《面向对象程序设计》大作业

(2022-2023 第 1 学期)

教职工信息管理系统

评分：_____

班级：计算机科学与技术 2102 班
姓名：邓富丽
学号：20211003071

2022 年 12 月

报告内容

一、任务描述

大学里的工作人员（Employee）可以分为管理人员（Staff）和教师（Faculty）两类。运用 C++语言和面向对象编程思想，设计一个学校教职工管理系统，要求实现如下功能：创建职工信息数据，包括职工编号、姓名、 性别、出生时间、部门、参加工作时间、年龄和工龄等（须根据当前时间计算得到），初始模拟数据通过文本文件创建（记录不少于 6 个），通过程序读文件载入。能够查找、添加或删除一个员工；浏览全部教职工信息（管理人员和教师分开）；统计职工的平均年龄和工龄等（管理人员和教师可分开统计）。

二、源代码（必须提供完整的源代码）

说明：复制代码有序号，可以先复制到 WPS，去掉编号格式，再复制到编译器运行。

1．main.cpp

```
1.  #include <iostream>
2.  #include "method.h"
3.  using namespace std;
4.
5.  /*********************************************************
6.      主函数 main.cpp
7.      内容：菜单、全局变量
8.  *********************************************************/
9.
10. // 定义全局变量
11. vector<Employee *> emp;
12. vector<Employee *> getTxt();
13. fstream iofile;
14. string txtName = "/Users/m12j10/Code/CPPFinalProject/employee.txt";
15.
16. void menu()
17. {
18.     cout << "+-----------------------------------------------+\n";
19.     cout << "|                教职工信息管理系统             |\n";
20.     cout << "+-----------------------------------------------+\n";
21.     cout << "|            1、添加教职工信息                  |\n";
22.     cout << "|            2、删除教职工信息                  |\n";
23.     cout << "|            3、按姓名搜索教职工                |\n";
24.     cout << "|            4、浏览所有教职工信息              |\n";
25.     cout << "|            5、统计教职工平均年龄和工龄        |\n";
26.     cout << "+-----------------------------------------------+\n";
27.     cout << "|                  0、退出                      |\n";
28.     cout << "+-----------------------------------------------+\n";
29. }
```

```
30.
31.    int main()
32.    {
33.        while (1)
34.        {
35.            menu();
36.            int choice;
37.            cout << "请输入操作序号: ";
38.            cin >> choice;
39.            switch (choice)
40.            {
41.            case 0:
42.                cout << "0 退出系统" << endl;
43.                exit(1);
44.                break;
45.
46.            // 添加【按职工类型添加】
47.            case 1:
48.                int select0;
49.                cout << "请选择:1、添加教师信息, 2、添加管理员信息, 0、返回  \n";
50.                cin >> select0;
51.                if (select0 == 0)
52.                    ;
53.                else
54.                {
55.                    cout << "输入格式: 【职工类别 职工编号 姓名 性别 出生日期 部门 参加
    工作时间 职称或级别】\n";
56.                    addData(select0);
57.                }
58.                break;
59.
60.            // 删除【输入编号, 打印删除信息, 再次确认是否删除】
61.            case 2:
62.                int select;
63.                cout << "请选择:1、按编号删除, 0、返回  \n";
64.                cin >> select;
65.                if (select == 1)
66.                    deleteData();
67.                else if (select == 0)
68.                    ;
69.                else
70.                    cout << " 选择错误。\n";
71.                break;
72.
73.            // 搜索【按姓名搜索】
74.            case 3:
75.                int select1;
76.                cout << "请选择:1、按姓名搜索, 0、返回  \n";
77.                cin >> select1;
78.                if (select1 == 1)
79.                    search();
80.                else if (select1 == 0)
81.                    ;
82.                else
83.                    cout << " 选择错误。\n";
84.                break;
85.
86.            // 查看【可选择查看教师/管理员信息, 打印显示为: 基本信息加上年龄和工龄】
87.            case 4:
```

```
                int select2;
                cout << "请选择:1、查看所有教师信息, 2、查看所有管理员信息, 0、返回  \n";
                cin >> select2;
                if (select2 == 0)
                    ;
                else
                {
                    cout << "【职工类别 职工编号 姓名 性别 出生日期 部门 参加工作时间 职称或级别 年龄 工龄】\n";
                    show(select2);
                }
                break;

            // 统计【可分别统计教师/管理员的平均年龄和工龄】
            case 5:
                int select3;
                cout << "请选择:1、统计教师的平均年龄和工龄, 2、统计管理员的平均年龄和工龄, 0、返回  \n";
                cin >> select3;
                if (select3 == 0)
                    ;
                else
                {
                    cout << "【平均年龄 平均工龄】\n";
                    Statistics(select3);
                }
                break;
            default:
                printf("无此选项,请重新输入! \n");
            }
        }
        return 0;
    }
```

