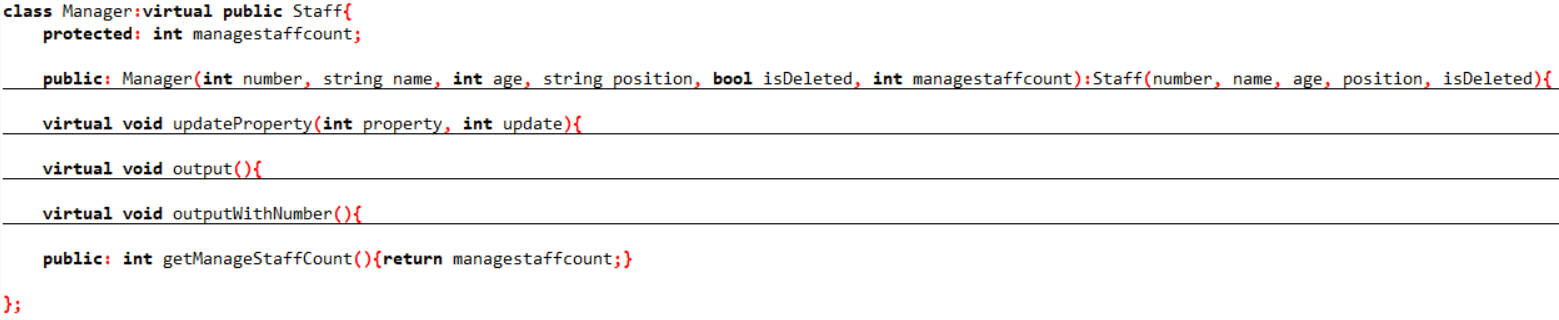
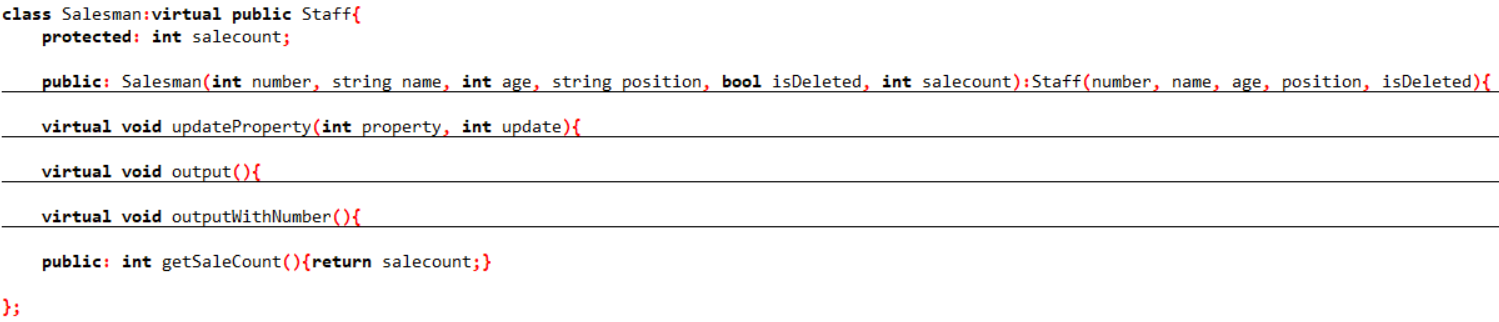
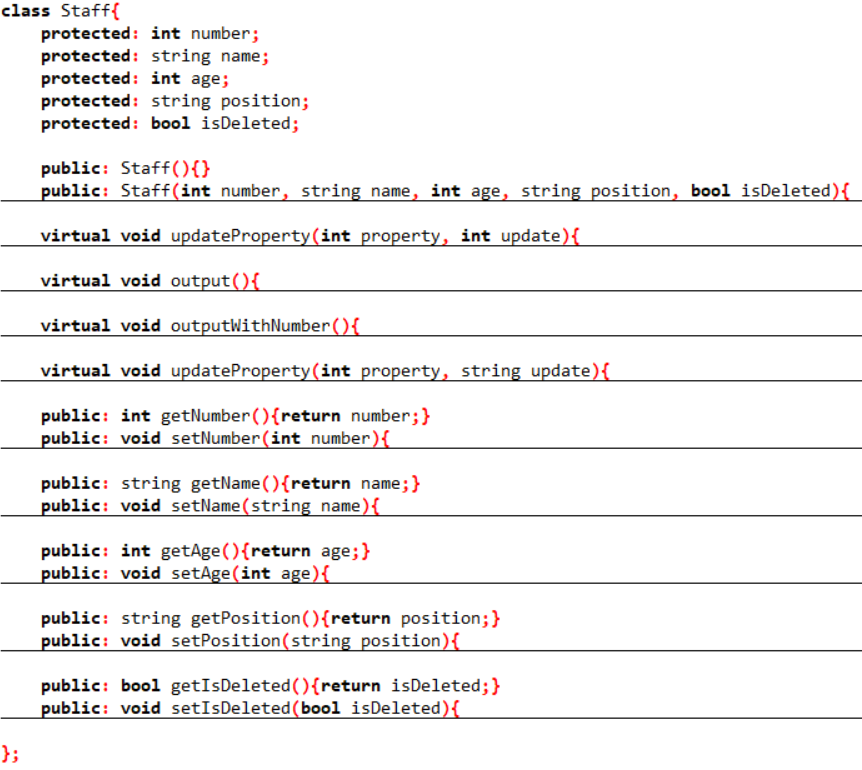
# 题目要求

