项目说明文档

数据结构课程设计

——考试报名系统

作 者 姓 名： 曹峰源

学 号： 1951328

指 导 教 师： 张颖

学院、 专业： 软件学院 软件工程

同济大学

Tongji University

目录

[1. 分析 3](#_Toc59901498)

[1.1 背景分析 3](#_Toc59901499)

[1.2 功能分析 3](#_Toc59901500)

[2. 设计 3](#_Toc59901501)

[2.1 数据结构设计 3](#_Toc59901502)

[2.2 类成员 3](#_Toc59901503)

[2.2.1 结点student类 3](#_Toc59901504)

[2.2.2 链表examination类 3](#_Toc59901505)

[2.3 系统设计 4](#_Toc59901506)

[3. 实现 4](#_Toc59901507)

[3.1 构造函数的实现 4](#_Toc59901508)

[3.2 插入函数的实现 5](#_Toc59901509)

[3.3 删除功能实现 7](#_Toc59901510)

[3.4 查找功能实现 9](#_Toc59901511)

[3.5 修改功能实现 10](#_Toc59901512)

[3.6 统计功能实现 12](#_Toc59901513)

[3.7 主函数功能实现 13](#_Toc59901514)

[4. 易错部分测试 16](#_Toc59901515)

[4.1 边界测试 16](#_Toc59901516)

[4.1.1 初始无输入数据 16](#_Toc59901517)

[4.1.2 删除头结点 16](#_Toc59901518)

[4.1.3 删除后链表为空 16](#_Toc59901519)

[4.2 错误输入测试 17](#_Toc59901520)

[4.2.1 考生人数错误 17](#_Toc59901521)

[4.2.2 操作码错误 17](#_Toc59901522)

[4.2.3 插入位置不存在 17](#_Toc59901523)

[4.2.4 删除、查找、修改的考号不存在 18](#_Toc59901524)

[4.2.5 要插入的数据考号与已存在的重复 18](#_Toc59901525)

1. 分析
   1. 背景分析

在现在这个计算机快速发展的时代，学校的工作也离不开计算机的帮助，考试报名系统就是其中之一，它能很好的帮助学校的管理者管理学生的考试信息，相对于手工统计更不容易出错。所以说，考试报名系统十分重要，这份报告是通过c++语言实现考试报名系统的一些操作。

* 1. 功能分析

在考试报名系统中，首先要能生成一份报名的名单且能够显示。其次它应该拥有插入、删除、修改等功能。此外，我们认为姓名在现实生活中可能相同，但是考号应当是独有的，所以设计了对于出现相同考号的情况下的处理。，代码与相关内容放到了错误输入测试的部分。最后，程序应该能正常关闭。

1. 设计
   1. 数据结构设计

要实现如上功能，可以设计链表结构进行存储并操作。对于每一位同学的参考数据，可以使用结点类来存储，考试报名系统则使用链表类来实现。链表添加头结点使得操作更加方便。

* 1. 类成员
     1. 结点student类

私有成员：

int ticket\_number=0;

string name="0";

string gender="0";

int age=0;

string job = "0";

student\* next=NULL;

* + 1. 链表examination类

私有成员：

student\* first;

student\* current;

公有成员函数：

examination()//构造函数

~examination()//析构函数

void insert();//插入函数

void display();//打印函数

void find(int target);//通过考号来查询

void change(int target);//通过考号来替换

void del(int target);//通过考号来删除

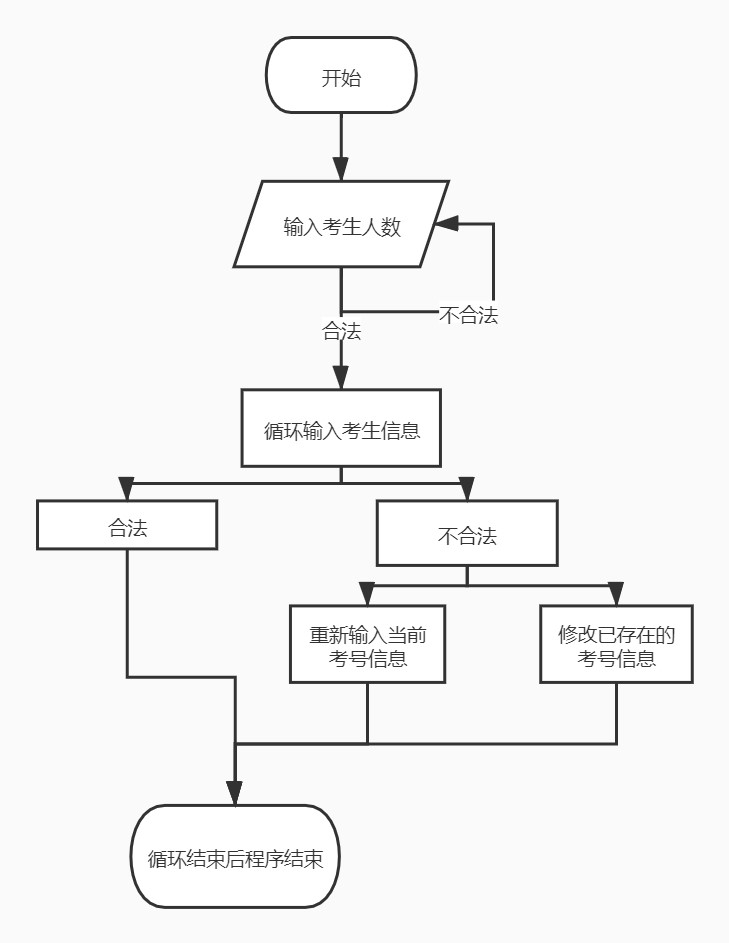
bool judge(student\* p);//用于判断插入的考号是否与已存在的重复

