

职工管理系统

1、管理系统需求

职工管理系统可以用来管理公司内所有员工的信息

本教程主要利用C++来实现一个基于多态的职工管理系统

公司中职工分为三类：普通员工、经理、老板，显示信息时，需要显示职工编号、职工姓名、职工岗位、以及职责

普通员工职责：完成经理交给的任务

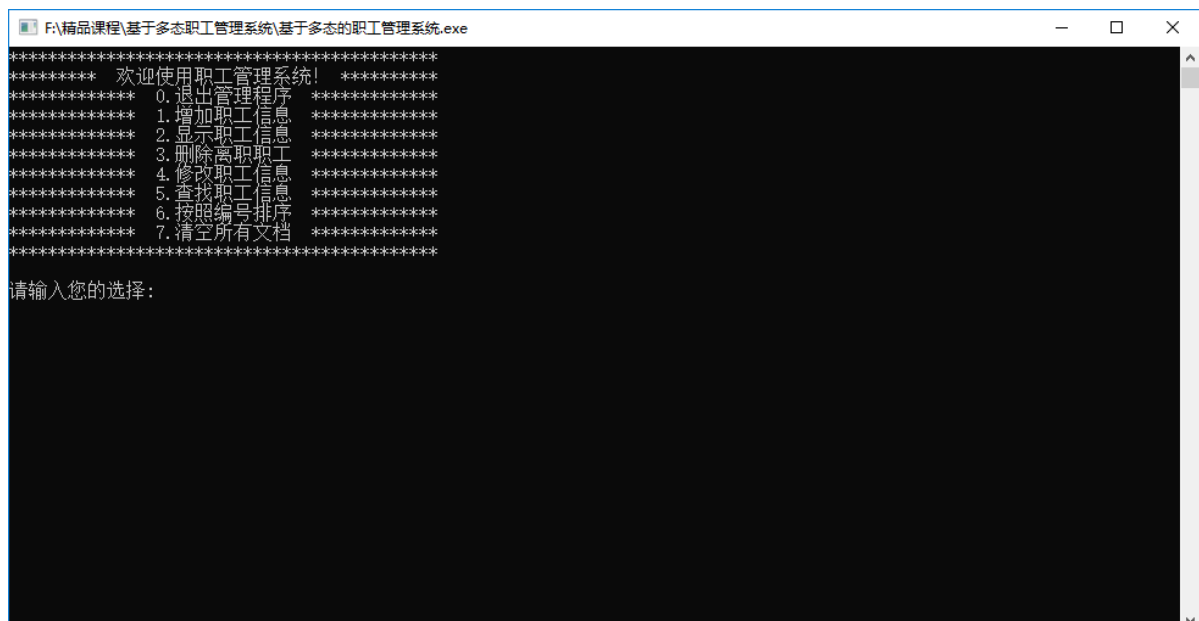
经理职责：完成老板交给的任务，并下发任务给员工

老板职责：管理公司所有事务

管理系统中需要实现的功能如下：

- 退出管理程序：退出当前管理系统
- 增加职工信息：实现批量添加职工功能，将信息录入到文件中，职工信息为：职工编号、姓名、部门编号
- 显示职工信息：显示公司内部所有职工的信息
- 删除离职职工：按照编号删除指定的职工
- 修改职工信息：按照编号修改职工个人信息
- 查找职工信息：按照职工的编号或者职工的姓名进行查找相关的人员信息
- 按照编号排序：按照职工编号，进行排序，排序规则由用户指定
- 清空所有文档：清空文件中记录的所有职工信息（清空前需要再次确认，防止误删）

