

## 摘 要

本实验的目的是开发一套职工管理系统，供公司管理职工信息。实验设计是基于 C++ 的多态特性完成运行时的动态绑定，并通过文件操作保存和管理数据。该系统可以完成的功能包括对职工信息的批量增加、显示、修改、删除、查找、排序。

**关键词：**职工管理系统；多态；文件操作

# 目 录

1. 选题及功能描述
2. 分析与设计
3. 程序实现及重难点
  - 3.1 继承关系的实现
  - 3.2 主体程序的编写
  - 3.3 main 函数的调用
  - 3.4 重难点小结
4. 演示（部分功能）
5. 总结
6. 附录

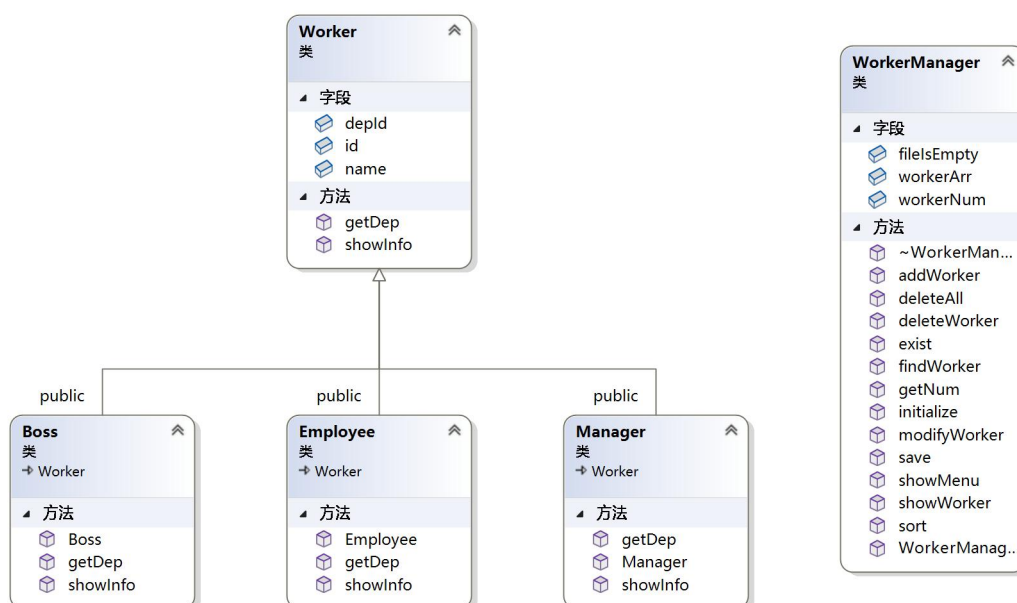
## 1. 选题及功能描述

本次实验选择“职工管理系统”作为选题，该系统的功能包括：

- (1) 添加职工信息：批量添加职工信息，包括编号、姓名、部门编号，并将这些信息录入文本文件
- (2) 显示职工信息：显示已添加的所有职工信息
- (3) 删除职工信息：输入编号，删除该编号的职工信息，删除前有确认操作
- (4) 修改职工信息：输入编号，修改该编号的职工的姓名和部门，修改前有确认操作
- (5) 查找职工信息：按照编号或姓名查找相关人员并输出该人员信息，编号无重复，姓名可以有重复
- (6) 按照编号排序：根据用户选择按照职工编号的升序或者降序排列
- (7) 清空文档内容：清空所有职工信息，清空前有确认操作

选择该项目作为选题是因为实现上述功能体现了 C++ 的多态特性，实现了文件交互功能，涉及到排序算法，包含多处复杂的选择和分支结构，能够很好地夯实对 C++ 的基础学习，锻炼逻辑思维能力。

## 2. 分析与设计



该公司中的职工有三类，分别为 Employee，Manager 和 Boss。为了实现方法调用和数据访问时的动态绑定，需要首先创建一个抽象类 Worker，将其中的成员函数设计成纯虚函数。让上述三类职工继承 Worker 基类并重写其成员函数，这样就可以通过父类指针访问子类成员。

WorkerManager 类是该程序的主体部分。在该类的头文件中声明了一个指针数组用于存放 Worker 类指针，每个指针指向一个具体的对象，通过对该数组及数组内容的操作实现该系统的大部分功能。为了实现文件交互，在 WorkerManager 类的编程中多次使用 fstream 文件流，使用数据的读写功能。

在 main 函数中设计一个 switch 语句，分别调用 WorkerManager 类中封装好的函数，即可实现职工信息管理的各种功能。

### 3. 程序实现及重难点

#### 3.1 继承关系的实现

根据上述的分析与设计，首先创建一个抽象类 Worker，其成员变量包括编号、姓名、部门编号，成员函数包括 getDep（获取部门），showInfo（展示信息），都声明为纯虚函数。