* 1. 系统设计

主函数中首先创建一个链表，然后根据用户输入的order\_number来选择对应操作，执行完后进行下一次循环。

1. 实现
   1. 构造函数的实现

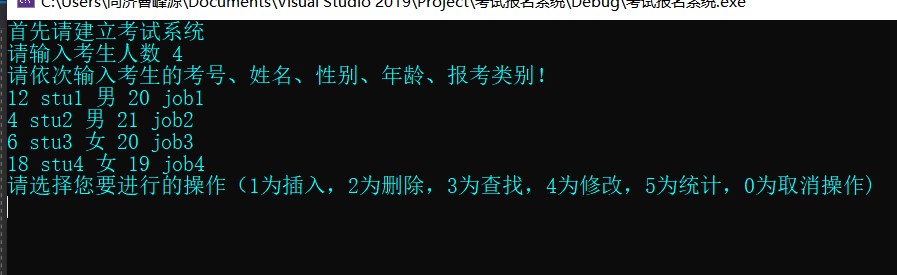
流程图：



代码实现：

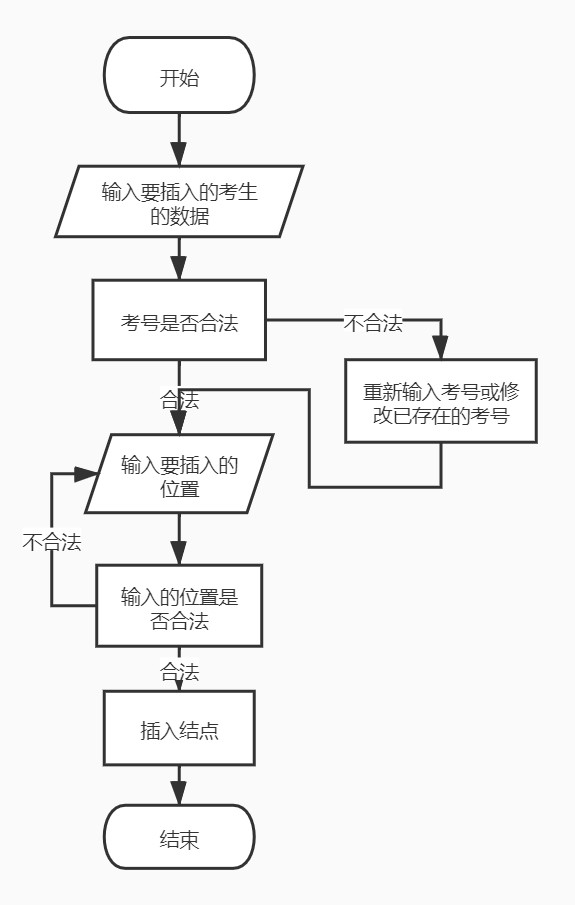
1. first = **new** student;
2. **int** number;
3. cout << "首先请建立考试系统" << endl;
4. cout << "请输入考生人数" << " ";
5. **while** (cin >> number)
6. {
7. **if** (number > 0) {
8. cout << "请依次输入考生的考号、姓名、性别、年龄、报考类别！" << endl;
9. **break**;
10. }
11. **else** {
12. cout << "请输入一个正整数！" << endl;
13. cout << "请输入考生人数" << " ";
14. }
15. }
16. **for** (**int** i=0;i<number;i++)
17. {
18. //每一个新节点开创空间
19. student\* p = **new** student;
20. cin >> p->ticket\_number;
21. cin >> p->name;
22. cin >> p->gender;
23. cin >> p->age;
24. cin >> p->job;
25. judge(p);
26. current = first;
27. **while** (current->next!=NULL)
28. {
29. current = current->next;
30. }
31. current->next = p;
32. }

