

<< "\t职工姓名：" << this->m\_Name

<< "\t职工工资：" << this->m\_Wages

<< "\t\t\t职工岗位：" << this->getDeptName()

<< "\t职工年龄：" << this->m\_Age

<< "\t职工性别：" << this->m\_Sex

/\*<< "\t计算方式：固定月薪为 8000" \*/<< endl;

}

//获取岗位名称

string Manager::getDeptName()

{

return string("经理");

}

**salesManager.cpp：**

#include "salesManager.h"

//构造函数

SalesManager::SalesManager(int id, int dId, string name, int department\_M, int age, string sex)

{

this->m\_Id = id;

this->m\_Name = name;

this->m\_Department\_M = department\_M;

this->m\_DeptId = dId;

this->m\_Age = age;

this->m\_Sex = sex;

this->m\_AllSalesVolume = 0;

this->m\_Wages\_M = 0;/\*5000 + (m\_AllSalesVolume \* 0.005);\*/

}

//显示个人信息

void SalesManager::showInfo()

{

cout << "职工编号：" << this->m\_Id

<< "\t职工姓名：" << this->m\_Name

<< "\t职工工资：" << this->m\_Wages\_M

<< "\t\t\t职工岗位：" << this->getDeptName()

<< "\t职工年龄：" << this->m\_Age

<< "\t职工性别：" << this->m\_Sex

<< "\t所属部门：" << this->m\_Department\_M

<< "\t销售总额：" << this->m\_AllSalesVolume

/\*<< "\t计算方式：底薪（5000）＋所辖部门销售额总额×0.5%" \*/<< endl;

}

//获取岗位名称

string SalesManager::getDeptName()

{

return string("销售经理");

}

**salesperson.cpp：**

#include "salesperson.h"

//构造函数

Salesperson::Salesperson(int id, int dId, string name, double salesVolume, int department\_S, int age, string sex)

{

this->m\_Id = id;

this->m\_Name = name;

this->m\_SalesVolume = salesVolume;

this->m\_Wages = salesVolume\*0.04;

this->m\_DeptId = dId;

this->m\_Department\_S = department\_S;

this->m\_Age = age;

this->m\_Sex = sex;

}

//显示个人信息

void Salesperson::showInfo()

{

cout << "职工编号：" << this->m\_Id

<< "\t职工姓名：" << this->m\_Name

<< "\t职工工资：" << this->m\_Wages

<< "\t\t\t职工岗位：" << this->getDeptName()

<< "\t职工年龄：" << this->m\_Age

<< "\t职工性别：" << this->m\_Sex

<< "\t所属部门：" << this->m\_Department\_S

<< "\t销售额：" << this->m\_SalesVolume

/\*<< "\t计算方式：销售额×4%提成" \*/<< endl;

}

//获取岗位名称

string Salesperson::getDeptName()

{

return string("销售员");

}

**technician.cpp：**

#include "technician.h"

//构造函数

Technician::Technician(int id, int dId, string name, double time, int age, string sex)

{

this->m\_Id = id;

this->m\_Name = name;

this->m\_Time = time;

this->m\_Wages = time\*100;

this->m\_DeptId = dId;

this->m\_Age = age;

this->m\_Sex = sex;

}

//显示个人信息

void Technician::showInfo()

{

cout << "职工编号：" << this->m\_Id

<< "\t职工姓名：" << this->m\_Name

<< "\t职工工资：" << this->m\_Wages

<< "\t\t\t职工岗位：" << this->getDeptName()

<< "\t职工年龄：" << this->m\_Age

<< "\t职工性别：" << this->m\_Sex

<< "\t工作时长：" << this->m\_Time

/\*<< "\t计算方式：工作时间×小时工资（100 元每小时）"\*/ << endl;

}

//获取岗位名称

string Technician::getDeptName()

{

return string("技术员");

}

**workerManager.cpp：**

#include "workerManager.h"

WorkerManager::WorkerManager()

{

ifstream ifs;

if (ifs.is\_open())

{

//文件不存在

ifs.open(FILENAME, ios::in);//读文件

/\*cout << "文件不存在！" << endl;\*/

this->m\_EmpNum = 0;

this->m\_EmpArray = NULL;

this->m\_FileIsEmpty = true;

ifs.close();

return;