2. Employee.h

```
    #include <iostream>
    #include <string>
    using namespace std;

    /*************************************************************
        头文件 Employee.h
        内容: Employee、Faculty、Staff 类
    *************************************************************/

    // 声明全局函数
    extern void addData(int);
    extern void deleteData();
    extern void search();
    extern void show(int);
    extern void Statistics(int);

    // Employee 基类
    class Employee
    {
    protected:
```

```
21.         int type;           // 职工类别(1 表示教师，2 表示管理者)
22.         string id;          // 职工编号
23.         string name;        // 姓名
24.         string sex;         // 性别
25.         int birth;          // 出生年份
26.         string department; // 部门
27.         int workDay;        // 参加工作的年份
28.         int age;            // 年龄
29.         int workTime;       // 工龄
30.
31.     public:
32.         // 构造函数与析构函数
33.         Employee(int t, string i, string n, string s, int b, string d, int w)
34.         {
35.             type = t;
36.             id = i;
37.             name = n;
38.             sex = s;
39.             birth = b;
40.             department = d;
41.             workDay = w;
42.             time_t now = time(0); // 系统头文件<ctime>中的类和函数
43.             tm *ltm = localtime(&now);
44.             age = (1900 + ltm->tm_year) - b;
45.             workTime = (1900 + ltm->tm_year) - w;
46.         }
47.         Employee(){};
48.         virtual ~Employee(){}; // 涉及指针的虚函数的动态联编会出现 delete 指针的警告，
    设置为虚析构函数好的解决方法
49.
50.         friend void addData(int);
51.         friend void deleteData();
52.         friend void search();
53.         friend void show(int);
54.         friend void Statistics(int);
55.         virtual void display() = 0;
56.
57.         int getAge() // 加上 getAge(),getWorkTime()函数，否则 Employee 为虚基类，无法
    完成后续的指针类型转换
58.         {
59.             return age;
60.         }
61.         int getWorkTime()
62.         {
63.             return workTime;
64.         }
65.     };
66.
67.     // Faculty 类
68.     class Faculty : public Employee
69.     {
70.         string title; // 职称
71.     public:
72.         // 构造函数与析构函数
73.         Faculty(int t, string i, string n, string s, int b, string d, int w,
    string ti)
```

```
74.         {
75.             type = t;
76.             id = i;
77.             name = n;
78.             sex = s;
79.             birth = b;
80.             department = d;
81.             workDay = w;
82.             title = ti;
83.             time_t now = time(nullptr);
84.             tm *ltm = localtime(&now);
85.             age = (1900 + ltm->tm_year) - birth;
86.             workTime = (1900 + ltm->tm_year) - workDay;
87.         }
88.         Faculty(){};
89.         ~Faculty() {}
90.
91.         // 声明友元函数
92.         friend void addData(int);
93.         friend void deleteData();
94.         friend void search();
95.         friend void show(int);
96.         friend void Statistics(int);
97.
98.         // 重载输入输出运算符
99.         friend ostream &operator<<(ostream &os, const Faculty &s)
100.        {
101.            os << s.type << " " << s.id << " " << s.name << " " << s.sex << "
     " << s.birth << " " << s.department
102.                << " " << s.workDay << " " << s.title;
103.            return os;
104.        }
105.        friend istream &operator>>(istream &is, Faculty &s)
106.        {
107.            is >> s.type >> s.id >> s.name >> s.sex >> s.birth >> s.departmen
    t >> s.workDay >> s.title;
108.            return is;
109.        }
110.
111.        // 实现虚函数
112.        virtual void display()
113.        {
114.            cout << " " << type << " " << id << " " << name << " " << sex <<
    " " << birth << " " << department
115.                << " " << workDay << " " << title << " " << age << " " << wo
    rkTime << endl;
116.        }
117.    };
118.
119.    // Staff 类
120.    class Staff : public Employee
121.    {
122.        string level; // 级别
123.    public:
124.        // 构造函数与析构函数
125.        Staff(int t, string i, string n, string s, int b, string d, int w, st
    ring l)
126.        {
127.            type = t;
128.            id = i;
129.            name = n;
130.            sex = s;
131.            birth = b;
```

```
            department = d;
            workDay = w;
            level = l;
            time_t now = time(nullptr);
            tm *ltm = localtime(&now);
            age = (1900 + ltm->tm_year) - birth;
            workTime = (ltm->tm_year + 1900) - workDay;
        }
        Staff(){};
        ~Staff(){};

        // 声明友元函数
        friend void addData(int);
        friend void deleteData();
        friend void search();
        friend void show(int);
        friend void Statistics(int);

        // 重载输入输出运算符
        friend ostream &operator<<(ostream &os, const Staff &s)
        {
            os << s.type << " " << s.id << " " << s.name << " " << s.sex << " " << s.birth << " " << s.department
               << " " << s.workDay << " " << s.level;
            return os;
        }
        friend istream &operator>>(istream &is, Staff &s)
        {
            is >> s.type >> s.id >> s.name >> s.sex >> s.birth >> s.department >> s.workDay >> s.level;
            return is;
        }

        // 实现虚函数
        virtual void display()
        {
            cout << " " << type << " " << id << " " << name << " " << sex << " " << birth << " " << department
                 << " " << workDay << " " << level << " " << age << " " << workTime << endl;
        }
    };
```