其次分别创建 Employee，Manager 和 Boss 三类职工，继承抽象类 Worker，编写各自的构造函数，重写父类的成员函数，实现“获取部门信息”和“展示信息”的功能。

#### 3.2 主体程序的编写

下面开始编写主体程序：WorkerManager 类。其成员变量包括 workerNum（职工人数）、Worker\*\* workerArr（用于存放 Worker 指针的数组）、bool 型变量 fileIsEmpty（用于判断文本文件是否为空）。由上文的分析可知，数组中存放抽象类指针，就可以通过抽象类指针指向子类对象，实现不同类型员工信息存放在同一个数组中，便于管理。

该类的构造函数实现较为复杂，因为构造函数初始化时存在三种情况：

（1）文件未创建：通过 ifstream 的对象调用 is\_open 函数用于判断文件打开是

否成功，若打开失败说明不存在这个文件，需要初始化

（2）文件存在，但内容被清空：通过 `ifstream` 的对象调用 `eof()` 函数用于测试文件是否为空，若文件为空则需要初始化

（3）文件存在，且保存了数据：通过 `ifstream` 的对象读取文本文件中的每一行数据，同时计数，作为职工人数。根据职工人数创建数组并在堆中开辟空间，用读取的数据初始化新创建的职工对象，再将对象保存在数组中。

该类的析构函数中，为了防止内存泄漏，需要手动释放堆区数据，即释放每一个对象以及堆区中的数组本身，其中还须判断指针是否为空。

在实现添加职工信息的 `addWorker` 函数中，要根据用户输入的添加人数创建扩容后的新数组。首先复制原来数组中的元素，其次根据用户输入添加新的数组元素。在用户添加职工编号时，加入了“判断该编号是否已经存在”的功能，主要由一个 `for` 循环实现，为了让用户重新输入，外层又添加了一个 `while` 的死循环，循环退出条件是用户输入的编号尚未被占用。用户输入完成后，依次执行释放原有数组空间、更改新的数组指向、更新员工个数和将信息保存到文件中。如果用户在一开始输入的添加人数不符合要求，系统将会输出提示并让其重新输入，为了实现这个功能，在本函数函数体的最外层添加了一个 `while` 的死循环，循环终止条件是用户输入合理的添加人数并成功录入信息。

在显示职工信息的 `showWorker` 函数中，首先判断文件是否不存在或内容为空，若非则遍历数组，用 `Worker` 父类指针调用子类的 `showInfo` 函数，输出不同类的职工信息。

在删除职工信息的 `deleteWorker` 函数中，在文件存在且内容不为空的基础上，遍历数组，寻找待删除的职工编号，确认删除后做删除操作，最后保存信息至文本文件。数组中删除元素的做法是：从下标为 `i+1` 的元素开始，把后面的数据依次向前移，覆盖掉下标为 `i` 的元素，并把数组中最后一个元素删除。若用户输入的职工编号不存在，则输出相应的提示信息并让用户重新输入，实现方法与 `addWorker` 函数中相似。

在修改职工信息的 `modifyWorker` 函数中，在文件存在且内容不为空的基础上，遍历数组，寻找待修改的职工编号，确认修改后根据用户输入修改该职工信息，

最后保存信息至文本文件。修改该职工信息的做法是：先释放该对象空间，根据用户输入创建新对象并将其指针放入数组中。若用户输入的职工编号不存在，则输出相应的提示信息并让用户重新输入，实现方法与 `addWorker` 函数中相似。

在查找职工信息的 `findWorker` 函数中，在文件存在且内容不为空的基础上，有两种查找方式，分别是按照编号查找和按照姓名查找。前一种比较简单，后一种需注意可能存在职工同名情况，故不可以一旦找到一个目标对象就退出 `for` 循环，而是要继续遍历数组，直到把所有姓名相同的员工找出再退出。此时用 `bool` 类型的 `have_found` 标签区分是否找到了相应的职工。

在按照编号排序的 `sort` 函数中，可以选择按照升序或者降序排列。本质上就是一个排序算法。我在实现排序时首先选择了冒泡排序，以升序排列为例，相邻的两个元素进行比较，如果前者大于后者，则交换位置，保证每一次的比较都将最大值向后移动。但是这种排序方式效率较低，因此我又用选择排序进行了实现，依然以升序排列为例，每一次从待排序的元素中选出最小值，存放在数组的起始位置，直到全部待排序的元素排完为止。排序后输出信息至屏幕并保存至文本文件中。在学习了 STL 标准模板库，领会容器与函数指针的使用，感受专业排序算法的高效率后，我思考能否用 STL 中的 `sort` 函数来实现排序功能。实现该功能的过程是：首先编写两种比较方式的函数；其次把 `workerArr` 数组中的内容复制到 `vector` 中；再次，将不同比较方式的函数作为参数分别传入 `sort` 函数中实现升序或者降序排列；最后把 `vector` 中的数据复制回 `workerArr` 数组中，保存至文本文件。事实证明，利用专业排序算法所需要的代码量远小于自己编写排序算法，而且在数据规模较大的情况下效率远高于后者。