系统界面效果图如下：



需根据用户不同的选择，完成不同的功能！

2、创建项目

创建项目步骤如下：

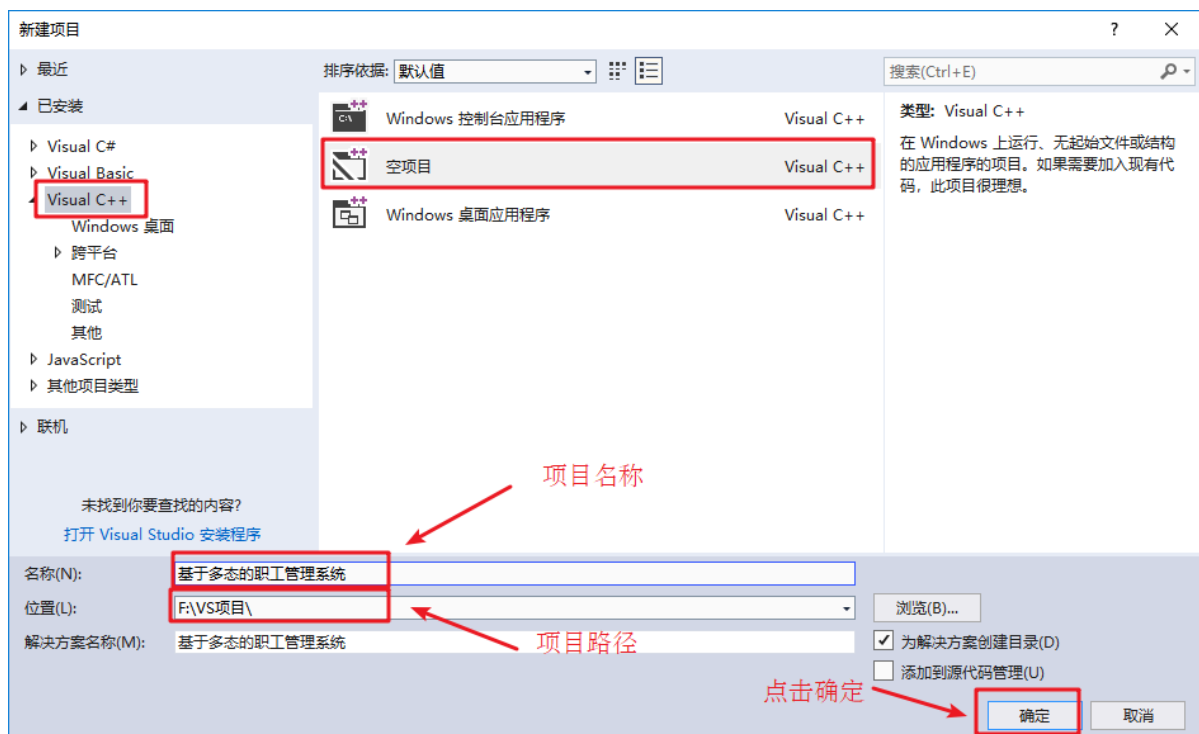
- 创建新项目
- 添加文件

2.1 创建项目

打开vs2017后，点击创建新项目，创建新的C++项目

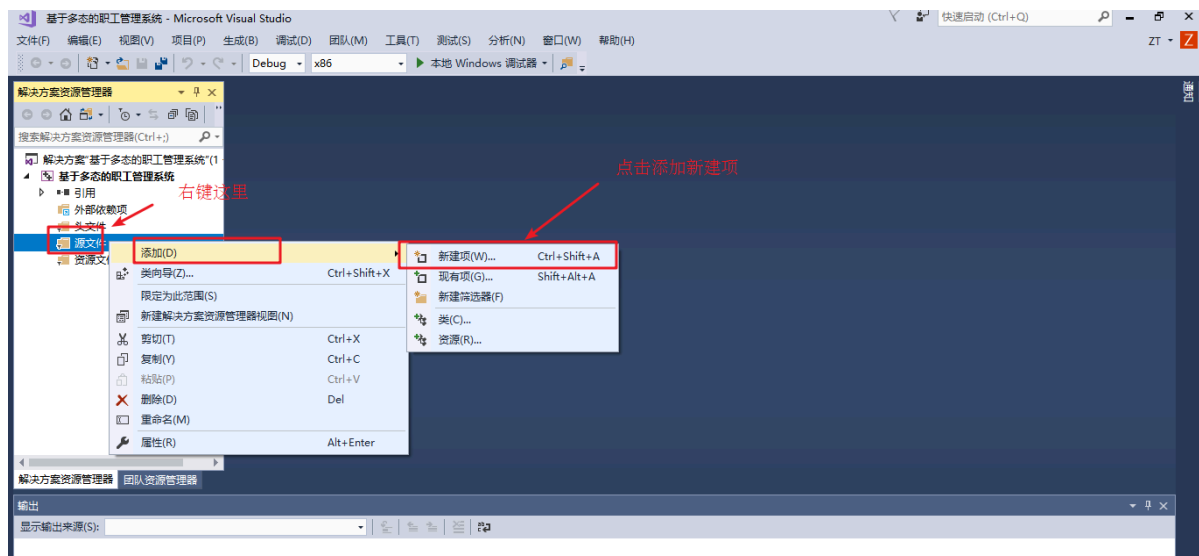


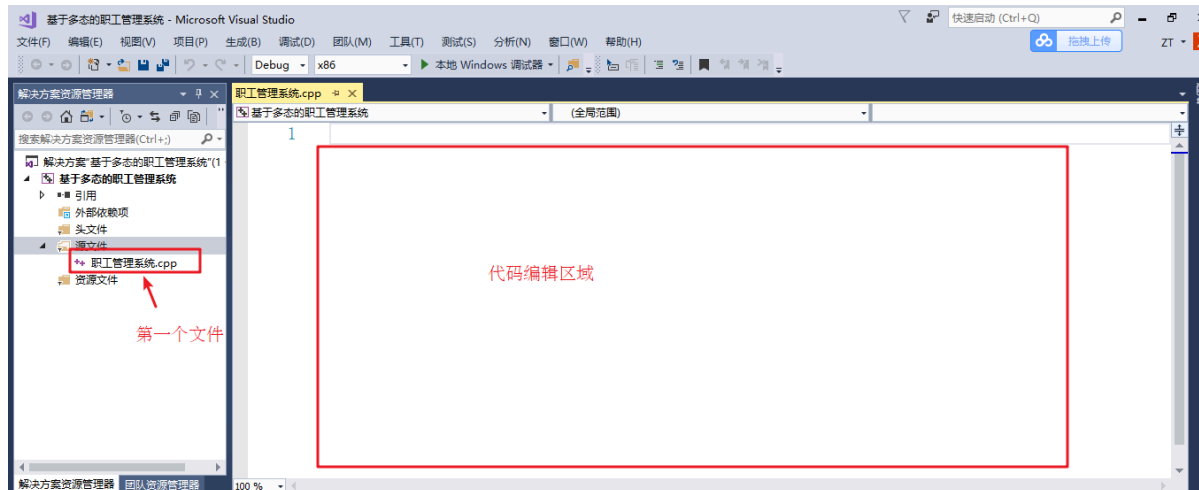
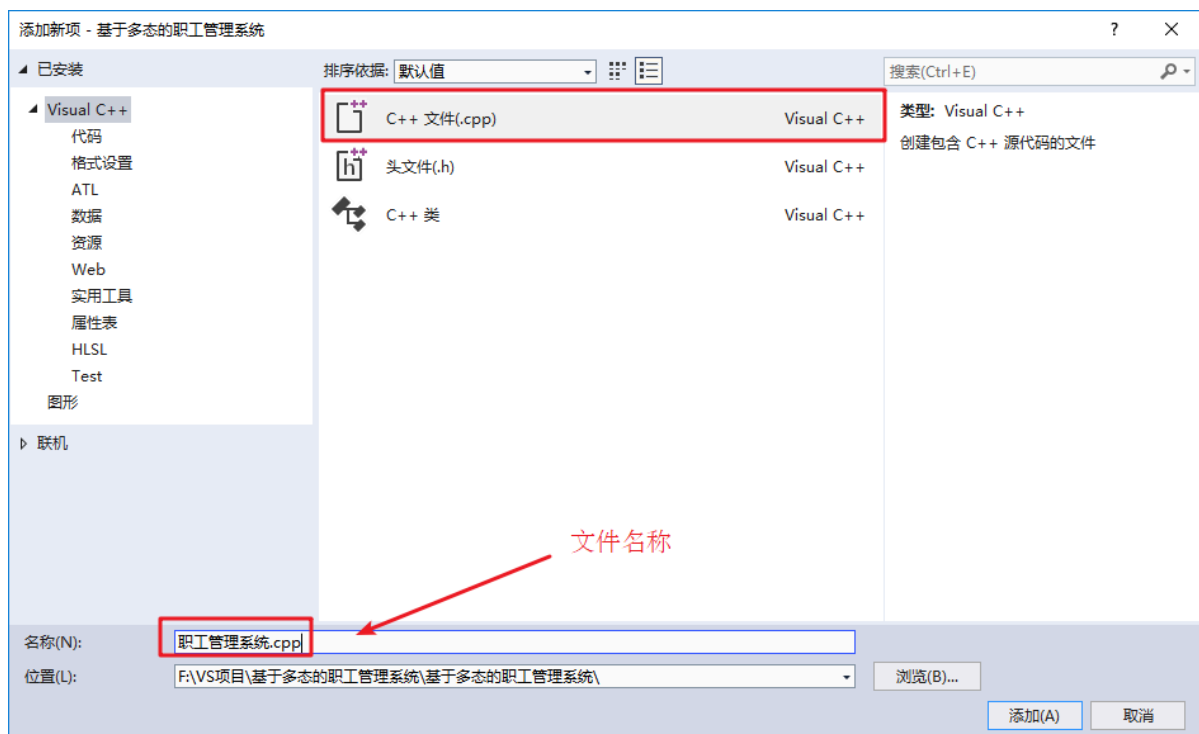
填写项目名称以及项目路径，点击确定



2.2 添加文件

右键源文件, 进行添加文件操作





至此，项目已创建完毕

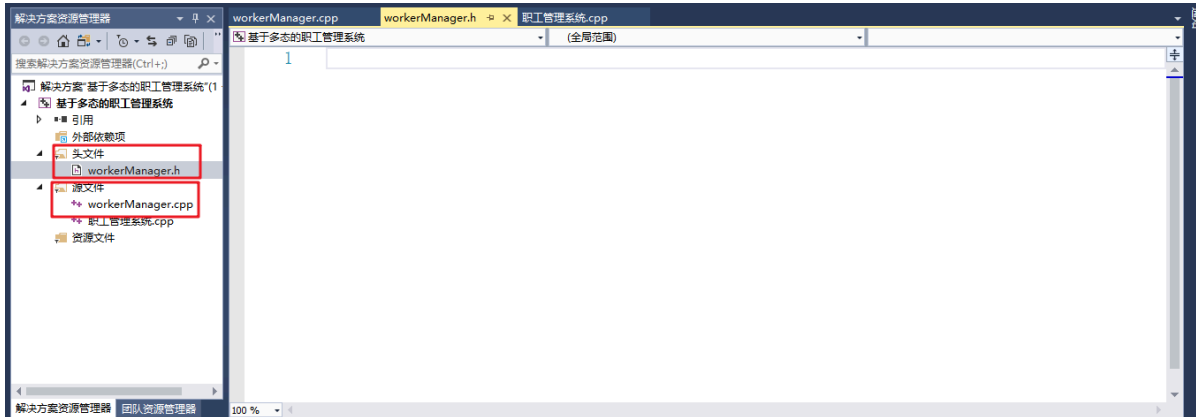
3、创建管理类

管理类负责的内容如下：

- 与用户的沟通菜单界面
- 对职工增删改查的操作
- 与文件的读写交互

3.1 创建文件

在头文件和源文件的文件夹下分别创建workerManager.h 和 workerManager.cpp文件



3.2 头文件实现

在workerManager.h中设计管理类

代码如下：

```
1  #pragma once
2  #include<iostream>
3  using namespace std;
4
5
6  class workerManager
7  {
8  public:
9
10     //构造函数
11     workerManager();
12
13     //析构函数
14     ~workerManager();
15
16 };
```

3.3 源文件实现

在workerManager.cpp中将构造和析构函数空实现补全

```
1  #include "workerManager.h"
2
3  workerManager::workerManager()
4  {
5  }
6
7  workerManager::~~workerManager()
8  {
9  }
10
```

至此职工管理类以创建完毕

4、菜单功能

功能描述：与用户的沟通界面

4.1 添加成员函数

在管理类workerManager.h中添加成员函数 `void Show_Menu();`

```
6 class WorkerManager
7 {
8 public:
9
10 //构造函数
11 WorkerManager();
12
13 //展示菜单
14 void Show_Menu();
15
16 //析构函数
17 ~WorkerManager();
18
19 };
```



4.2 菜单功能实现

在管理类workerManager.cpp中实现 `Show_Menu()`函数

```
1 void WorkerManager::Show_Menu()
2 {
3     cout << "*****" << endl;
4     cout << "***** 欢迎使用职工管理系统! *****" << endl;
5     cout << "***** 0.退出管理程序 *****" << endl;
6     cout << "***** 1.增加职工信息 *****" << endl;
7     cout << "***** 2.显示职工信息 *****" << endl;
8     cout << "***** 3.删除离职职工 *****" << endl;
9     cout << "***** 4.修改职工信息 *****" << endl;
10    cout << "***** 5.查找职工信息 *****" << endl;
11    cout << "***** 6.按照编号排序 *****" << endl;
12    cout << "***** 7.清空所有文档 *****" << endl;
13    cout << "*****" << endl;
14    cout << endl;
15 }
```

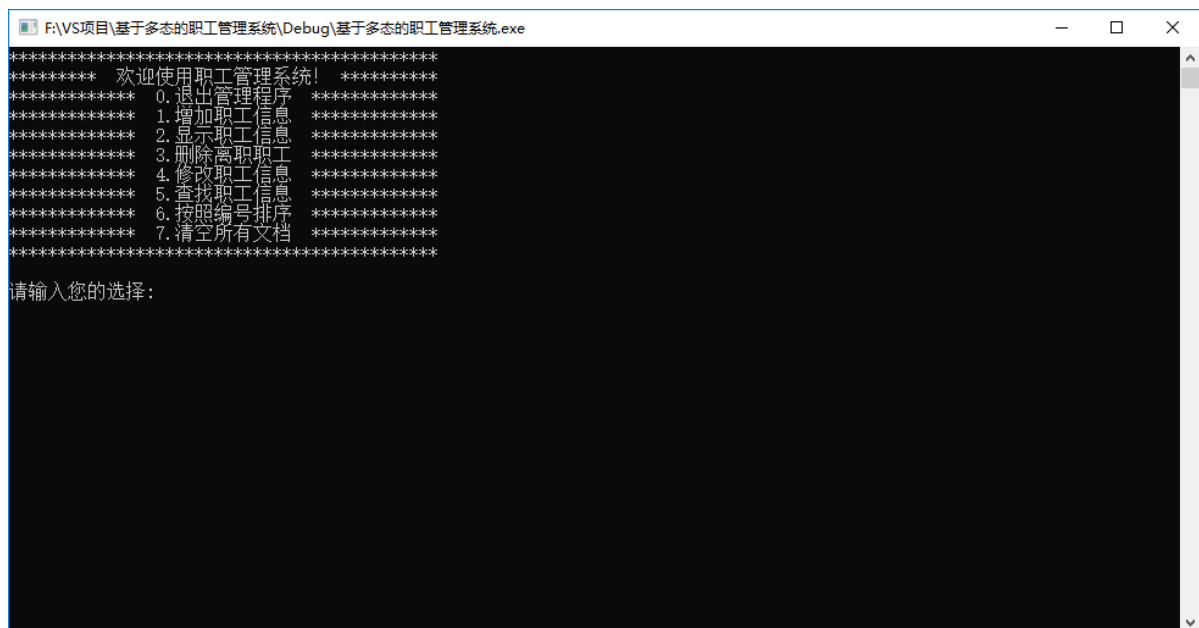
4.3 测试菜单功能

在职工管理系统.cpp中测试菜单功能

代码：

```
1  #include<iostream>
2  using namespace std;
3  #include "workerManager.h"
4
5  int main() {
6
7      WorkerManager wm;
8
9      wm.Show_Menu();
10
11     system("pause");
12
13     return 0;
14 }
```

运行效果如图：



5、退出功能

5.1 提供功能接口

在main函数中提供分支选择，提供每个功能接口

代码：

```
1  int main() {
2
3      WorkerManager wm;
4      int choice = 0;
5      while (true)
6      {
7          //展示菜单
8          wm.Show_Menu();
9          cout << "请输入您的选择:" << endl;
10         cin >> choice;
11
12         switch (choice)
13         {
14             case 0: //退出系统
15                 break;
16             case 1: //添加职工
17                 break;
18             case 2: //显示职工
19                 break;
20             case 3: //删除职工
21                 break;
22             case 4: //修改职工
23                 break;
24             case 5: //查找职工
25                 break;
26             case 6: //排序职工
27                 break;
28             case 7: //清空文件
29                 break;
30             default:
31                 system("cls");
32                 break;
33         }
34     }
35
36     system("pause");
37     return 0;
38 }
```


