1.Problem Description

Faculty management is not only an indispensable part of School operation, but also an important guarantee of school's stability. How to realize convenient and permanent faculty management, on the one hand, it is convenient for school leaders to operate teachers, on the other hand, it is my original intention to bid farewell to the past file type faculty management and make the storage of faculty records more reliable.

Faculty information management system is used to manage faculty information Be able to accurately query employee information according to job number, name and department Be able to count the professional titles by department, calculate the number of people with each professional title, sort and output according to the professional titles of employees and modify or delete employee information according to job number

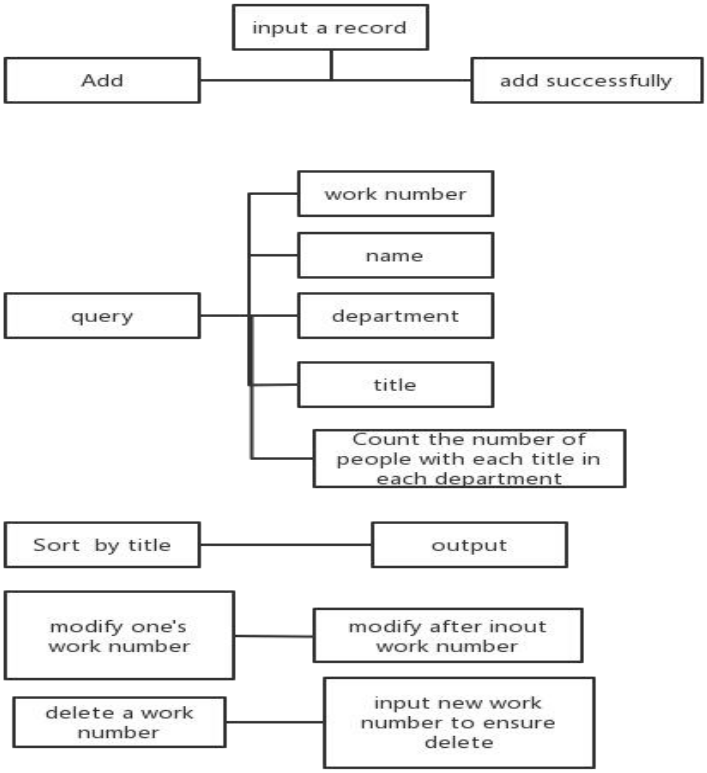
For C + + beginners, they lack database related knowledge and only use files and C + + basic knowledge to develop the system, which is very helpful to consolidate C + + basic knowledge and improve their overall ability this semester.

2.Functional Requirements Analysis

2.1 main functions of faculty management system

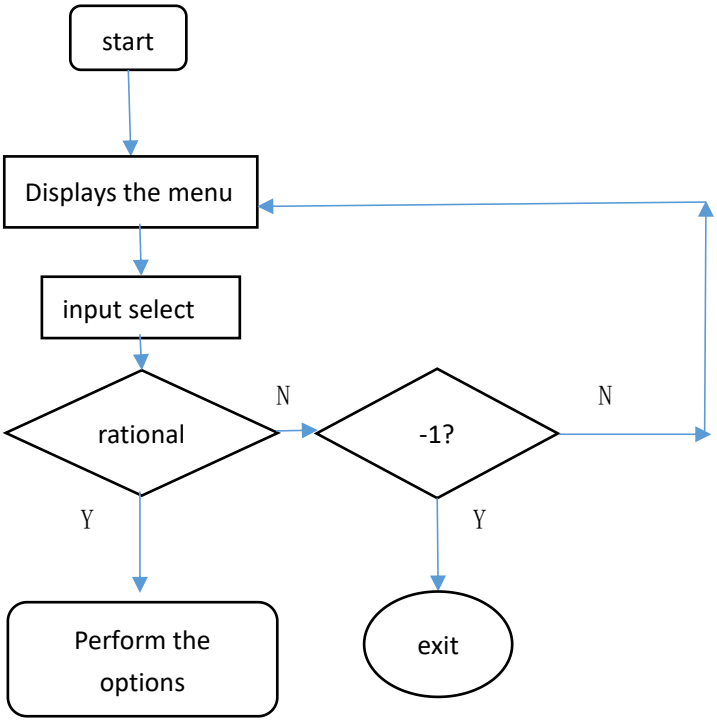
1.Design menu to realize function selection;
2.Input function: input employee information and save it to file;
3. Query function:
(1) Be able to accurately query employee information according to job number;
(2) Be able to query employee information according to name and department
(3) The Department shall make statistics of professional titles and calculate the number of people with each professional title
4. Sort the output according to the employee's professional title
5. Modify employee information according to job number
6. Delete employee information according to job number