在清空文档内容的 `deleteAll` 函数中，首先删除文件，即用 `ios::trunc` 方式打开文本文件，其功能为如果文件存在，则删除该文件并重新创建。然后手动释放堆区数据，即释放每一个对象以及堆区中的数组本身，实现方式与析构函数相似。

其成员函数中还有 `showMenu` 用于将菜单页面打印在屏幕上，`exist` 用于退出程序，比较简单，在此不过多赘述。此外，在实现上述函数时，为了方便调用，将 `save`（保存文件）、`getNum`（统计文本文件中的职工人数）、`initialize`（从文件中读取数据并初始化员工数组）进行了封装，主要用到了 `fstream` 文件流。

### 3.3 main 函数的调用

此部分较为简单，即在 main 函数中创建一个 WorkerManager 对象，根据用户的输入选择，通过 switch 语句调用 WorkerManager 的不同函数实现相应的功能。为了保证程序持续运行，在外层添加了一个 while 死循环，循环退出条件是用户选择“退出系统”，即调用 exit 函数退出。

### 3.4 重难点小结

- (1) 为了实现方法调用和数据访问时的动态绑定，创建了一个抽象类 Worker，将其中的成员函数设计成纯虚函数，再创建派生类继承这个抽象类；
- (2) 在 WorkerManager 中声明了一个指针数组用于存放 Worker 类指针，通过抽象类指针指向子类对象，实现不同类型员工信息存放在同一个数组中，便于管理；
- (3) 使用 fstream 文件流，完成数据的读写功能。比如构造函数中判断文件是否存在及内容是否为空、读取文件中的内容，save 函数中将数据保存至文本文件；
- (4) 析构函数中，为了防止内存泄漏，需要手动释放堆区数据，即释放每一个对象以及堆区中的数组本身；
- (5) 数组中删除元素的做法是从下标为 i+1 的元素开始，把后面的数据依次向前移，覆盖掉下标为 i 的元素；
- (6) 在查找职工信息时可能存在职工同名情况，需要全部输出；
- (7) 在实现按照编号排序用了冒泡排序和选择排序两种算法；
- (8) 使用了 STL 标准模板库中的 sort 函数来实现排序功能，其中涉及数组与 vector 的相互转化，并且极大地减少了代码量、提高了效率；
- (9) 判断“用户输入的是否是有效选项”、“用户输入数据是否合理”、“编号是否已存在”时用了嵌套的 if 条件判断、for 循环、while 循环以及 break 语句、goto 语句。