5.2 实现退出功能

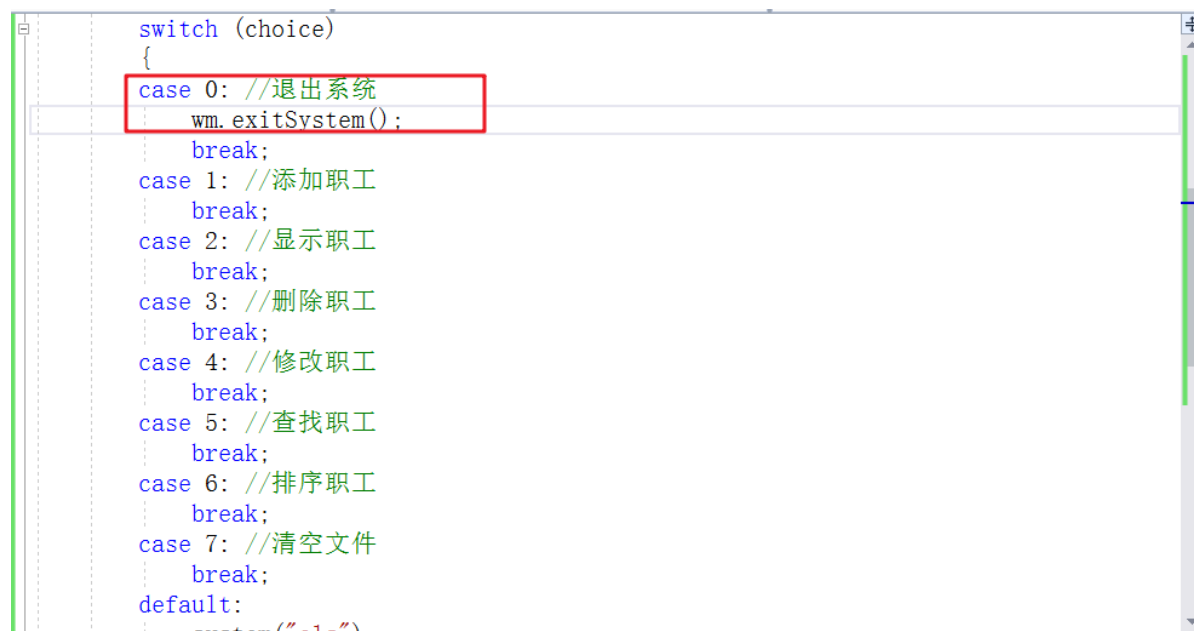
在workerManager.h中提供退出系统的成员函数 `void exitSystem();`

在workerManager.cpp中提供具体的功能实现

```
1 void WorkerManager::exitSystem()
2 {
3     cout << "欢迎下次使用" << endl;
4     system("pause");
5     exit(0);
6 }
```

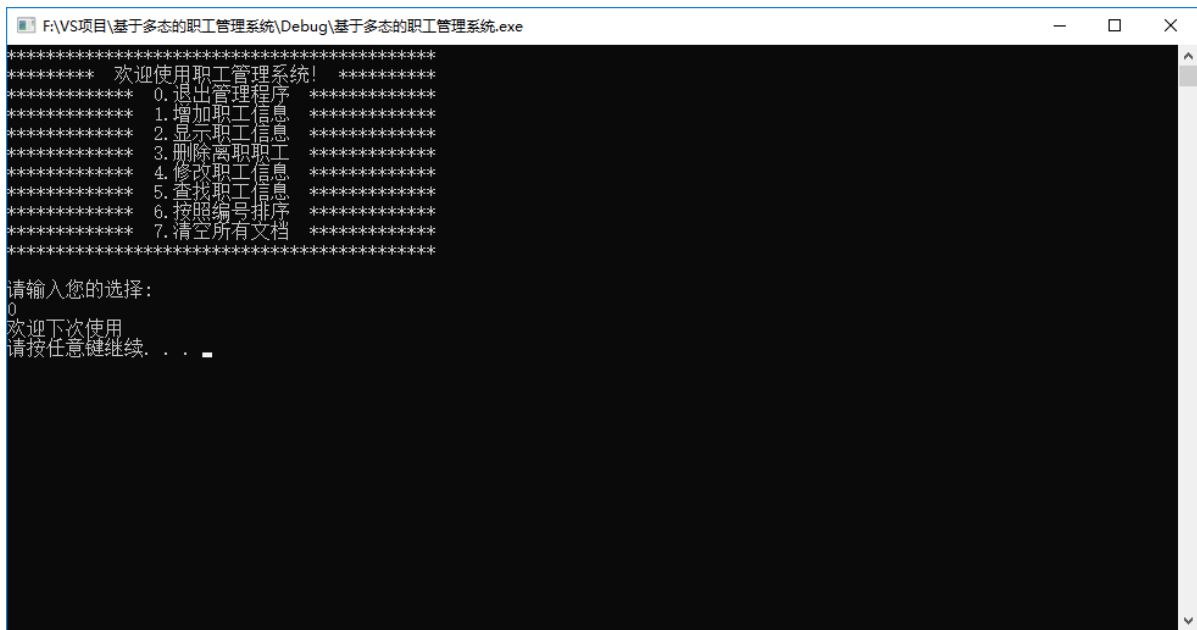
5.3 测试功能

在main函数分支 0 选项中，调用退出程序的接口



```
switch (choice)
{
    case 0: //退出系统
        wm.exitSystem();
        break;
    case 1: //添加职工
        break;
    case 2: //显示职工
        break;
    case 3: //删除职工
        break;
    case 4: //修改职工
        break;
    case 5: //查找职工
        break;
    case 6: //排序职工
        break;
    case 7: //清空文件
        break;
    default:
        system("-1-")
}
```

运行测试效果如图：



6、创建职工类

6.1 创建职工抽象类

职工的分类为：普通员工、经理、老板

将三种职工抽象到一个类（worker）中,利用多态管理不同职工种类

职工的属性为：职工编号、职工姓名、职工所在部门编号

职工的行为为：岗位职责信息描述，获取岗位名称

头文件文件夹下 创建文件worker.h 文件并且添加如下代码：

```
1  #pragma once
2  #include<iostream>
3  #include<string>
4  using namespace std;
5
6  //职工抽象基类
7  class Worker
8  {
9  public:
10
11      //显示个人信息
12      virtual void showInfo() = 0;
13      //获取岗位名称
14      virtual string getDeptName() = 0;
15
```

```

16     int m_Id; //职工编号
17     string m_Name; //职工姓名
18     int m_DeptId; //职工所在部门名称编号
19 };

```

6.2 创建普通员工类

普通员工类继承职工抽象类，并重写父类中纯虚函数

在头文件和源文件的文件夹下分别创建employee.h 和 employee.cpp文件

employee.h中代码如下：

```

1  #pragma once
2  #include<iostream>
3  using namespace std;
4  #include "worker.h"
5
6  //员工类
7  class Employee :public Worker
8  {
9  public:
10
11     //构造函数
12     Employee(int id, string name, int dId);
13
14     //显示个人信息
15     virtual void showInfo();
16
17     //获取职工岗位名称
18     virtual string getDeptName();
19 };

```

employee.cpp中代码如下：

```

1  #include "employee.h"
2
3  Employee::Employee(int id, string name, int dId)
4  {
5      this->m_Id = id;
6      this->m_Name = name;
7      this->m_DeptId = dId;
8  }
9
10 void Employee::showInfo()
11 {
12     cout << "职工编号: " << this->m_Id
13         << " \t职工姓名: " << this->m_Name
14         << " \t岗位: " << this->getDeptName()
15         << " \t岗位职责: 完成经理交给的任务" << endl;
16 }

```

```

17
18
19 string Employee::getDeptName()
20 {
21     return string("员工");
22 }
23
24

```

6.3 创建经理类

经理类继承职工抽象类，并重写父类中纯虚函数，和普通员工类似

在头文件和源文件的文件夹下分别创建manager.h 和 manager.cpp文件

manager.h中代码如下：

```

1  #pragma once
2  #include<iostream>
3  using namespace std;
4  #include "worker.h"
5
6  //经理类
7  class Manager :public Worker
8  {
9  public:
10
11      Manager(int id, string name, int dId);
12
13      //显示个人信息
14      virtual void showInfo();
15
16      //获取职工岗位名称
17      virtual string getDeptName();
18  };

```

manager.cpp中代码如下：

```

1  #include "manager.h"
2
3  Manager::Manager(int id, string name, int dId)
4  {
5      this->m_Id = id;
6      this->m_Name = name;
7      this->m_DeptId = dId;
8
9  }
10
11 void Manager::showInfo()
12 {

```

```

13     cout << "职工编号: " << this->m_Id
14         << " \t职工姓名: " << this->m_Name
15         << " \t岗位: " << this->getDeptName()
16         << " \t岗位职责: 完成老板交给的任务,并下发任务给员工" << endl;
17 }
18
19 string Manager::getDeptName()
20 {
21     return string("经理");
22 }
23
24

```

6.4 创建老板类

老板类继承职工抽象类，并重写父类中纯虚函数，和普通员工类似

在头文件和源文件的文件夹下分别创建boss.h 和 boss.cpp文件

boss.h中代码如下：

```

1  #pragma once
2  #include<iostream>
3  using namespace std;
4  #include "worker.h"
5
6  //老板类
7  class Boss :public worker
8  {
9  public:
10
11     Boss(int id, string name, int dId);
12
13     //显示个人信息
14     virtual void showInfo();
15
16     //获取职工岗位名称
17     virtual string getDeptName();
18 };

```

boss.cpp中代码如下：

```

1  #include "boss.h"
2
3  Boss::Boss(int id, string name, int dId)
4  {
5      this->m_Id = id;
6      this->m_Name = name;
7      this->m_DeptId = dId;
8
9  }

```

```

10
11 void Boss::showInfo()
12 {
13     cout << "职工编号: " << this->m_Id
14         << " \t职工姓名: " << this->m_Name
15         << " \t岗位: " << this->getDeptName()
16         << " \t岗位职责: 管理公司所有事务" << endl;
17 }
18
19 string Boss::getDeptName()
20 {
21     return string("总裁");
22 }
23