结果：



* 1. 插入函数的实现

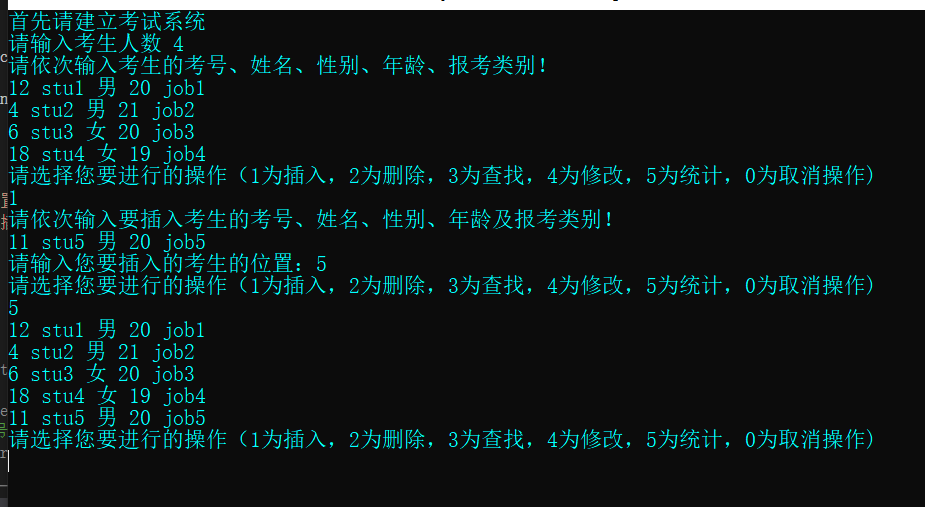
流程图：



代码实现：

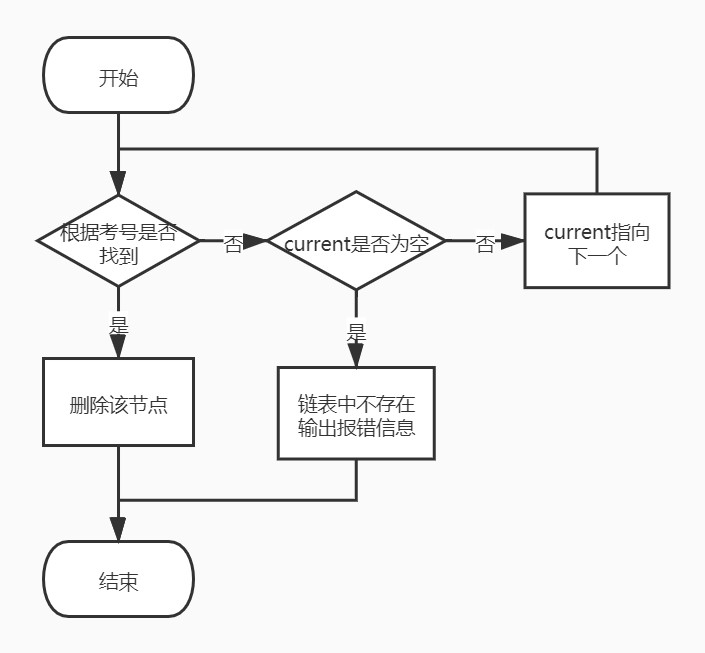
1. cout << "请依次输入要插入考生的考号、姓名、性别、年龄及报考类别！"<<endl;
2. student\* p = **new** student;
3. cin >> p->ticket\_number;
4. cin >> p->name;
5. cin >> p->gender;
6. cin >> p->age;
7. cin >> p->job;
8. judge(p);
9. **int** location;
10. cout << "请输入您要插入的考生的位置：";
11. **while** (cin >> location)
12. {
13. **int** i = 1;
14. current = first;
15. **while** (i < location && current != NULL)
16. {
17. current = current->next;
18. i++;
19. }
20. **if** (current == NULL)
21. {
22. cout << "你输入的位置不合法，请重新输入！" << endl;
23. cout << "请输入您要插入的考生的位置：";
24. }
25. **else** **break**;
26. }
27. p->next = current->next;
28. current->next = p;