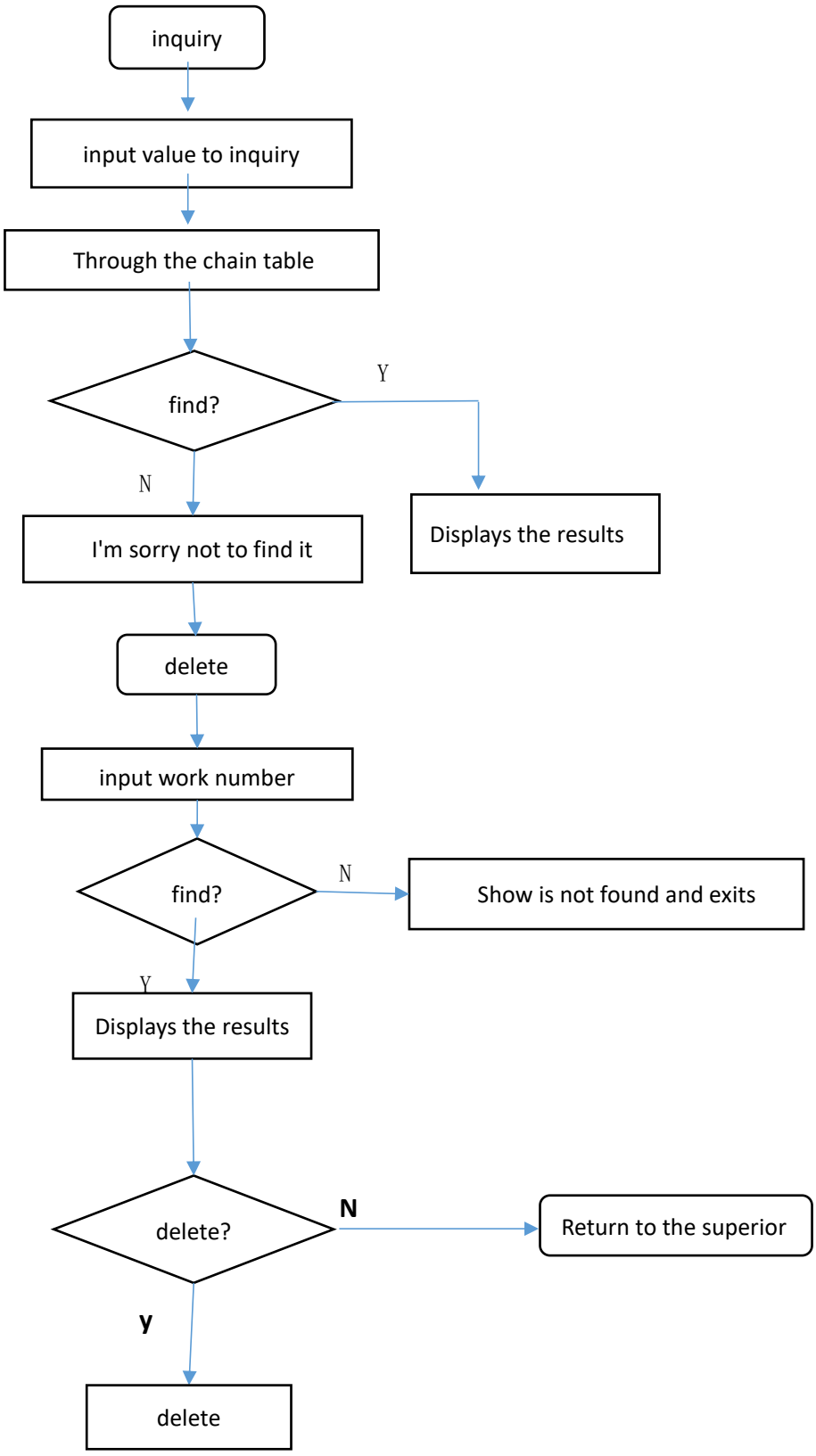
2.2 Program menu items



3 .Overall design of the system

3.1 Basic process





3.2 Use of linked list

The linked list is used to store the information of each faculty member during the program operation Considering the large amount of information, the linked list is used It is easy to add or delete

3.3 Program process

1. Read the file (read the data in the data.txt file into the linked list)
2. Start with small welcome interface(simple text effect)
3. Display menu (signal1 as the selection signal of the primary menu)

You can select the corresponding function from the first level menu. If it is illegal, the program will prompt

- **Add**

After selecting, you can enter the staff information according to the system prompts

- **Search**

Jump to the secondary menu and select the specific search method

- **Modify**

Press the prompt to enter the job number. If the job number exists, it will be modified. If it does not exist, it will return to the first level menu

- **Sort by professional title and output**

The system sorts and outputs all teaching staff by level

- **Delete**

Enter the job number. If the job number exists, delete the corresponding entry. If it does not exist, call back to the first level menu

3.4 Level 2 query menu

The next process depends on the user's choice, and the user can freely choose various functions
The program obtains input from the user, queries or adds, deletes and modifies the linked list, and then displays the processing results