1. 设计一个虚基类员工类Staff，包含编号、姓名和年龄等保护数据成员以及相关的成员函数。
2. 由Staff派生出销售员类Salesman，包含销售额等保护数据成员以及相关的成员函数。
3. 由Staff派生出经理类Manager。
4. 由Salesman和Manager派生出销售经理类SalesManager。
5. 利用文件处理方式实现对公司人员（包括销售员、经理和销售经理）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。（删除数据在记录中做删除标志，重组文件指在物理上删除有删除标志的记录）另外还要分别统计这三类员工的人数以及所有员工的总数。

# 设计思路

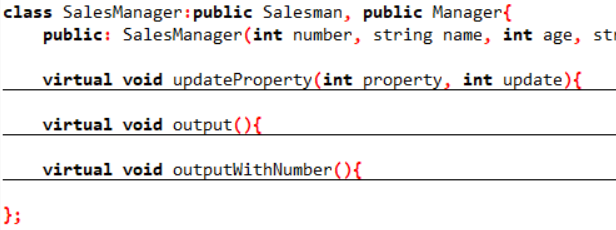
1. 分析题目要求，分别建立了四个类：虚基类Staff、公有继承类Salesman、公有继承类Manager和公有继承类SalesManager。
2. 将所有员工的记录都放在同一个txt文件中，每个员工记录占用一行。
3. 文件在程序一开始全部读入vector容器中，并在程序结束前统一写回文件。
4. 所有字符串统一采用string类，从而简化程序设计。
5. 在虚基类Staff中定义虚函数updateProperty（更新属性值函数）、虚函数output（输出函数）和虚函数outputWithNumber（输出函数），并在派生类中进行了覆盖以达到动态多态性。
6. 使用虚基类和虚函数的特性，利用动态联编使基类指针所要调用的派生类函数。
7. 为了在主程序少写if-else语句块和switch语句块，将要更新的属性类型和属性值均作为参数传入函数updateProperty（更新属性值函数）并对该函数进行了重载。

