同济大学

TONGJI UNIVERSITY

数据结构课程设计

项目名称	考试报名系统
学　　院	计算机科学与技术学院
专　　业	软件工程
学生姓名	杨瑞晨
学　　号	2351050
指导教师	张颖
日　　期	2024 年 12 月 1 日

目 录

1 项目分析

1.1 项目背景分析

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。考试报名系统对于学校加强考试管理有极其重要的作用。随着学生数量和考试数量的日益庞大，如何管理如此庞大的数据显得极为复杂，传统的手工管理工作量大且容易出错。

随着计算机科学技术的不断成熟，使用计算机对考试报名系统进行管理，具有手工管理所无法比拟的优势。这些优点能够极大地提高学校和学生的效率，也是学校走向信息化、科学化、国际化的重要条件。因此，开发一套考试报名系统具有十分重要的意义。

1.2 项目功能分析

1.2.1 功能要求

本项目是对考试报名系统的简单模拟，用控制台选项的选择方式完成以下功能：建立考试信息系统；输入考生信息；输出考生信息；插入考生信息；删除考生信息；修改考生信息；查询考生信息；统计考生人数等等。

本项目的实质是完成对考生信息的建立，查找，插入，修改，删除等功能。其中考生信息包括准考证号，姓名，性别，年龄和报考类别等信息。项目在设计时应首先确定系统的数据结构，定义类的成员变量和成员函数；然后实现各成员函数以完成对数据操作的相应功能；最后完成主函数以验证各个成员函数的功能并得到运行结果。

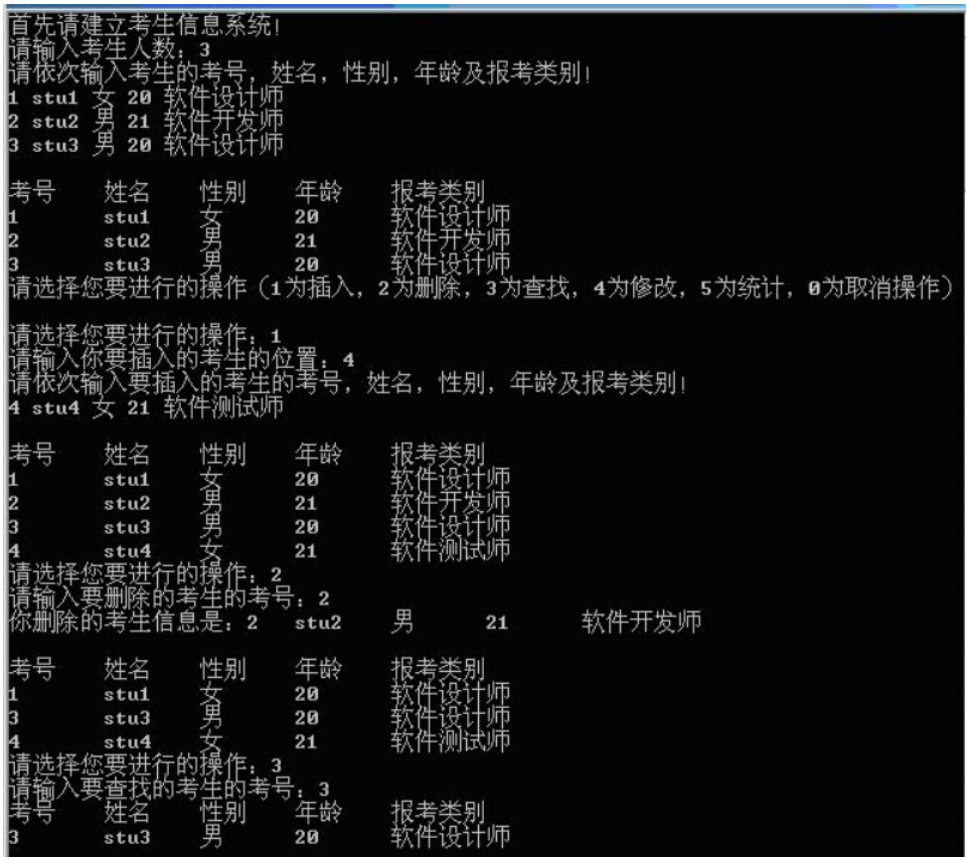
1.2.2 输入要求

每次操作都输入相对应的操作数字，同时输入正确的考生信息。(包括考号、姓名、性别、年龄和报考类型这五个基本的考生信息)

1.2.3 输出要求

输出建立后的考试信息库中的考生信息，进行一系列操作后（插入、删除、修改）的考试信息库中的考生信息，查询某一学号所对应的考生的基本信息，统计考生人数和男生女生分别人数的考生信息等等。