}

//文件存在，但为空

else if (ifs.eof())

{

//文件不存在

ifs.open(FILENAME, ios::in);//读文件

/\*cout << "文件为空！" << endl;\*/

this->m\_EmpNum = 0;

this->m\_EmpArray = NULL;

this->m\_FileIsEmpty = true;

ifs.close();

return;

}

else

{

//文件存在，不为空

int num = this->get\_EmpNum();

//cout << "职工人数为：" << num << endl;

this->m\_EmpNum = num;

//开辟空间并继承文件内容

this->m\_EmpArray = new Worker \* [this->m\_EmpNum];

this->init\_Emp();

}

//for (int i = 0; i < this->m\_EmpNum; i++)

//{

// cout << "职工编号：" << this->m\_EmpArray[i]->m\_Id << endl

// << "姓名：" << this->m\_EmpArray[i]->m\_Name << endl;

//}

}

//初始化员工

void WorkerManager::init\_Emp()

{

ifstream ifs;

ifs.open(FILENAME, ios::in);

int id;//职工编号

string name;//职工姓名

int dId;//部门编号

int age;//职工年龄

int time;//工作时长

string sex;//职工性别

int salesVolume;//销售额

int department;//工作部门

int index = 0;//索引

char c\_id[256];

ifs.seekg(0, ios::beg);

while (!ifs.eof())

{

Worker \* worker = NULL;

ifs >> id;

ifs >> dId;

ifs >> name;

if (dId == 1)//经理

{

ifs >> age;

ifs >> sex;

worker = new Manager(id, 1, name, age, sex);

}

else if (dId == 2)//技术员

{

ifs >> time;

ifs >> age;

ifs >> sex;

worker = new Technician(id, 2, name, time, age, sex);

}

else if (dId == 3)//销售员

{

ifs >> salesVolume;

ifs >> department;

ifs >> age;

ifs >> sex;

worker = new Salesperson(id, 3, name, salesVolume, department, age, sex);

}

else if (dId == 4)//销售经理

{

ifs >> department;

ifs >> age;

ifs >> sex;

worker = new SalesManager(id, 4, name, department, age, sex);

}

ifs.getline(c\_id, 256, 't');

this->m\_EmpArray[index] = worker;

index++;

}

ifs.close();

}

//显示员工

void WorkerManager::Show\_Emp()

{

//判断文件是否为空or存在

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

this->Sm\_Wages();

for (int i = 0; i < m\_EmpNum ; i++)

{

this->m\_EmpArray[i]->showInfo();

}

}

system("pause");

system("cls");

}

//菜单

void WorkerManager::Show\_Menu()

{

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\* 欢迎使用职工工资管理系统！ \*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0.退出管理程序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.增加职工信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2.显示职工信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3.删除离职职工 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 4.修改职工信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 5.查找职工信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6.按照编号排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 7.最终统计排序 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\* 8.清空所有数据 \*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << endl;

}

//退出功能

void WorkerManager::ExitSystem()

{

cout << "欢迎下次使用！" << endl;

system("pause");

exit(0);

}

//添加职工

void WorkerManager::Add\_Emp()

{

cout << "请输入添加职工的数量！" << endl;

int addNum = 0;//输入数量

cin >> addNum;

if (addNum > 0)

{

int newSize = this->m\_EmpNum + addNum;//新空间大小

Worker\*\* newSpace = new Worker \* [newSize];

//拷贝旧数据

if (this->m\_EmpArray != NULL)

{

for (int i = 0; i < this->m\_EmpNum; i++)

{

newSpace[i] = this->m\_EmpArray[i];

}

}

//批量添加新数据

for (int i = 0; i < addNum; i++)

{

int id;//职工编号

string name;//职工姓名

int dId;//部门编号

int age;//职工年龄

int time;//工作时长

string sex;//职工性别

int salesVolume;//销售额

int department\_S;//工作部门

int department\_M;

cout << "请输入第" << i + 1 << "个职工的编号！(编号大于零)" << endl;

//判断编号

while (true)

{

ifstream ifs;

ifs.open(FILENAME, ios::in);

if ((ifs.eof())|| (!ifs.is\_open()))

{

while (true)

{

cin >> id;

if (id < 0)

{

cout << "编号小于零！" << endl;

cout << "请重新输入！" << endl;

}

else

{

break;