# 类的UML图



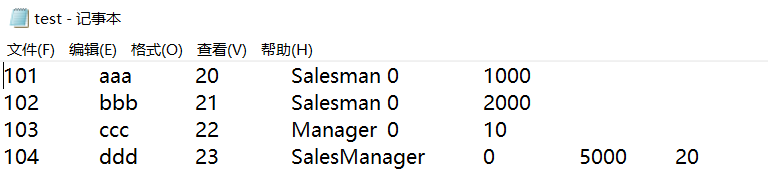
virtual

virtual

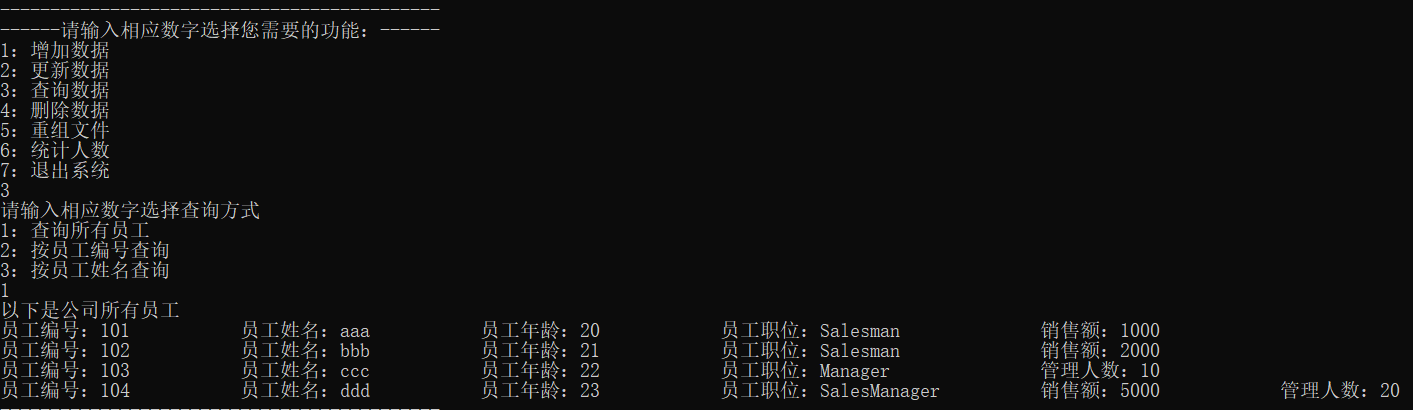


# 运行结果

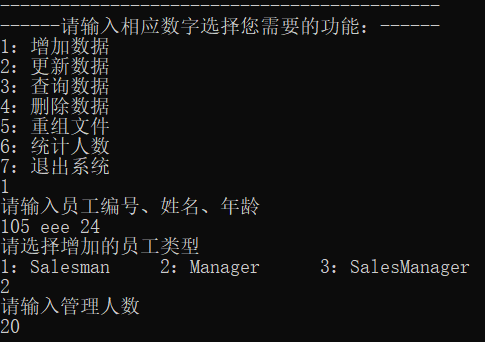
## 文件初始内容



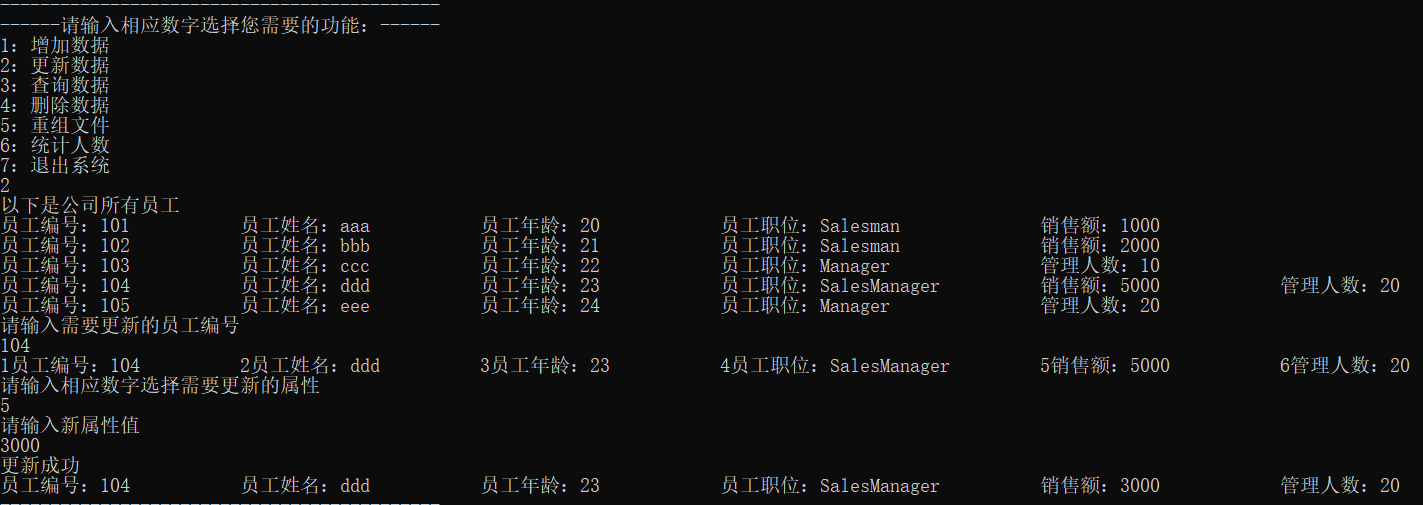
## 查询未修改前的数据



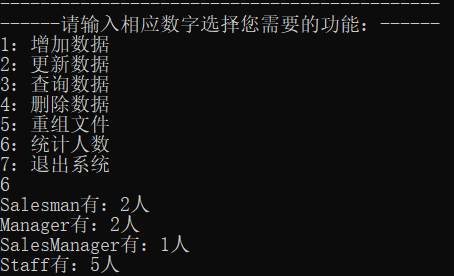
## 添加数据



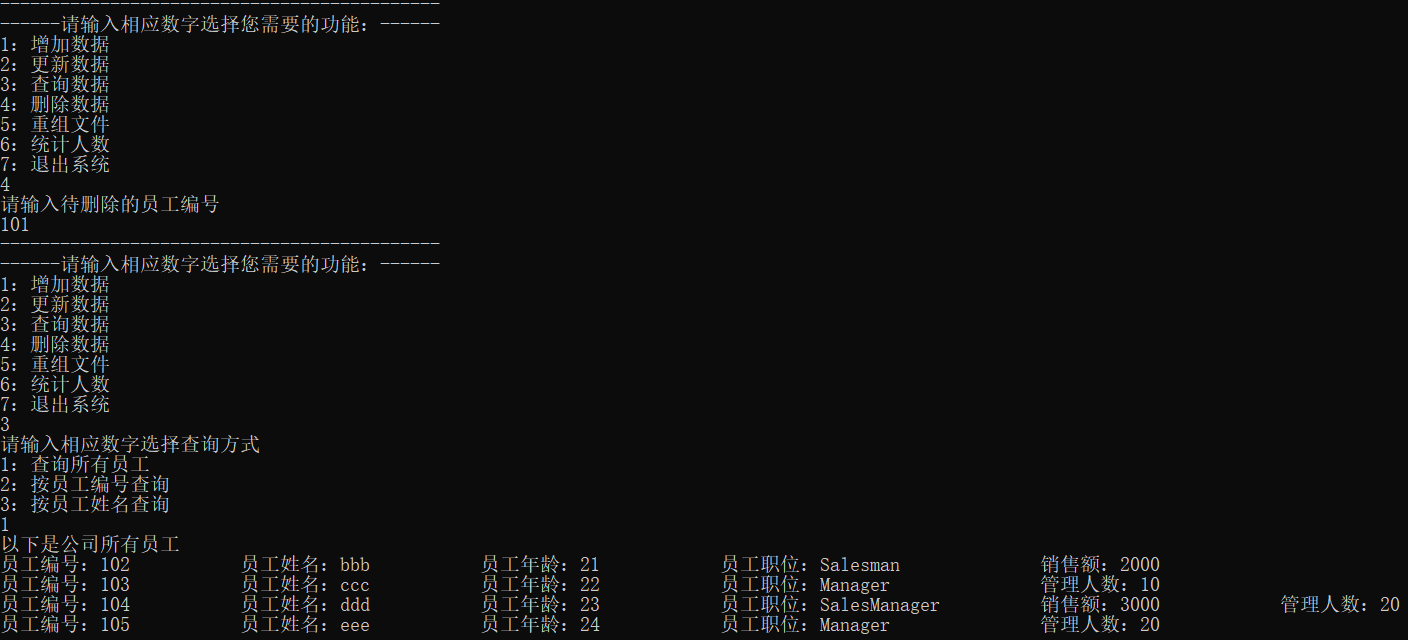
## 更新数据



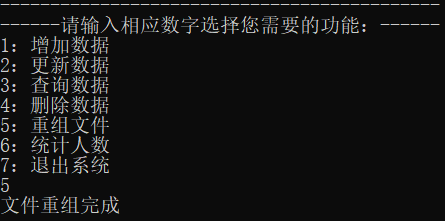
## 统计人数



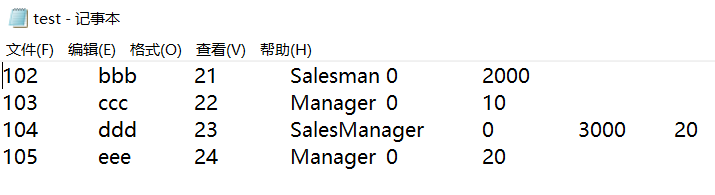
## 删除数据



## 重组文件



## 再次查询文件内容



# 遇到的问题及解决方法

1. 利用fstream读入文件时，总是会把文件内容最后一行读入两次。经过查阅资料和调试后发现是因为在最后一行后还有回车，所以文件的结束标志EOF在文件内容的下一行。该问题实际上是因为最后一行没有数据造成读入失败，但是在创建对象时用了上一次赋值给变量的数值。最终通过fstream的fail()方法来判断读入是否成功从而解决该问题。
2. 在将vector容器中的内容写回文件时发现基类指针无法调用派生类的函数。将基类指针强制转换为派生类指针后，编译无法通过。经查阅资料后发现，C++必须使用dynamic\_cast关键字进行动态类型转换，问题得以解决。
3. 第一次设计时，所有字符串均采用char类型数组，对其进行各种操作（赋值、更新等）都非常繁琐，后经查阅资料发现C++使用string类代替C中的字符串，赋值仅需使用“=”即可，大大简化了代码。

附：源程序

#include<iostream>

#include<fstream>

#include<vector>

#include<string>

using namespace std;

class Staff{

protected: int number;

protected: string name;

protected: int age;

protected: string position;

protected: bool isDeleted;

public: Staff(){}

public: Staff(int number, string name, int age, string position, bool isDeleted){

this->number = number;

this->name = name;

this->age = age;

this->position =position;

this->isDeleted = isDeleted;

}