1.2.4 项目实例


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：1
请输入你要插入的考生的位置：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
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
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

2 项目设计

2.1 数据结构设计

该考试报名系统要求快速地完成考生信息的插入、删除、修改等等操作，因此要选对合适的数据结构来进行编写程序。本程序采用了带头结点的单链表作为最基本的数据结构。

带头结点的单链表具有以下一系列优点：

（1）链表是一个动态的数据结构，可以在运行时根据用户的需要来分配内存从而实现链表长度的增加或者缩短，无需初始链表长度；

（2）链表结点的插入、删除、修改等等操作比较方便。在进行这一系列操作的时候，我们只需通过相应函数找到要操作链表节点的地址即可，然后对其进行适当的操作，无需移动其他元素，这样也使操作的时间复杂度减小为 $O(n)$；

（3）链表可以在运行的时候可以根据用户的需要自动地增加和缩短长度，可以减少内存的浪费。

2.2 类设计

经典的链表一般包括两个抽象数据类型 (ADT) ——链表结点类 (ListNode) 与链表类 (LinkList)，而两个类之间的耦合关系可以采用嵌套、继承等多种关系。本程序也构造了这两个基本的链表类来进行操作。

2.2.1 Student 类

每一个链表结点储存着一名考生的信息和下一名考生信息的地址。

```
class Student
{
public:
    char id[ID_LEN];              // 学号
    char name[NAME_LEN];          // 姓名
    char gender[GENDER_LEN];      // 性别
    int age;                      // 年龄
    char examType[EXAMTYPE_LEN]; // 报考类别
    Student *next;                // 指向下一个学生的指针

    // 构造函数
    Student(const char id[], const char name[], const char gender[], int age, const char
    ↪  examType[])
    {
```

```
        strcpy(this->id, id);
        strcpy(this->name, name);
        strcpy(this->gender, gender);
        this->age = age;
        strcpy(this->examType, examType);
        this->next = nullptr;
    }

    // 析构函数
    ~Student() {}
};
```

