

高级语言程序设计课程实践报告

项 目 名 称**:**  职工工资信息管理系统

所 在 班 级： 2022级计算机科学与技术三班

小 组 成 员： 薛琦嘉 侯栩涛

指 导 教 师： 焦清局

起 止 时 间： 2023.11.21—2023.12.9

项目基本信息

|  |  |
| --- | --- |
| 项目名称 |  |
| 项目简介 | 1．问题描述  职工工资信息管理系统。  2．基本要求  创建存储职工工资信息的存储文件。添加某职工的工资信息。删除某职工的工资信息。修改某职工的部分工资信息（当月开始增加或减少某些项工资或扣款数变化）。输出指定编号职工的工资信息（查询用）。输出全体职工的工资信息（发工资用）。  3．测试数据  0 aa 10000 100  1 bb 3000 900  0 cc 4000 500  1 dd 5000 600  0 ee 6000 500  0 ff 10000 900  1 gg 3000 800  1 hh 4000 700  0 ii 5000 600  1 jj 6000 500 |
| 小组成员 | 薛琦嘉 侯栩涛 |
| 任务分工 | 薛琦嘉 侯栩涛 程序设计  薛琦嘉 侯栩涛 实验报告分析 |

**一、问题描述及分析**

创建存储职工工资信息的存储文件。添加某职工的工资信息。删除某职工的工资信息。修改某职工的部分工资信息（当月开始增加或减少某些项工资或扣款数变化）。输出指定编号职工的工资信息（查询用）。输出全体职工的工资信息（发工资用）。

