****

项目说明文档

**数据结构课程设计**

**——考试报名系统**

**培养单位：软件学院**

**本 科 生：蓝 盛 霖**

**学 号：1951096**

**指导老师：张 颖**

二○二○年十二月

目录

[第1章 分析 1](#_Toc58711260)

[1.1背景分析 1](#_Toc58711261)

[1.2 功能分析 1](#_Toc58711262)

[第2章 设计 2](#_Toc58711263)

[2.1 数据结构设计 2](#_Toc58711264)

[2.2 类结构设计 2](#_Toc58711265)

[2.3 成员与操作设计 2](#_Toc58711266)

[2.4 系统设计 3](#_Toc58711267)

[第3章 实现 4](#_Toc58711268)

[3.1 插入功能的实现 4](#_Toc58711269)

[3.1.1 插入功能流程图 4](#_Toc58711270)

[3.1.2 插入功能核心代码 5](#_Toc58711271)

[3.1.3 插入功能截屏示例 6](#_Toc58711272)

[3.2 删除功能的实现 7](#_Toc58711273)

[3.2.1 删除功能流程图 7](#_Toc58711274)

[3.2.2 删除代码实现 7](#_Toc58711275)

[3.2.3 删除功能截屏示例 8](#_Toc58711276)

[3.3 查找功能的实现 9](#_Toc58711277)

[3.3.1 查找功能流程图 9](#_Toc58711278)

[3.3.2 查找代码实现 9](#_Toc58711279)

[3.3.3 查找功能截屏示例 10](#_Toc58711280)

[3.4 修改功能的实现 11](#_Toc58711281)

[3.4.1 修改功能流程图 11](#_Toc58711282)

[3.4.2 修改代码实现 11](#_Toc58711283)

[3.4.3 修改功能截屏示例 12](#_Toc58711284)

[3.5 统计功能的实现 13](#_Toc58711285)

[3.5.1 统计功能流程图 13](#_Toc58711286)

[3.5.2 统计代码实现 13](#_Toc58711287)

[3.5.3 统计功能截屏示例 14](#_Toc58711288)

[3.6 总体系统的实现 15](#_Toc58711289)

[3.6.1 总体系统流程图 15](#_Toc58711290)

[3.6.2 总体系统代码实现 15](#_Toc58711291)

[3.6.3 统计功能截屏示例 18](#_Toc58711292)

[第4章 测试 19](#_Toc58711293)

[4.1 功能测试 19](#_Toc58711294)

[4.1.1 插入功能测试 19](#_Toc58711295)

[4.1.2 删除功能测试 19](#_Toc58711296)

[4.1.3 查找功能测试 20](#_Toc58711297)

[4.1.4 修改功能测试 20](#_Toc58711298)

[4.1.5 统计功能测试 21](#_Toc58711299)

[4.2 边界测试 21](#_Toc58711300)

[4.2.1 初始化无输入数据 21](#_Toc58711301)

[4.2.2 删除头结点 22](#_Toc58711302)

[4.2.3 删除后链表为空 22](#_Toc58711303)

[4.3 出错测试 22](#_Toc58711304)

[4.3.1 考生人数错误 22](#_Toc58711305)

[4.3.2 操作码错误 23](#_Toc58711306)

[4.3.3 插入位置不存在 23](#_Toc58711307)

[4.3.4 删除考号不存在 23](#_Toc58711308)

[4.3.5 查找考号不存在 24](#_Toc58711309)

[4.3.6 修改考号不存在 24](#_Toc58711310)

# 第1章 分析

## 1.1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 功能分析

作为一个最简易的考试报名系统，首先应该有的功能就是输入同学们的考试报名情况并且可以予以显示。其次，考试报名系统还应该具有插入、删除、修改功能，以保证同学可以随时更改自己的考试报名情况。最后，考试报名系统软件还应该确保软件可以正常关闭。

综上所述，一个考试报名系统至少应该具有输入、输出、插入、删除、修改、退出的功能。

# 第2章 设计

## 2.1 数据结构设计

如上功能分析所述，该系统要求大量的增加、删除、修改操作，而链表进行增加、删除等操作十分简便，因此考虑使用链表数据结构。同时，为了实现简易，在第一个结点之前附加一个头结点，这样就使得增加或者删除头结点与处理其他结点方法相同，使得程序简洁。