3. method.h

```
    #include "Employee.h"
    #include <iostream>
    #include <ctime>
    #include <string>
    #include <vector>
    #include <fstream>
    #include <sstream>
    using namespace std;

    /*************************************************************
        头文件 method.h
        内容：方法的实现
    *************************************************************/

    // 声明全局变量
    extern vector<Employee *> emp;
```

```
17.    extern vector<Employee *> getTxt();
18.    extern fstream iofile;
19.    extern string txtName;
20.
21.    // 将 txt 文件内容读入数组
22.    vector<Employee *> getTxt()
23.    {
24.        vector<Employee *> tmp; // 定义临时的 vector 数组 tmp，用于数据传递
25.        ifstream fp(txtName);
26.        if (!fp.is_open())
27.        {
28.            printf("出错了");
29.            exit(-1);
30.        }
31.        string text;
32.        while (!fp.eof())
33.        {
34.            getline(fp, text);
35.            vector<string> v; // 通过临时的 vector 数组 v 将每一行数据的元素存入
36.            istringstream in(text);
37.            string temp;
38.            while (in >> temp)
39.            {
40.                v.push_back(temp);
41.            }
42.            if (text.length()) // 将数据对应转换为 Faculty 类型或 Staff 类型
43.            {
44.                if (v[0] == "1")
45.                {
46.                    int type = stoi(v[0]);
47.                    int bir = stoi(v[4]);
48.                    int work = stoi(v[6]);
49.                    tmp.push_back(new Faculty(type, v[1], v[2], v[3], bir, v[
    5], work, v[7]));
50.                }
51.                else
52.                {
53.                    int type = stoi(v[0]);
54.                    int bir = stoi(v[4]);
55.                    int work = stoi(v[6]);
56.                    tmp.push_back(new Staff(type, v[1], v[2], v[3], bir, v[5],
     work, v[7]));
57.                }
58.            }
59.        }
60.        fp.close();
61.        return tmp; // 将 tmp 的内容传递给 emp
62.    }
63.
64.    // 添加信息
65.    void addData(int s)
66.    {
67.        emp = getTxt(); // 将文本中的内容存入 vector 数组 emp
68.        Faculty newFac;
69.        Staff newSta;
70.        if (s == 1)
71.            cin >> newFac;
72.        else
73.            cin >> newSta;
74.        for (int i = 0; i < emp.size(); i++)
75.        {
76.            if (emp[i]->id == newFac.id || emp[i]->id == newSta.id)
```

```
77.             {
78.                 cout << "该职工编号已存在!不能重复添加." << endl;
79.                 return;
80.             }
81.         }
82.         iofile.open(txtName, ios::out); // 打开文本文件，同时清空文本的所有内容
83.         if (s == 1)
84.         {
85.             emp.push_back(new Faculty(newFac.type, newFac.id, newFac.name, ne
    wFac.sex, newFac.birth, newFac.department, newFac.workDay, newFac.title));
86.         }
87.         else
88.         {
89.             emp.push_back(new Staff(newSta.type, newSta.id, newSta.name, newS
    ta.sex, newSta.birth, newSta.department, newSta.workDay, newSta.level));
90.         }
91.         for (int i = 0; i < emp.size(); i++) // 重新将数组 emp 内容写入文本
92.         {
93.             if (emp[i]->type == 1) // 必须区分类型存入，因为基类无法获取到派生类的特有
    数据，无法写入职称和级别
94.             {
95.                 Faculty *temp = (Faculty *)emp[i]; // 将 Employee*指针类型强制转
    换为 Faculty*指针类型
96.                 iofile << *temp << endl;
97.             }
98.             else
99.             {
100.                Staff *temp = (Staff *)emp[i]; // 将 Staff*指针类型强制转换为
    Staff*指针类型
101.                iofile << *temp << endl;
102.            }
103.        }
104.        iofile.close();
105.        cout << "添加完成" << endl;
106.    }
107.
108.    // 删除信息
109.    void deleteData()
110.    {
111.        emp = getTxt();
112.        string id;
113.        bool flag; // 用于标记是否确认删除
114.        cout << "请输入要删除的职工编号: \n";
115.        cin >> id;
116.        for (int i = 0; i < emp.size(); i++)
117.        {
118.            if (emp[i]->id == id)
119.            {
120.                // 为了保证删除无误，先打印该职工信息
121.                cout << "职工信息:";
122.                emp[i]->display(); // 无需判断 Faculty 类还是 Staff 类，直接调用，因为
    display()是虚函数
123.                cout << "确认删除?(1.确认 2.取消)\n";
124.                int t;
125.                cin >> t;
126.                if (t == 1)
```

```
127.                 {
128.                     delete emp[i]; // erase 不能彻底销毁指针，需要 delete!
129.                     emp.erase(emp.begin() + i);
130.                     flag = true;
131.                     break;
132.                 }
133.                 else
134.                     return;
135.             }
136.         }
137.         if (flag) // 确认删除
138.         {
139.             iofile.open(txtName, ios::out);
140.             for (int i = 0; i < emp.size(); i++)
141.             {
142.                 if (emp[i]->type == 1) // 同样的需要区分类型，再写入文件
143.                 {
144.                     Faculty *temp = (Faculty *)emp[i];
145.                     iofile << *temp << endl;
146.                 }
147.                 else
148.                 {
149.                     Staff *temp = (Staff *)emp[i];
150.                     iofile << *temp << endl;
151.                 }
152.             }
153.             cout << "删除成功" << endl;
154.             iofile.close(); // 要记得关闭文件，否则无法保存修改
155.         }
156.         else
157.             cout << "未找到该职工编号，删除失败\n";
158.     }
159.
160.     // 按姓名搜索
161.     void search()
162.     {
163.         emp = getTxt();
164.         string name;
165.         bool flag = false;
166.         cout << "请输入要查找的职工姓名: \n";
167.         cin >> name;
168.         for (int i = 0; i < emp.size(); i++)
169.         {
170.             if (emp[i]->name == name)
171.             {
172.                 emp[i]->display();
173.                 flag = true;
174.             }
175.         }
176.         if (flag == false)
177.             cout << "未查找到该职工\n";
178.     }
179.
180.     // 查看所有教职工信息
181.     void show(int sel) // 传入需要浏览的职工类型
182.     {
183.         emp = getTxt();
184.         for (int i = 0; i < emp.size(); i++)
185.         {
186.             if (emp[i]->type == sel) // 筛选
```

```
            {
                emp[i]->display(); // 调用虚函数 display()
            }
        }
    }

    // 统计平均年龄和工龄
    void Statistics(int ty) // 传入需要统计数据的职工类型
    {
        emp = getTxt();
        int totalAge = 0;       // 统计总年龄
        int totalWorkTime = 0; // 统计总工龄
        int goal = 0;           // 统计人数
        for (int i = 0; i < emp.size(); i++)
        {
            if (emp[i]->type == ty) // 对职工类型进行筛选
            {
                totalAge += emp[i]->age;
                totalWorkTime += emp[i]->workTime;
                goal++;
            }
        }
        cout << "   " << (double)totalAge / goal << "     " << (double)totalW
    orkTime / goal << endl;
    }
```