结果：



* 1. 删除功能实现

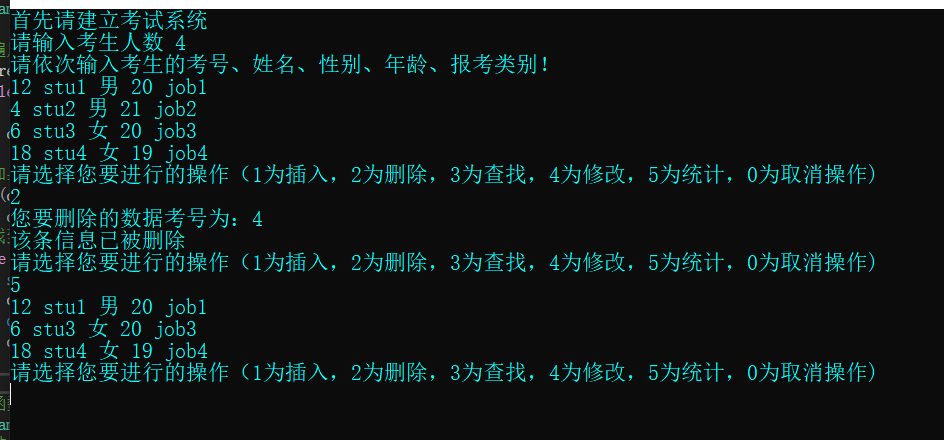
流程图：



代码实现：

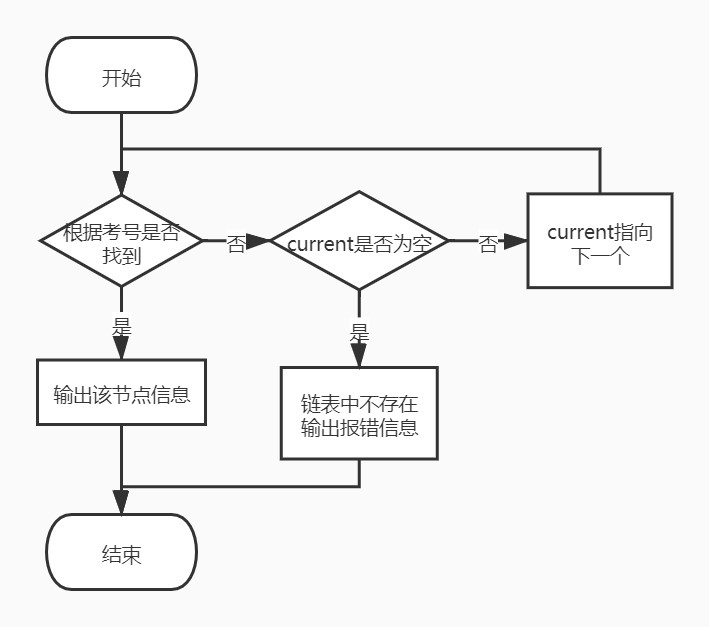
1. //遍历链表找到要删除的节点的上一个节点
2. current = first;
3. **while** (current->next != NULL && current->next->ticket\_number != target)
4. {
5. current = current->next;
6. }
7. //如果没有找到输出提示语句
8. **if** (current->next == NULL)
9. cout << "找不到匹配的数据" << endl;
10. //找到符合的元素后，将符合条件的节点删除
11. **else** {
12. student\* p = current->next;
13. current->next = current->next->next;
14. **delete** p;
15. cout << "该条信息已被删除" << endl;
16. }