## 2.2 类结构设计

经典的链表一般包括两个抽象数据类型（ADT）——链表结点类（Node）与链表类（List），而两个类之间的耦合关系可以采用嵌套、继承等多种关系。为方便处理，本系统采用模板struct结构体描述链表结点类（Node<T>），这样使得模板链表结点类（List<T>）可以访问链表结点。并且将学生数据统一存储为struct结构体，方便调用模板类。

## 2.3 成员与操作设计

**学生信息结构体（Student）**

struct Student {

    int \_id = 0;          //学号

    int \_age = 0;         //年龄

    std::string \_sex;     //性别

    std::string \_name;    //姓名

    std::string \_choice;  //志愿

    friend std::ostream &operator<<(std::ostream &output, const Student &D);

    friend std::istream &operator>>(std::istream &input, const Student &D);

    bool operator==(const Student D);

    void operator=(const Student D);

};

**模板链表节点结构体（Node）**

template <class T>

struct Node {

    T \_data;      //数据

    Node \*\_next;  //指针域

    Node();

};

**模板链表类（Node）**

template <class T>

class List {

   private:

    Node<T> \*\_head;  //表头结点

    int \_len;        //链表长度

   public:

    void listIns(int n, T data);  //将数据插入第n个节点中

    T listPop(T data);    //查找链表中的数据返回并删除节点

    int listFind(T id);   //查找数据在链表中的位置

    int listLen() const;  //返回链表长度

    bool empty() const;   //返回是否为空

    void listChange(int n, T data);      //修改链表第n个节点的值

    std::string listPrint(int n) const;  //将链表输出

    List();

    ~List();

};

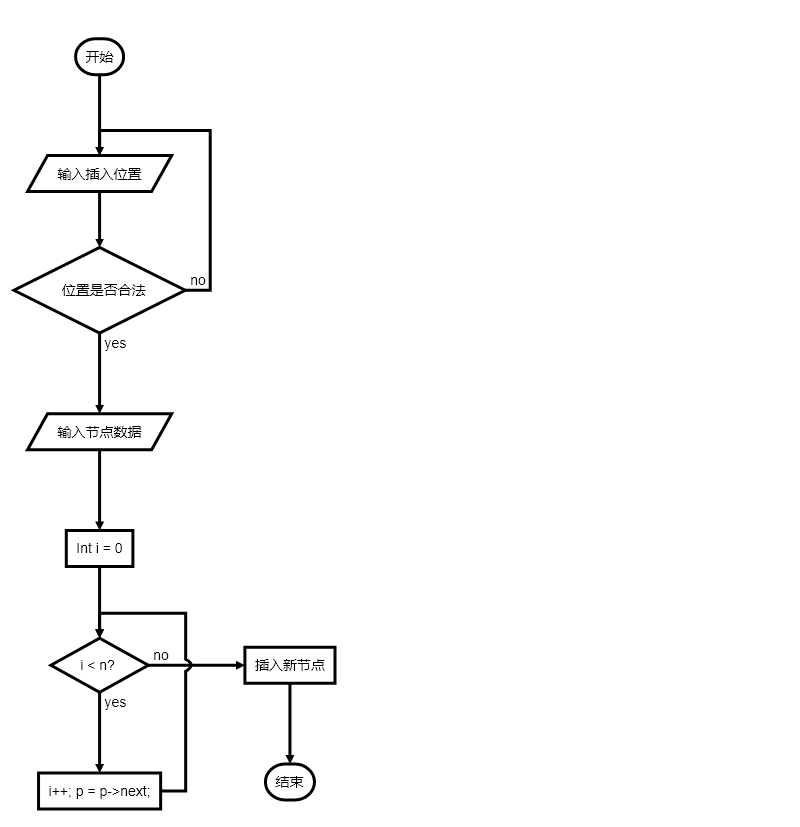
## 2.4 系统设计

系统首先调用opening ()函数实现对屏幕的初始化，完成对链表L的创建和输入数据工作，然后进入loop ()循环，根据用户所输入的操作码（operatorCode）执行链表L对应的成员函数。

# 第3章 实现

## 3.1 插入功能的实现

### 3.1.1 插入功能流程图



### 3.1.2 插入功能核心代码

template <class T>

void List<T>::listIns(int n, T data) {

    Node<T> \*p = this->\_head;

    for (int i = 0; i < n; i++) {

        if (p->\_next == nullptr) throw "Error: Not so many nodes.";

        p = p->\_next;

    }

    Node<T> \*node = new Node<T>;

    node->\_data = data;

    node->\_next = p->\_next;

    p->\_next = node;

    this->\_len++;

}

void insList(List<Student> &list) {

    std::cout << "Please enter the student location to insert: ";

    int i;

    std::cin >> i;

    if (i < 0 || i > list.listLen()) throw "Error: Invalid input.";

    std::cout << "Please input the student's id, name, gender, age and first "

                 "choice in turn."

