考试报名系统

1651573 刘客

功能分析

- 考试报名系统是一个学校不可缺少的部分，它对于学校的管理者和学生来说都至关重要，所以一个良好的考试报名系统应该能够为用户提供充足的信息和功能。一个基本的考试报名系统应该具备**输入**、**输出**、**插入**、**删除**、**修改**、**退出**的功能。

编译说明

- 在windows平台下的.exe文件
- 在Linux平台下的out文件

一 . 设计

1. 数据结构设计

    考试报名系统内含大量信息，一个考生的信息可以视作一个单元，其中需要对大量的单元进行插入,删除,修改等操作。众所周知，链表的特性则在于它的插入、删除以及修改操作非常简便，时间复杂度为o(1)，而数组虽然其查找功能非常简便，但对于一个数组进行插入、删除以及修改的话会耗费大量的资源。而考试报名系统信息量巨大，因此从性能方面考虑，应该使用链表进行设计。本课程项目设计即是使用了studentNode(存储学生信息，是一个节点类)和studentList(带有头指针的链表)来实现。

2. 类的设计

- ### studentNode 类
    - 成员变量

成员名称	属性	类型	描述
testNumber	private	int	学号
stuName	private	string	姓名
stuSex	private	string	性别
stuAge	private	int	年龄
stuApplyType	private	string	报考类型
stuNext	private	studentNode*	指向下一个学生节点的指针

- 成员函数

函数名称	返回值类型	描述
studentNode	无	构造函数
getNext	studentNode*	获得下一个学生节点
setNext	void	设置下一个学生节点
getName	string	获得当前学生的姓名
getNum	int	获得当前学生的学号
setNum	void	设置当前学生的学号
setName	void	设置当前学生的姓名
getSex	string	获得当前学生的性别
setSex	void	设置当前学生的性别
getAge	int	获得当前学生的年龄
setAge	void	设置当前学生的年龄
getType	string	获得当前学生的报考类型
setType	void	设置当前学生的报考类型

StudentList(链表类)

- 成员变量

成员名称	属性	类型	描述
first	private	studentNode*	头结点

- 成员函数

函数名称	返回值类型	描述
studentList	无	构造函数
~studentList	无	析构函数
insertStu	bool	插入学生信息
findPosByPos	studentNode*	根据在表中的位置查找节点
findPosByNum	studentNode*	根据考号查找节点
fixStuInfo	bool	修改学生信息
deleteStu	bool	删除一条学生信息
outPutStuInfo	void	输出表中全部学生信息
outPutStuInfoOneLine	void	输出表中一条特定的学生信息

二. 类的具体实现

- 构造函数

> 重写构造函数,当开辟空间失败时,输出错误

```
studentList() {
    first = new studentNode(0, "", "", 0, "");
    if (first == NULL) {
        cerr << "动态分配首节点失败" << endl;
    }
}
```

- 析构函数

> 按照顺序,将生成的节点一个一个删除,当开辟首节点失败时,则不进行这些操作,以防失败。将动态生成的内存清除,防止内存泄漏

```
~studentList() {
    if (first != NULL) {
        studentNode *temp = first->getNext();
        delete first;
        while (temp != NULL) {
            studentNode* tempNode = temp->getNext();
            delete temp;
            temp = tempNode;
        }
    }
}
```

- 搜索函数
  - 按位置来搜索

> 当输入pos不在范围之内时,输出错误信息,并且返回NULL
> 当pos==1时,直接返回头指针的后记节点
> 当pos!=1时,可能存在找不到位置的情况,此时仍输出错误信息,并返回NULL
> 若搜索到,则直接返回节点。

```
//以在表中的位置为依据来搜索特定位置的节点
studentNode* studentList::findPosByPos(int pos) {
    //如果请求的位置索引错误
    if (pos <= 0) {
        cerr << "无效的位置索引,位置索引应为大于0的自然数" << endl;
        return NULL;
    }
    //如果请求的位置恰好为first的直接后驱节点;
    else if (pos == 1) {
        return first->getNext();
    }
    else {
        int count = 1;
        studentNode* temp = first->getNext();
        //如果不为首节点,则循环获取
        while (count != pos) {
            //如果请求的位置超出了表的范围
            if (temp == NULL) {
                cerr << "无效的位置索引,该索引指示的位置超出了信息系统的范围" << endl;
                return NULL;
            }
            else {
                temp = temp->getNext();
                count++;
            }
        }
        return temp;
    }
}
```