All inputs are filtered to prevent program crash caused by illegal characters. Before deletion, the user will be asked to confirm again. Some display results will automatically jump to the superior menu within one to two seconds, and some will be returned after confirmation
Until the user exits the first level menu by entering - 1 in the first level menu, the content in the linked list will be updated to data after exiting the menu Txt file

4 Class and object design

The faculty management system includes two classes: Employee class and empnode class.
Employee contains all the basic information of the faculty and all the methods to change and view the basic information

```
class employee
{
public:
 employee();
 //默认构造函数
 employee(string na, bool sex, long long jobNu, long long phoneNu, string depart, string title);
 ~employee();
 //显示教职工信息
 void display();
 //返回工号
 long long getJobNumber();
 //返回科室
 string getDepartment();
 //返回姓名
 string getName();
 //返回职称
 string getTitle();
 //返回性别
 bool getSex();
 //返回电话号码
 long long getPhoneNumber();
 //返回职称等级
 int getlevelOfTitle();
 //用于修改姓名
 void setName(const string a);
 //修改性别
 void setSex(const bool a);
 //修改工号
 void setJobNumber(const long long a);
 //修改电话号码
 void setPhoneNumber(const long long  a);
 //修改所属部门
 void setDepartment(const string a);
 //修改职称
 void setTitle(const string a);
 //输入所有信息
 void setAllFromeCin();
private:
 string name;//姓名
 bool sex;//性别  1---男  0----女
 long long jobNumber;//工号
 long long phoneNumber;//电话号码
 string department;//所在部门科室
 string title;//职称
```

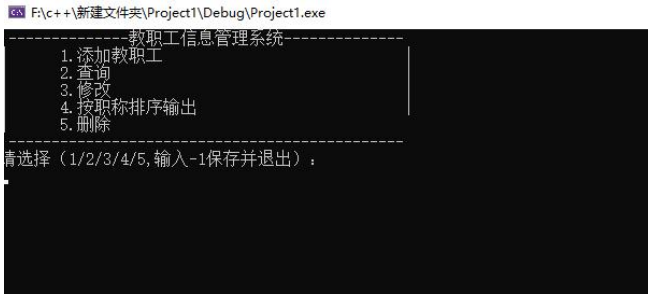
```
    int levelOftitle;//职称等级
};
```

The empnode class is a linked list. The node value is an employee object and the pointer points to the next node