              << std::endl;

    Student temp;

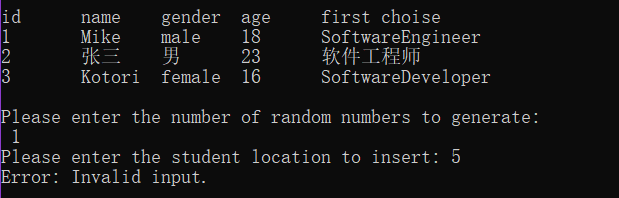
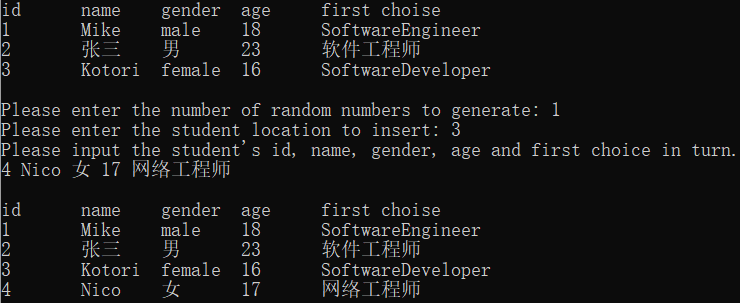
    std::cin >> temp;

    list.listIns(i, temp);

    printList(std::cout, list);

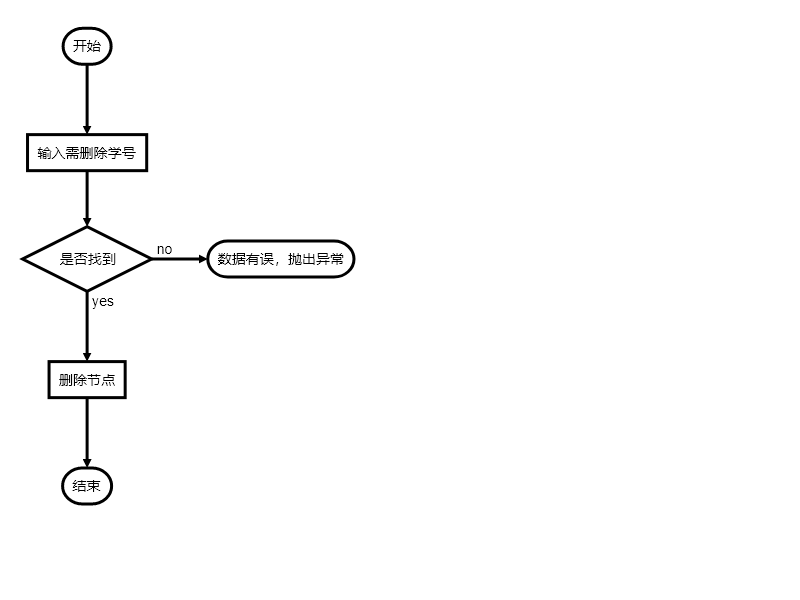
}

### 3.1.3 插入功能截屏示例



## 3.2 删除功能的实现

### 3.2.1 删除功能流程图



### 3.2.2 删除代码实现

Template <class T>

T List<T>::listPop(T data) {

if (this->empty()) throw "Error: The list is empty.";

    int n = this->listFind(data);

    if (n == -1) throw "Error: Data not found.";

    Node<T> \*p = this->\_head;

    for (int i = 0; i < n; i++) {

        if (p->\_next == nullptr) throw "Error: Not so many nodes.";

        p = p->\_next;

    }

    Node<T> \*node = p->\_next;

    p->\_next = p->\_next->\_next;

    auto i = node->\_data;

delete node;

    this->\_len--;

    return i;

}

void delList(List<Student> &list) {

    if (list.empty()) throw "Error: The list is empty.";

    std::cout << "Please enter the id of the student to delete: ";

    int i;

    std::cin >> i;

    Student temp;

    temp.\_id = i;