三、运行界面（包括文本文件截图和运行结果截图，最好对每个菜单都有测试）

employee.txt ×

CPPFinalProject > employee.txt

```
1   1 201610100 黄石 男 1982 计算机系 2005 副教授
2   1 201311067 绿叶 女 1986 国关系 2000 副教授
3   2 201210121 大地 男 1975 财务处 1999 管理3级
4   1 201610102 蓝天 男 1963 会计系 1985 教师
5   1 201710800 月亮 女 1990 国关系 2016 教师
6   2 201519100 星辰 男 1987 人事处 2014 管理5级
7   1 201410600 太阳 女 1982 外国语系 2010 副教授
8   1 201910789 柳树 女 1994 西班牙语系 2020 教师
9   2 201011029 白云 女 1978 人事处 2000 管理5级
10
```

图 1 文本文件截图

```
+-------------------------------------------------+
|                教职工信息管理系统               |
+-------------------------------------------------+
|              1、添加教职工信息                  |
|              2、删除教职工信息                  |
|              3、按姓名搜索教职工                |
|              4、浏览所有教职工信息              |
|              5、统计教职工平均年龄和工龄        |
+-------------------------------------------------+
|                  0、退出                        |
+-------------------------------------------------+
请输入操作序号：1
请选择:1、添加教师信息，2、添加管理员信息，0、返回
1
输入格式：【职工类别 职工编号 姓名 性别 出生日期 部门 参加工作时间 职称或级别】
1 201310980 鲜花 女 1993 土木系 2019 教师
添加完成
```

图 2 测试添加教师功能

```
+-------------------------------------------------+
|                教职工信息管理系统               |
+-------------------------------------------------+
|              1、添加教职工信息                  |
|              2、删除教职工信息                  |
|              3、按姓名搜索教职工                |
|              4、浏览所有教职工信息              |
|              5、统计教职工平均年龄和工龄        |
+-------------------------------------------------+
|                  0、退出                        |
+-------------------------------------------------+
请输入操作序号：1
请选择:1、添加教师信息，2、添加管理员信息，0、返回
2
输入格式：【职工类别 职工编号 姓名 性别 出生日期 部门 参加工作时间 职称或级别】
2 201210790 大海 男 1986 后勤处 2017 管理4级
添加完成
```

图 3 测试添加管理员功能

employee.txt

CPPFinalProject > employee.txt

```
1   1 201610100 黄石 男 1982 计算机系 2005 副教授
2   1 201311067 绿叶 女 1986 国关系 2000 副教授
3   2 201210121 大地 男 1975 财务处 1999 管理3级
4   1 201610102 蓝天 男 1963 会计系 1985 教师
5   1 201710800 月亮 女 1990 国关系 2016 教师
6   2 201519100 星辰 男 1987 人事处 2014 管理5级
7   1 201410600 太阳 女 1982 外国语系 2010 副教授
8   1 201910789 柳树 女 1994 西班牙语系 2020 教师
9   2 201011029 白云 女 1978 人事处 2000 管理5级
10  1 201310980 鲜花 女 1993 土木系 2019 教师
11  2 201210790 大海 男 1986 后勤处 2017 管理4级
12
```