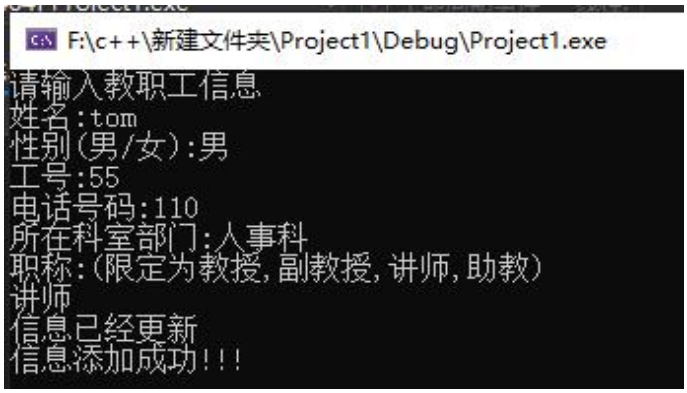
```
    class empNode
    {
    public:
     empNode();
     empNode(const employee& item, empNode* nextNode = NULL) :emp(item), next(nextNode) {}
     //find***按照相应的变量查找
     void findName(string name);
     void findJobNumber(long long jobNumber);
     void findDepartment(string department);
     void findTitle(string title);
     //统计概览 各个职称人数与各个部门人数
     void overView();
     //将链表中的值写入文件
     void write();
     //修改某个工号的个人信息
     void change(long long);
     //删除某个工号
     void erease(long long);
     //按照职称排序输出
     void paixushuchu();
     ~empNode();
     employee emp;//节点值
     empNode* next;
    private:


};
```

5. test

display welcome

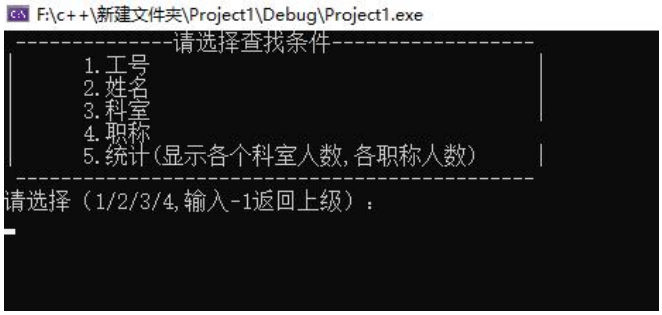


main menu



add employee

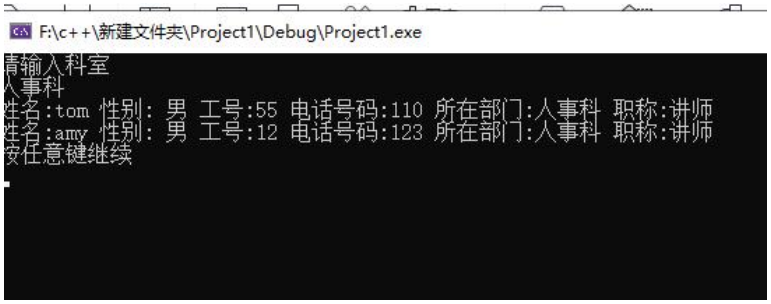
sort by work number



search interface

search by work number

search by name



search by department

statistics

delete

6 Core code and comments

6.1 To filter the illegal input

```
//非法字符输入过滤
    while (!(cin >> signal1))
    {
        cin.clear();
        cin.ignore();
        cout << "请选择（1/2/3/4/5,输入-1 保存并退出）: " << endl;

    }
```