- 按考号来搜索

> 逻辑和按学号搜索相同
>
> **返回的值是所搜索到的学号的前一个节点,方便进行插入操作**

```
//以考号为依据来搜索特定位置的节点
studentNode* studentList::findPosByNum(int num) {
    if (num <= 0) {
        cerr << "无效的考号,考号应为大于0的自然数" << endl;
        return NULL;
    }
    auto node = first->getNext();
    //用来存储节点
    auto temp = node;
    if (node == NULL) {
        cerr << "此考号不存在!" << endl;
```

```
        return NULL;
    }
    //如果第一个学生节点的考号符合要求,则返回头指针
    if (node->getNum() == num) {
        return first;
    }
    while (node->getNum() != num) {
        //temp用来存储node的前一个节点
        temp = node;
        node = node->getNext();
        if (node == NULL) {
            cerr << "此考号不存在!" << endl;
            return NULL;
        }
    }
    return temp;
}
```

- 插入函数

> 由于课程项目提供的demo界面中，插入是按位置插入的(因此特意写了上述的按位置查找的函数)此时要获取的,应该是插入位置前的节点,因此使用findPosByPos(pos-1)

```
bool studentList::insertStu(studentNode* node, int pos) {
    if (first->getNext() == NULL) {
        first->setNext(node);
    }
    else {
        //如果要插入的位置在first节点之后
        if (pos == 1) {
            //暂存first后的节点,进行节点之间的更换
            auto tempNode = first->getNext();
            first->setNext(node);
            node->setNext(tempNode);
        }//如果不是处在首节点之后的位置,则使用findPos进行插入位置前置节点的获取
        else {
            auto beforeNode = findPosByPos(pos - 1);
            //如果找到了,则进行插入,否则将生成的节点删除,防止内存泄漏
            if (beforeNode != NULL) {
                auto tempNode = beforeNode->getNext();
                beforeNode->setNext(node);
                node->setNext(tempNode);
            }
            else {
                delete node;
                return false;
            }
        }
    }
    return true;
}
```

- 修改信息函数

> 利用findPosByNum函数进行考号查找,并修改。 这里使用的是节点替换的方法,将新信息的节点替换老信息的节点并将老信息的内存空间释放.

```
bool studentList::fixStuInfo(int num, studentNode* node) {
    //寻找当前考号的前置节点
    auto fNode = findPosByNum(num);
    //如果找到,则将node节点与考号节点进行替换,并将delete掉考号节点防止内存泄漏
    if (fNode != NULL) {
        auto tempNode = fNode->getNext()->getNext();
        auto deleteNode = fNode->getNext();
        fNode->setNext(node);
        node->setNext(tempNode);
        delete deleteNode;
        cout << "修改后的考生信息是: ";
        outPutStuInfoOneLine(node);
        return true;
    }
    return false;
}
```

- 删除信息函数

> 同样使用findPosByNum函数,获得要删除节点的前驱节点,然后对其进行删除

```
bool studentList::deleteStu(int num) {
    auto beforeNode = findPosByNum(num);
    if (beforeNode != NULL) {
        auto node = beforeNode->getNext()->getNext();
        auto deleteNode = beforeNode->getNext();
        beforeNode->setNext(node);
        cout << "你所删除的考生信息是: ";
        outPutStuInfoOneLine(deleteNode);
        delete deleteNode;
        return true;
    }
    return false;
}
```

- 输出函数

> 一个为输出全部信息 另一个为输出一条信息 简化了在主程序中的代码重复程度

```
//输出表中的学生信息
void studentList::outPutStuInfo() {
    cout << "\n";
    cout<<"考号\t"<<"姓名\t"<<"性别\t"<<"年龄\t"<<"报考类别\n";
    auto node = first->getNext();
    while (node != NULL) {
        outPutStuInfoOneLine(node);
        node = node->getNext();
    }
}
```

```
//输出表中的一条学生信息
void studentList::outPutStuInfoOneLine(studentNode* node) {
    cout << node->getNum() << '\t'
        << node->getName() << '\t'
        << node->getSex() << '\t'
        << node->getAge() << '\t'
        << node->getType() << endl;
}
```

三. 主程序实现

- 函数

函数名称	返回值类型	描述
InitStuInfo	void	初始化学生数据库函数
CreateStuInfo	studentNode*	生成一个学生节点
OutOperation	int	操作逻辑函数
insertStuInfo	void	插入一条学生信息节点
deleteStuInfo	void	删除一条学生信息节点
findStuInfo	void	寻找一条学生信息
fixStuInfo	void	修改一条学生信息
outAllInfo	void	输出学生信息表
main	int	主函数