结果：



* 1. 查找功能实现

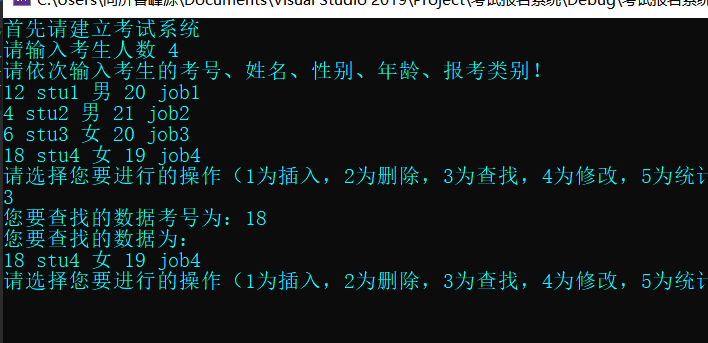
流程图：



代码实现：

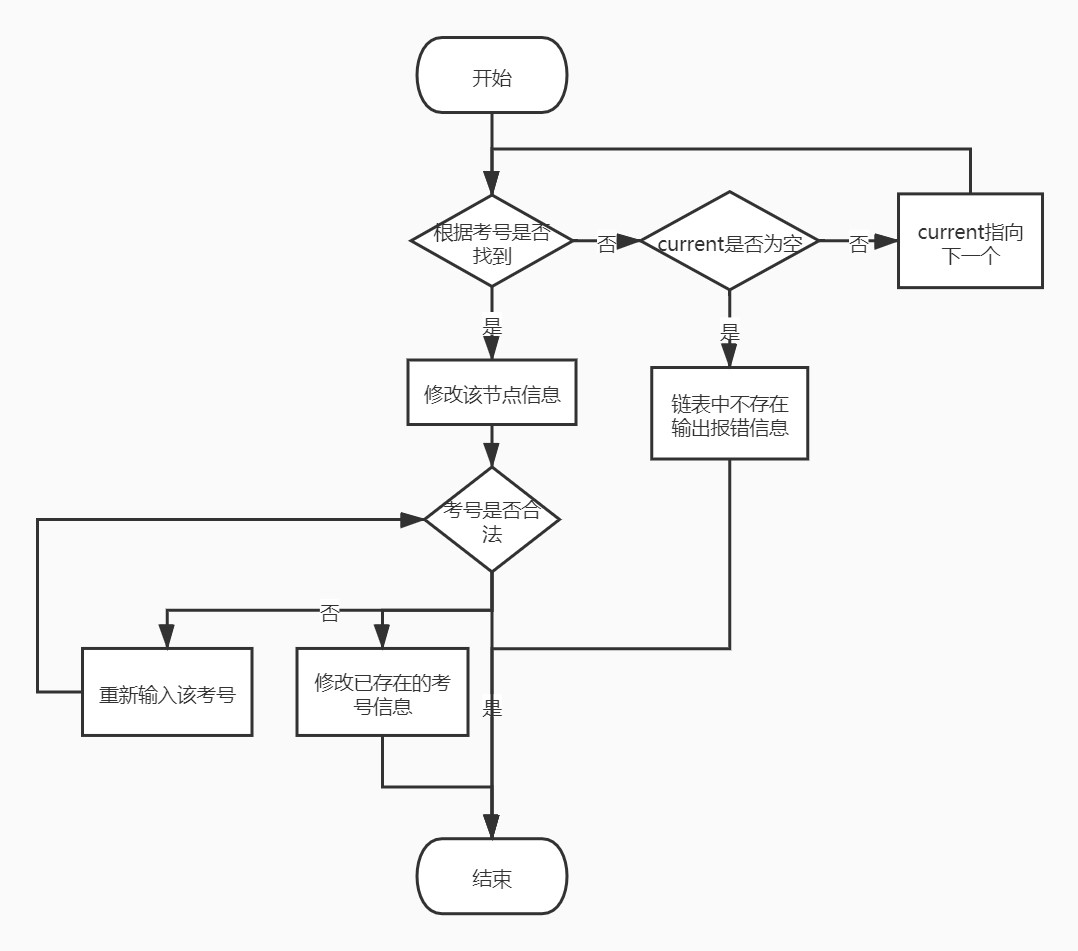
1. //对链表进行遍历，找到符合条件的节点再跳出循环，如果最后没有发现符合条件的，输出提示信息促使用户检查输入的信息无误
2. current = first->next;
3. **while** (current != NULL && current->ticket\_number != target)
4. {
5. current = current->next;
6. **if** (current == NULL)
7. {
8. cout << "没有找到该对象，请确认输入的信息无误" << endl;
9. }
10. }
11. //找到符合条件的节点后，将该节点的数据输出
12. **if** (current != NULL)
13. {
14. cout << "您要查找的数据为：" << endl;
15. cout << current->ticket\_number << " ";
16. cout << current->name << " ";
17. cout << current->gender << " ";
18. cout << current->age << " ";
19. cout << current->job << endl;
20. }