6.2 • empNode(use of Linked list)

```
employee::employee(string na, bool se, long long jobNu, long long phoneNu, string depart, string tit)
{
string TITLE[4] = { "教授","副教授","讲师","助教" };//职称列表
name = na;
sex = se;
jobNumber = jobNu;
phoneNumber = phoneNu;
department = depart;
title = tit;
//通过职称得到职称等级
for (int i = 0; i < 4; i++)
{
    if (title == TITLE[i])
    {
        levelOftitle = i;
```

```
            break;
        }
    }

    }

    employee::~employee()
    {
    }
    //展示职称信息
    void employee::display()
    {
    cout << "姓名:" << name << " 性别: ";
    if (sex)
        cout << "男";
    else
        cout << "女";
    cout << " 工号:" << jobNumber << " 电话号码:" << phoneNumber << " 所在部门:" << department << "
职称:" << title << endl;
    }
    //以下为返回 employee 的各项值
    long long employee::getJobNumber()
    {
    return jobNumber;
    }

    string employee::getDepartment()
    {
    return department;
    }

    string employee::getName()
    {
    return name;
    }

    string employee::getTitle()
    {
    return title;
    }

    bool employee::getSex()
    {
    return sex;
```

```
}

long long employee::getPhoneNumber()
{
return phoneNumber;
}

int employee::getlevelOfTitle()
{
return levelOftitle;
}

void employee::setName(const string a)
{
name = a;
}

void employee::setSex(const bool a)
{
sex = a;
}

void employee::setJobNumber(const long long a)
{
jobNumber = a;
}

void employee::setPhoneNumber(const long long a)
{
phoneNumber = a;
}

void employee::setDepartment(const string a)
{
department = a;
}

void employee::setTitle(const string a)
{
title = a;
}
//从输入设定值
void employee::setAllFromeCin()
{
```

```
string TITLE[4] = { "教授","副教授","讲师","助教" };//职称列表
cout << "姓名:";
cin >> name;

string SEX;
//限定输入自能为男或女
do
{
    cout << "性别(男/女):";
    cin >> SEX;
    if (SEX == "男")
        sex = 1;
    else if (SEX == "女")
        sex = 0;

} while (SEX != "男" && SEX != "女");
cout << "工号:";
while (!(cin >> jobNumber))
{
    cout << "工号:";
    cin.clear();
    cin.ignore();
}
cout << "电话号码:";
while (!(cin >> phoneNumber))
{
    cout << "电话号码:";
    cin.clear();
    cin.ignore();
}
cout << "所在科室部门:";
cin >> department;
bool isTitalIllegal = false;
//职称限定
while (isTitalIllegal == false)
{
    cout << "职称:(限定为教授,副教授,讲师,助教)" << endl;;
    cin >> title;
    for (int i = 0; i < 4; i++)
    {
        if (title == TITLE[i])
        {
            isTitalIllegal = true;
            levelOftitle = i;
```

```
                break;
            }
        }
    }
    cout << "信息已经更新" << endl;
}
```