virtual void updateProperty(int property, int update){

if(property == 1)number = update;

else if(property == 3)age = update;

}

virtual void output(){

cout<<"员工编号："<<number;

cout<<"\t\t员工姓名："<<name;

cout<<"\t\t员工年龄："<<age;

cout<<"\t\t员工职位："<<position<<endl;

}

virtual void outputWithNumber(){

cout<<"1员工编号："<<number;

cout<<"\t\t2员工姓名："<<name;

cout<<"\t\t3员工年龄："<<age;

cout<<"\t\t4员工职位："<<position<<endl;

}

virtual void updateProperty(int property, string update){

if(property == 2)name = update;

else if(property == 4)position = update;

}

public: int getNumber(){return number;}

public: void setNumber(int number){

this->number = number;

}

public: string getName(){return name;}

public: void setName(string name){

this->name = name;

}

public: int getAge(){return age;}

public: void setAge(int age){

this->age = age;

}

public: string getPosition(){return position;}

public: void setPosition(string position){

this->position = position;

}

public: bool getIsDeleted(){return isDeleted;}

public: void setIsDeleted(bool isDeleted){

this->isDeleted = isDeleted;

}

};

class Salesman:virtual public Staff{

protected: int salecount;

public: Salesman(int number, string name, int age, string position, bool isDeleted, int salecount):Staff(number, name, age, position, isDeleted){

this->salecount = salecount;