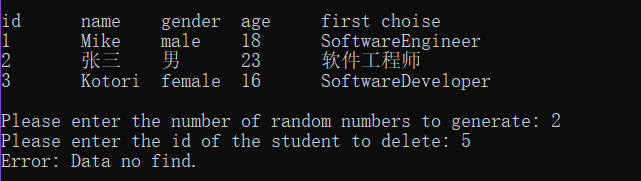
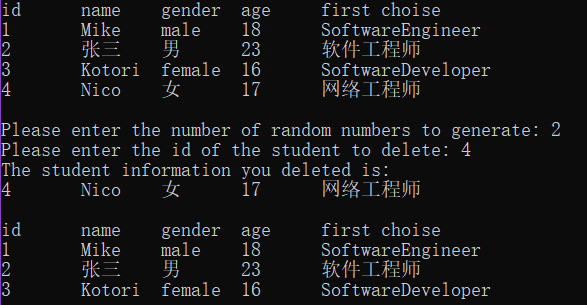
auto s = list.listPop(temp);

    std::cout << "The student information you deleted is: \n" << s;

    if (!list.empty()) printList(std::cout, list);

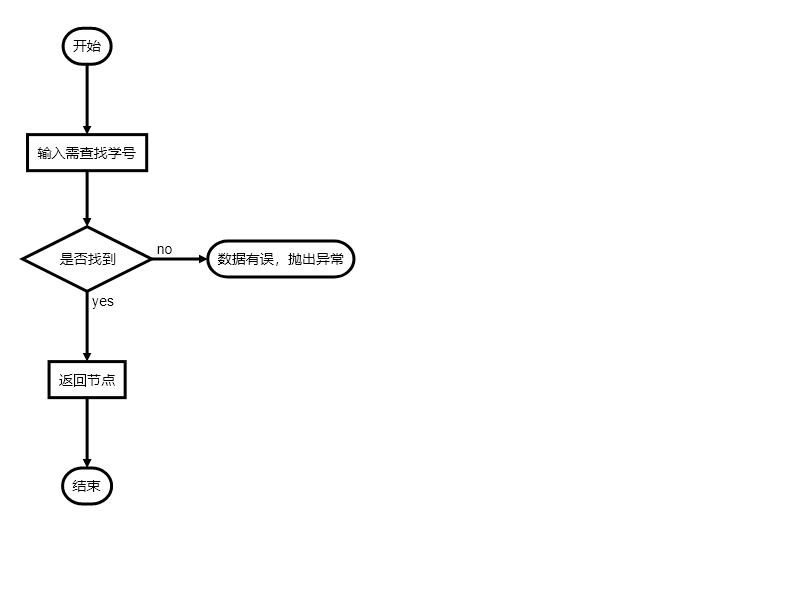
}

### 3.2.3 删除功能截屏示例



## 3.3 查找功能的实现

### 3.3.1 查找功能流程图



### 3.3.2 查找代码实现

template <class T>

int List<T>::listFind(T data) {

    if (this->empty()) return -1;

    Node<T> \*p = this->\_head;

    p = p->\_next;

    int i = 0;

    while (true) {

        if (p->\_data == data) return i;

        if (p->\_next == nullptr) return -1;

        p = p->\_next;

        i++;

    }

}

void findList(List<Student> &list) {

    if (list.empty()) throw "Error: The list is empty.";

    std::cout << "Please enter the id of the student you are looking for: ";

    int i;

    std::cin >> i;

    Student temp;

    temp.\_id = i;

    int n = list.listFind(temp);

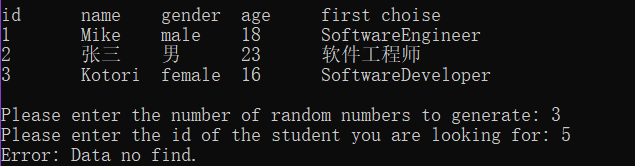
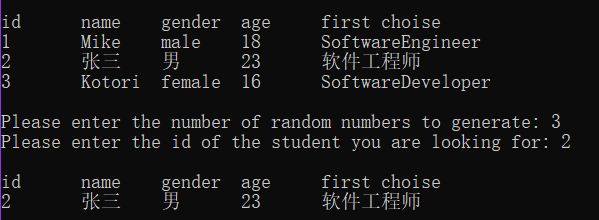
    std::cout << "\nid" << '\t' << "name" << '\t' << "gender" << '\t' << "age"

              << '\t' << "first choise" << std::endl

              << list.listPrint(n);