}

}

}

else

{

while (true)

{

cin >> id;

int j\_id = -1;

char c\_id[256];

ifs.seekg(0, ios::beg);

while (true)

{

if (id < 0)

{

cout << "编号小于零！" << endl;

cout << "请重新输入！" << endl;

}

else

{

break;

}

cin >> id;

}

while (!ifs.eof())

{

ifs >> j\_id;

if (j\_id == id)

{

cout << "已存在此编号！" << endl;

cout << "请重新输入！" << endl;

break;

}

ifs.getline(c\_id, 256, 't');

}

if (j\_id != id)

{

break;

}

}

}

ifs.close();

break;

}

cout << "请输入第 " << i+1 << " 个新职工的姓名！" << endl;

cin >> name;

cout << "请输入第 " << i+1 << " 个新职工的年龄！" << endl;

cin >> age;

cout << "请输入第 " << i+1 << " 个新职工的性别！" << endl;

cin >> sex;

cout << "请选择该职工的岗位！" << endl;

cout << "1、经理" << endl;

cout << "2、技术员" << endl;

cout << "3、销售员" << endl;

cout << "4、销售经理" << endl;

cin >> dId;

Worker\* worker = NULL;

switch (dId)

{

case 1:

worker = new Manager(id, 1, name, age, sex);

break;

case 2:

cout << "请输入该技术员的工作时长！" << endl;

cin >> time;

worker = new Technician(id, 2, name, time, age, sex);

break;

case 3:

cout << "请输入该销售员的销售额！" << endl;

cin >> salesVolume;

cout << "请输入该销售员的工作部门！" << endl;

cin >> department\_S;

worker = new Salesperson(id, 3, name, salesVolume, department\_S, age, sex);

break;

case 4:

cout << "请输入该销售经理的工作部门！" << endl;

cin >> department\_M;

worker = new SalesManager(id, 4, name, department\_M, age, sex);

break;

default:

break;

}

//保存入数组

newSpace[this->m\_EmpNum + i] = worker;

}

//释放

this->m\_EmpArray = new Worker \* [newSize];

//更新指向

this->m\_EmpArray = newSpace;

//更新人数

this->m\_EmpNum = newSize;

cout << "成功添加了" << addNum << "名新职工！" << endl;

this->save();

this->m\_FileIsEmpty = false;

}

else

{

cout << "请输入数据有误！" << endl;

}

system("pause");

system("cls");

}

//删除职工

void WorkerManager::Del\_Emo()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

//按照职工编号删除职工信息

cout << "请输入想要删除职工的编号！" << endl;

int id = -1;

cin >> id;

int index = this->IsExist(id);

if (index != -1)//职工存在

{

for (int i = index; i < this->m\_EmpNum - 1; i++)

{

//数据前移

this->m\_EmpArray[i] = this->m\_EmpArray[i + 1];

}

this->m\_EmpNum--;//更新所记录的个数

//同步更新文件

this->save();

cout << "删除成功！" << endl;

}

else

{

cout << "删除失败，未找到该职工！" << endl;

}

}

system("pause");

system("cls");

}

//修改职工

void WorkerManager::Mod\_Emp()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

cout << "请输入想要修改的职工编号！" << endl;

int select;

int id;

cin >> id;

int index = this->IsExist(id);

if (index != -1)

{

int NewId;//职工编号

string NewName;//职工姓名

int NewDId;//部门编号

int NewAge;//职工年龄

int NewTime;//工作时长

string NewSex;//职工性别

int NewSalesVolume;//销售额

int NewDepartment;//工作部门

while (true)

{

//找到该职工

cout << "查找到ID为" << id << "号的职工！" << endl;

cout << "该职工的信息为：" << endl;

this->m\_EmpArray[index]->showInfo();

cout << "是否删除此职工信息？" << endl;

cout << "1、是" << endl;

cout << "2、否" << endl;

cin >> select;

if (select == 1)

{

delete this->m\_EmpArray[index];

cout << "请输入新的职工编号：" << endl;

//判断编号

while (true)

{

ifstream ifs;

ifs.open(FILENAME, ios::in);

if ((ifs.eof()) || (!ifs.is\_open()))

{

while (true)

{

cin >> NewId;

if (id < 0)

{

cout << "编号小于零！" << endl;

cout << "请重新输入！" << endl;