图 4 添加成功

```
+-------------------------------------------------+
|                教职工信息管理系统               |
+-------------------------------------------------+
|              1、添加教职工信息                  |
|              2、删除教职工信息                  |
|              3、按姓名搜索教职工                |
|              4、浏览所有教职工信息              |
|              5、统计教职工平均年龄和工龄        |
+-------------------------------------------------+
|                  0、退出                        |
+-------------------------------------------------+
请输入操作序号：2
请选择:1、按编号删除, 0、返回
1
请输入要删除的职工编号：
201311067
职工信息：1 201311067 绿叶 女 1986 国关系 2000 副教授 36 22
确认删除?(1.确认 2.取消)
2
+-------------------------------------------------+
|                教职工信息管理系统               |
+-------------------------------------------------+
|              1、添加教职工信息                  |
|              2、删除教职工信息                  |
|              3、按姓名搜索教职工                |
|              4、浏览所有教职工信息              |
|              5、统计教职工平均年龄和工龄        |
+-------------------------------------------------+
|                  0、退出                        |
+-------------------------------------------------+
请输入操作序号：2
请选择:1、按编号删除, 0、返回
1
请输入要删除的职工编号：
201311067
职工信息：1 201311067 绿叶 女 1986 国关系 2000 副教授 36 22
确认删除?(1.确认 2.取消)
1
删除成功
```

图 5 测试删除功能（可取消删除）

图 6 删除成功

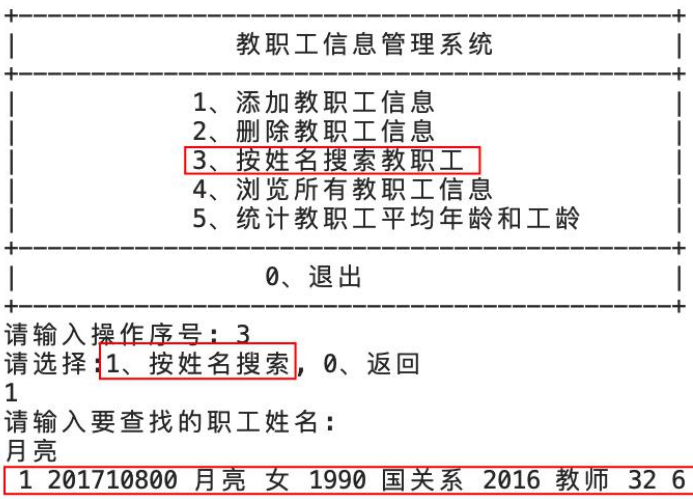

图 7 测试查找功能

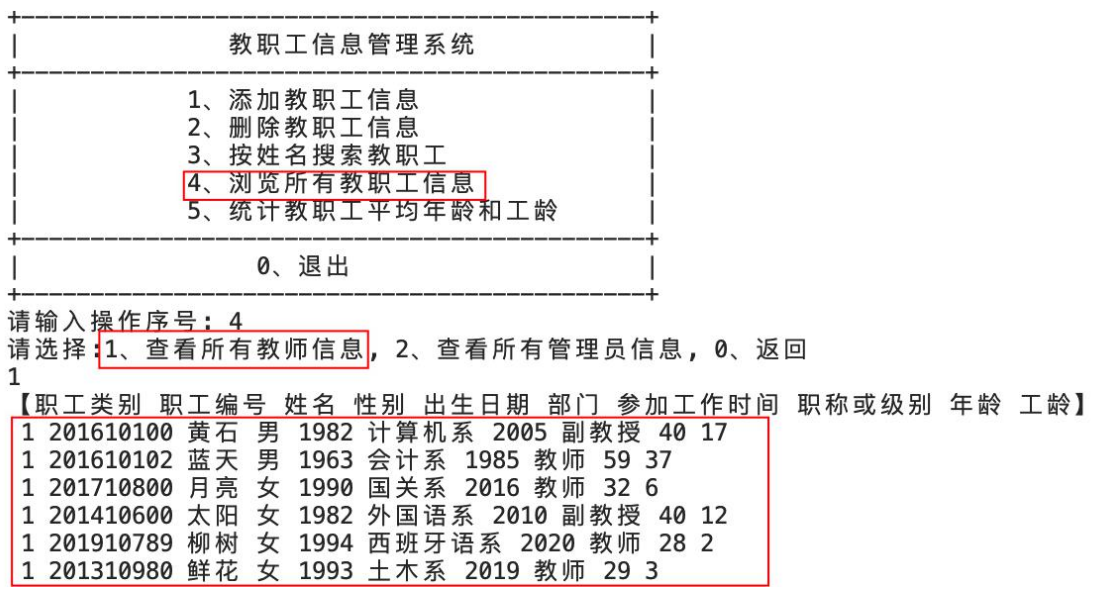

图 8 测试查看教师信息功能

```
+-----------------------------------------------+
|                教职工信息管理系统             |
+-----------------------------------------------+
|              1、添加教职工信息                |
|              2、删除教职工信息                |
|              3、按姓名搜索教职工              |
|              4、浏览所有教职工信息            |
|              5、统计教职工平均年龄和工龄      |
+-----------------------------------------------+
|                  0、退出                      |
+-----------------------------------------------+
请输入操作序号：4
请选择:1、查看所有教师信息，2、查看所有管理员信息，0、返回
2
【职工类别 职工编号 姓名 性别 出生日期 部门 参加工作时间 职称或级别 年龄 工龄】
2 201210121 大地 男 1975 财务处 1999 管理3级 47 23
2 201519100 星辰 男 1987 人事处 2014 管理5级 35 8
2 201011029 白云 女 1978 人事处 2000 管理5级 44 22
2 201210790 大海 男 1986 后勤处 2017 管理4级 36 5
```