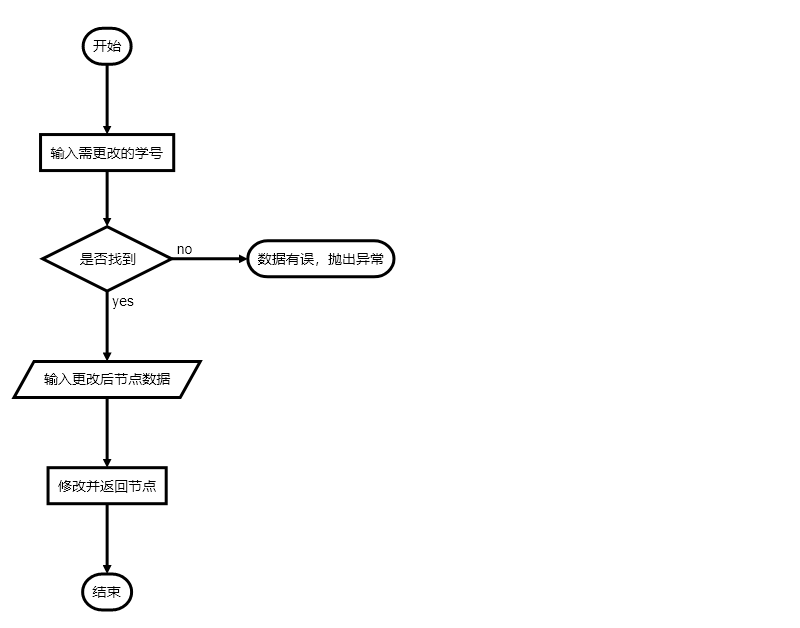
}

### 3.3.3 查找功能截屏示例



## 3.4 修改功能的实现

### 3.4.1 修改功能流程图



### 3.4.2 修改代码实现

template <class T>

void List<T>::listChange(int n, T data) {

    if (this->empty()) throw "Error: The list is empty.";

    auto p = this->\_head->\_next;

    for (int i = 0; i < n; i++) {

        if (p->\_next == nullptr) throw "Error: Not so many nodes.";

        p = p->\_next;

    }

    p->\_data = data;

}

void changeList(List<Student> &list) {

    if (list.empty()) throw "Error: The list is empty.";

    std::cout << "Please input the id of the student to be changed: ";

    int i;

    std::cin >> i;

    Student temp;

    temp.\_id = i;

int n = list.listFind(temp);

if (n == -1) throw "Error: Data no find.";

    std::cout << "Please enter the change student's id, name, gender, age and "

                 "first choise in turn"

              << std::endl;

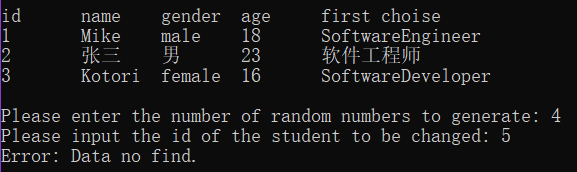
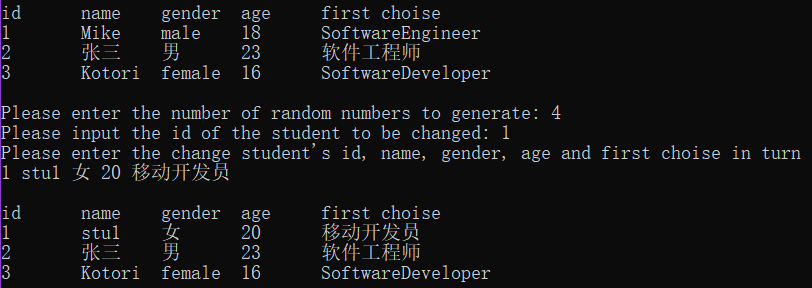
    std::cin >> temp;

    list.listChange(n, temp);

    printList(std::cout, list);

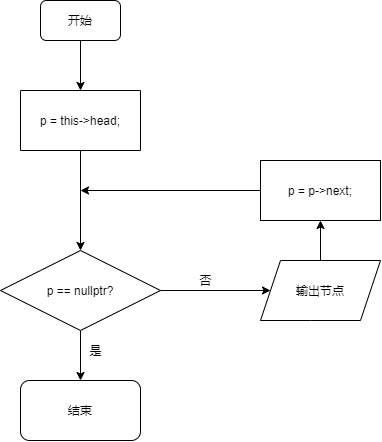
}

### 3.4.3 修改功能截屏示例



## 3.5 统计功能的实现

### 3.5.1 统计功能流程图



### 3.5.2 统计代码实现

template <class T>

std::string List<T>::listPrint(int n) const {

    if (this->empty()) throw "Error: The list is empty.";

    auto p = this->\_head->\_next;

    for (int i = 0; i < n; i++) {

        if (p->\_next == nullptr) throw "Error: Not so many nodes.";

        p = p->\_next;