}

else

{

break;

}

}

}

else

{

while (true)

{

cin >> NewId;

int j\_id = -1;

char c\_id[256];

ifs.seekg(0, ios::beg);

while (true)

{

if (NewId < 0)

{

cout << "编号小于零！" << endl;

cout << "请重新输入！" << endl;

}

else

{

break;

}

cin >> NewId;

}

while (!ifs.eof())

{

ifs >> j\_id;

if (j\_id == NewId)

{

cout << "已存在此编号！" << endl;

cout << "请重新输入！" << endl;

break;

}

ifs.getline(c\_id, 256, 't');

}

if (j\_id != NewId)

{

break;

}

}

}

ifs.close();

break;

}

cout << "请输入职工的新姓名！" << endl;

cin >> NewName;

cout << "请输入职工的新年龄！" << endl;

cin >> NewAge;

cout << "请输入职工的新性别！" << endl;

cin >> NewSex;

cout << "请选择该职工的新岗位！" << endl;

cout << "1、经理" << endl;

cout << "2、技术员" << endl;

cout << "3、销售员" << endl;

cout << "4、销售经理" << endl;

cin >> NewDId;

Worker\* worker = NULL;

switch (NewDId)

{

case 1:

{

worker = new Manager(NewId, 1, NewName, NewAge, NewSex);

break;

}

case 2:

{

cout << "请输入职工新的工作时长！" << endl;

cin >> NewTime;

worker = new Technician(NewId, 2, NewName, NewTime, NewAge, NewSex);

break;

}

case 3:

{

cout << "请输入职工新的销售额！" << endl;

cin >> NewSalesVolume;

cout << "请输入职工新的工作部门！" << endl;

cin >> NewDepartment;

worker = new Salesperson(NewId, 3, NewName, NewSalesVolume, NewDepartment, NewAge, NewSex);

break;

}

case 4:

{

cout << "请输入职工的新工作部门！" << endl;

cin >> NewDepartment;

worker = new SalesManager(NewId, 4, NewName, NewDepartment, NewAge, NewSex);

break;

}

default:

break;

}

this->m\_EmpArray[index] = worker;

cout << "修改成功！" << endl;

this->save();

break;

}

else if (select == 2)

{

break;

}

else

{

cout << "输入选项有误，请重新输入！" << endl;

system("cls");

}

}

}

else

{

cout << "修改失败，未找到该职工！" << endl;

}

}

system("pause");

system("cls");

}

//查找职工

void WorkerManager::Find\_Emp()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

}

else

{

while (true)

{

cout << "请输入查找的方式！" << endl;

cout << "1、按职工编号查找" << endl;

cout << "2、按职工姓名查找" << endl;

int select = 0;

cin >> select;

if (select == 1)

{

//按编号查找

int id;

cout << "请输入所查找的员工的编号是多少！" << endl;

cin >> id;

int index = IsExist(id);

if (index != -1)

{

//成功

cout << "查找成功！该职工的信息如下：" << endl;

this->m\_EmpArray[index]->showInfo();

}

else

{

//失败

cout << "查找失败，查无此人！" << endl;

}

break;

}

else if(select == 2)

{

//按姓名查找

string name;

cout << "请输入查找的姓名！" << endl;

cin >> name;

//判断

bool flag = false;//默认失败

for (int i = 0; i < this->m\_EmpNum; i++)

{

if (this->m\_EmpArray[i]->m\_Name == name)

{

cout << "查找成功，职工编号为："

<< this->m\_EmpArray[i]->m\_Id

<< "号的职工，信息如下：" << endl;

this->m\_EmpArray[i]->showInfo();

flag = true;

}

}

if (flag == false)

{

//失败

cout << "查找失败，查无此人！" << endl;

}

break;

}

else

{

cout << "输入选项有误，请重新输入！" << endl;

}

}

}

system("pause");

system("cls");

}

//编号排序

void WorkerManager::Sort\_1\_Emp()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

system("pause");

system("cls");

}

else

{

this->Sm\_Wages();

cout << "请选择排序方式：" << endl;

cout << "1、升序！" << endl;

cout << "2、降序！" << endl;

int select = 0;

int index = 1;

for (int i = 0; i < this->m\_EmpNum; i++)