**二、功能模块及结构描述**

一、功能模块：职工工资信息管理系统主要提供以下功能：

①增加员工信息

②修改员工姓名

③查找员工信息

④删除员工信息

⑤显示员工信息

⑥统计员工数量

⑦统计总发放工资

**二、结构描述**.

职工工资信息管理系统：

（1）本职工工资信息管理系统下整体通过类来完成，以实现更好的封装性，每一个类对象即为一个独立的账户，在此对象中含有该员工的全部信息，包括姓名，基本工资,奖金等。

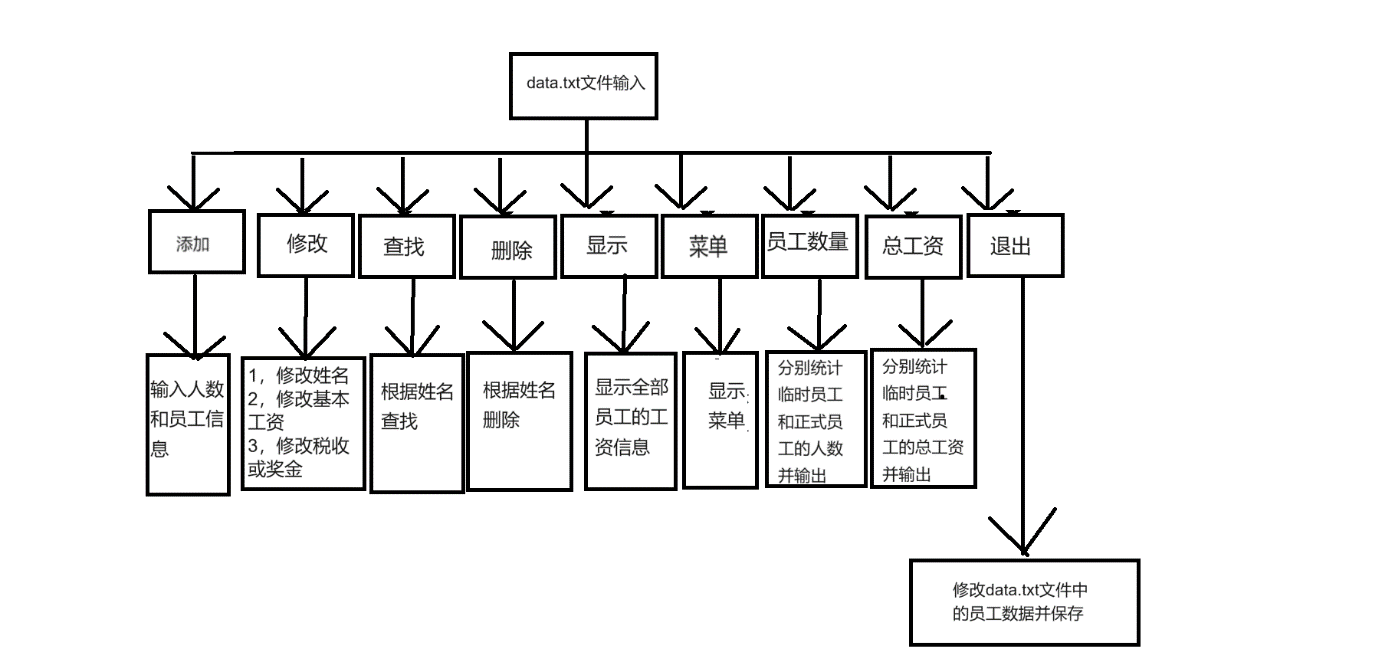
（2）使用指针动态开辟对象空间，以节省空间。

（3）将每个功能独立化，将其设置为单一函数如输入函数、输出函数、查找函数、删除函数、计算总工资函数、修改信息函数、显示员工信息函数、计算发放总工资函数。

（4）在设计函数时，将用户的所有误操作进行考虑，即使是用户存在误操作，也会人性化的给予提示，提醒用户再次输入正确操作，使程序更加完整、全面和人性化。

**三、主要流程描述**

流程图：