    }

    std::stringstream ss;

    ss << p->\_data;

    return ss.str();

}

void printList(std::ostream &os, const List<Student> &list) {

    if (list.empty()) throw "Error: The list is empty.";

    os << "\nid" << '\t' << "name" << '\t' << "gender" << '\t' << "age" << '\t'

       << "first choise" << std::endl;

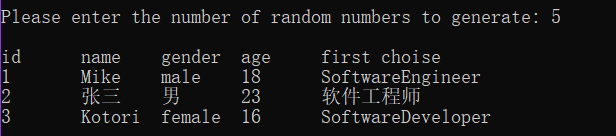
    for (int i = 0; i < list.listLen(); i++) {

        os << list.listPrint(i);

    }

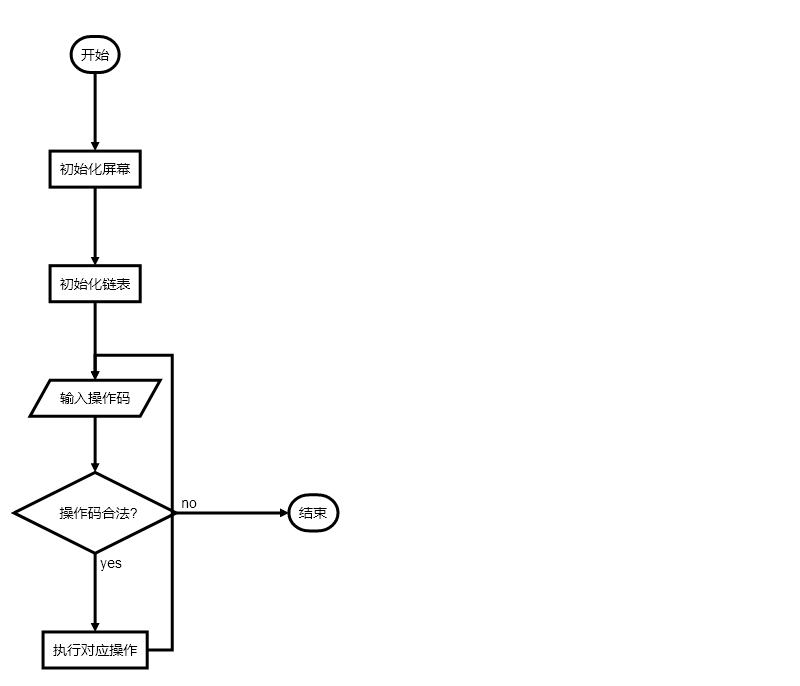
}

### 3.5.3 统计功能截屏示例



## 3.6 总体系统的实现

### 3.6.1 总体系统流程图



### 3.6.2 总体系统代码实现

int opening(List<Student> &list) {

    std::cout << "First set up the examinee information system" << std::endl;

    int num = 0;

    do {

        std::cout << "Please enter the number of students: ";

        std::cin >> num;

        if (num == 0) return num;

        if (num < 0)

            std::cerr << "Error: Invalid input, Please try again." << std::endl;

    } while (num < 0);

    std::cout << "Please input the student's id, name, gender, age and first "

                 "choice in turn. (separated by ' ')"

              << std::endl;

    for (int i = 0; i < num; i++) {

        Student temp;

        std::cin >> temp;

        list.listIns(i, temp);

    }

    return num;

}

bool loop(List<Student> &list) {

    std::cout << std::endl

              << "Please enter the number of random numbers to generate: ";

    char i = '\0';

    std::cin >> i;

    try {

        switch (i) {

            case '1':

                insList(list);

                return true;

            case '2':

                delList(list);

                return true;

            case '3':

                findList(list);

                return true;

            case '4':

                changeList(list);

                return true;

            case '5':

                printList(std::cout, list);

                return true;

            default:

                return false;

        }

    } catch (const char \*e) {

        std::cerr << e << std::endl;

    }

    return true;

}

int main() {

    List<Student> list;

    std::cout << "\*\*             Examination Enrollment System            \*\*"

              << std::endl

              << "=========================================================="

              << std::endl

              << "\*\*         Please select the action to perform:         \*\*"

              << std::endl

              << "\*\*           1 --- Insert the students                  \*\*"

              << std::endl

              << "\*\*           2 --- Delete the students                  \*\*"

              << std::endl

              << "\*\*           3 --- Search the students                  \*\*"

              << std::endl

              << "\*\*           4 --- change the students                  \*\*"