## 4. 演示（部分功能）

```
D:\C++code\职工管理系统v6 x + v
欢迎使用职工管理系统
0.退出
1.增加职工信息
2.显示职工信息
3.删除离职职工
4.修改职工信息
5.查找职工信息
6.按照编号排序
7.清空所有文档

请输入您的选择
5
请选择查找方式：
1.按照编号查找
2.按照姓名查找
1
请输入待查找编号：6
查无此人！请重新查找
请选择查找方式：
1.按照编号查找
2.按照姓名查找
1
请输入待查找编号：3
职工编号为：3 职工姓名为：周卉馨 岗位为：总裁 工作内容为：统筹公司各项事务

请按任意键继续... |
```

```
D:\C++code\职工管理系统v6 x + v
欢迎使用职工管理系统
0.退出
1.增加职工信息
2.显示职工信息
3.删除离职职工
4.修改职工信息
5.查找职工信息
6.按照编号排序
7.清空所有文档

请输入您的选择
4
请输入需要修改的职工编号：4
确定修改下列职工？y/n
职工编号为：4 职工姓名为：王珊凌 岗位为：员工 工作内容为：完成经理分配的任务，勤勤恳恳996
y
请输入职工姓名：王珊凌
请选择职工职位：
1.普通员工
2.经理
3.总裁
2

该职工信息已成功修改

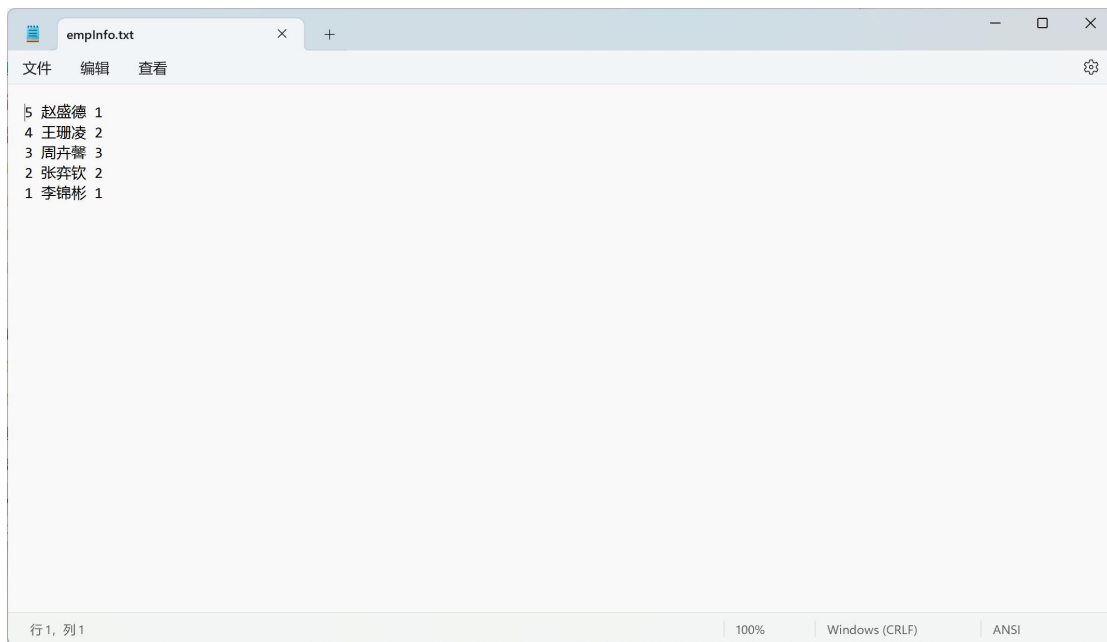
请按任意键继续... |
```

```
D:\C++code\职工管理系统v6 x + v
欢迎使用职工管理系统
0.退出
1.增加职工信息
2.显示职工信息
3.删除离职职工
4.修改职工信息
5.查找职工信息
6.按照编号排序
7.清空所有文档

请输入您的选择
6
请选择排序方式：
1.升序排列
2.降序排列
2
排序已完成，排序后的员工信息如下
职工编号为：5 职工姓名为：赵盛德 岗位为：员工 工作内容为：完成经理分配的任务，勤勤恳恳996
职工编号为：4 职工姓名为：王珊凌 岗位为：经理 工作内容为：接受总裁指派的任务，分配给员工
职工编号为：3 职工姓名为：周卉馨 岗位为：总裁 工作内容为：统筹公司各项事务
职工编号为：2 职工姓名为：张奔钦 岗位为：经理 工作内容为：接受总裁指派的任务，分配给员工
职工编号为：1 职工姓名为：李锦彬 岗位为：员工 工作内容为：完成经理分配的任务，勤勤恳恳996

请按任意键继续... |
```





The screenshot shows a text editor window titled 'empInfo.txt'. The window has a menu bar with '文件' (File), '编辑' (Edit), and '查看' (View). The main text area contains the following content:

```
5 赵盛德 1
4 王珊凌 2
3 周卉馨 3
2 张弈钦 2
1 李锦彬 1
```

The status bar at the bottom indicates '行 1, 列 1' (Line 1, Column 1), '100%', 'Windows (CRLF)', and 'ANSI'.

## 5. 总结

该银行职工管理系统的整体设计难度不是很大，程序结构和继承关系较为清晰，主要涉及到 C++ 的动态绑定机制、文件交互功能和一些简单算法。此外在实现一些具体的功能，如判断用户输入是否符合要求时，逻辑思维和程序设计较为繁琐。

当然该程序依然有很大的改进空间，比如由于删除、修改、查找中都需要根据编号寻找职工，可以把这个操作封装成一个函数，方便调用、减少代码的重复和冗余。又如实现排序功能时我自己编写的排序代码依然比较繁琐，有些步骤可以进行合并。

## 6. 附录

如果老师能够看到这里那真是奇迹再现，因为出于“想把我自己编程过程中遇到的困难、解决方式和收获感悟全都展现出来”的私心，本文写的可谓是废话连篇、详略不当。可能“职工管理系统”这个程序的难度和技术含金量都不高，我也承认其中的一些实现方式确实借鉴了网上的资料，但我是根据需求分析先自己编程，遇到困难再去查找资料，弄懂后再写入自己的代码中而非直接照抄，看在这种真诚的态度上希望老师能够手下留情。