```

6.5 测试多态

在职工管理系统.cpp中添加测试函数，并且运行能够产生多态

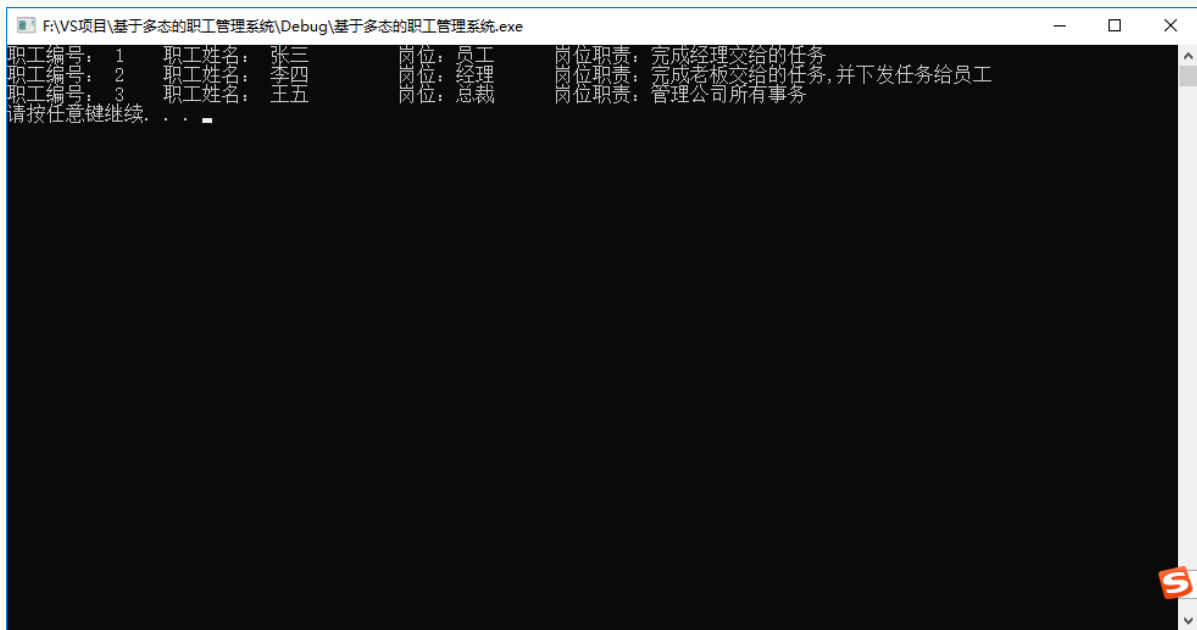
测试代码如下：

```

1  #include "worker.h"
2  #include "employee.h"
3  #include "manager.h"
4  #include "boss.h"
5
6
7  void test()
8  {
9      worker * worker = NULL;
10     worker = new Employee(1, "张三", 1);
11     worker->showInfo();
12     delete worker;
13
14     worker = new Manager(2, "李四", 2);
15     worker->showInfo();
16     delete worker;
17
18     worker = new Boss(3, "王五", 3);
19     worker->showInfo();
20     delete worker;
21 }
22

```

运行效果如图：



测试成功后，测试代码可以注释保留，或者选择删除

7、添加职工

功能描述：批量添加职工，并且保存到文件中

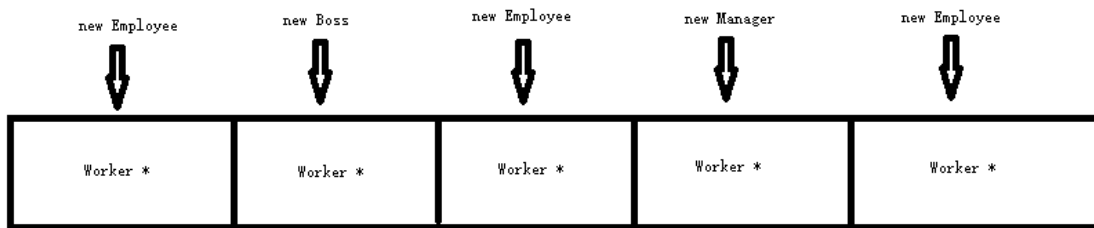
7.1 功能分析

分析：

用户在批量创建时，可能会创建不同种类的职工

如果想将所有不同种类的员工都放入到一个数组中，可以将所有员工的指针维护到一个数组里

如果想在程序中维护这个不定长度的数组，可以将数组创建到堆区，并利用Worker **的指针维护



堆区开辟数组

```
new Worker*[ 5 ]
```

7.2 功能实现

在WokerManager.h头文件中添加成员属性 代码：

```
1 //记录文件中的人数个数
2 int m_EmpNum;
3
4 //员工数组的指针
5 worker ** m_EmpArray;
```

在WorkerManager构造函数中初始化属性

```
1 WorkerManager::WorkerManager()
2 {
3     //初始化人数
4     this->m_EmpNum = 0;
5
6     //初始化数组指针
7     this->m_EmpArray = NULL;
8 }
```

在workerManager.h中添加成员函数

```
1 //增加职工
2 void Add_Emp();
```

workerManager.cpp中实现该函数

```
1 //增加职工
2 void WorkerManager::Add_Emp()
3 {
4     cout << "请输入增加职工数量: " << endl;
```



```

5
6     int addNum = 0;
7     cin >> addNum;
8
9     if (addNum > 0)
10    {
11        //计算新空间大小
12        int newSize = this->m_EmpNum + addNum;
13
14        //开辟新空间
15        worker ** newSpace = new worker*[newSize];
16
17        //将原空间下内容存放到新空间下
18        if (this->m_EmpArray != NULL)
19        {
20            for (int i = 0; i < this->m_EmpNum; i++)
21            {
22                newSpace[i] = this->m_EmpArray[i];
23            }
24        }
25
26        //输入新数据
27        for (int i = 0; i < addNum; i++)
28        {
29            int id;
30            string name;
31            int dSelect;
32
33            cout << "请输入第 " << i + 1 << " 个新职工编号: " << endl;
34            cin >> id;
35
36
37            cout << "请输入第 " << i + 1 << " 个新职工姓名: " << endl;
38            cin >> name;
39
40
41            cout << "请选择该职工的岗位: " << endl;
42            cout << "1、普通职工" << endl;
43            cout << "2、经理" << endl;
44            cout << "3、老板" << endl;
45            cin >> dSelect;
46
47
48            worker * worker = NULL;
49            switch (dSelect)
50            {
51                case 1: //普通员工
52                    worker = new Employee(id, name, 1);
53                    break;
54                case 2: //经理
55                    worker = new Manager(id, name, 2);
56                    break;
57                case 3: //老板
58                    worker = new Boss(id, name, 3);
59                    break;
60                default:

```

```

61         break;
62     }
63
64
65     newSpace[this->m_EmpNum + i] = worker;
66 }
67
68 //释放原有空间
69 delete[] this->m_EmpArray;
70
71 //更改新空间的指向
72 this->m_EmpArray = newSpace;
73
74 //更新新的个数
75 this->m_EmpNum = newSize;
76
77 //提示信息
78 cout << "成功添加" << addNum << "名新职工! " << endl;
79 }
80 else
81 {
82     cout << "输入有误" << endl;
83 }
84
85 system("pause");
86 system("cls");
87 }

```

在WorkerManager.cpp的析构函数中，释放堆区数据

```

1  WorkerManager::~~WorkerManager()
2  {
3      if (this->m_EmpArray != NULL)
4      {
5          delete[] this->m_EmpArray;
6      }
7  }
8

```

7.3 测试添加

在main函数分支 1 选项中，调用添加职工接口

```

switch (choice)
{
case 0: //退出系统
    wm.exitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    break;
case 3: //删除职工
    break;
case 4: //修改职工
    break;
case 5: //查找职工
    break;
case 6: //排序职工
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
}

```

效果如图：

```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
1
请输入增加职工数量:
1
请输入第 1 个新职工编号:
1
请输入第 1 个新职工姓名:
张三
请选择该职工的岗位:
1. 普通职工
2. 经理
3. 老板
1
成功添加1名新职工!
请按任意键继续. . .

```

至此，添加职工到程序中功能实现完毕

8、文件交互 - 写文件

功能描述：对文件进行读写

在上一个添加功能中，我们只是将所有数据添加到了内存中，一旦程序结束就无法保存了

因此文件管理类中需要一个与文件进行交互的功能，对于文件进行读写操作

8.1 设定文件路径

首先我们将文件路径，在workerManager.h中添加宏常量,并且包含头文件 fstream

```
1 #include <fstream>
2 #define FILENAME "empFile.txt"
```

8.2 成员函数声明

在workerManager.h中类里添加成员函数 `void save()`

```
1 //保存文件
2 void save();
```

8.3 保存文件功能实现

```
1 void WorkerManager::save()
2 {
3     ofstream ofs;
4     ofs.open(FILENAME, ios::out);
5
6
7     for (int i = 0; i < this->m_EmpNum; i++)
8     {
9         ofs << this->m_EmpArray[i]->m_Id << " "
10            << this->m_EmpArray[i]->m_Name << " "
11            << this->m_EmpArray[i]->m_DeptId << endl;
12     }
13
14     ofs.close();
15 }
```