结果：



* 1. 修改功能实现

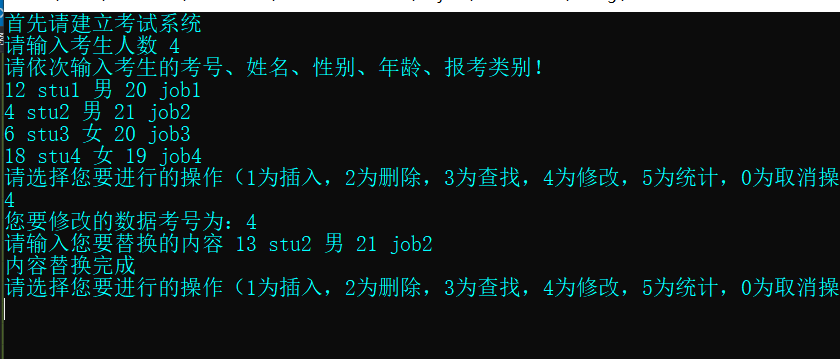
流程图：



代码实现：

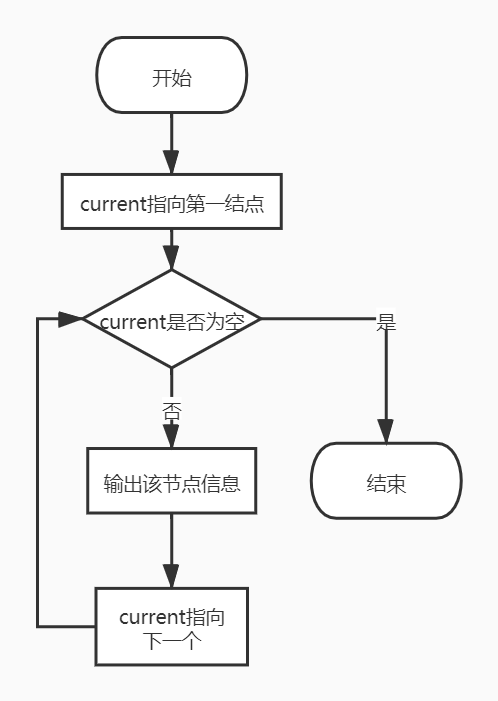
1. //遍历链表找到符合条件的节点的上一个节点
2. current = first;
3. **while** (current->next != NULL && current->next->ticket\_number != target)
4. {
5. current = current->next;
6. }
7. //如果指向NULL，说明该链表中没有该元素
8. **if** (current->next == NULL)
9. cout << "找不到匹配的数据" << endl;
10. //找到符合的节点后，将节点的数据放到新开辟的空间中，将新开辟的空间插入链表，delete掉原节点
11. **else** {
12. student\* p = **new** student;
13. student\* q = current->next;
14. p->next = q->next;
15. current->next = p;
16. **if** (q != NULL) **delete** q;
17. cout << "请输入您要替换的内容" << " ";
18. cin >> p->ticket\_number;
19. cin >> p->name;
20. cin >> p->gender;
21. cin >> p->age;
22. cin >> p->job;
23. judge(p);
24. cout << "内容替换完成" << endl;
25. }

结果：



* 1. 统计功能实现

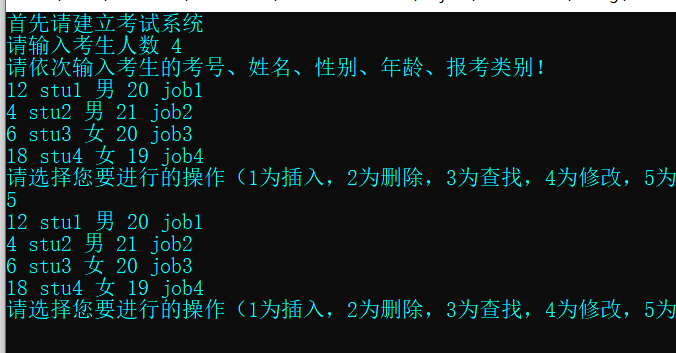
流程图：



代码实现：

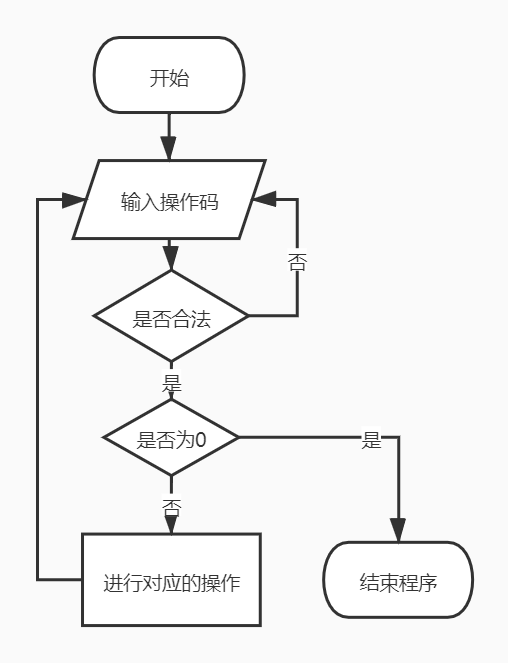
1. current = first->next;
2. **while** (current != NULL)
3. {
4. cout << current->ticket\_number << " ";
5. cout << current->name << " ";
6. cout << current->gender << " ";
7. cout << current->age << " ";
8. cout << current->job << endl;
9. current = current->next;
10. }