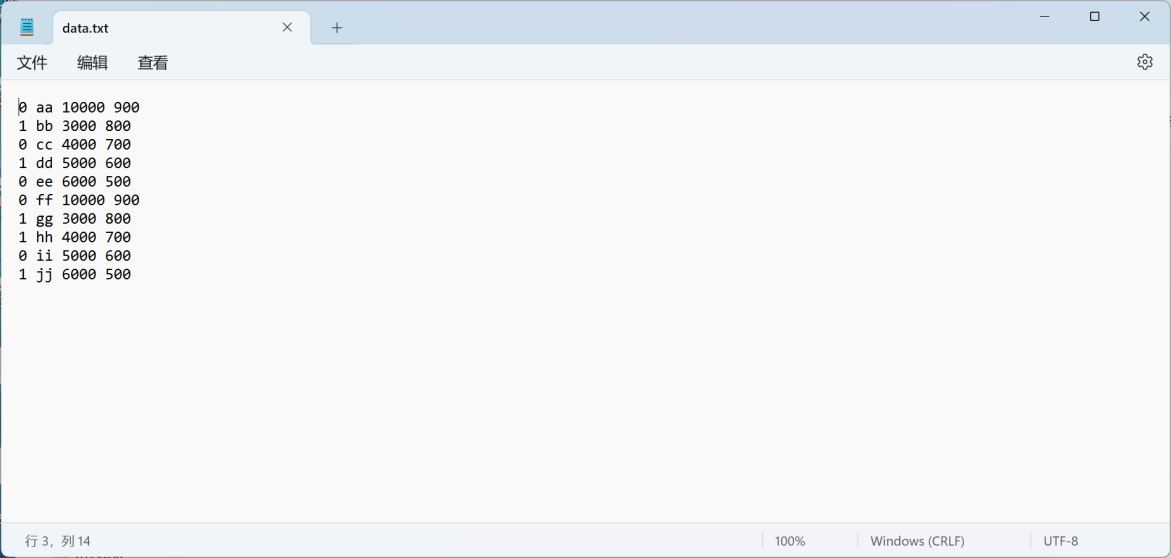


**四、使用说明**

程序运行后，进入界面：

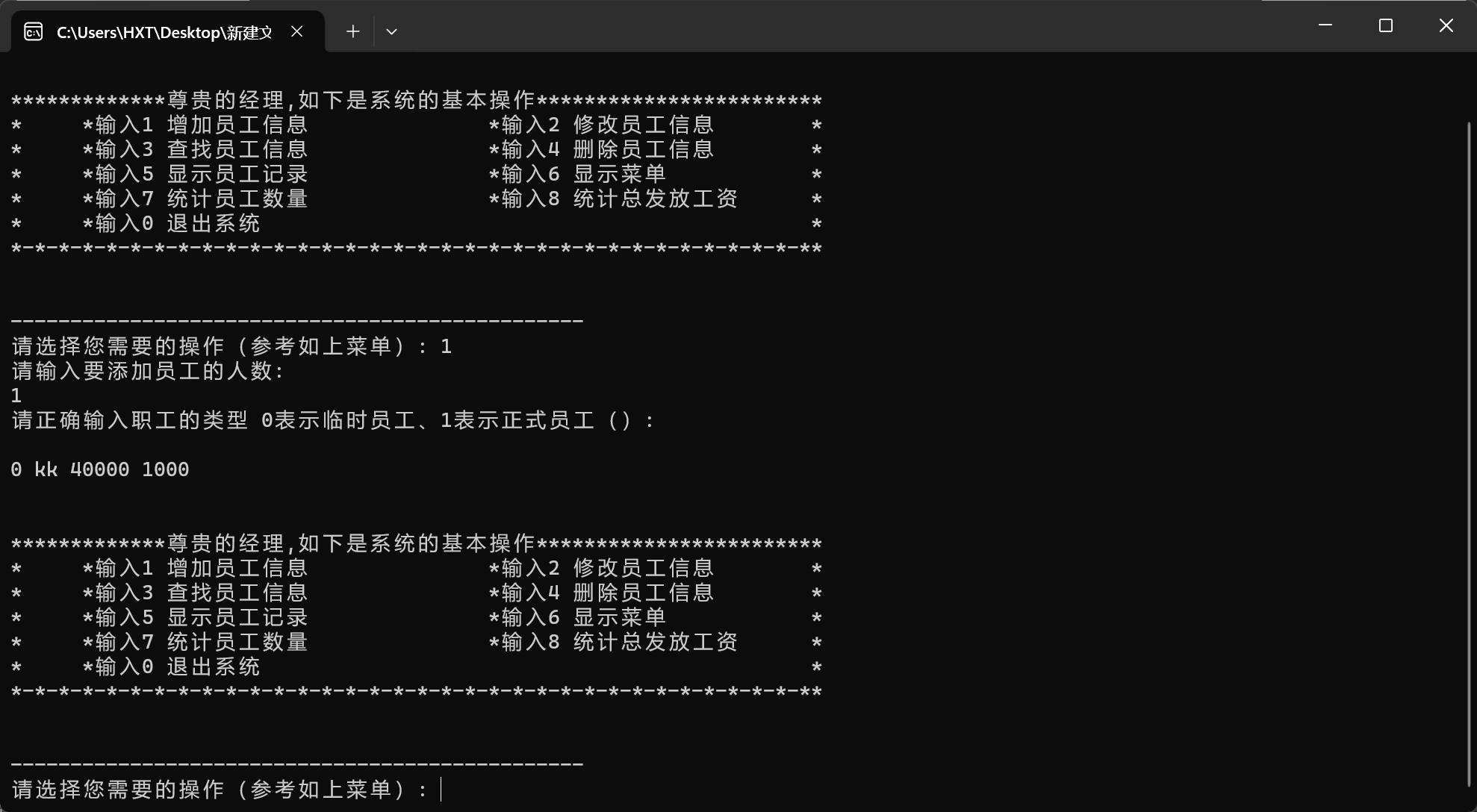
进入职工工资信息管理系统：

在文件data.txt文件中输入员工人数，及他们的信息，使用文件输入流存到数组中：



  
在如上所示的界面下，根据提示进行基本的操作。

下面输入“1”，进行员工信息添加（输入要添加的人数，和工资信息）：



输入“2”：对员工的信息进行修改（包括姓名，基本工资，税和奖金）：



输入“3”：查找员工基本工资信息（根据名字查找）：



输入“4”：删除员工信息（根据名字删除）：



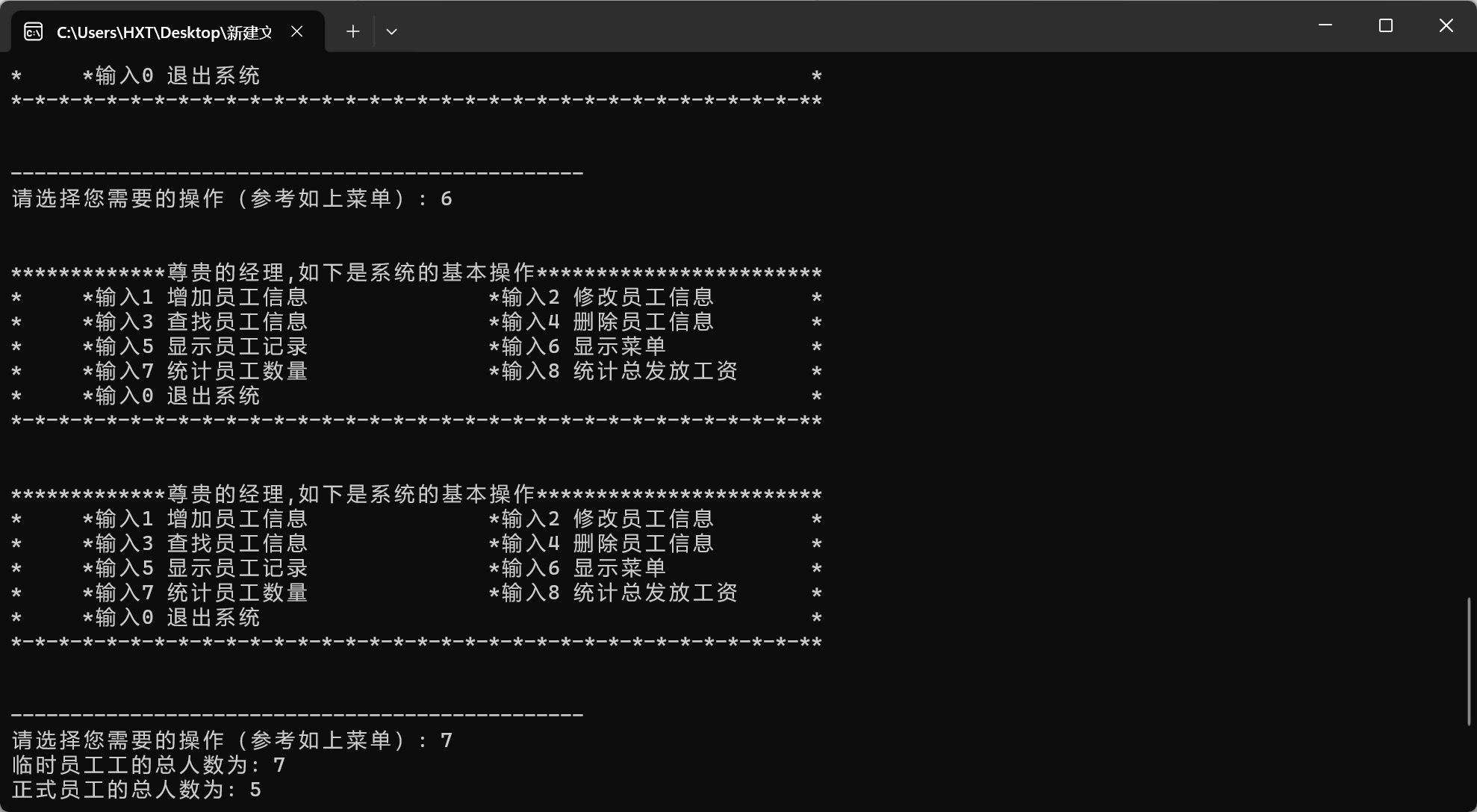
输入“5”：显示所有员工的工资信息：



输入“6”：显示菜单：



输入“7”：统计并输出员工数量：



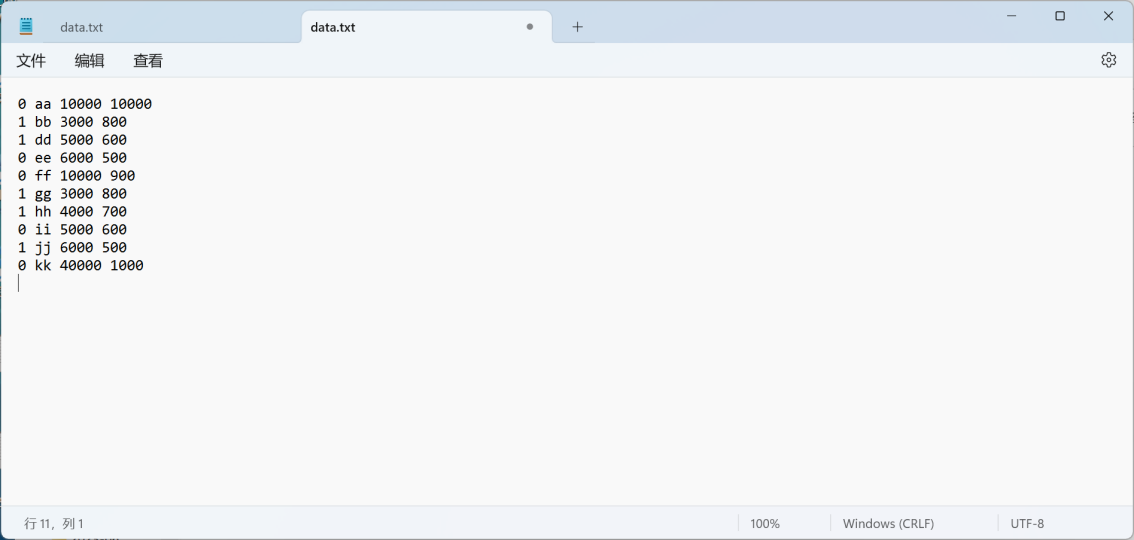
输入“8”：统计并输出总发放工资：



输入“0”：退出系统



最后文件data.txt文件中的数据已被更改



**五、问题及解决方法**

问题1：

添加新员工的时候容易覆盖掉之前的员工信息，造成错误

解决1：定义一个全局变量cnt，用来统计员工的人数，这样在添加新员工是数组角标从cnt开始，避免了覆盖。

问题2：

修改员工信息的时候需要判断是正式员工还是临时员工，对他们的信息进行不同的修改，不加判断会混淆。

解决2：

使用dynamic\_cast函数把基类转化为派生类，如果失败则转化为NULL，让p1等于转化为正式工p2转化为临时工，判断NULL来判断是临时工还是正式工。

**六、总结**

在编写程序时，将各个功能模块化有助于提高代码的可读性和可维护性。每个函数和模块应该专注于实现特定的功能，这样在出现问题时能更容易定位和解决错误。模块化编程还有助于团队合作，因为不同的开发者可以独立地工作在不同的模块上，最后将它们整合在一起。