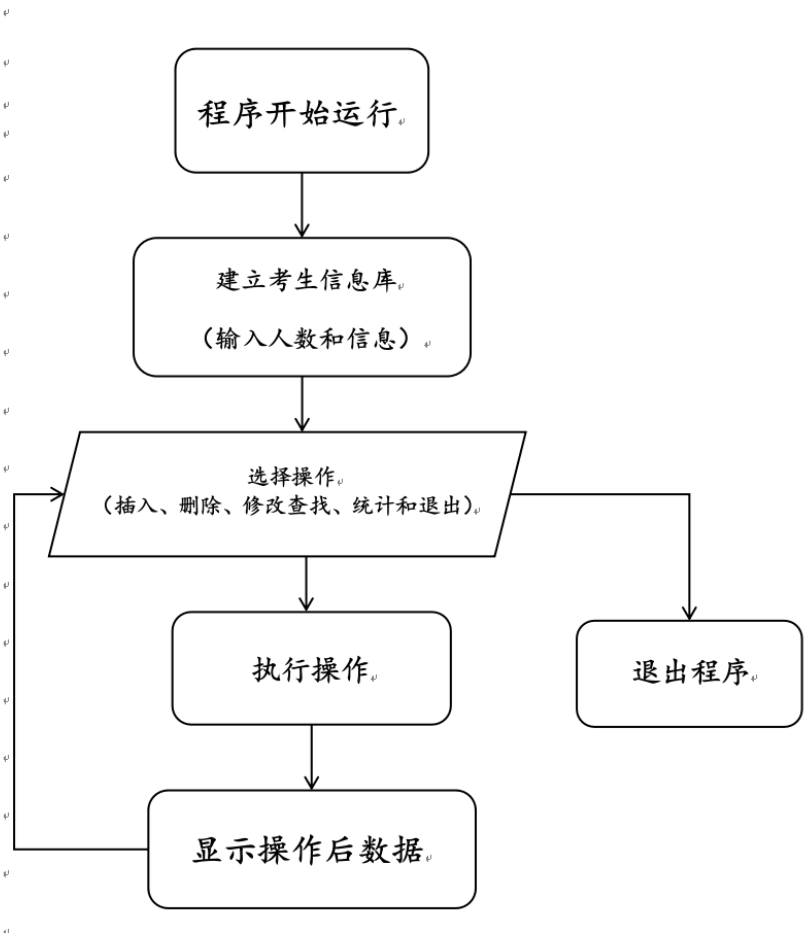
2.2.2 Student Manager 类

用于管理考生数据。它包括用于输入、输出、查询、添加、修改和删除考生信息的方法，允许用户有效地存储和操作考生数据。链表结构的灵活性使得数据的增加和删除操作更加高效通过这个链表类，可以轻松维护和操作大量考生信息。

```
class StudentManager
{
private:
    Student *head; // 链表头指针
public:
    StudentManager() : head(nullptr) {}// 构造函数初始化链表为空
    ~StudentManager()// 析构函数释放链表内存
    {
        while (head)
        {
            Student *temp = head;
            head = head->next;
            delete temp;
        }
    }
    bool checkIDExist(const char id[]) const; // 检查考号是否存在
    void addStudent();// 添加学生
    void insertStudent();// 插入学生
    void displayStudents() const;// 显示所有学生信息
    void findStudent() const;// 查找学生
```

```
    void deleteStudent();// 删除学生
    void updateStudent();// 更新学生信息
    void analyseStudents() const;// 统计功能
};
```

2.2.3 系统流程设计



3 项目实现

3.1 基本功能的实现

3.1.1 输出考生信息

```
// 显示所有学生信息
void displayStudents() const
    {
        if (!head)
        {
            cout << " 暂无考生信息" << endl;
            return;
        }
        cout << endl
             << " 考号     姓名     性别     年龄     报考类别" << endl;
        Student *temp = head;
        while (temp)
        {
            printf("%-8s%-8s%-8s%-8d%-8s\n", temp->id, temp->name, temp->gender,
            ↪  temp->age, temp->examType);
            temp = temp->next;
        }
    }
```

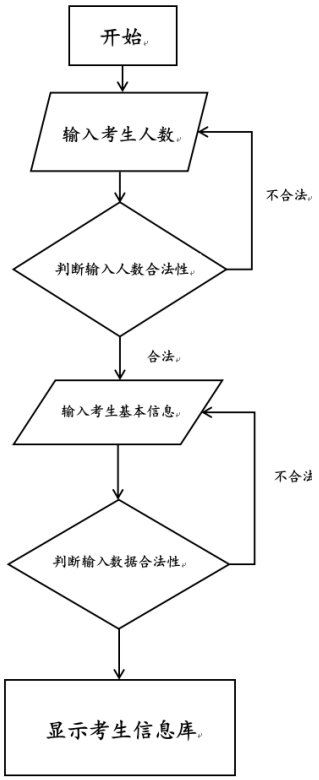
3.1.2 查找考号

根据输入的考号查找考生信息，如果考号存在，返回 true，否则返回 false。

```
bool checkIDExist(const char id[]) const
{
    Student *temp = head;
    while (temp)
    {
        if (strcmp(temp->id, id) == 0) { return true; }
        temp = temp->next;
    }
    return false;
}
```

3.2 建立考生信息库的实现

3.2.1 建立考生信息库的流程



3.2.2 建立考生信息库的核心代码

在创建考生信息库的时候有着健全的健壮性检验。如果输入的新的考生信息正确，则会将其插入到链表的末尾，从而完成考试报名系统的构建。

```
// 添加学生
void addStudent()
{
    char id[ID_LEN];
    char name[NAME_LEN];
    char gender[GENDER_LEN];
    int age;
    char examType[EXAMTYPE_LEN];

    cin >> id >> name >> gender >> age >> examType;

```

```
        Student *newStudent = new Student(id, name, gender, age, examType);

        if (!head) { head = newStudent; }
        else
        {
            Student *temp = head;
            while (temp->next)
            {
                temp = temp->next;
            }
            temp->next = newStudent;
        }
    }
```

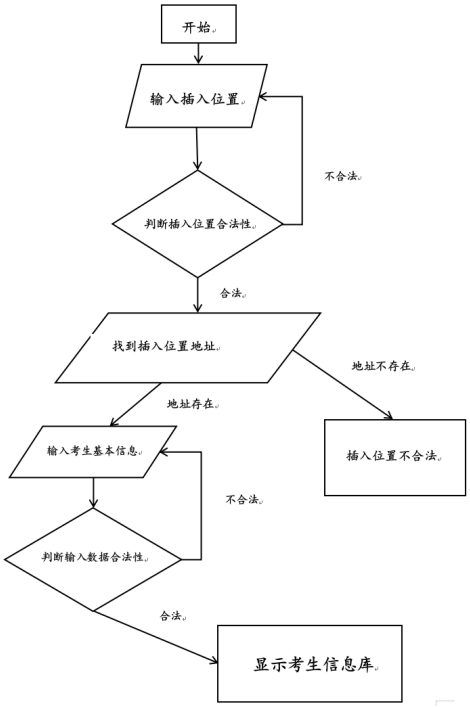
3.2.3 建立考生信息库的实例

```
首先请建立考生信息系统!
请输入考生人数: 3
请输入考号、姓名、性别、年龄、报考类别!
2351050 stu1 F 20 Designer
2350947 stu2 M 21 Developer
2252552 stu3 M 20 Developer

考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
```