{

int MinOrMax = i;//声明最大最小值

for (int j = i + 1; j < this->m\_EmpNum; j++)

{

while (true)

{

if (index == 1)

{

cin >> select;

}

if (select == 1)//升序

{

if (this->m\_EmpArray[MinOrMax]->m\_Id > this->m\_EmpArray[j]->m\_Id)

{

MinOrMax = j;

}

index = -1;

break;

}

else if (select == 2)//降序

{

if (this->m\_EmpArray[MinOrMax]->m\_Id < this->m\_EmpArray[j]->m\_Id)

{

MinOrMax = j;

}

index = -1;

break;

}

else

{

cout << "输入选项有误，请重新输入！" << endl;

}

}

}

//判断

if (i != MinOrMax)

{

Worker\* temp = this->m\_EmpArray[i];

this->m\_EmpArray[i] = this->m\_EmpArray[MinOrMax];

this->m\_EmpArray[MinOrMax] = temp;

}

}

cout << "排序成功！排序之后的结果为：" << endl;

this->save();//保存

this->Show\_Emp();//展示

}

}

//工资排序

void WorkerManager::Sort\_2\_Emp()

{

if (this->m\_FileIsEmpty)

{

cout << "文件不存在或记录为空！" << endl;

system("pause");

system("cls");

}

else

{

this->Sm\_Wages();

cout << "请选择排序方式：" << endl;

cout << "1、升序！" << endl;

cout << "2、降序！" << endl;

int select = 0;

int index = 1;

for (int i = 0; i < this->m\_EmpNum; i++)

{

int MinOrMax = i;//声明最大最小值

for (int j = i + 1; j < this->m\_EmpNum; j++)

{

while (true)

{

if (index == 1)

{

cin >> select;

}

if (select == 1)//升序

{

if ((this->m\_EmpArray[MinOrMax]->m\_DeptId == 4) && (this->m\_EmpArray[j]->m\_DeptId != 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages\_M > this->m\_EmpArray[j]->m\_Wages)

{

MinOrMax = j;

}

index = -1;

break;

}

else if ((this->m\_EmpArray[MinOrMax]->m\_DeptId != 4) && (this->m\_EmpArray[j]->m\_DeptId == 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages > this->m\_EmpArray[j]->m\_Wages\_M)

{

MinOrMax = j;

}

index = -1;

break;

}

else if ((this->m\_EmpArray[MinOrMax]->m\_DeptId == 4) && (this->m\_EmpArray[j]->m\_DeptId == 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages\_M > this->m\_EmpArray[j]->m\_Wages\_M)

{

MinOrMax = j;

}

index = -1;

break;

}

else

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages > this->m\_EmpArray[j]->m\_Wages)

{

MinOrMax = j;

}

index = -1;

break;

}

}

else if (select == 2)//降序

{

if ((this->m\_EmpArray[MinOrMax]->m\_DeptId == 4) && (this->m\_EmpArray[j]->m\_DeptId != 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages\_M < this->m\_EmpArray[j]->m\_Wages)

{

MinOrMax = j;

}

index = -1;

break;

}

else if ((this->m\_EmpArray[MinOrMax]->m\_DeptId != 4) && (this->m\_EmpArray[j]->m\_DeptId == 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages < this->m\_EmpArray[j]->m\_Wages\_M)

{

MinOrMax = j;

}

index = -1;

break;

}

else if ((this->m\_EmpArray[MinOrMax]->m\_DeptId == 4) && (this->m\_EmpArray[j]->m\_DeptId == 4))

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages\_M < this->m\_EmpArray[j]->m\_Wages\_M)

{

MinOrMax = j;

}

index = -1;

break;

}

else

{

if (this->m\_EmpArray[MinOrMax]->m\_Wages < this->m\_EmpArray[j]->m\_Wages)

{

MinOrMax = j;

}

index = -1;

break;

}

}

else

{

cout << "输入选项有误，请重新输入！" << endl;

}

}

}

//判断

if (i != MinOrMax)

{

Worker\* temp = this->m\_EmpArray[i];

this->m\_EmpArray[i] = this->m\_EmpArray[MinOrMax];

this->m\_EmpArray[MinOrMax] = temp;

}

}

cout << "排序成功！排序之后的结果为：" << endl;

this->save();//保存

this->Show\_Emp();//展示

}

}

//销售经理工资计算

void WorkerManager::Sm\_Wages()