通过模块化编程，我们可以更好地组织代码，使得代码结构更加清晰，易于理解和修改。当一个模块出现问题时，我们只需要关注这个模块的代码，而不必深入整个程序的逻辑，这样能够大大加快故障排查和修复的速度。

此外，模块化编程也有利于代码的复用。当我们需要实现类似的功能时，可以直接引用已经编写好的模块，而不必重复编写相同的代码，这样可以提高开发效率并减少出错的可能性。

总之，模块化编程是一种良好的编程习惯，可以使代码更加清晰、易于维护和复用，有助于提高开发效率和代码质量。因此，在编写程序时，我们应该尽量去将各个功能模块化，这样能够更好地管理和维护我们的代码。

**七、源代码**

#include<bits/stdc++.h>

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

int cnt=0;

class Employee //创建Employee类

{

protected:

string name; // 员工姓名

double BasicWage; // 员工的基本工资

public:

bool flag; // 0表示不存在,不可以访问; 1表示存在,可以访问

Employee(); //无参数的构造函数

Employee(string name, double BasicWage); //带有参数的构造函数

virtual ~Employee(); //析构函数

double getBasicSalary(); //显示基本工资函数

string getName(); //显示名字函数

void setName(string name); //修改名字函数

void setBasicWage(double wage);

virtual void show(); //输出对应职工的属性

virtual void showCount(); //输出对应职工的人数

virtual void showSalary(); //输出对应职工的总工资、平均工资

};

Employee::Employee()

{

}

Employee::Employee(string name, double BasicWage)

{

this->name = name;

this->BasicWage = BasicWage;

}

Employee::~Employee() {

}

double Employee::getBasicSalary()

{

return BasicWage;

}

string Employee::getName()

{

return name;

}

void Employee::setName(string name)

{

this->name = name;

}

void Employee::setBasicWage(double wage){

this->BasicWage=wage;

}

void Employee::show() {

}

void Employee::showCount() {

}

void Employee::showSalary(){

}

class TemporaryEmployee : public Employee //创建临时员工类,继承Employee类

{

private:

double bonus; // 奖金

double realSalary; // 实发工资

static int tempCount; // 静态数据类型: 临时员工总人数

static double totalSalary; // 静态数据类型: 临时员工总工资

public:

TemporaryEmployee();

TemporaryEmployee(string name, double BasicWage, double bonus);

virtual ~TemporaryEmployee();

double getBonus();

double getRealSalary(); // 获得临时工的实发工资

void show(); // 输出临时工的姓名、基本工资、所得税、实发工资

void showCount(); // 输出临时员工的总数

void showSalary(); // 输出临时员工的总工资、平均工资

void setbonus(double bon);

void setrealSalary();

};

int TemporaryEmployee::tempCount = 0; //定义临时员工的人数赋初值为0

double TemporaryEmployee::totalSalary = 0;// 定义临时员工的总工资赋初值为0

TemporaryEmployee::TemporaryEmployee()

{

this->bonus = 0;

this->realSalary = getBasicSalary();

tempCount++; // 临时员工个数加1

totalSalary += 0;

}

TemporaryEmployee::TemporaryEmployee(string name, double BasicWage, double bonus) : Employee(name, BasicWage)

{

this->bonus = bonus;

this->realSalary = getBasicSalary() + bonus;

tempCount++;

totalSalary += realSalary;

}

TemporaryEmployee::~TemporaryEmployee()

{

tempCount--; // 临时员工个数减1

totalSalary -= realSalary;

}

double TemporaryEmployee::getBonus()

{

return bonus;

}

double TemporaryEmployee::getRealSalary()

{

return realSalary;

}

void TemporaryEmployee::show()

{

cout << getName() << "\t" << getBasicSalary() << "\t" << getBonus() << "\t" << getRealSalary() << endl;

}

void TemporaryEmployee::showCount()

{

cout << "临时职工的总人数为: " << tempCount << endl;

}

void TemporaryEmployee::showSalary()

{

cout << "临时职工的总工资为: " << totalSalary << endl;

cout << "临时职工的平均工资为: " << totalSalary / tempCount << endl << endl;

}

ostream& operator<<(ostream& cout, TemporaryEmployee& t) //重载左移运算符 (用于输出一个对象的信息)

{

cout << t.getName() << "\t" << t.getBasicSalary() << "\t"

<< t.getBonus() << "\t" << t.getRealSalary() << endl;

return cout;

}

void TemporaryEmployee::setbonus(double bon){

this->bonus=bon;