- main函数实现

```
int main() {
    studentList * studentSystem = new studentList();

    int studentNum;
    cout << "首先请建立考生系统!\n";
    do {
        cout << "请输入考生人数:";
        cin >> studentNum;
        if (studentNum < 0) {
            cout << "考生人数不可为负!! ";
        }
    } while (studentNum < 0);
    initStuInfo(studentNum, studentSystem);
    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;

    while (outOperation(studentSystem) != 0);
```

```
    return 0;

}
```

- InitStuInfo函数实现

> 利用createStuInfo生成节点 利用链表类的插入操作,插入节点 最后再将信息表输出

```
  //初始化信息
  void initStuInfo(int num,studentList* studentSystem) {
      cout << "请以此输入考生的考号、姓名、性别、年龄以及报考类别!" << endl;

      for (auto i = 0; i < num; i++) {
          auto tempNode = createStuInfo();
          if (tempNode == NULL) {
              cout << "插入学生信息失败,没有足够的资源动态生成学生节点" << endl;
              return;
          }
          studentSystem->insertStu(tempNode,i + 1);
      }

      studentSystem->outPutStuInfo();
  }
```

- CreateStuInfo函数实现

> 将询问操作放在函数中,减少代码重复冗余程度

```
//创建一条学生信息节点
studentNode* createStuInfo() {
    int testNum;
    string name;
    string sex;
    int age;
    string type;
    cin >> testNum >> name >> sex >> age >> type;
    auto tempNode = new studentNode(testNum, name, sex, age, type);
    return tempNode;
}
```

- outOperation函数实现

> 在此其中,进行命令判断并执行相应函数

```
  //操作序列
  int outOperation(studentList* studentSystem) {
      int num;
      do {
          cout << "请选择您要进行的操作:";
          cin >> num;
          switch (num)
```

```
            {
            case 1:
                insertStuInfo(studentSystem);
                break;
            case 2:
                deleteStuInfo(studentSystem);
                break;
            case 3:
                findStuInfo(studentSystem);
                break;
            case 4:
                fixStuInfo(studentSystem);
                break;
            case 5:
                outAllInfo(studentSystem);
                break;
            case 0:
                return 0;
            default:
                cout << "您的操作码错误,请重新输入 ";
                break;
            }
        } while (num < 0 || num > 5);
        return 1;
    }
```

- insertStuInfo函数实现

```
//插入一条学生信息节点
void insertStuInfo(studentList* studentSystem) {
    int pos;
    cout << "请输入您要插入的考生的位置:";
    cin >> pos;
    cout << "请以此输入要插入的考生的考号,姓名,性别,年龄以及报考类别\n";
    auto tempNode = createStuInfo();
    if (tempNode == NULL) {
        cout << "插入学生信息失败,没有足够的资源动态生成学生节点" << endl;
        return;
    }
    if (studentSystem->insertStu(tempNode, pos)) {
        outAllInfo(studentSystem);
    }
    else {
        delete tempNode;//释放内
        cerr << "插入失败\n";
    }
}
```

- deleteStuInfo函数实现

```
//删除一条学生信息节点
void deleteStuInfo(studentList* studentSystem) {
    int num;
    cout << "请输入要删除的考生的考号:";
    cin >> num;
    if (studentSystem->deleteStu(num)) {
        outAllInfo(studentSystem);
    }
}
```

- findStuInfo函数实现

```
//寻找一条学生信息节点
void findStuInfo(studentList* studentSystem) {
    int num;
    cout << "请输入要查询的考生的考号";
    cin >> num;
    auto node = studentSystem->findPosByNum(num);
    if (node == NULL) {
        return;
    }
    cout << "考号\t" << "姓名\t" << "性别\t" << "年龄\t" << "报考类别\n";
    studentSystem->outPutStuInfoOneLine(node->getNext());
    cout << "\n";
}
```

- fixStuInfo函数实现

```
//修改一条学生信息节点
void fixStuInfo(studentList* studentSystem) {
    int num;
    cout << "请输入要修改的学生的考号:";
    cin >> num;
    cout << "请依次输入修改后的考生的考号,姓名,性别,年龄以及报考类型\n";
    auto tempNode = createStuInfo();
    if (tempNode == NULL) {
        cout << "新建学生信息失败,没有足够的资源动态生成学生节点" << endl;
        return;
    }
    if (studentSystem->fixStuInfo(num, tempNode)) {
        outAllInfo(studentSystem);
    }
    else {
        //释放内存
        delete tempNode;
    }
}
```