图 9 测试查看管理员信息功能

```
+-----------------------------------------------+
|                教职工信息管理系统             |
+-----------------------------------------------+
|              1、添加教职工信息                |
|              2、删除教职工信息                |
|              3、按姓名搜索教职工              |
|              4、浏览所有教职工信息            |
|              5、统计教职工平均年龄和工龄      |
+-----------------------------------------------+
|                  0、退出                      |
+-----------------------------------------------+
请输入操作序号：5
请选择:1、统计教师的平均年龄和工龄，2、统计管理员的平均年龄和工龄，0、返回
1
【平均年龄 平均工龄】
  38      12.8333
+-----------------------------------------------+
|                教职工信息管理系统             |
+-----------------------------------------------+
|              1、添加教职工信息                |
|              2、删除教职工信息                |
|              3、按姓名搜索教职工              |
|              4、浏览所有教职工信息            |
|              5、统计教职工平均年龄和工龄      |
+-----------------------------------------------+
|                  0、退出                      |
+-----------------------------------------------+
请输入操作序号：5
请选择:1、统计教师的平均年龄和工龄，2、统计管理员的平均年龄和工龄，0、返回
2
【平均年龄 平均工龄】
  40.5      14.5
```

图 10 测试统计功能

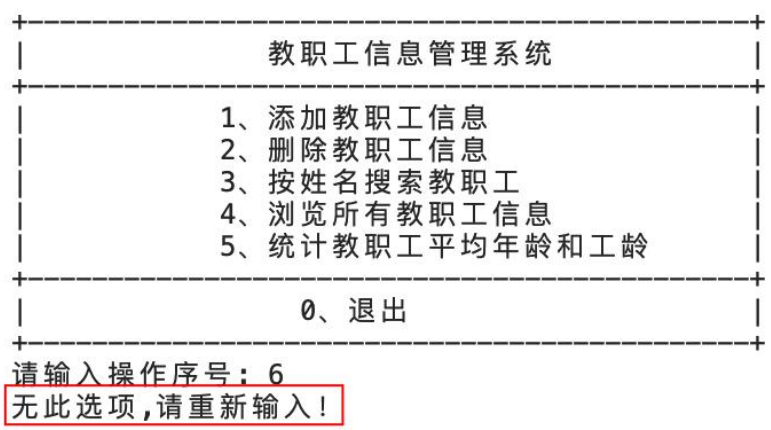

图 11 测试异常选择的提示功能

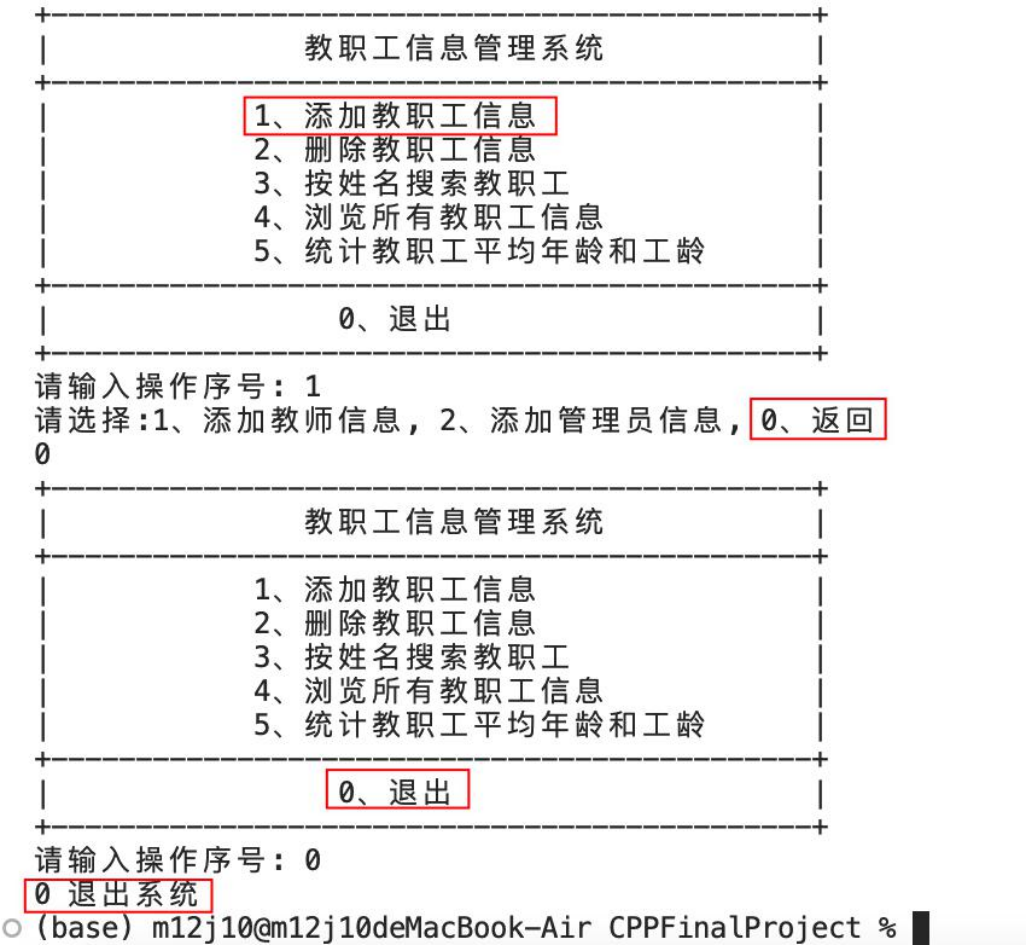

图 12 测试返回和退出系统功能

四、总结与体会（在此描述你是如何做的，中间遇到了什么问题，如何解决的，有何收获和体会）

一、功能图和类图的绘制

在开始着手做大作业之前，根据要求，粗略地绘制了功能图和类图，保证自己在完成大作业的过程中，有整体的框架，和实现思路。当然，在完成了代码的实现后，我又再次整理了类图，这样的好处是，既可以帮助自己思考代码的优化，又可以便于以后自己和他人对代码和理解，提高项目的可读性。