              << std::endl

              << "\*\*           5 --- Print the list                       \*\*"

              << std::endl

              << "\*\*           Others --- Exit                            \*\*"

              << std::endl

              << std::endl;

    int num = opening(list);

    if (num != 0) printList(std::cout, list);

    while (loop(list))

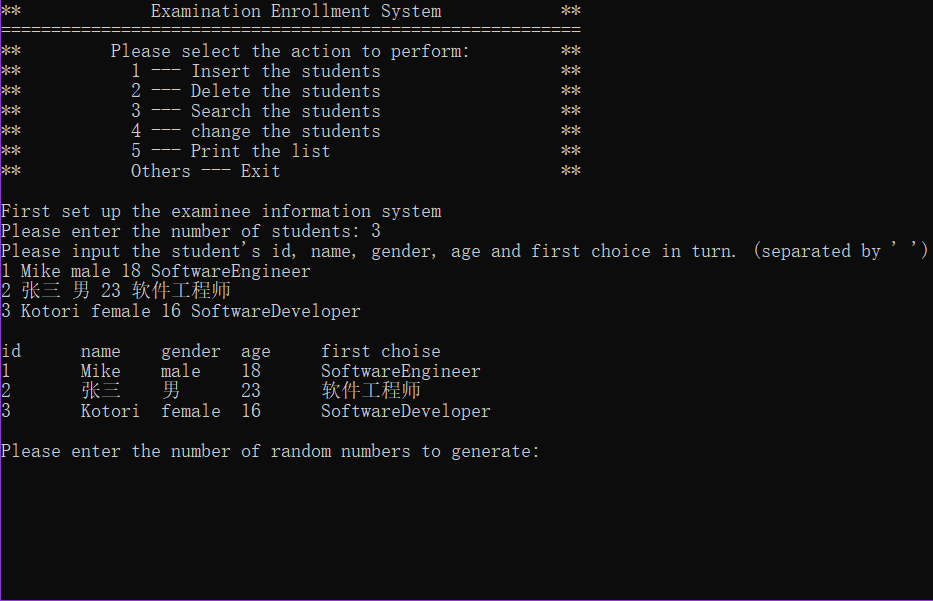
        ;

    std::cout << "Thanks for using it. See you next time! " << std::endl;

    return 0;

