考试报名系统

问题描述

考试报名工作给各高校报名工作带来了新的挑战，给教务管理部门增加了很大的工作量。本项目是对考试报名管理的简单模拟，用控制台选项的选择方式完成下列功能：输入考生信息；输出考生信息；查询考生信息；添加考生信息；修改考生信息；删除考生信息。

功能实现及代码分析

1、结构体设计

使用单向无环链表，将考生数据封装为一个结构体，包含考号、姓名、性别、年龄、报考类型以及指向下一节点的指针。

```
struct examinee {
    int examNo;             //考号
    string name;            //姓名
    string gender;          //性别
    float age;              //年龄
    string examType;        //报考类别
    struct examinee *next;  //指针域
};
```

2、类设计

将考试报名系统的各功能封装在类System中。

```
class System {
public:
    System();                       //Constructor
    void add(int num);              //添加元素到链表,默认到尾部添加
    Examinee *search(int stuNum);   //搜索考号为stuNum的元素
    Examinee *locate(int stuNum);   //搜索第stuNum个元素的地址
    void find(int stuNum);          //查找考号为stuNum的考生
    bool remove(int stuNum);        //删除第stuNum个位置的元素
    bool insert(int stuNum);        //在第stuNum个位置将信息插入到System
    void display();                 //对System的元素进行显示
    void modify(int stuNum);        //更改第stuNum个位置的data
    int getLength() const;          //得到System的长度
    void count();                   //统计某类别考生人数
    ~System();                      //Destructor

private:
    struct Examinee *head;
};
```

add()函数

初始化链表中添加元素到链表，保留头指针，使用后插入法，方便后续操作中结点的删除和添加。

```
void System::add(int num) {
    struct Examinee *newStu, *end;
    end = head;
    while (num != getLength()) {
        newStu = new struct Examinee;
        cin >> *newStu;
        if (newStu == NULL) {
            cerr << "存储分配错误！" << endl;
            exit(1);
        }
        end->next = newStu;
        end = newStu;
    }
    end->next = NULL;
}
```

3、建立考生系统

进入系统，根据用户初次输入建立初始考生系统链表，并支持用户多次进行各种操作（插入、删除、查找、修改、统计）的跳转，最终输入0退出系统。

```
int main() {
    System stuList;

    int num;
    cout << "首先请建立考生信息系统！\n"
         << "请输入考生人数：";
    cin >> num;
    if (num == 0) return 0;
    cout << "请依次输入考生的考号，姓名，性别，年龄及报考类别！" << endl;
    stuList.add(num);
    stuList.display();
    cout << "请选择您要进行的操作（1为插入，2为删除，3为查找，4为修改，5为统计，0为取消操作)" << endl;
    int ops = -1; //将要进行的操作
    int stuNum; //考生位置
    while (ops) {
        cout << "请选择您要进行的操作：";
        cin >> ops;
        switch (ops) {
            case 0:                                         //取消操作，退出程序
                cout << "已退出";
                break;
            case 1:                                         //插入操作
                cout << "请输入你要插入的考生位置：";
                cin >> stuNum;
                stuList.insert(stuNum);
                stuList.display();
                break;
            case 2:                                         //删除操作
                cout << "请输入要删除的考生的考号：";
                cin >> stuNum;
                stuList.remove(stuNum);
                stuList.display();
                break;
            case 3:                                         //查找操作
                cout << "请输入要查找的考生的考号：";
                cin >> stuNum;
                stuList.find(stuNum);

                break;
            case 4:                                         //修改操作
                cout << "请输入你要修改的考生的考号：";
                cin >> stuNum;
                stuList.modify(stuNum);
                stuList.display();
                break;
            case 5:                                         //统计操作
                stuList.count();
                break;

            default:
                cout << "该操作不存在，请重新输入!" << endl;
                break;
        }
    }
    return 0;
}
```

4、插入考生信息

顺序遍历定位所要插入的考生信息位置，若不能找到要求位置，进行报错并退出操作；若成功定位，则向链表添加新的结点，输入相关考生信息。locate函数作用是定位所要求考生位置，并返回指针，具体实现见后。

```
bool System::insert(int stuPos) {
    Examinee *cur = locate(stuPos - 1);
    if (cur == NULL) {                    //插入不成功
        cout << "你插入的位置不正确！" << endl;
        return false;
    }
```

```
    cout << "请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别！" << endl;
    struct Examinee *newStu = new struct Examinee;
    cin >> *newStu;
    if (newStu == NULL) {
        cerr << "存储分配错误！" << endl;
        exit(1);
    }
    newStu->next = cur->next;
    cur->next = newStu;
    return true;                            //插入成功
}
```

5、删除考生信息

根据输入的考生号，顺序遍历链表搜索考生，若搜索失败，即考生号不存在则报错并退出操作；若搜索成功，删除该考生结点。

```
bool System::remove(int stuPos) {
    Examinee *cur = search(stuPos);
    if (cur == NULL) {            //删除不成功
        cout << "你删除的考生号不存在！" << endl;
        return false;
    }
    cout << "你删除的考生信息是："; //删除成功
    cout << *cur;
    Examinee *temp = head;
    while (temp->next != cur) {
        temp = temp->next;
    }
    temp->next = cur->next;
    delete cur;
    return true;
}
```