}

virtual void updateProperty(int property, int update){

Staff::updateProperty(property, update);

if(property == 5)salecount = update;

}

virtual void output(){

cout<<"员工编号："<<number;

cout<<"\t\t员工姓名："<<name;

cout<<"\t\t员工年龄："<<age;

cout<<"\t\t员工职位："<<position;

cout<<"\t\t销售额："<<salecount<<endl;

}

virtual void outputWithNumber(){

cout<<"1员工编号："<<number;

cout<<"\t\t2员工姓名："<<name;

cout<<"\t\t3员工年龄："<<age;

cout<<"\t\t4员工职位："<<position;

cout<<"\t\t5销售额："<<salecount<<endl;

}

public: int getSaleCount(){return salecount;}

};

class Manager:virtual public Staff{

protected: int managestaffcount;

public: Manager(int number, string name, int age, string position, bool isDeleted, int managestaffcount):Staff(number, name, age, position, isDeleted){

this->managestaffcount = managestaffcount;

}

virtual void updateProperty(int property, int update){

Staff::updateProperty(property, update);

if(property == 6)managestaffcount = update;

}

virtual void output(){

cout<<"员工编号："<<number;

cout<<"\t\t员工姓名："<<name;

cout<<"\t\t员工年龄："<<age;

cout<<"\t\t员工职位："<<position;

cout<<"\t\t管理人数："<<managestaffcount<<endl;

}

virtual void outputWithNumber(){

cout<<"1员工编号："<<number;

cout<<"\t\t2员工姓名："<<name;

cout<<"\t\t3员工年龄："<<age;

cout<<"\t\t4员工职位："<<position;

cout<<"\t\t6管理人数："<<managestaffcount<<endl;