6.3 • employee(class)

```
empNode::empNode()
{
next = NULL;
}

void empNode::findName(string nam)
{
empNode* curr = this;
bool bfound = false;//标记是否找到

while (curr != NULL)
{
    if ((curr->emp.getName()) == nam)
    {
        curr->emp.display();
        bfound = true;
    }
    curr = curr->next;
}
//如果没找到就提示一下
if (!bfound)
{
    cout << "未找到所需信息" << endl;
}
delete curr;
}
void empNode::findJobNumber(long long jobNumber)
{
empNode* curr = this;
bool bfound = false;
while (curr != NULL)
{
    if (curr->emp.getJobNumber() == jobNumber)
    {
        curr->emp.display();
        bfound = true;
```

```
        }
        curr = curr->next;
    }
    if (!bfound)
    {
        cout << "未找到所需信息" << endl;
    }
    delete curr;
}
void empNode::findDepartment(string department)
{
    empNode* curr = this;
    bool bfound = false;
    while (curr != NULL)
    {
        if (curr->emp.getDepartment() == department)
        {
            curr->emp.display();
            bfound = true;
        }
        curr = curr->next;
    }
    if (!bfound)
    {
        cout << "未找到所需信息" << endl;
    }
    delete curr;
}
void empNode::findTitle(string title)
{
    empNode* curr = this;
    bool bfound = false;
    while (curr != NULL)
    {
        if (curr->emp.getTitle() == title)
        {
            curr->emp.display();
            bfound = true;
        }
        curr = curr->next;
    }
    if (!bfound)
    {
        cout << "未找到所需信息" << endl;
```

```
}
delete curr;
}
//显示统计信息
void empNode::overView()
{
/*string TITLE[4] = {"教授","副教授","讲师","助教"};//职称列表
empNode* curr = this;
int number[4];
for (int i = 0; i < 4; i++)//初始化
{
    number[i] = 0;
}//计数器初始化
//统计各个职称的人数
while (curr != NULL)
{
    int level = curr->emp.getlevelOfTitle();
    if (level != -1 && level >= 0 && level < 4)
    {
        number[level]++;
    }
    curr = curr->next;
}
delete curr;
//显示统计结果
for (int i = 0; i < 4; i++)
{
    cout << TITLE[i] << ": " << number[i] << "人" << endl;
}

map<string, int> T;
T.clear();
empNode* front = this;
while (front != NULL)
{
    string str;
    str = front->emp.getDepartment();
    if (str != "无")//过滤掉链表的末节点
        T[str]++;
    front = front->next;//指针指向下一个节点
}
map<string, int>::iterator it = T.begin();//容器指针
for (; it != T.end(); ++it)
    cout << it->first << " : " << it->second << "人" << endl;
```

```
    cout << "按任意键继续" << endl;
    cin.get();
    cin.get();*/
    }
    //写入文件
    void empNode::write()
    {
    ofstream out("data.txt");
    empNode* curr = this;
    while (curr->next != NULL)
    {
        out << curr->emp.getName() << " " << curr->emp.getSex() << " " << curr->emp.getJobNumber()
<< " " << curr->emp.getPhoneNumber() << " "
            << curr->emp.getDepartment() << " " << curr->emp.getTitle() << " " <<
curr->emp.getlevelOfTitle() << endl;
        curr = curr->next;
    }
    delete curr;
    }

    empNode::~empNode()
    {
    }

    void empNode::change(long long Jobnumber)
    {
    empNode* curr;
    curr = this;
    while (curr != NULL)
    {
        if (curr->emp.getJobNumber() == Jobnumber)
        {
            //显示原本的信息
            curr->emp.display();
            cout << "请输入" << endl;
            curr->emp.setAllFromeCin();
        }
        curr = curr->next;
    }
    delete curr;
    }

    void empNode::erease(long long target)
    {
```

```
empNode* curr = this;
empNode* pre = NULL;
bool bFound = false;//是否找到
while (curr != NULL && !bFound)
{
    if (curr->emp.getJobNumber() == target)
    {
        curr->emp.display();
        string confirm;
        do
        {
            cout << "确认删除?(Y/N)" << endl;
            cin >> confirm;
            if (confirm == "Y")
            {
                pre->next = curr->next;
                cout << "成功删除 :)" << endl;
            }
            else if (confirm == "N")
            {
                cout << "已经取消删除" << endl;
            }
        } while (confirm != "Y" && confirm != "N");
        delete curr;
        bFound = true;
    }
    else
    {
        pre = curr;
        curr = curr->next;
    }

}

if (!bFound)
{
    cout << "没有找到相应的工号" << endl;
}

}

void empNode::paixushuchu()
{
```

```
    for (int i = 0; i < 4; i++)
    {
        empNode* curr = this;
        while (curr != NULL)
        {
            if (curr->emp.getlevelOfTitle() == i)
            {
                curr->emp.display();
            }
            curr = curr->next;
        }

    }
}
```