3.3 插入考生信息功能的实现

3.3.1 插入考生信息功能的流程



3.3.2 插入考生信息功能的核心代码

插入功能允许用户将新的考生信息添加到链表中。此功能首先要求用户输入考生的位置，然后根据输入的位置将新节点插入到链表中。

```
// 插入学生
void insertStudent()
{
    char id[ID_LEN];
    char name[NAME_LEN];
    char gender[GENDER_LEN];
    int age;
    char examType[EXAMTYPE_LEN];

    int position;
    cout << " 请输入你要插入的考生的位置: ";
    while (true)
    {
```

```
            cin >> position;
            if (cin.fail() || position <= 0)
            {
                cout << " 无效操作，请重新输入" << endl;
                cin.clear();
                cin.ignore(1024, '\n');
            }
            else break;
        }

        cout << " 请依次输入要插入考生的考号、姓名、性别、年龄、报考类别！" << endl;
        do
        {
            cin >> id >> name >> gender >> age >> examType;
            if (checkIDExist(id))
            {
                cout << " 考号已存在，请重新输入" << endl;
            }
        } while (checkIDExist(id));

        Student *newStudent = new Student(id, name, gender, age, examType);

        // 如果链表为空，直接将新学生作为头节点
        if (!head)
        {
            head = newStudent;
            return;
        }
        // 如果插入位置为 1，则在链表头部插入
        if (position == 1)
        {
            newStudent->next = head;
            head = newStudent;
            return;
        }
        // 插入到链表的中间或尾部
        Student *temp = head;
        int currentPos = 1;
        // 找到指定位置的前一个节点
```

```
53          while (temp->next && currentPos < position - 1)
54          {
55              temp = temp->next;
56              currentPos++;
57          }
58          // 将新学生插入到链表
59          newStudent->next = temp->next;
60          temp->next = newStudent;
61      }
```

3.3.3 插入考生信息功能的实例

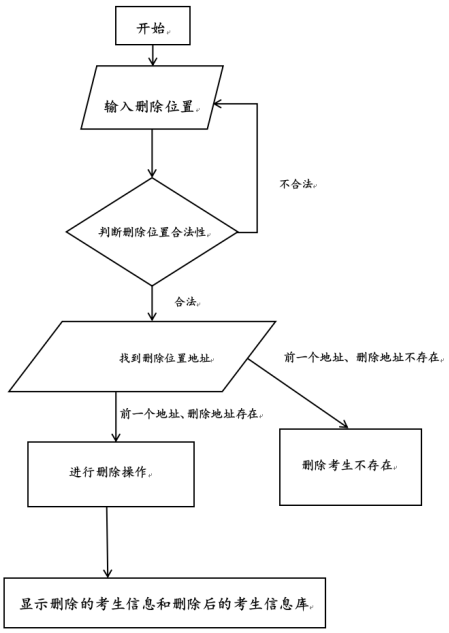
```
考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
请选择您要进行的操作（1为插入，2为删除，3为查找，4为修改，5为统计，0为取消操作，Q/q为退出程序）：
请选择您要进行的操作：1
请输入你要插入的考生的位置：2
请依次输入要插入考生的考号、姓名、性别、年龄、报考类别!
1954444 stu4 F 21 Tester

考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
请选择您要进行的操作：1
请输入你要插入的考生的位置：1
请依次输入要插入考生的考号、姓名、性别、年龄、报考类别!
2337088 stu5 M 18 Designer

考号    姓名    性别    年龄    报考类别
2337088 stu5    M       18      Designer
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
```

3.4 删除考生信息功能的实现

3.4.1 删除考生信息功能的流程



3.4.2 删除考生信息功能的核心代码

删除功能允许用户从链表中移除指定的考生信息。这一操作首先要求用户输入要删除的考生的考号。然后，程序遍历链表，寻找与输入的考号匹配的节点。如果找到匹配的考号，该节点将被删除。如果输入的考号不存在于链表中，系统将提示用户该考号未找到。

```
1   // 删除学生
2   void deleteStudent()
3   {
4       char id[ID_LEN];
5       cout << " 请输入要删除的考生的考号: ";
6       cin >> id;
7
8       Student *temp = head;
9       Student *prev = nullptr;
10
11      while (temp)
12      {
13          if (strcmp(temp->id, id) == 0)
```

```
            {
                cout << " 你删除的考生信息是: ";
                printf("%-8s%-8s%-8s%-8d%-8s\n", temp->id, temp->name, temp->gender,
                ↪  temp->age, temp->examType);
                if (prev) { prev->next = temp->next; }
                else { head = temp->next; }
                delete temp;
                return;
            }
            prev = temp;
            temp = temp->next;
        }
        cout << " 未找到该学生" << endl;
    }
```