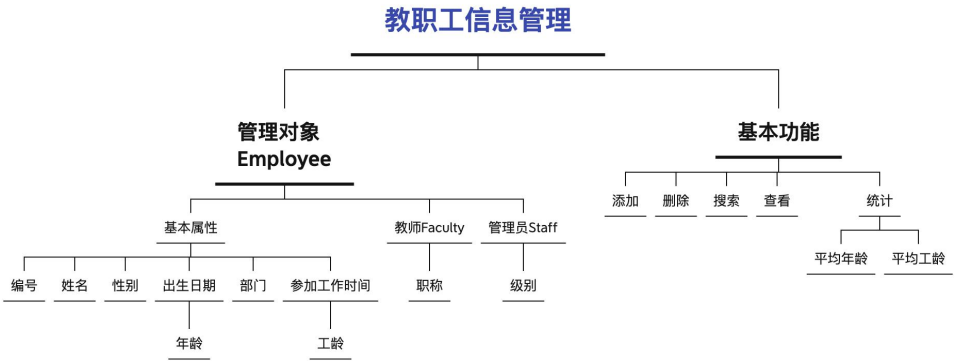

图 13 功能图

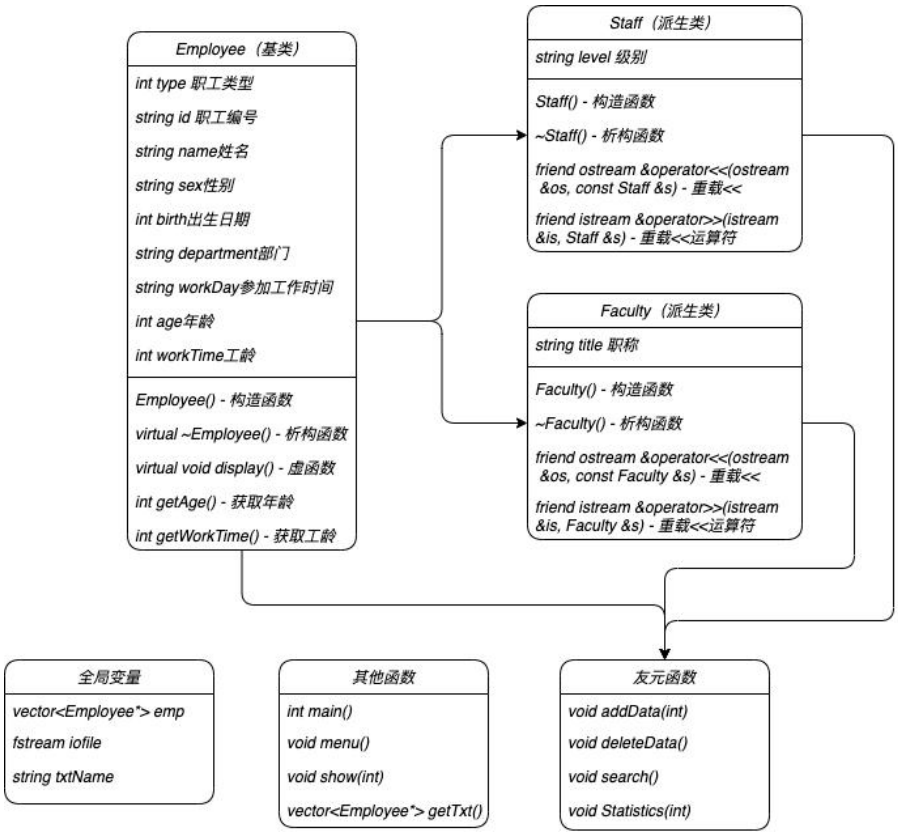

图 14 UML 图

二、程序整体的设计

1. 代码的复用性问题

这个项目有一个基类两个派生类，整体上有五个主要功能需要实现，如果为每一个派生类设置一个成员函数来实现，代码的复用性较低。

因此我的设计是，将这五大功能设置为友元函数。因为不同的类可以有同一个友元函数，代码的复用性得到了提高。

2. 代码的可维护性

一个完整的项目，为了调试和后续新增功能的便利，需要提高它的可维护性，所以我将这个项目整体分成了三个部分：主函数 main.cpp、封装类的头文件 Employee.h 和实现功能的函数的头文件 method.h。

3. 数据存储的设计

按照大作业的要求，需要使用一个<Employee*>vector 数组存储所有的数据，则需要思考在读入文本数据、添加信息、删除信息、写入到文本这些情况下，数据应该如何存储。我的处理方案是：