}

void TemporaryEmployee::setrealSalary(){

this->realSalary=this->BasicWage+this->bonus;

}

class RegularEmployees : public Employee

{

private:

double tax; // 税收

double realSalary; // 实发工资

static int formalCount; // 静态数据类型: 正式职工总人数

static double totalSalary; // 静态数据类型: 正式职工总工资、平均工资

public:

RegularEmployees();

RegularEmployees(string name, double BasicWage, double tax);

virtual ~RegularEmployees();

double getTax();

double getRealSalary();

void show(); // 输出正式职工的姓名、基本工资、所得税、实发工资

void showCount(); // 输出正式职工的总数

void showSalary(); // 输出正式职工的总工资、平均工资

void settax(double tax);

void setrealSalary();

};

int RegularEmployees::formalCount = 0; // 定义临时员工的人数赋初值为0

double RegularEmployees::totalSalary = 0; //定义临时员工的总工资赋初值为0

RegularEmployees::RegularEmployees()

{

this->tax = 0;

this->realSalary = getBasicSalary();

formalCount++;

totalSalary += 0;

}

RegularEmployees::RegularEmployees(string name, double BasicWage, double tax) : Employee(name, BasicWage)

{

this->tax = tax;

this->realSalary = getBasicSalary() - tax;

formalCount++;

totalSalary += realSalary;

}

RegularEmployees::~RegularEmployees()

{

formalCount--;

totalSalary -= realSalary;

}

double RegularEmployees::getRealSalary()

{

return realSalary;

}

double RegularEmployees::getTax()

{

return tax;

}

void RegularEmployees::show()

{

cout << getName() << "\t" << getBasicSalary() << "\t" << getTax() << "\t" << getRealSalary() << endl;

}

void RegularEmployees::showCount()

{

cout << "正式职工的总人数为: " << formalCount << endl;

}

void RegularEmployees::showSalary()

{

cout << "正式职工的总工资为: " << totalSalary << endl;

cout << "正式职工的平均工资为: " << totalSalary / formalCount << endl << endl;

}

ostream& operator<<(ostream& cout, RegularEmployees& f) //重载左移运算符 (用于输出一个对象的信息)

{

cout << f.getName() << "\t" << f.getBasicSalary() << "\t"

<< f.getTax() << "\t" << f.getRealSalary() << endl;

return cout;

}

void RegularEmployees::settax(double tax){

this->tax=tax;

}

void RegularEmployees::setrealSalary(){

this->realSalary=this->BasicWage-this->tax;

}

const int MAX = 100; // 职工的最大人数

void Menu(); // 菜单

void output(Employee\* pEmployee[], int num); // 输出员工工资信息

void queryEmployeeByName(Employee\* pEmployee[], int num); // 根据姓名来查找员工信息

void modifyEmployeeByName(Employee\* pEmployee[], int num); // 根据姓名来修改员工信息

void clearEmployeeByName(Employee\* pEmployee[], int& num); // 根据姓名来删除员工信息

void deleteAllEmployee(Employee\* pEmployee[], int num); // 用循环来删除指针数组

void showEmployeeNum(Employee\* pEmployee[], int num); // 显示两种员工的人数

void showEmployeeSalary(Employee\* pEmployee[], int num); // 显示员工总的实发工资、平均实发工资

void input(Employee\* pEmployee[], int n); //添加数据

void Init(Employee\* pEmployee[]); //读取文件

void write(Employee\* pEmployee[]); //写入文件

int main()

{

int n;

cout<<"------尊贵的经理,欢迎来到员工工资信息管理系统------"<<endl;

Employee\* pEmployee[MAX];

memset(pEmployee,0,MAX); // 将pEmployee的现有元素个数调至num个, 其值为0

Init(pEmployee);

int ch;

while (true)

{

Menu();

cout << "\n\n------------------------------------------------";

cout << "\n请选择您需要的操作（参考如上菜单）: ";

cin >> ch;

switch (ch)

{

case 1:

cout<<"请输入要添加员工的人数:"<<endl;

cin>>n;

input(pEmployee, n);

break;

case 2:

modifyEmployeeByName(pEmployee, cnt);

break;

case 3:

queryEmployeeByName(pEmployee, cnt);

system("pause");

break;

case 4:

clearEmployeeByName(pEmployee, cnt);

break;

case 5:

output(pEmployee, cnt);

break;

case 6:

Menu();

break;

case 7:

showEmployeeNum(pEmployee, cnt);

break;

case 8:

showEmployeeSalary(pEmployee, cnt);

break;

case 0:

write(pEmployee) ;

deleteAllEmployee(pEmployee, cnt); // 退出程序

cout << "成功退出系统" << endl;

return 0;

default:

cout << "您的输入有误, 请重新输入" << endl;

break;

}

}

return 0;