8.4 保存文件功能测试

在添加职工功能中添加成功后添加保存文件函数

```
        newSpace[this->m_EmpNum + i] = worker;
    }

    //释放原有空间
    delete[] this->m_EmpArray;

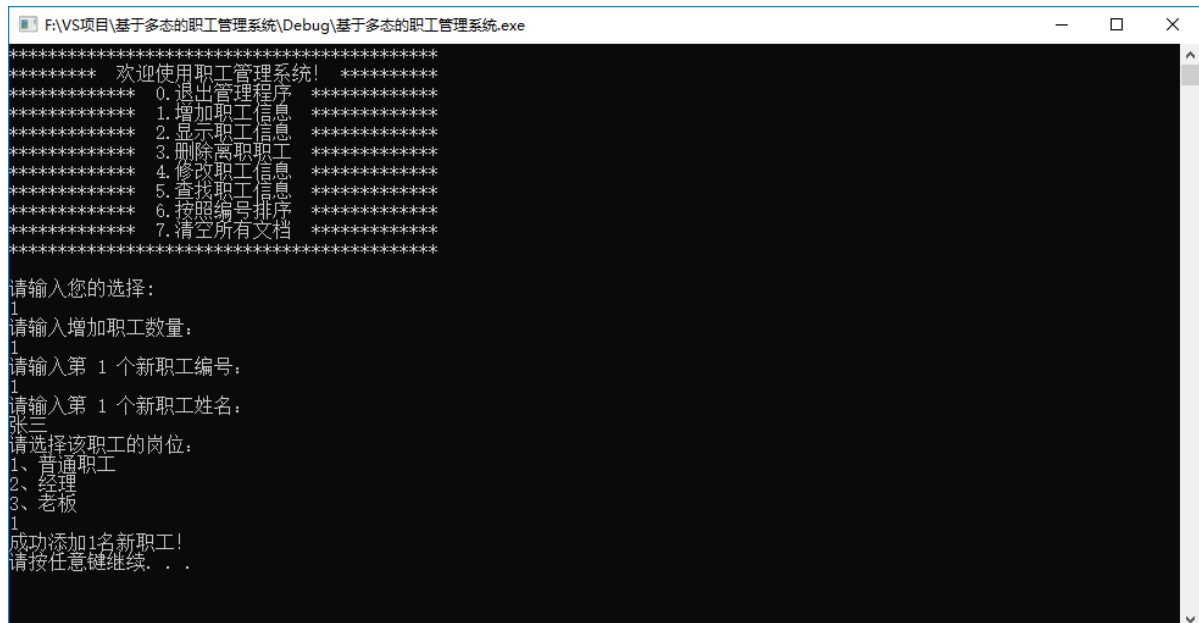
    //更改新空间的指向
    this->m_EmpArray = newSpace;

    //更新新的个数
    this->m_EmpNum = newSize;

    //提示信息
    cout << "成功添加" << addNum << "名新职工!" << endl;

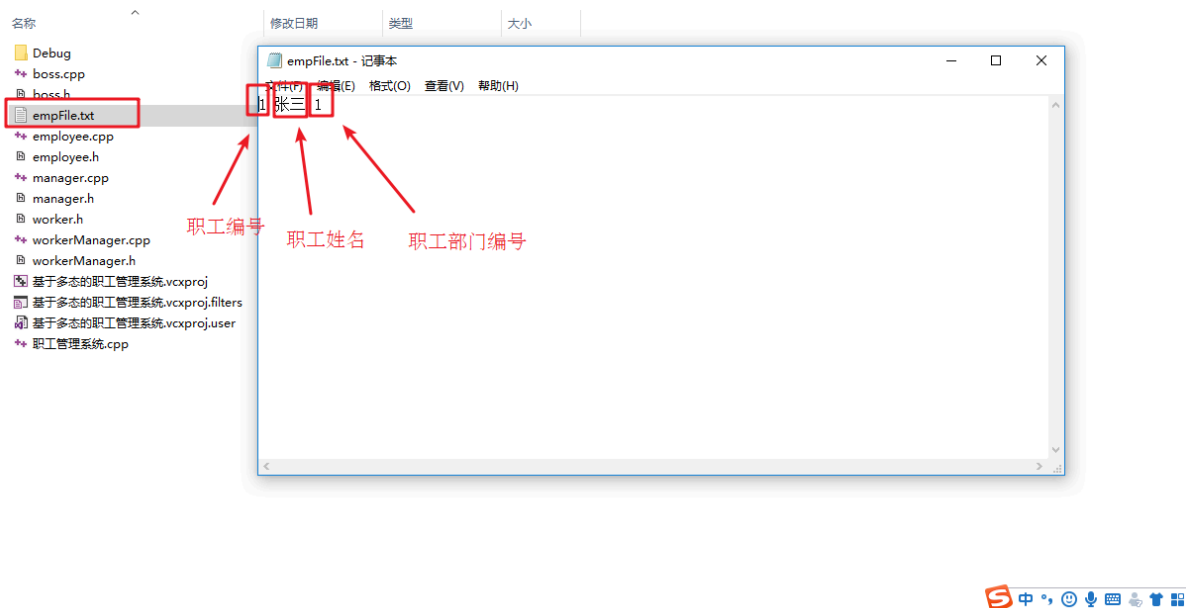
    //保存到文件中
    this->save();
```

再次运行代码，添加职工



```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
1
请输入增加职工数量:
1
请输入第 1 个新职工编号:
1
请输入第 1 个新职工姓名:
张三
请选择该职工的岗位:
1、普通职工
2、经理
3、老板
1
成功添加1名新职工!
请按任意键继续...
```

同级目录下多出文件，并且保存了添加的信息



9、文件交互 - 读文件

功能描述：将文件中的内容读取到程序中

虽然我们实现了添加职工后保存到文件的操作，但是每次开始运行程序，并没有将文件中数据读取到程序中

而我们的程序功能中还有清空文件的需求

因此构造函数初始化数据的情况分为三种

1. 第一次使用，文件未创建
2. 文件存在，但是数据被用户清空
3. 文件存在，并且保存职工的所有数据

9.1 文件未创建

在workerManager.h中添加新的成员属性 m_FileIsEmpty标志文件是否为空

```
1 //标志文件是否为空
2 bool m_FileIsEmpty;
```

修改WorkerManager.cpp中构造函数代码

```
1 WorkerManager::WorkerManager()
2 {
3     ifstream ifs;
4     ifs.open(FILENAME, ios::in);
5
6     //文件不存在情况
7     if (!ifs.is_open())
```

```

8      {
9          cout << "文件不存在" << endl; //测试输出
10         this->m_EmpNum = 0; //初始化人数
11         this->m_FileIsEmpty = true; //初始化文件为空标志
12         this->m_EmpArray = NULL; //初始化数组
13         ifs.close(); //关闭文件
14         return;
15     }
16 }

```

删除文件后，测试文件不存在时初始化数据功能

9.2 文件存在且数据为空

在workerManager.cpp中的构造函数追加代码：

```

1      //文件存在，并且没有记录
2      char ch;
3      ifs >> ch;
4      if (ifs.eof())
5      {
6          cout << "文件为空!" << endl;
7          this->m_EmpNum = 0;
8          this->m_FileIsEmpty = true;
9          this->m_EmpArray = NULL;
10         ifs.close();
11         return;
12     }

```

追加代码位置如图：

```

//文件不存在情况
if (!ifs.is_open())
{
    cout << "文件不存在" << endl;
    this->m_EmpNum = 0; //初始化人数
    this->m_FileIsEmpty = true; //初始化文件为空标志
    this->m_EmpArray = NULL; //初始化数组
    ifs.close(); //关闭文件
    return;
}

//文件存在，并且没有记录
char ch;
ifs >> ch;
if (ifs.eof())
{
    cout << "文件为空!" << endl;
    this->m_EmpNum = 0;
    this->m_FileIsEmpty = true;
    this->m_EmpArray = NULL;
    ifs.close();
    return;
}

```

将文件创建后清空文件内容，并测试该情况下初始化功能

我们发现文件不存在或者为空清空 m_FileIsEmpty 判断文件是否为空的标志都为真，那何时为假？

成功添加职工后，应该更改文件不为空的标志

在 void WorkerManager::Add_Emp() 成员函数中添加：

```

1 //更新职工不为空标志
2 this->m_FileIsEmpty = false;

```



```

//更改新空间的指向
this->m_EmpArray = newSpace;

//更新新的个数
this->m_EmpNum = newSize;

//更新职工不为空标志
this->m_FileIsEmpty = false;

//提示信息
cout << "成功添加" << addNum << "名新职工!" << endl;

//保存到文件中
this->save();

```

9.3 文件存在且保存职工数据

9.3.1 获取记录的职工人数

在workerManager.h中添加成员函数 `int get_EmpNum();`

```

1 //统计人数
2 int get_EmpNum();

```

workerManager.cpp中实现

```

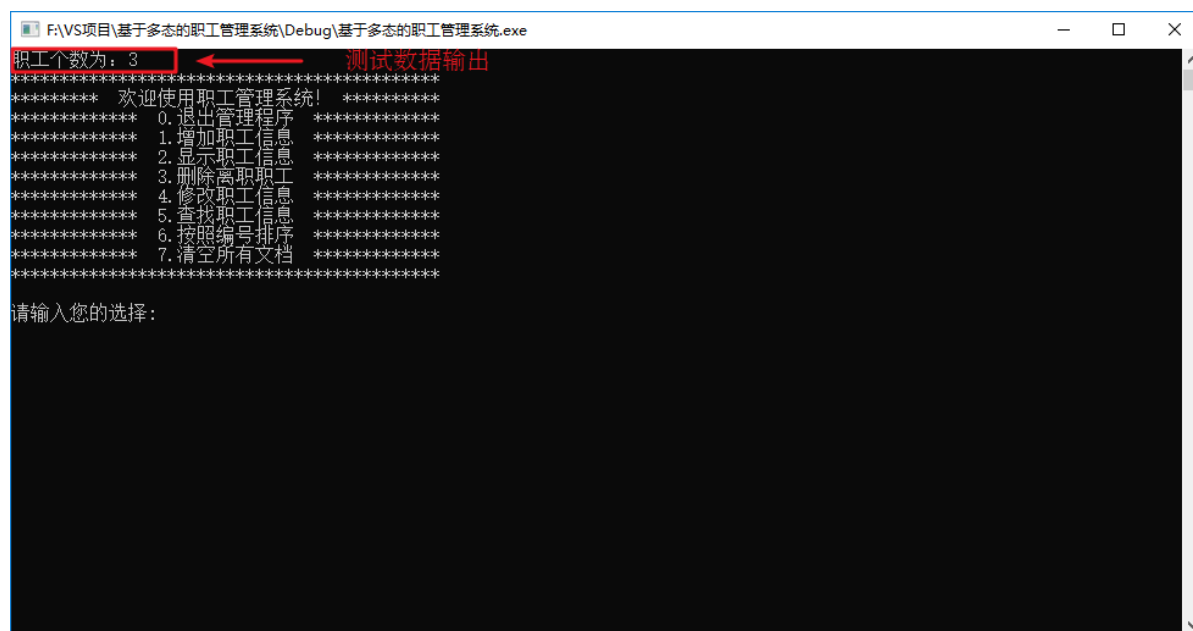
1 int WorkerManager::get_EmpNum()
2 {
3     ifstream ifs;
4     ifs.open(FILENAME, ios::in);
5
6     int id;
7     string name;
8     int did;
9
10    int num = 0;
11
12    while (ifs >> id && ifs >> name && ifs >> did)
13    {
14        //记录人数
15        num++;
16    }
17    ifs.close();
18
19    return num;
20 }

```

在workerManager.cpp构造函数中继续追加代码：

```
1   int num = this->get_EmpNum();
2   cout << "职工个数为：" << num << endl; //测试代码
3   this->m_EmpNum = num; //更新成员属性
```

手动添加一些职工数据，测试获取职工数量函数



9.3.2 初始化数组

根据职工的数据以及职工数据，初始化workerManager中的Worker ** m_EmpArray 指针

在WorkerManager.h中添加成员函数 `void init_Emp();`

```
1 //初始化员工
2 void init_Emp();
```

在WorkerManager.cpp中实现

```
1 void WorkerManager::init_Emp()
2 {
3     ifstream ifs;
4     ifs.open(FILENAME, ios::in);
5
6     int id;
7     string name;
8     int dId;
9
10    int index = 0;
11    while (ifs >> id && ifs >> name && ifs >> dId)
12    {
13        worker * worker = NULL;
14        //根据不同的部门Id创建不同对象
15        if (dId == 1) // 1普通员工
16        {
17            worker = new Employee(id, name, dId);
18        }
19        else if (dId == 2) //2经理
20        {
21            worker = new Manager(id, name, dId);
22        }
23        else //总裁
24        {
25            worker = new Boss(id, name, dId);
26        }
27        //存放在数组中
28        this->m_EmpArray[index] = worker;
29        index++;
30    }
31 }
```

在workerManager.cpp构造函数中追加代码

```

1      //根据职工数创建数组
2      this->m_EmpArray = new worker *[this->m_EmpNum];
3      //初始化职工
4      init_Emp();
5
6      //测试代码
7      for (int i = 0; i < m_EmpNum; i++)
8      {
9          cout << "职工号: " << this->m_EmpArray[i]->m_Id
10             << " 职工姓名: " << this->m_EmpArray[i]->m_Name
11             << " 部门编号: " << this->m_EmpArray[i]->m_DeptId << endl;
12      }