6、查找考生信息

根据输入的考生号，顺序遍历链表，若搜索失败，则报错并退出操作；若搜索成功，输出该考生相关信息。

```
void System::find(int stuNum) {
    Examinee *cur = search(stuNum);
    if (cur == NULL) {                          //查找失败
        cout << "你查找的考生号不存在！" << endl;
    } else {                                    //查找成功
        cout << "考号\t" << "姓名\t" << "性别\t" << "年龄\t" << "报考类别\t" << endl;
        cout << *cur;
    }
}
```

7、修改考生信息

根据输入的考生号，若考生号不符合要求或者不在系统中，报错并退出操作；若搜索到考生号，根据用户输入的内容修改考生信息。

```
void System::modify(int stuNum) {
    if (stuNum <= 0) {                          //考生号不符合要求
        cout << "输入的考生号不存在！" << endl;
        return;
    }
    Examinee *cur = search(stuNum);
    if (cur == NULL) {                          //所修改考生不在系统中
        cout << "输入的考生号不存在！" << endl;
        return;
    } else {
        cout << "请依次输入要修改的考生的考号，姓名，性别，年龄及报考类别！" << endl;
        cin >> *cur;
    }
}
```

8、统计考生信息

统计信息包含2个类型，考生总人数以及某一报考类型的总人数。

```
void System::count() {
    cout << "报名系统现有考生" << getLength() << "人。" << endl;    //系统中考生总人数
    cout << "请输入你要统计的考生报考类型：";                        //相应报考类型人数
    string type;
    cin >> type;
    int iCount = 0;
    Examinee *u = head->next;
    while (u) {
        if (type == u->examType) iCount++;
        u = u->next;
    }
    cout << "报考" << type << "类型的考生共有" << iCount << "人。" << endl;
}
```

9、辅助函数

展示考生列表

顺序遍历列表，输出所有考生信息。

```
void System::display() //打印整条链表
{
    cout << "考号\t" << "姓名\t" << "性别\t" << "年龄\t" << "报考类别\t" << endl;
    Examinee *cur = head->next;
    while (cur != NULL) {
        cout << *cur;
        cur = cur->next;
    }
}
```

重载输入输出函数

为了简化输入输出过程，重载考生信息的输入输出函数。

```
istream &operator>>(istream &in, Examinee &u) {
    cin >> u.examNo >> u.name >> u.sex >> u.age >> u.examType;
    return in;
}

ostream &operator<<(ostream &out, Examinee &u) {
    cout << u.examNo << '\t'
         << u.name << '\t'
         << u.sex << '\t'
         << u.age << '\t'
         << u.examType
         << endl;
    return out;
}
```

搜索指定元素

根据考号搜索系统链表中的考生，返回指向该考生的指针，若未找到返回NULL。

```
Examinee *System::search(int stuNum) {
    Examinee *cur = head->next;
    while (cur != NULL) {
        if (cur->examNo == stuNum) break; //循链找考号为pos的结点
        else cur = cur->next;
    }
    return cur;
}
```

定位指定元素

根据输入确定考生插入的位置，若查找失败，返回NULL。

```
Examinee *System::locate(int stuNum) {
    if (stuNum < 0) return NULL;
    Examinee *cur = head;
    int count = 0;
    while (cur != NULL && count < stuNum) {
        cur = cur->next;
        count++;
    }
    return cur;
}
```

获得链表长度

顺序遍历获得链表长度。

```
int System::getLength() const {
    Examinee *u = head->next;
    int count = 0;
    while (u) {
        count++;
        if (u->next == NULL) break;
        u = u->next;
    }
    return count;
}
```

用例演示

进入系统

```
首先请建立考生信息系统!
请输入考生人数: 3
请依次输入考生的考号，姓名，性别，年龄及报考类别!
1 stu1 女 20 软件设计师
2 stu2 男 21 软件开发师
3 stu3 男 20 软件设计师
考号   姓名   性别   年龄   报考类别
1    stu1     女    20   软件设计师
2    stu2     男    21   软件开发师
3    stu3     男    20   软件设计师
请选择您要进行的操作（1为插入，2为删除，3为查找，4为修改，5为统计，0为取消操作)
请选择您要进行的操作:
```

插入考生信息

```
请选择您要进行的操作: 1
请输入你要插入的考生位置: 4
请依次输入要插入的考生的考号，姓名，性别，年龄及报考类别!
4 stu4 女 21 软件测试师
考号   姓名   性别   年龄   报考类别
1    stu1     女    20   软件设计师
2    stu2     男    21   软件开发师
3    stu3     男    20   软件设计师
4    stu4     女    21   软件测试师
```

删除考生信息

```
请选择您要进行的操作: 2
请输入要删除的考生的考号: 2
你删除的考生信息是: 2 stu2     男    21   软件开发师
考号   姓名   性别   年龄   报考类别
1    stu1     女    20   软件设计师
3    stu3     男    20   软件设计师
4    stu4     女    21   软件测试师
```

查找考生信息

```
请选择您要进行的操作：3
请输入要查找的考生的考号：3
考号   姓名   性别   年龄   报考类别
3   stu3    男   20  软件设计师
```

修改考试信息

```
请选择您要进行的操作：4
请输入你要修改的考生的考号：1
请依次输入要修改的考生的考号，姓名，性别，年龄及报考类别！
1 stu1 男 20 产品经理
考号   姓名   性别   年龄   报考类别
1   stu1    男   20  产品经理
3   stu3    男   20  软件设计师
4   stu4    女   21  软件测试师
```

统计考生信息

```
请选择您要进行的操作：5
报名系统现有考生3人。
请输入你要统计的考生报考类型：软件设计师
报考软件设计师类型的考生共有1人。
```