}

void Init(Employee\* pEmployee[]){ //输入流函数

ifstream in;

in.open("data.txt",ios::in);

int flag;

string name;

double BasicWage,other;

while(!(in.eof())){

in>>flag>>name>>BasicWage>>other;

if(flag){

pEmployee[cnt]=new RegularEmployees(name,BasicWage,other);

}else{

pEmployee[cnt]=new TemporaryEmployee(name,BasicWage,other);

}

pEmployee[cnt]->flag=1;

cnt++;

}

in.close();

}

void write(Employee\* pEmployee[]){ //写数据函数

ofstream out;

out.open("data.txt",ios::out);

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

if (pEmployee[i]->flag)

{

RegularEmployees\* p1 = dynamic\_cast<RegularEmployees\*>(pEmployee[i]);

TemporaryEmployee\* p2 = dynamic\_cast<TemporaryEmployee\*>(pEmployee[i]);

if (p1 != NULL && p2 == NULL)

{

out<<1<<" "<<p1->getName()<<" "<<p1->getBasicSalary()<<" "<<p1->getTax();

}

else if (p1 == NULL && p2 != NULL)

{

out<<0<<" "<<p2->getName()<<" "<<p2->getBasicSalary()<<" "<<p2->getBonus();

}

else

{

cout << "写入文件失败" << endl;

exit(1);

}

out<<endl;

}

}

}

void Menu() //提示菜单

{

cout << endl << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*尊贵的经理,如下是系统的基本操作\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

cout << "\* \*输入1 增加员工信息 \*输入2 修改员工信息 \*" << endl;

cout << "\* \*输入3 查找员工信息 \*输入4 删除员工信息 \*" << endl;

cout << "\* \*输入5 显示员工记录 \*输入6 显示菜单 \*" << endl;

cout << "\* \*输入7 统计员工数量 \*输入8 统计总发放工资 \*" << endl;

cout << "\* \*输入0 退出系统 \*" << endl;

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

}

void input(Employee\* pEmployee[], int n) //输入员工信息

{

string name; // 员工姓名

double BasicWage; // 员工基本工资

double tax; // 员工税收

double bonus; // 员工奖金

char choice;

cout << "请正确输入员工的类型 0表示临时员工、1表示正式员工（）: \n" << endl;

for (int i = cnt; i < cnt+n; i++)

{

cin >> choice;

if (choice == '1')

{

cin >> name >> BasicWage >> tax;

pEmployee[i] = new RegularEmployees(name, BasicWage, tax);

}

else if (choice == '0')

{

cin >> name >> BasicWage >> bonus;

pEmployee[i] = new TemporaryEmployee(name, BasicWage, bonus);

}

else

{

cout << "您的输入有误!!!" << endl;

pEmployee[i] = new TemporaryEmployee();

}

pEmployee[i]->flag = true; // 可以访问

}

cnt+=n;

}

void output(Employee\* pEmployee[], int num) // 输出员工信息

{

cout << endl << endl;

cout << "姓名 基本工资 税收或奖金 实际工资" << endl;

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

{

if (pEmployee[i]->flag)

{

RegularEmployees\* p1 = dynamic\_cast<RegularEmployees\*>(pEmployee[i]);

TemporaryEmployee\* p2 = dynamic\_cast<TemporaryEmployee\*>(pEmployee[i]);

if (p1 != NULL && p2 == NULL)

{

cout << \*p1;

}

else if (p1 == NULL && p2 != NULL)

{

cout << \*p2;

}

else

{

cout << "输出错误！" << endl;

exit(1);

}

}

}

}

void queryEmployeeByName(Employee\* pEmployee[], int num) //根据姓名来查找员工信息

{

string queryName;

cout << "\n请输入要查询的员工的姓名: " << endl;

cin >> queryName;

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

{

if (pEmployee[i]->getName() == queryName && pEmployee[i]->flag)

{

cout << "员工姓名为: " << queryName << ", 员工基本工资是: " << pEmployee[i]->getBasicSalary() << endl;

return; // 查找成功

}

}

cout << "查找失败 !!!" << endl;

}

void modifyEmployeeByName(Employee\* pEmployee[], int num) // 根据姓名来修改员工信息

{

string queryName, newName;

cout << "\n请输入要修改的员工的姓名: " << endl;

cin >> queryName;

int flag;

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

{

if (pEmployee[i]->getName() == queryName && pEmployee[i]->flag) // 查找成功

{

RegularEmployees\* p1 = dynamic\_cast<RegularEmployees\*>(pEmployee[i]);

TemporaryEmployee\* p2 = dynamic\_cast<TemporaryEmployee\*>(pEmployee[i]);

if (p1 != NULL && p2 == NULL)

{

cout << endl << endl;