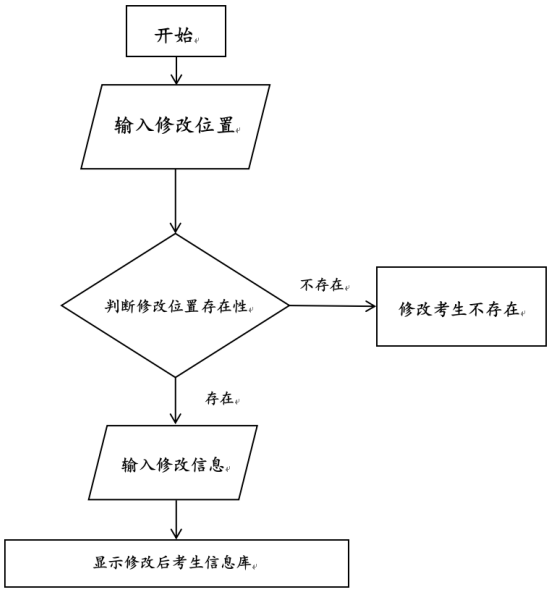
3.4.3 删除考生信息功能的实例

```
考号     姓名    性别    年龄    报考类别
2337088 stu5    M       18      Designer
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
请选择您要进行的操作: 2
请输入要删除的考生的考号: 2350947
你删除的考生信息是: 2350947 stu2    M       21      Developer

考号     姓名    性别    年龄    报考类别
2337088 stu5    M       18      Designer
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2252552 stu3    M       20      Developer
```

3.5 修改考生信息功能的实现

3.5.1 修改考生信息功能的流程



3.5.2 修改考生信息功能的核心代码

修改功能允许用户修改链表中指定考生的信息。首先要求用户输入要修改的考生的考号。然后，程序遍历链表，寻找与输入的考号匹配的节点。如果找到匹配的考号，程序将提示用户输入新的考生信息。如果输入的考号不存在于链表中，系统将提示用户该考号未找到。

```
// 更新学生信息
void updateStudent()
{
    char id[ID_LEN];
    cout << " 输入要修改的考生的考号: ";
    cin >> id;

    Student *temp = head;
    while (temp)
    {
        if (strcmp(temp->id, id) == 0)
        {
            cout << " 输入新姓名: ";
            cin >> temp->name;
```

```
15                  cout << " 输入新性别: ";
16                  cin >> temp->gender;
17                  cout << " 输入新年龄: ";
18                  cin >> temp->age;
19                  cout << " 输入新报考类别: ";
20                  cin >> temp->examType;
21                  cout << " 学生信息已更新" << endl;
22                  return;
23              }
24              temp = temp->next;
25          }
26          cout << " 未找到该学生" << endl;
27      }
```

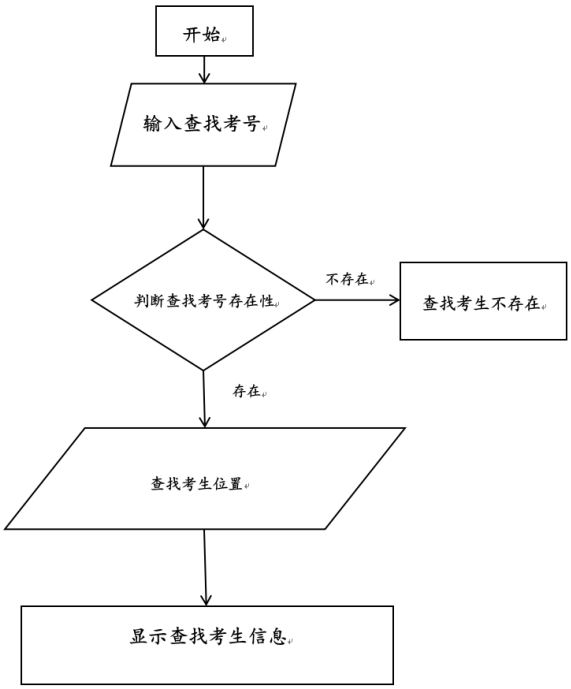
3.5.3 修改考生信息功能的实例

```
考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu1     F        20       Designer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
请选择您要进行的操作: 4
输入要修改的考生的考号: 2351050
输入新姓名: stu8
输入新性别: M
输入新年龄: 22
输入新报考类别: Programmer
学生信息已更新

考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu8     M        22       Programmer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
```