}

public: int getManageStaffCount(){return managestaffcount;}

};

class SalesManager:public Salesman, public Manager{

public: SalesManager(int number, string name, int age, string position, bool isDeleted, int salecount, int managestaffcount):Staff(number, name, age, position, isDeleted), Salesman(number, name, age, position, isDeleted, salecount), Manager(number, name, age, position, isDeleted, managestaffcount){}

virtual void updateProperty(int property, int update){

Staff::updateProperty(property, update);

if(property == 5)salecount = update;

else if(property == 6)managestaffcount = update;

}

virtual void output(){

cout<<"员工编号："<<number;

cout<<"\t\t员工姓名："<<name;

cout<<"\t\t员工年龄："<<age;

cout<<"\t\t员工职位："<<position;

cout<<"\t\t销售额："<<salecount;

cout<<"\t\t管理人数："<<managestaffcount<<endl;

}

virtual void outputWithNumber(){

cout<<"1员工编号："<<number;

cout<<"\t\t2员工姓名："<<name;

cout<<"\t\t3员工年龄："<<age;

cout<<"\t\t4员工职位："<<position;

cout<<"\t\t5销售额："<<salecount;

cout<<"\t\t6管理人数："<<managestaffcount<<endl;

}

};

int findByNumber(vector<Staff\*> obj, int number, int sum){

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

if(obj[i]->getNumber() == number)return i;

}

return -1;

}

int findByName(vector<Staff\*> obj, string name, int sum){

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

if(obj[i]->getName() == name)return i;

}

return -1;

}

int main(){

ifstream infile;

infile.open("test.txt", ios::in);

int number;

string name;

int age;

string position;

bool isDeleted;

int salecount;

int managestaffcount;

int sum = 0;

vector<Staff\*> obj;

int id;

int property;

int choice;

while(!infile.eof()){

infile>>number>>name>>age>>position>>isDeleted;

if(infile.fail())break;

else if(position == "Staff"){

Staff\* temp = new Staff(number, name, age, position, isDeleted);

obj.push\_back(temp);

sum++;

}

else if(position == "Salesman"){

infile>>salecount;

Salesman\* temp = new Salesman(number, name, age, position, isDeleted, salecount);

obj.push\_back(temp);

sum++;

}

else if(position == "Manager"){

infile>>managestaffcount;

Manager\* temp = new Manager(number, name, age, position, isDeleted, managestaffcount);

obj.push\_back(temp);

sum++;

}

else if(position == "SalesManager"){

infile>>salecount>>managestaffcount;

SalesManager\* temp = new SalesManager(number, name, age, position, isDeleted, salecount, managestaffcount);

obj.push\_back(temp);

sum++;

}

}

infile.close();

int mode = 0;

while(mode != 8){

cout<<"--------------------------------------------"<<endl;

cout<<"------请输入相应数字选择您需要的功能：------"<<endl;

cout<<"1：增加数据"<<endl;

cout<<"2：更新数据"<<endl;

cout<<"3：查询数据"<<endl;

cout<<"4：删除数据"<<endl;

cout<<"5：重组文件"<<endl;

cout<<"6：统计人数"<<endl;

cout<<"7：退出系统"<<endl;

cin>>mode;

switch(mode){

case 1:{

cout<<"请输入员工编号、姓名、年龄"<<endl;

cin>>number>>name>>age;

cout<<"请选择增加的员工类型"<<endl;

cout<<"1：Salesman 2：Manager 3：SalesManager"<<endl;

cin>>choice;

switch(choice){

case 1:{

cout<<"请输入销售额"<<endl;

cin>>salecount;

Salesman\* temp = new Salesman(number, name, age, "Salesman", false, salecount);

obj.push\_back(temp);

sum++;

break;

}

case 2:{

cout<<"请输入管理人数"<<endl;

cin>>managestaffcount;

Manager\* temp = new Manager(number, name, age, "Manager", false, managestaffcount);

obj.push\_back(temp);

sum++;

break;

}

case 3:{

cout<<"请输入销售额"<<endl;

cin>>salecount;

cout<<"请输入管理人数"<<endl;

cin>>managestaffcount;