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

cout << "\* \*输入1 修改名字 \*" << endl;

cout << "\* \*输入2 修改基础工资 \*" << endl;

cout << "\* \*输入3 修改所得税 \*" << endl;

cout << "\* \*输入0 退出系统 \*" << endl;

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

cin>>flag;

string name;

double basewage;

double tax;

switch(flag){

case 1:

cout<<"请输入新的名字"<<endl;

cin>>name;

p1->setName(name);

break;

case 2:

cout<<"请输入新的基本工资"<<endl;

cin>>basewage;

p1->setBasicWage(basewage);

p1->setrealSalary();

break;

case 3:

cout<<"请输入新的所得税数额"<<endl;

cin>>tax;

p1->settax(tax);

p1->setrealSalary();

break;

break;

case 0:

return;

break;

}

}

else if (p1 == NULL && p2 != NULL)

{

cout << endl << endl;

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

cout << "\* \*输入1 修改名字 \*" << endl;

cout << "\* \*输入2 修改基础工资 \*" << endl;

cout << "\* \*输入3 修改奖金 \*" << endl;

cout << "\* \*输入0 退出系统 \*" << endl;

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

cin>>flag;

string name;

double basewage;

double bonus;

switch(flag){

case 1:

cout<<"请输入新的名字"<<endl;

cin>>name;

p2->setName(name);

break;

case 2:

cout<<"请输入新的基本工资"<<endl;

cin>>basewage;

p2->setBasicWage(basewage);

p2->setrealSalary();

break;

case 3:

cout<<"请输入新的奖金数额"<<endl;

cin>>bonus;

p2->setbonus(bonus);

p2->setrealSalary();

break;

break;

case 0:

return;

break;

}

}

else

{

cout << "输出错误！" << endl;

exit(1);

}

return; // 查找成功, 返回

}

}

cout << "查找失败, 该姓名无法修改!" << endl;

}

void clearEmployeeByName(Employee\* pEmployee[], int& num) // 根据姓名来删除员工信息

{

string Name;

cout << "\n请输入要删除的员工的姓名: " << endl;

cin >> Name;

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

{

if (pEmployee[i]->getName() == Name) // 查找成功

{

for(int j=i;j<num-1;j++) // 删除pEmployee[i]元素

{

pEmployee[j]=pEmployee[j+1];

}

num--;

cnt--;

return; // 查找成功, 返回

}

}

cout << "查无此人, 无法删除!" << endl;

}

void showEmployeeNum(Employee\* pEmployee[], int num) //显示各种员工人数

{

int formalEmployeeNum = 0;

int tempEmployeeNum = 0;

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

{

if (pEmployee[i]->flag)

{

RegularEmployees\* p1 = dynamic\_cast<RegularEmployees\*>(pEmployee[i]);

TemporaryEmployee\* p2 = dynamic\_cast<TemporaryEmployee\*>(pEmployee[i]);

if (p1 != NULL && p2 == NULL)

{

formalEmployeeNum++;

}

else if (p1 == NULL && p2 != NULL)

{

tempEmployeeNum++;

}

else

{

cout << "输出错误！" << endl;

exit(1);

}

}

}

cout << "临时员工工的总人数为: " << tempEmployeeNum << endl;

cout << "正式员工的总人数为: " << formalEmployeeNum << endl;

}

void showEmployeeSalary(Employee\* pEmployee[], int num) // 显示员工总的实发工资、平均实发工资

{

int formalEmployeeNum = 0;

int tempEmployeeNum = 0;

int formalEmployeeSalary = 0;

int tempEmployeeSalary = 0;

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

{

if (pEmployee[i]->flag)

{

RegularEmployees\* p1 = dynamic\_cast<RegularEmployees\*>(pEmployee[i]);

TemporaryEmployee\* p2 = dynamic\_cast<TemporaryEmployee\*>(pEmployee[i]);

if (p1 != NULL && p2 == NULL)

{

formalEmployeeNum++;

formalEmployeeSalary += p1->getRealSalary();

}

else if (p1 == NULL && p2 != NULL)

{

tempEmployeeNum++;

tempEmployeeSalary += p2->getRealSalary();

}

else

{

cout << "输出错误！" << endl;

exit(1);

}

}

}

cout << "临时员工的总的实际发放的工资: " << tempEmployeeSalary << endl;

cout << "临时员工的平均实际发放工资: " << tempEmployeeSalary / tempEmployeeNum << endl;

cout << "正式员工的总实实际发放工资: " << formalEmployeeSalary << endl;

cout << "正式员工的平均实际发放工资: " << formalEmployeeSalary / formalEmployeeNum << endl;

}

void deleteAllEmployee(Employee\* pEmployee[], int num) //用循环来删除指针数组

{

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

{

delete pEmployee[i];

}

}