```

运行程序，测试从文件中获取的数据

```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
职工号: 1 职工姓名: 张三 部门编号: 1
职工号: 2 职工姓名: 李四 部门编号: 2
职工号: 3 职工姓名: 王五 部门编号: 3
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:

```

至此初始化数据功能完毕，测试代码可以注释或删除掉！

10、显示职工

功能描述：显示当前所有职工信息

10.1 显示职工函数声明

在workerManager.h中添加成员函数 `void Show_Emp();`

```

1      //显示职工
2      void Show_Emp();

```

10.2 显示职工函数实现

在workerManager.cpp中实现成员函数 `void Show_Emp();`

```
1 //显示职工
2 void WorkerManager::Show_Emp()
3 {
4     if (this->m_FileIsEmpty)
5     {
6         cout << "文件不存在或记录为空!" << endl;
7     }
8     else
9     {
10         for (int i = 0; i < m_EmpNum; i++)
11         {
12             //利用多态调用接口
13             this->m_EmpArray[i]->showInfo();
14         }
15     }
16
17     system("pause");
18     system("cls");
19 }
```

10.3 测试显示职工

在main函数分支2 选项中，调用显示职工接口

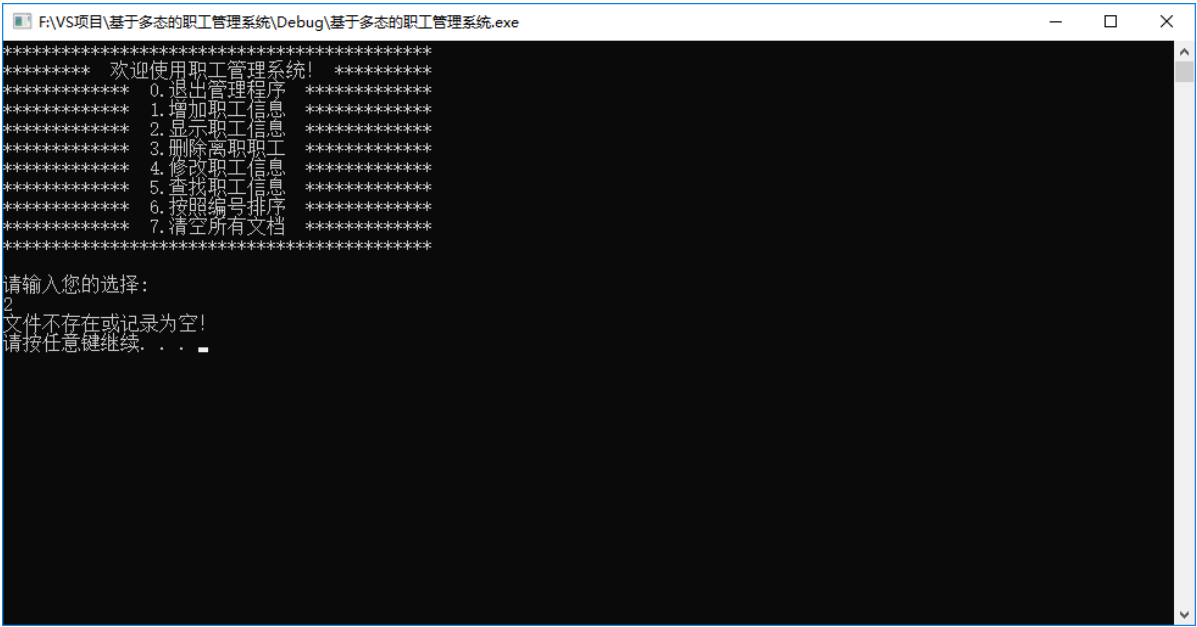
```
switch (choice)
{
case 0: //退出系统
    wm.exitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    break;
case 4: //修改职工
    break;
case 5: //查找职工
    break;
case 6: //排序职工
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
}
```

添加显示职工接口

测试时分别测试 文件为空和文件不为空两种情况

测试效果：

测试1-文件不存在或者为空情况



测试2 - 文件存在且有记录情况



测试完毕，至此，显示所有职工信息功能实现

11、删除职工

功能描述：按照职工的编号进行删除职工操作

11.1 删除职工函数声明

在workerManager.h中添加成员函数 `void Del_Emp();`

```
1 //删除职工
2 void Del_Emp();
```

11.2 职工是否存在函数声明

很多功能都需要用到根据职工是否存在来进行操作如：删除职工、修改职工、查找职工

因此添加该函数，以便后续调用

在workerManager.h中添加成员函数 `int IsExist(int id);`

```
1 //按照职工编号判断职工是否存在,若存在返回职工在数组中位置，不存在返回-1
2 int IsExist(int id);
```

11.3 职工是否存在函数实现

在workerManager.cpp中实现成员函数 `int IsExist(int id);`

```
1 int WorkerManager::IsExist(int id)
2 {
3     int index = -1;
4
5     for (int i = 0; i < this->m_EmpNum; i++)
6     {
7         if (this->m_EmpArray[i]->m_Id == id)
8         {
9             index = i;
10
11             break;
12         }
13     }
14
15     return index;
16 }
```

11.4 删除职工函数实现

在workerManager.cpp中实现成员函数 `void Del_Emp();`

```
1  //删除职工
2  void WorkerManager::Del_Emp()
3  {
4      if (this->m_FileIsEmpty)
5      {
6          cout << "文件不存在或记录为空!" << endl;
7      }
8      else
9      {
10         //按职工编号删除
11         cout << "请输入想要删除的职工号: " << endl;
12         int id = 0;
13         cin >> id;
14
15         int index = this->IsExist(id);
16
17         if (index != -1) //说明index上位置数据需要删除
18         {
19             for (int i = index; i < this->m_EmpNum - 1; i++)
20             {
21                 this->m_EmpArray[i] = this->m_EmpArray[i + 1];
22             }
23             this->m_EmpNum--;
24
25             this->save(); //删除后数据同步到文件中
26             cout << "删除成功!" << endl;
27         }
28         else
29         {
30             cout << "删除失败, 未找到该职工" << endl;
31         }
32     }
33
34     system("pause");
35     system("cls");
36 }
```

11.5 测试删除职工

在main函数分支3 选项中, 调用删除职工接口

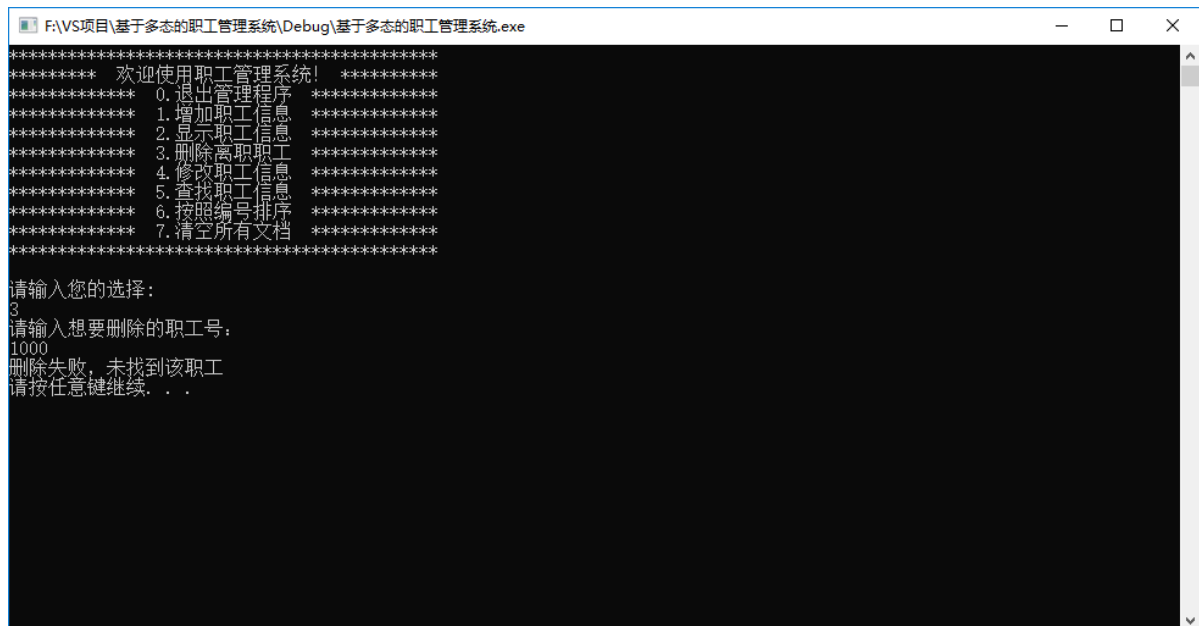

```

switch (choice)
{
case 0: //退出系统
    wm.ExitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    wm.Del_Emp();
    break;
case 4: //修改职工
    break;
case 5: //查找职工
    break;
case 6: //排序职工
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
}

```

调用删除职工接口

测试1 - 删除不存在职工情况



```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 1. 增加职工信息 *****
***** 2. 显示职工信息 *****
***** 3. 删除职工信息 *****
***** 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:
3
请输入想要删除的职工号:
1000
删除失败, 未找到该职工
请按任意键继续. . .