结果：



* 1. 主函数功能实现

流程图：

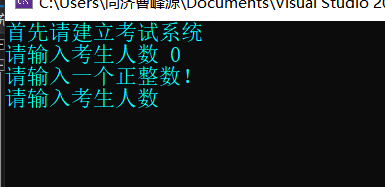


代码实现：

1. //创建链表类
2. examination tju;
3. **int** order\_number;
4. 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;
5. //while循环进行操作
6. **while** (cin >> order\_number)
7. {
8. **switch** (order\_number)
9. {
10. //输入1的时候进行插入操作
11. **case** 1: {
12. tju.insert();
13. **break**;
14. }
15. //输入2的时候进行删除操作，进一步分类通过考号或姓名进行操作
16. **case** 2:
17. {
18. cout << "您要删除的数据考号为：";
19. **int** number;
20. cin >> number;
21. tju.del(number);
22. **break**;
23. }
24. //输入3的时候进行查找操作
25. **case** 3:
26. {
27. cout << "您要查找的数据考号为：";
28. **int** number;
29. cin >> number;
30. tju.find(number);
31. **break**;
32. }
33. //输入4的时候进行修改操作
34. **case** 4:
35. {
36. cout << "您要修改的数据考号为：";
37. **int** number;
38. cin >> number;
39. tju.change(number);
40. **break**;
41. }
42. //输入5的时候将链表中数据打印出来
43. **case** 5:
44. {
45. tju.display();
46. **break**;
47. }
48. //输入0的时候取消操作
49. **case** 0:
50. {
51. cout << "取消操作" << endl;
52. exit(0);
53. system("pause");
54. **break**;
55. }
56. //输入这以外的其他数，判定为错误操作，输出提示语句报错
57. **default**:
58. {
59. cout << "您输入的操作有误，请重新选择您要进行的操作" << endl;
60. **break**;
61. }
62. }
63. 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;
64. }
65. **return** 0;
66. 易错部分测试
    1. 边界测试
       1. 初始无输入数据