3.6 查询考生信息功能的实现

3.6.1 查询考生信息功能的流程



3.6.2 查询考生信息功能的核心代码

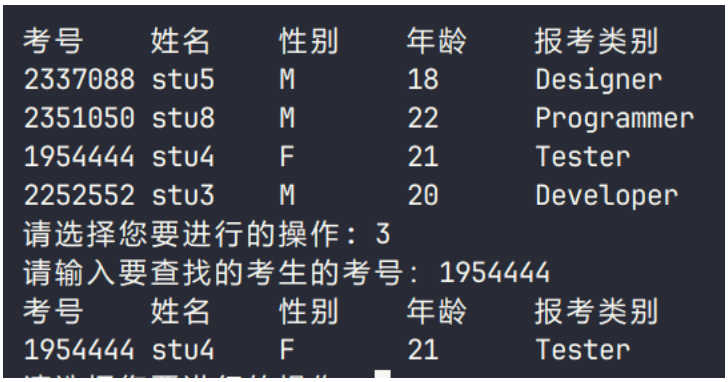
查询功能允许用户根据考号查询链表中的考生信息。首先要求用户输入要查询的考生的考号。然后，程序遍历链表，寻找与输入的考号匹配的节点。如果找到匹配的考号，程序将显示该考生的信息。如果输入的考号不存在于链表中，系统将提示用户该考号未找到。

```
// 查找学生
void findStudent() const
{
    char id[ID_LEN];
    cout << " 请输入要查找的考生的考号: ";
    cin >> id;

    Student *temp = head;
    while (temp)
    {
        if (strcmp(temp->id, id) == 0)
        {
```

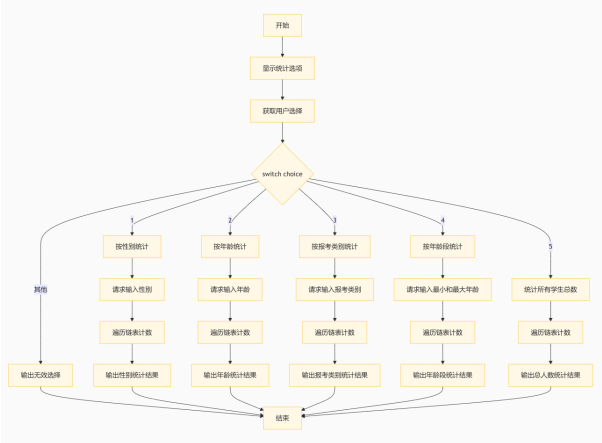
```
13                  cout << " 考号      姓名      性别      年龄      报考类别" << endl;
14                  printf("%-8s%-8s%-8s%-8d%-8s\n", temp->id, temp->name, temp->gender,
                    ↪  temp->age, temp->examType);
15                  return;
16              }
17              temp = temp->next;
18          }
19          cout << " 未找到该学生" << endl;
20      }
```

3.6.3 查询考生信息功能的实例



3.7 统计考生信息功能的实现

3.7.1 统计考生信息功能的流程



3.7.2 统计考生信息功能的核心代码

```
// 统计功能
void analyseStudents() const
{
    int choice;
    cout << " 请选择统计选项: \n1. 按性别统计\n2. 按年龄统计\n3. 按报考类别统计\n4. 按年龄段统计\n5. 统计所有学生总数\n";
    cin >> choice;

    switch (choice)
    {
    case 1:
    {
        char gender[GENDER_LEN];
        cout << " 请输入要统计的性别: ";
        cin >> gender;

        int count = 0;
        Student *temp = head;
        while (temp)
        {
            if (strcmp(temp->gender, gender) == 0)
            {
                count++;
            }
            temp = temp->next;
        }
        cout << " 性别为 " << gender << " 的学生人数为: " << count << endl;
        break;
    }
    case 2:
    {
        int age;
        cout << " 请输入要统计的年龄: ";
        cin >> age;

        int count = 0;
        Student *temp = head;
```

```
            while (temp)
            {
                if (temp->age == age)
                {
                    count++;
                }
                temp = temp->next;
            }
            cout << " 年龄为 " << age << " 的学生人数为: " << count << endl;
            break;
        }
        case 3:
        {
            char examType[EXAMTYPE_LEN];
            cout << " 请输入要统计的报考类别: ";
            cin >> examType;

            int count = 0;
            Student *temp = head;
            while (temp)
            {
                if (strcmp(temp->examType, examType) == 0)
                {
                    count++;
                }
                temp = temp->next;
            }
            cout << " 报考类别为 " << examType << " 的学生人数为: " << count << endl;
            break;
        }
        case 4:
        {
            int minAge, maxAge;
            cout << " 请输入要统计的年龄段 (例如 20 25) : ";
            cin >> minAge >> maxAge;

            int count = 0;
            Student *temp = head;
            while (temp)
```

```
76              {
77                  if (temp->age >= minAge && temp->age <= maxAge)
78                  {
79                      count++;
80                  }
81                  temp = temp->next;
82              }
83              cout << " 年龄在 " << minAge << " 到 " << maxAge << " 的学生人数为: " << count <<
                ↪  endl;
84              break;
85          }
86          case 5:
87          {
88              int count = 0;
89              Student *temp = head;
90              while (temp)
91              {
92                  count++;
93                  temp = temp->next;
94              }
95              cout << " 学生总人数为: " << count << endl;
96              break;
97          }
98          default:
99              cout << " 无效选择，请重试" << endl;
100             break;
101         }
102     }
```