- outAllInfo函数实现

```
//输出学生信息表
void outAllInfo(studentList* studentSystem) {
    studentSystem->outPutStuInfo();
    cout << "\n";
}
```

四. 程序运行效果

```
首先请建立考生系统!
请输入考生人数:4
请以此输入考生的考号、姓名、性别、年龄以及报考类别!
155 小欧话 男 21 前端工程师
233 德菲 女 20 初级会计师
168 欧澧 男 22 linux内核工程师
33 韦德 男 26 中级造价师

考号     姓名     性别     年龄     报考类别
155      小欧话   男       21       前端工程师
233      德菲     女       20       初级会计师
168      欧澧     男       22       linux内核工程师
33       韦德     男       26       中级造价师
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:1
请输入您要插入的考生的位置:1
请以此输入要插入的考生的考号,姓名,性别,年龄以及报考类别
188 莫非 男 24 电气工程师

考号     姓名     性别     年龄     报考类别
188      莫非     男       24       电气工程师
155      小欧话   男       21       前端工程师
233      德菲     女       20       初级会计师
168      欧澧     男       22       linux内核工程师
33       韦德     男       26       中级造价师

请选择您要进行的操作:2
请输入要删除的考生的考号:188
你所删除的考生信息是: 188      莫非     男       24       电气工程师
```

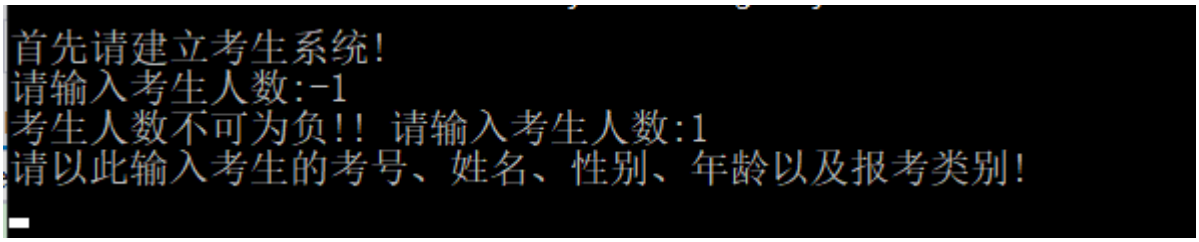
```
考号      姓名      性别      年龄      报考类别
155       小欧话    男        21        前端工程师
233       德菲      女        20        初级会计师
168       欧澧      男        22        linux内核工程师
33        韦德      男        26        中级造价师

请选择您要进行的操作:3
请输入要查询的考生的考号233
考号      姓名      性别      年龄      报考类别
233       德菲      女        20        初级会计师

请选择您要进行的操作:4
请输入要修改的学生的考号:168
请依次输入修改后的考生的考号,姓名,性别,年龄以及报考类型
168 欧鸿 女 18 初级经济学
修改后的考生信息是: 168 欧鸿      女        18        初级经济学

考号      姓名      性别      年龄      报考类别
155       小欧话    男        21        前端工程师
233       德菲      女        20        初级会计师
168       欧鸿      女        18        初级经济学
33        韦德      男        26        中级造价师

请选择您要进行的操作:5

考号      姓名      性别      年龄      报考类别
155       小欧话    男        21        前端工程师
```

```
修改后的考生信息是: 168 欧鸿      女        18        初级经济学

考号      姓名      性别      年龄      报考类别
155       小欧话    男        21        前端工程师
233       德菲      女        20        初级会计师
168       欧鸿      女        18        初级经济学
33        韦德      男        26        中级造价师

请选择您要进行的操作:5

考号      姓名      性别      年龄      报考类别
155       小欧话    男        21        前端工程师
233       德菲      女        20        初级会计师
168       欧鸿      女        18        初级经济学
33        韦德      男        26        中级造价师
```