{

for (int i = 0; i < this->m\_EmpNum; i++)

{

if (this->m\_EmpArray[i]->m\_DeptId == 4)

{

this->m\_EmpArray[i]->m\_AllSalesVolume = 0;

for (int j = 0; j < this->m\_EmpNum; j++)

{

if (this->m\_EmpArray[j]->m\_DeptId == 3)

{

if (this->m\_EmpArray[i]->m\_Department\_M == this->m\_EmpArray[j]->m\_Department\_S)

{

this->m\_EmpArray[i]->m\_AllSalesVolume = this->m\_EmpArray[i]->m\_AllSalesVolume + this->m\_EmpArray[j]->m\_SalesVolume;

}

}

}

this->m\_EmpArray[i]->m\_Wages\_M = 5000 + (0.005 \* this->m\_EmpArray[i]->m\_AllSalesVolume);

}

}

}

//判断职工是否存在

int WorkerManager::IsExist(int id)

{

int index = -1;

for (int i = 0; i < this->m\_EmpNum; i++)

{

if (this->m\_EmpArray[i]->m\_Id == id)

{

//职工存在

index = i;

break;

}

}

return index;

}

//统计人数

int WorkerManager::get\_EmpNum()

{

ifstream ifs;

ifs.open(FILENAME, ios::in);

int Number = -1;

char id[256];

ifs.seekg(0, ios::beg);

while (!ifs.eof())

{

if (ifs.getline(id, 256, 't'))

{

Number++;

}

}

ifs.close();

//cout << "人数为：" << Number << endl;

return Number;

}

//保存

void WorkerManager::save()

{

ofstream ofs;

ofs.open(FILENAME, ios::out);//写入

ofs.seekp(0, ios::beg);

int index = 0;

while (index < this->m\_EmpNum)

{

if (this->m\_EmpArray[index]->m\_DeptId == 1)

{

ofs << this->m\_EmpArray[index]->m\_Id << " "

<< this->m\_EmpArray[index]->m\_DeptId << " "

<< this->m\_EmpArray[index]->m\_Name << " "

<< this->m\_EmpArray[index]->m\_Age << " "

<< this->m\_EmpArray[index]->m\_Sex << " t" << endl;

}

else if (this->m\_EmpArray[index]->m\_DeptId == 2)

{

ofs << this->m\_EmpArray[index]->m\_Id << " "

<< this->m\_EmpArray[index]->m\_DeptId << " "

<< this->m\_EmpArray[index]->m\_Name << " "

<< this->m\_EmpArray[index]->m\_Time << " "

<< this->m\_EmpArray[index]->m\_Age << " "

<< this->m\_EmpArray[index]->m\_Sex << " t" << endl;

}

else if (this->m\_EmpArray[index]->m\_DeptId == 3)

{

ofs << this->m\_EmpArray[index]->m\_Id << " "

<< this->m\_EmpArray[index]->m\_DeptId << " "

<< this->m\_EmpArray[index]->m\_Name << " "

<< this->m\_EmpArray[index]->m\_SalesVolume << " "

<< this->m\_EmpArray[index]->m\_Department\_S << " "

<< this->m\_EmpArray[index]->m\_Age << " "

<< this->m\_EmpArray[index]->m\_Sex << " t" << endl;

}

else if (this->m\_EmpArray[index]->m\_DeptId == 4)

{

ofs << this->m\_EmpArray[index]->m\_Id << " "

<< this->m\_EmpArray[index]->m\_DeptId << " "

<< this->m\_EmpArray[index]->m\_Name << " "

<< this->m\_EmpArray[index]->m\_Department\_M << " "

<< this->m\_EmpArray[index]->m\_Age << " "

<< this->m\_EmpArray[index]->m\_Sex << " t" << endl;

}

ofs.seekp(0, ios::end);

index++;

}

ofs.close();

}

//清空文件

void WorkerManager::Clean\_File()

{

cout << "是否确定清空文件！" << endl;

cout << "1、是" << endl;

cout << "2、否" << endl;

int select = 0;

cin >> select;

if (select == 1)

{

//清空文件

ofstream ofs(FILENAME, ios::trunc);//删除文件后重新创建

ofs.close();

if (this->m\_EmpArray != NULL)