1) **读入文本数据的情况。**将文本文件的数据一一识别，逐个赋值给 Faculty 类或 Staff 类，再将 Faculty*和 Staff*存到<Employee*>vector 中。具体的实现是：设置两个临时的数组将数据先按行从文本文件中提取出来，再将每行的数据按空格分离出来，根据对职工类型的判断，可以区分数据是 Faculty 类或者是 Staff 类，再通过 new Faculty 或 new Staff 的方式将数据 push_back 到<Employee*>vector 数组中。
2) **添加和删除信息的情况。**添加信息同样需要区分添加的信息是 Faculty 类或 Staff 类，再将信息存入<Employee*>vector 数组；而删除信息时，无需区分，只需要释放指针即可。
3) **写入到文本的情况。**写入文本使用文件流 iofile << emp[i] << endl，显然这样是不行的。因为基类指针无法获取派生类的私有数据成员，所以派生类的职称/级别信息无法写入到文本中。解决的方法是将 emp[i]一一识别后强制转换为对应的 Faculty*和 Staff*，并且重载 Faculty 类和 Staff 类的“<<”运算符

图 15 vector 数组的存储情况

三、遇到的问题及解决

1. 虚基类报错问题

因为一开始在程序中并未给基类 Employee 定义非虚的函数，仅有一个虚函数 display()，导致基类成为了虚基类，而不能实现虚基类指针 Employee*到 Faculty*和 Staff*的强制转换。

我的解决方法比较直接，给基类定义非虚函数 getAge()和 getWorkTime()，从而使基类非虚，基类指针可以实现强制转换。

2. 指针的销毁问题

在删除信息后，vector 数组中的指针需要销毁，但是仅仅使用 vector 数组的 erase()函数是无法彻底销毁指针的，需要额外的将指针 delete 掉。但是使用 delete 时，出现了编译器的警告：warning: delete called on that is abstract but has non-virtual destructor [-Wdelete-non-virtual-dtor]。有内存泄漏的隐患。原因是虚函数调用动态联编时，释放内存时只调用了基类的析构函数，派生类的析构函数没有调用到，那么就会造成本来需要在派生类析构函数释放的资源没有得到正确释放，进而造成内存泄漏。

解决方法有两个：第一种是将指针强制转换为派生指针再 delete，第二种是将基类的析构函数定义为虚析构函数。我采用了后者，后者的代码更为简洁和安全。

3. 获取当前日期

为了计算教职工的年龄和工龄，需要获取当前的年份。获取当前时间也没有直接的方式获取，需要调用<ctime>头文件中的 time_t 类中的 time()函数和 tm 类的 localtime()函数获得 1900 年距今的年份，通过计算获得当前日期，进而计算年龄和工龄。

4. 文本文件的读写

对文本文件的读出和写入最大的问题是在增加数据时，不能覆盖了原有的信息；在删除数据后，需要调整数据之间的相对位置。

我的处理方法是：先打开文件调用定义的 getTxt()函数将所有数据存入 vector 数组中，在 vector 数组中完成数据的增删，再使用 ios::out 打开文件，将 vector 数组的内容写入文件中。

这里用 ios::out 的方式打开文件，会首先清空文本文件中所有内容，可以达到想要的目的。

四、收获

这次大作业从实践上考察了我们对本学期知识的运用，这种考试形式相比于以应试为目的的记背代码更加实用。

在大作业中，可供自己发挥的地方有很多。从整个系统的设计，到每一个功能的实现，都有自己的处理方式。

多次对代码的调试、对这个信息管理系统的使用，我感受到了设计一个项目需要有产品思维、用户思维。当用户错误输入了一项功能的编号，是否能够返回？当用户错误输入了一个职工的编号，或者用户不太确认是否想要删除的就是该编号的职工，是否能取消？

所以，我在每个功能下面增加了二级菜单，可以让用户选择返回上一级；在删除功能中，增加了显示该编号的职工信息、确认删除、取消删除功能。

这些功能的实现当然不复杂，但是这些设计在真正面向用户时是非常有用的。不断地思考如何才能将其打磨成一个真正能投入使用的项目，这会使每一行代码的实现都成为关键的一步。

我得到的第二点收获是：学习到了更多的方法和技术。当我有一个想法 —— 将文本文件的内容逐一识别为 Faculty 类或 Staff 类再读取到 vector 数组中 —— 的时候，我觉得实现这个功能比较复杂困难，但是通过在论坛中的搜索，我看到了许多人分享他们的方法和技巧，将这些方法收入囊中，也拓宽了我的知识面。当然还有 stoi()函数、time_t 类、istringstream 类等方法的运用。

当然，完成这项大作业的过程也不是一帆风顺。一开始我是将 vector 数组定义为 string 类型，但是仔细检查了大作业的要求后，需要将 vector 数组改为 Employee* 类型，这样的大改动花费了很多的时间和精力。但是这些付出是有益的，在重新修改代码的过程，我对代码进行了更多的完成，处理起来也比一开始编写代码快了许多；对虚函数、指针、基类派生类的理解更是上升了一个认知。

本次的大作业总的来说收获颇多，能将本学期学到的知识显性的运用起来，也让我更主动的学习到新的知识；最重要的，它激发了我的编程热情，给了我在即将到来的假期中，不断学习和尝试实践更多知识的期盼!