五. 边界测试

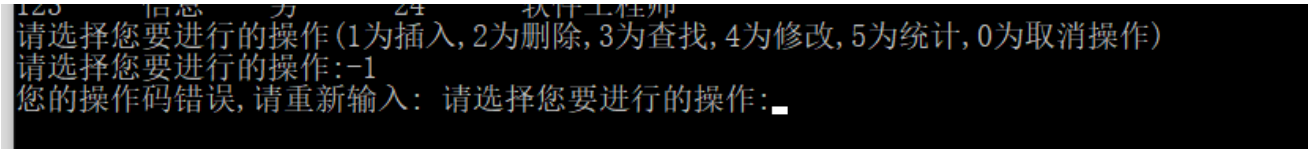
- 建立考试系统时输入考生人数错误

  > 给出警告并请求重新输入



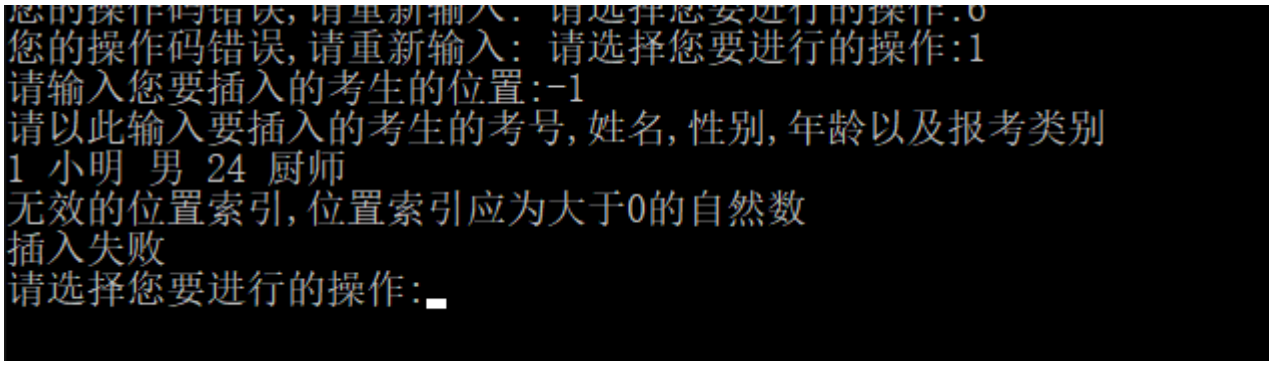
- 输入操作指令错误时

  > 给出警告并请求重新输入



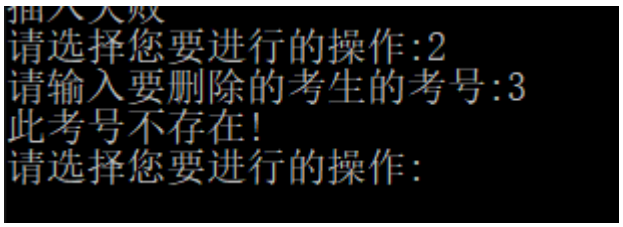
- 插入考生位置错误时

  > 给出警告并重置操作



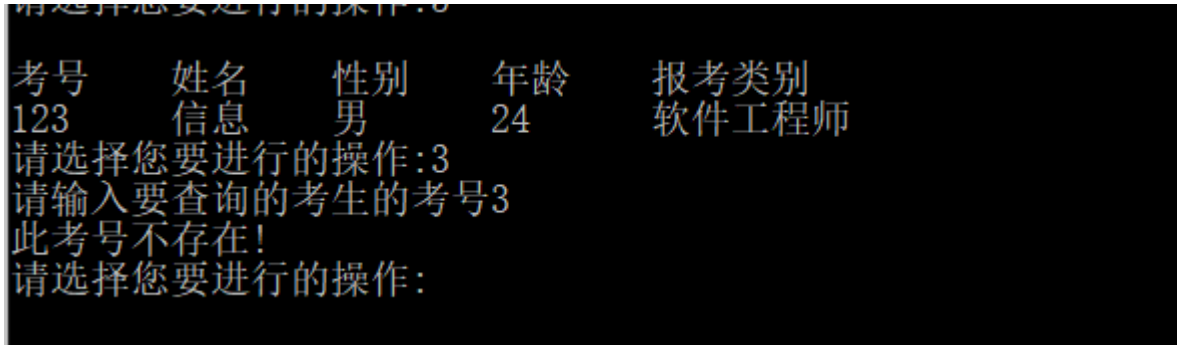
- 删除考生位置错误时

  > 给出警告并重置操作



- 查找考生不存在时

  > 给出警告并重置操作



- 修改信息考生不存在时

> 给出警告并重置操作