3.7.3 统计考生信息功能的实例

```
考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu8     M        22       Programmer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
请选择您要进行的操作：3
请输入要查找的考生的考号：1954444
考号     姓名     性别     年龄     报考类别
1954444 stu4     F        21       Tester
请选择您要进行的操作：5
请选择统计选项：
1. 按性别统计
2. 按年龄统计
3. 按报考类别统计
4. 按年龄段统计
5. 统计所有学生总数
1
请输入要统计的性别：F
性别为 F 的学生人数为：1
请选择您要进行的操作：5
请选择统计选项：
1. 按性别统计
2. 按年龄统计
3. 按报考类别统计
4. 按年龄段统计
5. 统计所有学生总数
2
请输入要统计的年龄：20
年龄为 20 的学生人数为：1
```

```
3
请输入要统计的报考类别：Tester
报考类别为 Tester 的学生人数为：1
请选择您要进行的操作：5
请选择统计选项：
1. 按性别统计
2. 按年龄统计
3. 按报考类别统计
4. 按年龄段统计
5. 统计所有学生总数
4
请输入要统计的年龄段（例如20 25）：19 21
年龄在 19 到 21 的学生人数为：2
请选择您要进行的操作：5
请选择统计选项：
1. 按性别统计
2. 按年龄统计
3. 按报考类别统计
4. 按年龄段统计
5. 统计所有学生总数
5
学生总人数为：4
```

3.8 主函数的实现

主函数首先实例化了一个 Student Manager 的链表类的对象，采用 switch-case 加循环语句来确定用户所需要的操作，在每一个操作中都调用链表类对象的函数，完成相应的操作，并最后输出操作过后的结果。

```
int main()
{
    StudentManager manager;
    char input;
    int choice;

    int numStudent;
    cout << " 首先请建立考生信息系统！" << endl;
    cout << " 请输入考生人数：";
    while (true)
    {
        cin >> numStudent;
        if (cin.fail() || numStudent <= 0)
        {
            cout << " 无效操作，请重新输入" << endl;
            cin.clear();
```

```
            cin.ignore(1024, '\n');
        }
        else break;
    }
    cout << " 请输入考号、姓名、性别、年龄、报考类别！" << endl;
    for (int i = 0; i < numStudent; i++)
    {
        manager.addStudent();
    }
    manager.displayStudents();

    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，Q/q 为退出程序）:" << endl;
    do
    {
        cout << " 请选择您要进行的操作: ";
        while (true)
        {
            cin >> input;
            if (cin.fail() || input < '0' || input > '5')
            {
                cout << " 无效操作，请重新输入" << endl;
                cin.clear();
                cin.ignore(1024, '\n');
            }
            else break;
        }

        // 判断是否退出
        if (input == 'Q' || input == 'q')
        {
            cout << " 程序已退出。" << endl;
            break;
        }

        choice = input - '0'; // 将输入转为数字

        switch (choice)
        {
```

```
55          case 1:
56              manager.insertStudent();
57              manager.displayStudents();
58              break;
59          case 2:
60              manager.deleteStudent();
61              manager.displayStudents();
62              break;
63          case 3:
64              manager.findStudent();
65              break;
66          case 4:
67              manager.updateStudent();
68              manager.displayStudents();
69              break;
70          case 5:
71              manager.analyseStudents();
72              break;
73          case 0:
74              break;
75          default:
76              cout << " 无效操作，请重新输入" << endl;
77              break;
78          }
79      } while (true);
80
81      return 0;
82  }
```

4 项目测试

4.1 建立考生信息库的测试

测试用例:

2351050	stu1	F	20	Designer
2350947	stu2	M	21	Developer
2252552	stu3	M	20	Developer

预期结果:

2351050	stu1	F	20	Designer
2350947	stu2	M	21	Developer
2252552	stu3	M	20	Developer

测试结果:

```
首先请建立考生信息系统!
请输入考生人数: 3
请输入考号、姓名、性别、年龄、报考类别!
2351050 stu1 F 20 Designer
2350947 stu2 M 21 Developer
2252552 stu3 M 20 Developer

考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
```

4.2 插入考生信息功能的测试

测试用例:

2

1954444	stu4	F	21	Tester

1

2337088	stu5	M	18	Designer

预期结果:

2337088	stu5	M	18	Designer
2351050	stu1	F	20	Designer
1954444	stu4	F	21	Tester
2350947	stu2	M	21	Developer
2252552	stu3	M	20	Developer

测试结果:

```
考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
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，Q/q为退出程序）：
请选择您要进行的操作：1
请输入你要插入的考生的位置：2
请依次输入要插入考生的考号、姓名、性别、年龄、报考类别！
1954444 stu4 F 21 Tester

考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
请选择您要进行的操作：1
请输入你要插入的考生的位置：1
请依次输入要插入考生的考号、姓名、性别、年龄、报考类别！
2337088 stu5 M 18 Designer

考号    姓名    性别    年龄    报考类别
2337088 stu5    M       18      Designer
2351050 stu1    F       20      Designer
1954444 stu4    F       21      Tester
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
```

4.3 删除考生信息功能的测试

测试用例:

2350947

预期结果:

2337088	stu5	M	18	Designer
2351050	stu1	F	20	Designer
1954444	stu4	F	21	Tester
2252552	stu3	M	20	Developer

测试结果:

```
考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu1     F        20       Designer
1954444 stu4     F        21       Tester
2350947 stu2     M        21       Developer
2252552 stu3     M        20       Developer
请选择您要进行的操作：2
请输入要删除的考生的考号：2350947
你删除的考生信息是：2350947 stu2    M       21      Developer

考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu1     F        20       Designer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
```

4.4 修改考生信息功能的测试

测试用例：

2351050

stu8	M	22	Programmer

预期结果：

2337088	stu5	M	18	Designer
2351050	stu8	M	22	Programmer
1954444	stu4	F	21	Tester
2252552	stu3	M	20	Developer

测试结果：

```
考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu1     F        20       Designer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
请选择您要进行的操作：4
输入要修改的考生的考号：2351050
输入新姓名：stu8
输入新性别：M
输入新年龄：22
输入新报考类别：Programmer
学生信息已更新

考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu8     M        22       Programmer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
```

4.5 查询考生信息功能的测试

测试用例:

1954444

预期结果:

1954444	stu4	F	21	Tester

测试结果:

```
考号     姓名     性别     年龄     报考类别
2337088 stu5     M        18       Designer
2351050 stu8     M        22       Programmer
1954444 stu4     F        21       Tester
2252552 stu3     M        20       Developer
请选择您要进行的操作：3
请输入要查找的考生的考号：1954444
考号     姓名     性别     年龄     报考类别
1954444 stu4     F        21       Tester
```

4.6 健壮性测试

在建立考生信息库过程中，人数输入非正整数会提示重新输入：

```
首先请建立考生信息系统！
请输入考生人数：0
无效操作，请重新输入
q
无效操作，请重新输入
2
```

在获取操作时输入非 0-5 的数字或‘Q/q’时会提示重新输入：

```
考号     姓名     性别     年龄     报考类别
2351050 stu1     F        20       Designer
2350947 stu2     M        21       Developer
2252552 stu3     M        20       Developer
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
请选择您要进行的操作：6
无效操作，请重新输入
f
无效操作，请重新输入
1
```

在插入考生位置为非正整数时（大于当前链表长度的会被自动加入链表末尾）会提示重新输入：

```
考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
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
请选择您要进行的操作：6
无效操作，请重新输入
f
无效操作，请重新输入
1
请输入你要插入的考生的位置：-2
无效操作，请重新输入
p
无效操作，请重新输入
2
请依次输入要插入考生的考号、姓名、性别、年龄、报考类别！
2337088 stu5 M 18 Designer
```

在删除、查找、修改操作时输入错误的学号会提示未找到该学生：

```
考号    姓名    性别    年龄    报考类别
2351050 stu1    F       20      Designer
2337088 stu5    M       18      Designer
2350947 stu2    M       21      Developer
2252552 stu3    M       20      Developer
请选择您要进行的操作：2
请输入要删除的考生的考号：2350000
未找到该学生
```

5 项目心得与体会

我至今为止编写的最长代码之一便是考试报名系统，它让我收获颇丰，特别是在编程经验和技巧上。首先，我对链表的基本操作如插入、删除等有了更深入的理解和实践；其次，这是我第一次用 C++ 类的方式编写程序，过程中我查阅了大量资料，深入学习了面向对象编程的许多方法，如类的定义、成员的选择以及类操作的调用等，这极大地提升了我对 C++ 面向对象编程的理解；再者，我编写了大量关于链表类的操作代码，如删除、插入、查找和计算长度等，这些函数可以保存并复用，为后续的编程工作提供了极大的便利。总的来说，这个项目让我获得了许多宝贵的编程思路和技巧。