```

测试2 - 删除存在的职工情况

删除成功提示图:

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe

***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
3
请输入想要删除的职工号:
1
删除成功!
请按任意键继续. . .
```

再次显示所有职工信息，确保已经删除

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe

***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
2
职工编号: 2    职工姓名: 李四    岗位: 经理    岗位职责: 完成老板交给的任务, 并下发任务给员工
职工编号: 3    职工姓名: 王五    岗位: 总裁    岗位职责: 管理公司所有事务
职工编号: 4    职工姓名: 赵六    岗位: 经理    岗位职责: 完成老板交给的任务, 并下发任务给员工
请按任意键继续. . .
```

查看文件中信息，再次核实员工已被完全删除

```
empFile.txt - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
2 李四 2
3 王五 3
4 赵六 2
```

至此，删除职工功能实现完毕！

12、修改职工

功能描述：能够按照职工的编号对职工信息进行修改并保存

12.1 修改职工函数声明

在workerManager.h中添加成员函数 `void Mod_Emp();`

```
1 //修改职工
2 void Mod_Emp();
```

12.2 修改职工函数实现

在workerManager.cpp中实现成员函数 `void Mod_Emp();`

```
1 //修改职工
2 void WorkerManager::Mod_Emp()
3 {
4     if (this->m_FileIsEmpty)
5     {
6         cout << "文件不存在或记录为空！" << endl;
7     }
8     else
9     {
10        cout << "请输入修改职工的编号：" << endl;
11        int id;
12        cin >> id;
13
14        int ret = this->IsExist(id);
15        if (ret != -1)
16        {
17            //查找到编号的职工
18
19            delete this->m_EmpArray[ret];
20
21            int newId = 0;
22            string newName = "";
23            int dSelect = 0;
24
25            cout << "查到：" << id << "号职工，请输入新职工号：" << endl;
26            cin >> newId;
27
28            cout << "请输入新姓名：" << endl;
29            cin >> newName;
```

```

30
31     cout << "请输入岗位:  " << endl;
32     cout << "1、普通职工" << endl;
33     cout << "2、经理" << endl;
34     cout << "3、老板" << endl;
35     cin >> dSelect;
36
37     Worker * worker = NULL;
38     switch (dSelect)
39     {
40     case1:
41         worker = new Employee(newId, newName, dSelect);
42         break;
43     case2:
44         worker = new Manager(newId, newName, dSelect);
45         break;
46     case 3:
47         worker = new Boss(newId, newName, dSelect);
48         break;
49     default:
50         break;
51     }
52
53     //更改数据 到数组中
54     this->m_EmpArray[ret]= worker;
55
56     cout << "修改成功! " << endl;
57
58     //保存到文件中
59     this->save();
60 }
61 else
62 {
63     cout << "修改失败, 查无此人" << endl;
64 }
65 }
66
67 //按任意键 清屏
68 system("pause");
69 system("cls");
70 }
71

```

12.3 测试修改职工

在main函数分支4 选项中, 调用修改职工接口

```

switch (choice)
{
case 0: //退出系统
    wm.exitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    wm.Del_Emp();
    break;
case 4: //修改职工
    wm.Mod_Emp();
    break;
case 5: //查找职工
    break;
case 6: //排序职工
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
}

```

调用修改职工接口

测试1 - 修改不存在职工情况

```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入修改职工的编号:
1000
修改失败，查无此人
请按任意键继续. . .

```

测试2 - 修改存在职工情况，例如将职工 "李四" 改为 "赵四"

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入修改职工的编号:
2
查到: 2号职工, 请输入新职工号:
2
请输入新姓名:
赵四
请输入岗位:
1、普通职工
2、经理
3、老板
2
修改成功
请按任意键继续. . .
```

修改后再次查看所有职工信息，并确认修改成功

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
2
职工编号: 2    职工姓名: 赵四    岗位: 经理    岗位职责: 完成老板交给的任务, 并下发任务给员工
职工编号: 3    职工姓名: 王五    岗位: 总裁    岗位职责: 管理公司所有事务
职工编号: 4    职工姓名: 赵六    岗位: 经理    岗位职责: 完成老板交给的任务, 并下发任务给员工
请按任意键继续. . .
```

再次确认文件中信息也同步更新

```
empFile.txt - 记事本
文件(F)  编辑(E)  格式(O)  查看(V)  帮助(H)
2 赵四 2
3 王五 3
4 赵六 2
```

至此，修改职工功能已实现！

13、查找职工

功能描述：提供两种查找职工方式，一种按照职工编号，一种按照职工姓名

13.1 查找职工函数声明

在workerManager.h中添加成员函数 `void Find_Emp();`

```
1 //查找职工
2 void Find_Emp();
```

13.2 查找职工函数实现

在workerManager.cpp中实现成员函数 `void Find_Emp();`

```
1 //查找职工
2 void WorkerManager::Find_Emp()
3 {
4     if (this->m_FileIsEmpty)
5     {
6         cout << "文件不存在或记录为空！" << endl;
7     }
8     else
9     {
10        cout << "请输入查找的方式：" << endl;
11        cout << "1、按职工编号查找" << endl;
12        cout << "2、按姓名查找" << endl;
13
14        int select = 0;
15        cin >> select;
16
17
18        if (select == 1) //按职工号查找
19        {
20            int id;
21            cout << "请输入查找的职工编号：" << endl;
22            cin >> id;
23
24            int ret = IsExist(id);
25            if (ret != -1)
26            {
27                cout << "查找成功！该职工信息如下：" << endl;
28                this->m_EmpArray[ret]->showInfo();
```

```

29         }
30         else
31         {
32             cout << "查找失败，查无此人" << endl;
33         }
34     }
35     else if(select == 2) //按姓名查找
36     {
37         string name;
38         cout << "请输入查找的姓名: " << endl;
39         cin >> name;
40
41         bool flag = false; //查找到的标志
42         for (int i = 0; i < m_EmpNum; i++)
43         {
44             if (m_EmpArray[i]->m_Name == name)
45             {
46                 cout << "查找成功,职工编号为: "
47                     << m_EmpArray[i]->m_Id
48                     << " 号的信息如下: " << endl;
49
50                 flag = true;
51
52                 this->m_EmpArray[i]->showInfo();
53             }
54         }
55         if (flag == false)
56         {
57             //查无此人
58             cout << "查找失败，查无此人" << endl;
59         }
60     }
61     else
62     {
63         cout << "输入选项有误" << endl;
64     }
65 }
66
67
68 system("pause");
69 system("cls");
70 }

```

13.3 测试查找职工

在main函数分支 5 选项中，调用查找职工接口

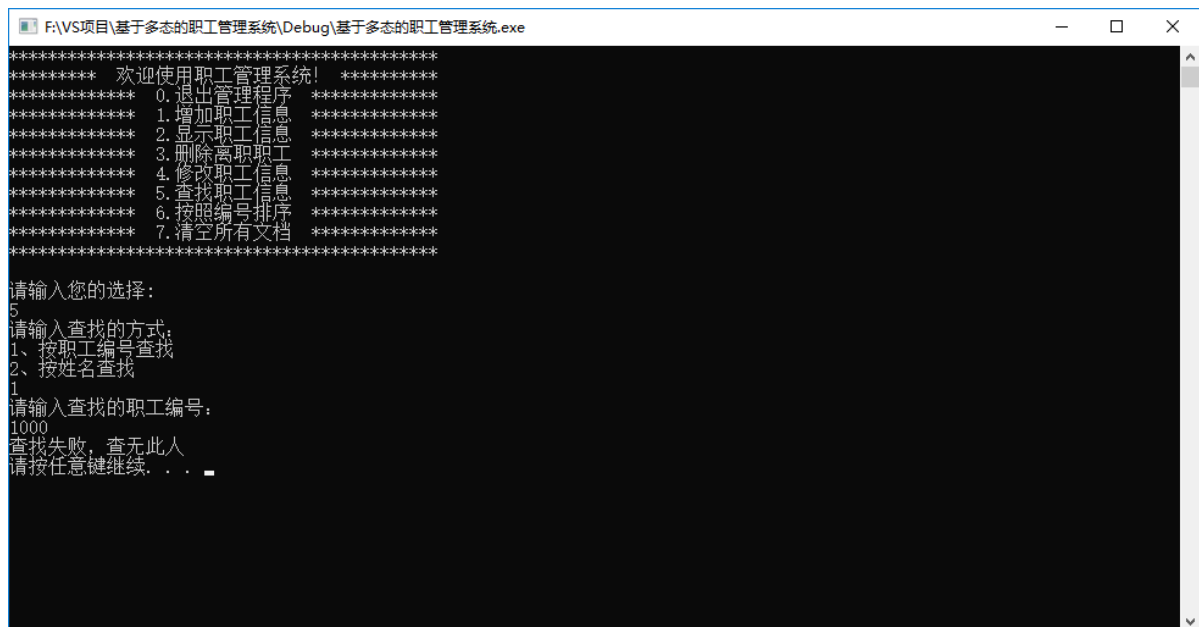

```

switch (choice)
{
case 0: //退出系统
    wm.ExitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    wm.Del_Emp();
    break;
case 4: //修改职工
    wm.Mod_Emp();
    break;
case 5: //查找职工
    wm.Find_Emp();
    break;
case 6: //排序职工
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
}

```

调用查找职工接口

测试1 - 按照职工编号查找 - 查找不存在职工



```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入查找的方式:
1、按职工编号查找
2、按姓名查找
1
请输入查找的职工编号:
1000
查找失败, 查无此人
请按任意键继续. . .

```

测试2 - 按照职工编号查找 - 查找存在职工

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入查找的方式:
1、按职工编号查找
2、按姓名查找
1
请输入查找的职工编号:
2
查找成功! 该职工信息如下:
职工编号: 2    职工姓名: 赵四    岗位: 经理    岗位职责: 完成老板交给的任务, 并下发任务给员工
请按任意键继续. . .
```

测试3 - 按照职工姓名查找 - 查找不存在职工

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入查找的方式:
1、按职工编号查找
2、按姓名查找
2
请输入查找的姓名:
张三丰
查找失败, 查无此人
请按任意键继续. . .
```

测试4 - 按照职工姓名查找 - 查找存在职工（如果出现重名，也一并显示，在文件中可以添加重名职工）

例如 添加两个王五的职工，然后按照姓名查找王五

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
2
职工编号: 2 职工姓名: 赵四 岗位: 经理 岗位职责: 完成老板交给的任务,并下发任务给员工
职工编号: 3 职工姓名: 王五 岗位: 总裁 岗位职责: 管理公司所有事务
职工编号: 4 职工姓名: 赵六 岗位: 经理 岗位职责: 完成老板交给的任务,并下发任务给员工
职工编号: 1 职工姓名: 王五 岗位: 员工 岗位职责: 完成经理交给的任务
请任意键继续...
```

同名职工

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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
请输入查找的方式:
1、按职工编号查找
2、按姓名查找
2
请输入查找的姓名:
王五
查找成功,职工编号为: 3 号的信息如下:
职工编号: 3 职工姓名: 王五 岗位: 总裁 岗位职责: 管理公司所有事务
查找成功,职工编号为: 1 号的信息如下:
职工编号: 1 职工姓名: 王五 岗位: 员工 岗位职责: 完成经理交给的任务
请任意键继续...
```