{

for (int i = 0; i < this->m\_EmpNum; i++)

{

delete this->m\_EmpArray[i];

this->m\_EmpArray[i] = NULL;

}

delete[] this->m\_EmpArray;

this->m\_EmpArray = NULL;

this->m\_EmpNum = 0;

this->m\_FileIsEmpty = true;

}

cout << "清空成功！" << endl;

}

system("pause");

system("cls");

}

//析构

WorkerManager::~WorkerManager()

{

if (this->m\_EmpArray != NULL)

{

for (int i = 0; i < this->m\_EmpNum; i++)

{

if (this->m\_EmpArray[i] != NULL)

{

delete this->m\_EmpArray[i];

}

}

this->m\_EmpArray = new Worker \* [this->m\_EmpNum];

this->m\_EmpArray = NULL;

}

}

**职工工资系统.cpp：**

#include <iostream>

using namespace std;

#include "workerManager.h"

#include "worker.h"

#include "manager.h"

#include "technician.h"

#include "salesperson.h"

#include "salesManager.h"

int main()

{

WorkerManager wm;

int choice = 0;

while (true)

{

wm.Show\_Menu();

cout << "请输入您的选择：" << endl;

cin >> choice;

switch (choice)

{

case 0://退出系统

wm.ExitSystem();

break;

case 1://增加职工

wm.Add\_Emp();

break;

case 2://显示职工

wm.Show\_Emp();

break;

case 3://删除职工

wm.Del\_Emo();

break;

case 4://修改职工

wm.Mod\_Emp();

break;

case 5://查找职工

wm.Find\_Emp();

break;

case 6://编号排序

wm.Sort\_1\_Emp();

break;

case 7://工资排序

wm.Sort\_2\_Emp();

break;

case 8://清空文档

wm.Clean\_File();

break;

default:

system("cls");

break;

}

}

system("pause");

return 0;

}

1. **参考文献：**

|  |  |  |
| --- | --- | --- |
| 教材名称： | 编者： | 出版社： |
| C++面向对象程序设计教程(第4版) | 陈维兴 | 清华大学出版社 |

1. **设计体会：**

这次C++课设做的是工资管理系统，在这次课设中我学到了许多书本上没有的知识我们首先很好的复习和巩固了自己所学的知识,提升了我们的实际操作能力。在这次实训中，我们学会了如何运用面向对象程序设计的思想，用C++语言编写代码，去设计程序。

在课设设计的过程中我认识到必须细致地设计程序中的每一个细节，准确的考虑程序运行时每一刻发生的事情。在课程设计过程中我也遇到了一些问题:

1. 在数据输入时，没有考虑到工资实际的情况，导致程序不完善。
2. 运行时有许多我们没有加以限制的条件，导致出入不符合情况使程序出现死循环。
3. 我们对程序的界面加以改观，使运行界面更加美丽。

但是通过查阅资料和请教老师和同学，终于解决了这些问题，使程序可以准确的运行出来。

这次课设使我对C++这门基础语言有了一定的了解，这些知识对我在今后的学习方面有很大的帮助，在这过程中不仅掌握了C++语言的编译方法，同时也了解到了在编译过程中应该注意的一些问题及其解决办法，为自己今后在其他语言的学习上做了很好的准备。有了这些知识为基础我相信自己以后的学习将变得更加容易。我深深体会到只是需要实践的重要性，通过实践我们进一步发现自己的不足，欠缺，进而知道我们怎样深化理解，真正吸收所学的知识。在这个过程中我们还了解了很多原本陌生的知识。我们深刻体会出学习C++我们还有很长的路要走，相信只要我们继续努力，多加练习，一定会更加出色。

在这个课程设计过程中，所学的专业知识有了一个较系统的综合，并对其中的许多问题进行了重新学习和加深理解，为将来实际工作的需要奠定了坚实的基础。课程设计是我们专业课程知识综合应用的实践训练，是我们迈向社会，从事职业工作前一个必不少的过程。”千里之行始于足下”，通过这次课程设计，我深深体会到这句千古名言的真正含义。我今天认真的进行课程设计，学会脚踏实地迈开这一步，就是为明天能稳健地在社会大潮中奔跑打下坚实的基础。这次课程设计真是让我受益匪浅。

**计算机2102**

**2106410224**

**洪利达**