SalesManager\* temp = new SalesManager(number, name, age, "SalesManager", false, salecount, managestaffcount);

obj.push\_back(temp);

sum++;

break;

}

}

break;

}

case 2:{

cout<<"以下是公司所有员工"<<endl;

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

obj[i]->output();

}

cout<<"请输入需要更新的员工编号"<<endl;

cin>>number;

id = findByNumber(obj, number, sum);

obj[id]->outputWithNumber();

cout<<"请输入相应数字选择需要更新的属性"<<endl;

cin>>property;

if(property == 2 || property == 4){

cout<<"请输入新属性值"<<endl;

string update;

cin>>update;

obj[id]->updateProperty(property, update);

}

else{

cout<<"请输入新属性值"<<endl;

int update;

cin>>update;

obj[id]->updateProperty(property, update);

}

cout<<"更新成功"<<endl;

obj[id]->output();

break;

}

case 3:{

cout<<"请输入相应数字选择查询方式"<<endl;

cout<<"1：查询所有员工"<<endl;

cout<<"2：按员工编号查询"<<endl;

cout<<"3：按员工姓名查询"<<endl;

cin>>choice;

switch(choice){

case 1:{

cout<<"以下是公司所有员工"<<endl;

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

if(obj[i]->getIsDeleted() != true){

obj[i]->output();

}

}

break;

}

case 2:{

cout<<"请输入待查询的员工编号"<<endl;

cin>>number;

id = findByNumber(obj, number, sum);

obj[id]->output();

break;

}

case 3:{

cout<<"请输入待查询的员工姓名"<<endl;

cin>>name;

id = findByName(obj, name, sum);

obj[id]->output();

break;

}

}

break;

}

case 4:{

cout<<"请输入待删除的员工编号"<<endl;

cin>>number;

id = findByNumber(obj, number, sum);

obj[id]->setIsDeleted(true);

break;

}

case 5:{

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

if(obj[i]->getIsDeleted() == true){

obj.erase(obj.begin()+i);

sum--;

}

}

cout<<"文件重组完成"<<endl;

break;

}

case 6:{

int salesmansum = 0;

int managersum = 0;

int salesmanagersum = 0;

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

if(obj[i]->getPosition() == "Salesman")salesmansum++;

else if(obj[i]->getPosition() == "Manager")managersum++;

else if(obj[i]->getPosition() == "SalesManager")salesmanagersum++;

}

cout<<"Salesman有："<<salesmansum<<"人"<<endl;

cout<<"Manager有："<<managersum<<"人"<<endl;

cout<<"SalesManager有："<<salesmanagersum<<"人"<<endl;

cout<<"Staff有："<<sum<<"人"<<endl;

break;

}

case 7:{

mode = 8;

break;

}

}

}

// 可以覆盖原有数据

ofstream outfile;

outfile.open("test.txt", ios::out);

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

if(obj[i]->getPosition() == "Staff"){

outfile<<obj[i]->getNumber()<<"\t"<<obj[i]->getName()<<"\t"<<obj[i]->getAge()<<"\t"<<obj[i]->getPosition()<<"\t"<<obj[i]->getIsDeleted()<<endl;

}

else if(obj[i]->getPosition() == "Salesman"){

Salesman\* temp = dynamic\_cast<Salesman\*>(obj[i]);

outfile<<temp->getNumber()<<"\t"<<temp->getName()<<"\t"<<temp->getAge()<<"\t"<<temp->getPosition()<<"\t"<<temp->getIsDeleted()<<"\t"<<temp->getSaleCount()<<endl;

}

else if(obj[i]->getPosition() == "Manager"){

Manager\* temp = dynamic\_cast<Manager\*>(obj[i]);

outfile<<temp->getNumber()<<"\t"<<temp->getName()<<"\t"<<temp->getAge()<<"\t"<<temp->getPosition()<<"\t"<<temp->getIsDeleted()<<"\t"<<temp->getManageStaffCount()<<endl;

}

else if(obj[i]->getPosition() == "SalesManager"){

SalesManager\* temp = dynamic\_cast<SalesManager\*>(obj[i]);

outfile<<temp->getNumber()<<"\t"<<temp->getName()<<"\t"<<temp->getAge()<<"\t"<<temp->getPosition()<<"\t"<<temp->getIsDeleted()<<"\t"<<temp->getSaleCount()<<"\t"<<temp->getManageStaffCount()<<endl;

}

}

}