希望结果：程序不崩溃

显示结果：



* + 1. 删除头结点

测试用例：

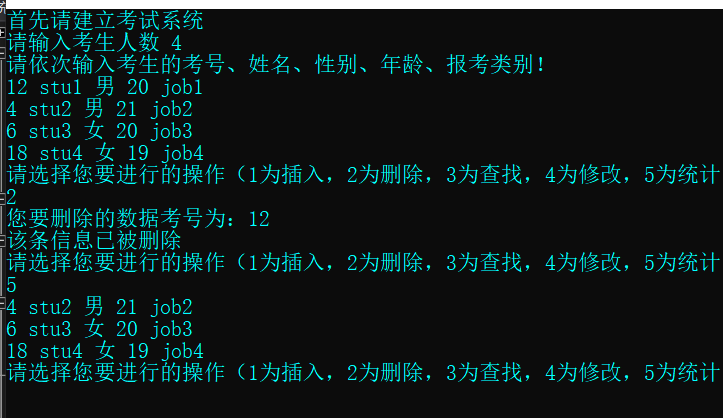
12 stu1 男 20 job1

4 stu2 男 21 job2

6 stu3 女 20 job3

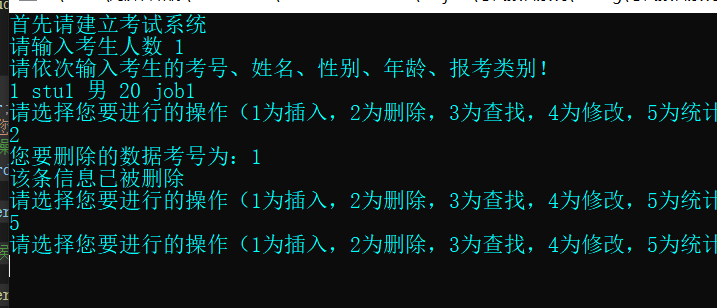
18 stu4 女 19 job4

结果：



* + 1. 删除后链表为空

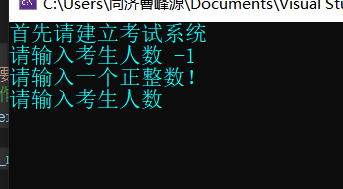
结果显示：



* 1. 错误输入测试
     1. 考生人数错误

希望结果：程序不崩溃

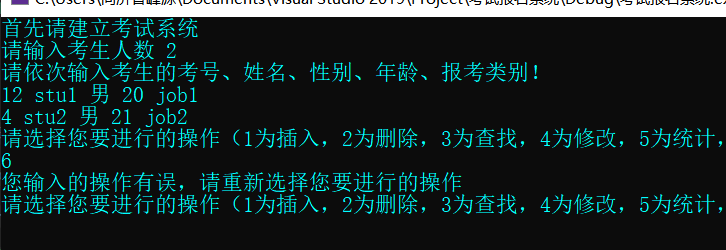
结果：



* + 1. 操作码错误

希望结果：程序不崩溃

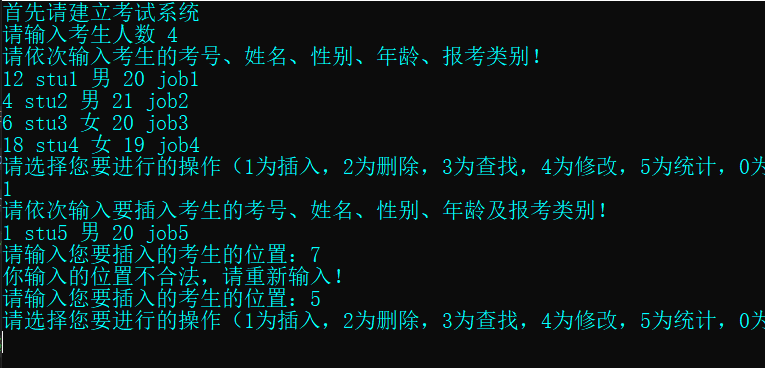
结果：



* + 1. 插入位置不存在

希望结果：程序不崩溃，能重新输入位置

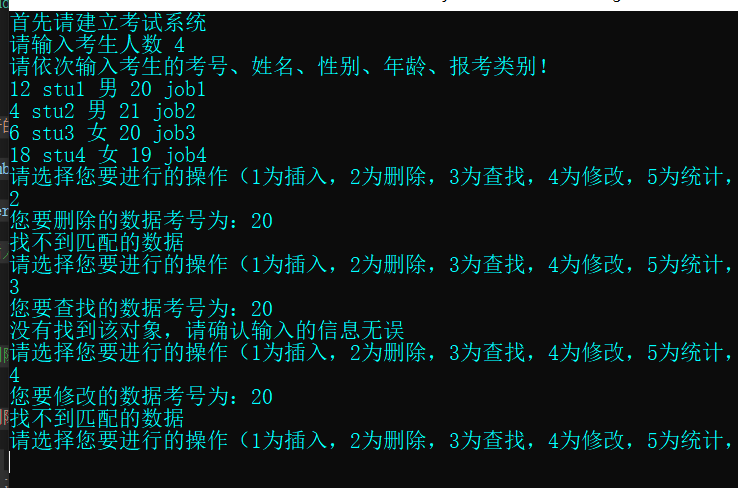
结果：



* + 1. 删除、查找、修改的考号不存在

希望结果：程序不崩溃

结果：



* + 1. 要插入的数据考号与已存在的重复

在现实生活中我们认为姓名可能会相同，但考号应当是独有的，所以在这段程序里，设计了如果出现了相同考号的情况，提供了修改已存在的考号或者重新输入考号的操作。

这种情况会在构造函数和插入函数以及修改函数的时候出现，利用judge函数来实现与原考号相同的判断。

1. **bool** examination::judge(student\* p)
2. {
3. current = first->next;
4. **while** (current != NULL)
5. {
6. **if** (current->ticket\_number == p->ticket\_number) **break**;
7. current = current->next;
8. }
9. **if** (current != NULL)
10. {
11. cout << "您输入的考号与已存在的重复" << endl;
12. cout << current->ticket\_number << " ";
13. cout << current->name << " ";
14. cout << current->gender << " ";
15. cout << current->age << " ";
16. cout << current->job << endl;
17. cout << "请选择：1.重新输入该学号 2.修改已存在的学号" << endl;
18. **int** num;
19. **while** (cin >> num)
20. {
21. **if** (num == 1)
22. {
23. cin >> p->ticket\_number;
24. **if**(judge(p)!=0)**break**;
25. }
26. **else** **if** (num == 2)
27. {
28. current = first->next;
29. **while** (current->ticket\_number != p->ticket\_number)
30. {
31. current = current->next;
32. }
33. cin >> current->ticket\_number;
34. student\* q = current;
35. **if** (judge(q)!=0&&q->ticket\_number!=p->ticket\_number) **break**;
36. }
37. **else**
38. {
39. cout << "输入操作无效" << endl;
40. cout << "请选择：1.重新输入该学号 2.修改已存在的学号";
41. }
42. }
43. **return** **true**;
44. }
45. }

结果显示：