同名人员信息全部显示

至此，查找职工功能实现完毕！

14、排序

功能描述：按照职工编号进行排序，排序的顺序由用户指定

14.1 排序函数声明

在workerManager.h中添加成员函数 `void Sort_Emp();`

```
1 //排序职工
2 void Sort_Emp();
```

14.2 排序函数实现

在workerManager.cpp中实现成员函数 `void Sort_Emp();`

```
1  //排序职工
2  void WorkerManager::Sort_Emp()
3  {
4      if (this->m_FileIsEmpty)
5      {
6          cout << "文件不存在或记录为空! " << endl;
7          system("pause");
8          system("cls");
9      }
10     else
11     {
12         cout << "请选择排序方式: " << endl;
13         cout << "1、按职工号进行升序" << endl;
14         cout << "2、按职工号进行降序" << endl;
15
16         int select = 0;
17         cin >> select;
18
19
20         for (int i = 0; i < m_EmpNum; i++)
21         {
22             int minOrMax = i;
23             for (int j = i + 1; j < m_EmpNum; j++)
24             {
25                 if (select == 1) //升序
26                 {
27                     if (m_EmpArray[minOrMax]->m_Id > m_EmpArray[j]->m_Id)
28                     {
29                         minOrMax = j;
30                     }
31                 }
32                 else //降序
33                 {
34                     if (m_EmpArray[minOrMax]->m_Id < m_EmpArray[j]->m_Id)
35                     {
36                         minOrMax = j;
37                     }
38                 }
39             }
40
41             if (i != minOrMax)
42             {
43                 worker * temp = m_EmpArray[i];
44                 m_EmpArray[i] = m_EmpArray[minOrMax];
45                 m_EmpArray[minOrMax] = temp;
46             }
47
48         }
49
50         cout << "排序成功,排序后结果为: " << endl;
51         this->save();
```

```
52         this->Show_Emp();
53     }
54
55 }
```

14.3 测试排序功能

在main函数分支 6 选项中，调用排序职工接口

```
switch (choice)
{
case 0: //退出系统
    wm.exitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    wm.Del_Emp();
    break;
case 4: //修改职工
    wm.Mod_Emp();
    break;
case 5: //查找职工
    wm.Find_Emp();
    break;
case 6: //排序职工
    wm.Sort_Emp();
    break;
case 7: //清空文件
    break;
default:
    system("cls");
    break;
```

调用排序职工接口



测试：

首先我们添加一些职工，序号是无序的，例如：

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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 *****
*****

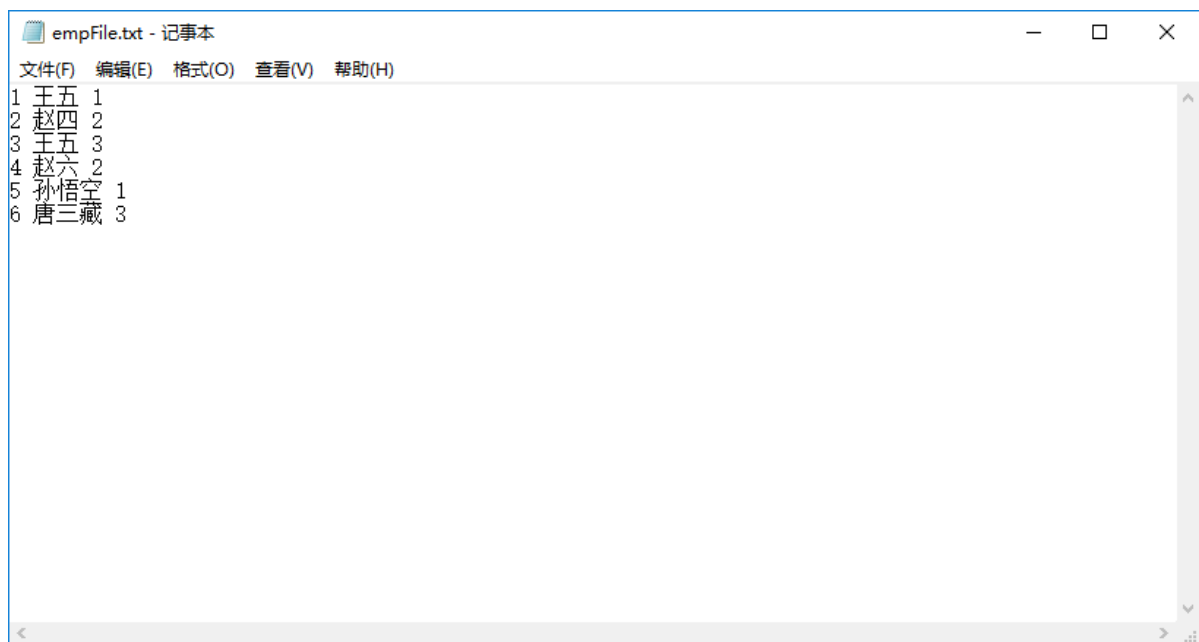
请输入您的选择:
2
职工编号: 2 职工姓名: 赵四 岗位: 经理 岗位职责: 完成老板交给的任务, 并下发任务给员工
职工编号: 3 职工姓名: 王五 岗位: 总裁 岗位职责: 管理公司所有事务, 并下发任务给员工
职工编号: 4 职工姓名: 赵六 岗位: 经理 岗位职责: 完成老板交给的任务, 并下发任务给员工
职工编号: 1 职工姓名: 王五 岗位: 员工 岗位职责: 完成经理交给的任务
职工编号: 5 职工姓名: 孙悟空 岗位: 员工 岗位职责: 完成经理交给的任务
职工编号: 6 职工姓名: 唐三藏 岗位: 总裁 岗位职责: 管理公司所有事务
请按任意键继续. . .
```

测试 - 升序排序

```
F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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、按职工号进行升序
2、按职工号进行降序
1
排序成功, 排序后结果为:
职工编号: 1 职工姓名: 王五 岗位: 员工 岗位职责: 完成经理交给的任务, 并下发任务给员工
职工编号: 2 职工姓名: 赵四 岗位: 总裁 岗位职责: 管理公司所有事务, 并下发任务给员工
职工编号: 3 职工姓名: 王五 岗位: 经理 岗位职责: 完成老板交给的任务, 并下发任务给员工
职工编号: 4 职工姓名: 赵六 岗位: 员工 岗位职责: 完成经理交给的任务
职工编号: 5 职工姓名: 孙悟空 岗位: 员工 岗位职责: 完成经理交给的任务
职工编号: 6 职工姓名: 唐三藏 岗位: 总裁 岗位职责: 管理公司所有事务
请按任意键继续. . .
```

文件同步更新



测试 - 降序排序



文件同步更新



至此，职工按照编号排序的功能实现完毕！

15、清空文件

功能描述：将文件中记录数据清空

15.1 清空函数声明

在workerManager.h中添加成员函数 `void Clean_File();`

```
1 //清空文件
2 void Clean_File();
```

15.2 清空函数实现

在workerManager.cpp中实现员函数 `void Clean_File();`

```
1 //清空文件
2 void WorkerManager::Clean_File()
3 {
4     cout << "确认清空? " << endl;
5     cout << "1、确认" << endl;
6     cout << "2、返回" << endl;
7 }
```



```

8      int select = 0;
9      cin >> select;
10
11     if (select == 1)
12     {
13         //打开模式 ios::trunc 如果存在删除文件并重新创建
14         ofstream ofs(FILENAME, ios::trunc);
15         ofs.close();
16
17         if (this->m_EmpArray != NULL)
18         {
19             for (int i = 0; i < this->m_EmpNum; i++)
20             {
21                 if (this->m_EmpArray[i] != NULL)
22                 {
23                     delete this->m_EmpArray[i];
24                 }
25             }
26             this->m_EmpNum = 0;
27             delete[] this->m_EmpArray;
28             this->m_EmpArray = NULL;
29             this->m_FileIsEmpty = true;
30         }
31         cout << "清空成功! " << endl;
32     }
33
34     system("pause");
35     system("cls");
36 }

```

15.3 测试清空文件

在main函数分支 7 选项中，调用清空文件接口

```

switch (choice)
{
case 0: //退出系统
    wm.ExitSystem();
    break;
case 1: //添加职工
    wm.Add_Emp();
    break;
case 2: //显示职工
    wm.Show_Emp();
    break;
case 3: //删除职工
    wm.Del_Emp();
    break;
case 4: //修改职工
    wm.Mod_Emp();
    break;
case 5: //查找职工
    wm.Find_Emp();
    break;
case 6: //排序职工
    wm.Sort_Emp();
    break;
case 7: //清空文件
    wm.Clean_File();
    break;
default:
    system("cls");
    break;
}

```

调用清空文件接口

测试：确认清空文件

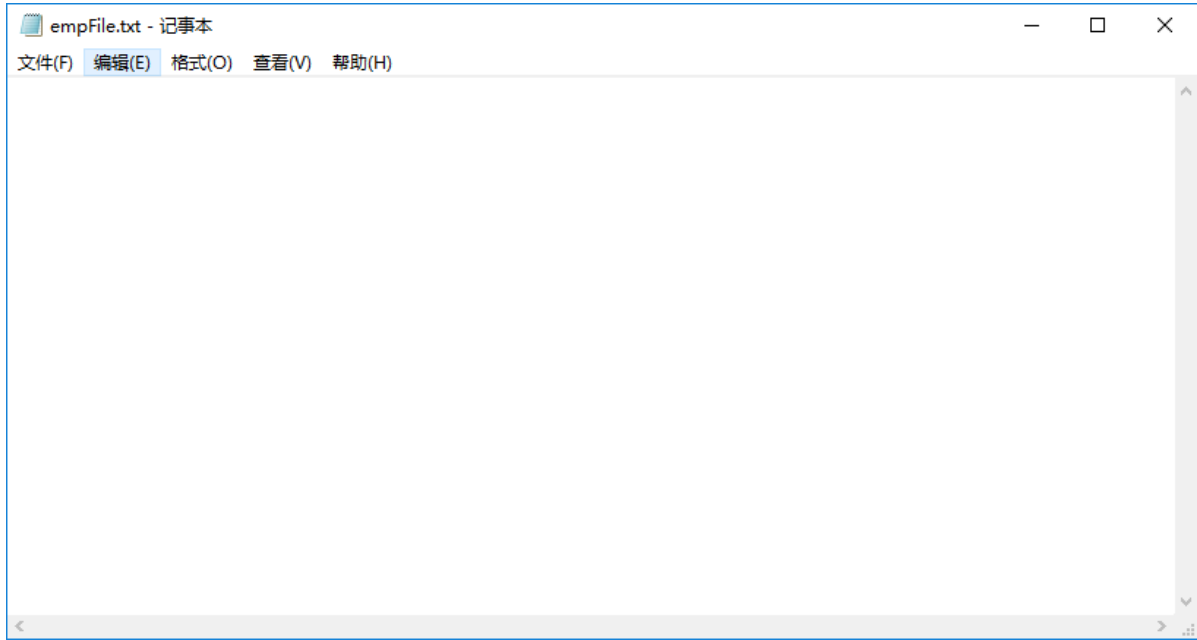
```

F:\VS项目\基于多态的职工管理系统\Debug\基于多态的职工管理系统.exe
***** 欢迎使用职工管理系统! *****
***** 0. 退出管理程序 *****
***** 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:
7
确认清空?
1、确认
2、返回
1
清空成功!
请按任意键继续. . .

```

再次查看文件中数据，记录已为空

打开文件，里面数据已确保清空，该功能需要慎用！



随着清空文件功能实现，本案例制作完毕 ^_^