}

### 3.6.3 总体功能截屏示例



# 第4章 测试

## 4.1 功能测试

### 4.1.1 插入功能测试

**测试用例**：4 Nico 女 17 网络工程师

**预期结果**：

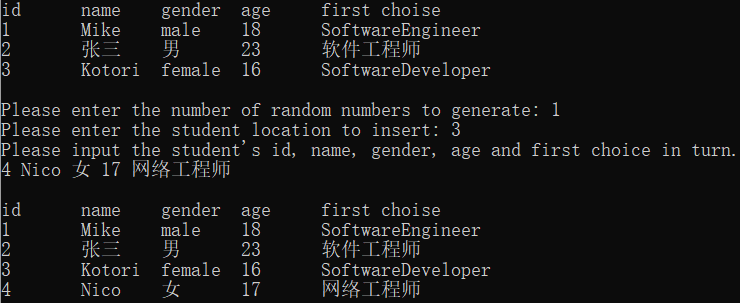
1 Mike male 18 SoftwareEngineer

2 张三 男 23 软件工程师

3 Kotori female 16 SoftwareDeveloper

4 Nico 女 17 网络工程师

**实验结果**



### 4.1.2 删除功能测试

**测试用例：**删除考号为4的考生

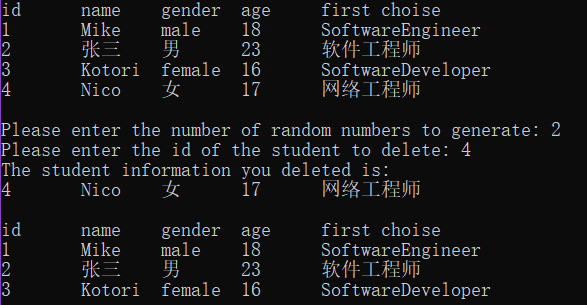
**预期结果：**

1 Mike male 18 SoftwareEngineer

2 张三 男 23 软件工程师

3 Kotori female 16 SoftwareDeveloper

**实验结果：**



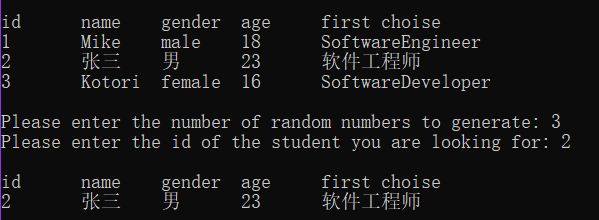
### 4.1.3 查找功能测试

**测试用例：**查找考号为2的考生

**预期结果：**

2 张三 男 23 软件工程师

**实验结果：**



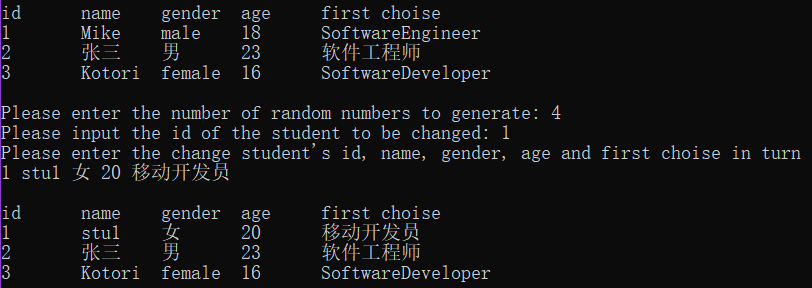
### 4.1.4 修改功能测试

**测试用例：**将考号1修改为性别女，年龄20，报考种类移动开发员。

**预期结果：**

1 stu1 女 20 移动开发员

**实验结果：**



### 4.1.5 统计功能测试

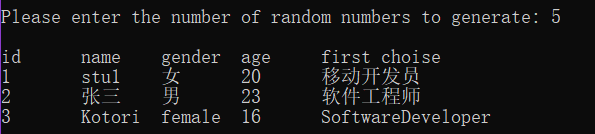
**测试用例：**统计当前数据

**预期结果：**

1 stu1 女 20 移动开发员

2 张三 男 23 软件工程师  
3 Kotori female 16 SoftwareDeveloper

**实验结果：**



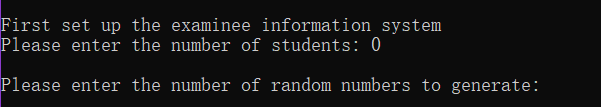
## 4.2 边界测试

### 4.2.1 初始化无输入数据

**测试用例：**初始无输入数据

**预期结果：**创建空链表，程序运行正常不崩溃。

**实验结果：**

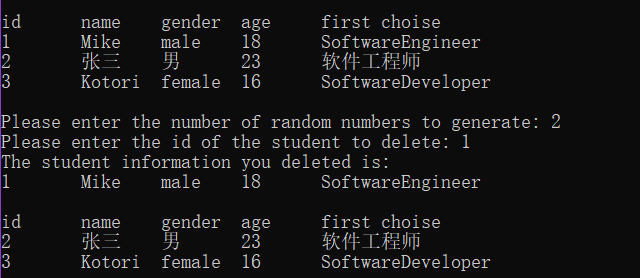


### 4.2.2 删除头结点

**测试用例：**删除头结点

**预期结果：**程序正常运行，不崩溃。

**实验结果：**

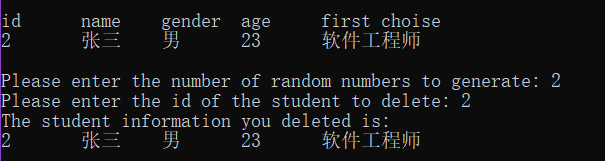


### 4.2.3 删除后链表为空

**测试用例：**删除前链表只有一个结点，删除后链表为空

**预期结果：**程序正常运行，不崩溃。

**实验结果：**



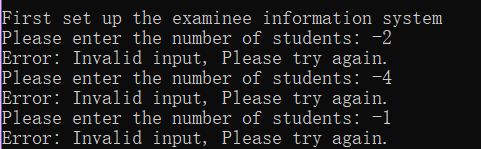
## 4.3 出错测试

### 4.3.1 考生人数错误

**测试用例：**输入考生人数为负数

**预期结果：**程序给出提示信息，程序正常运行不崩溃。

**实验结果：**

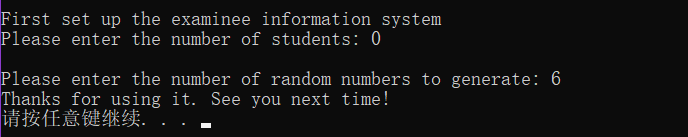


### 4.3.2 操作码错误

**测试用例：**输入操作码错误

**预期结果：**程序认为输入退出指令，自动退出。

**实验结果：**

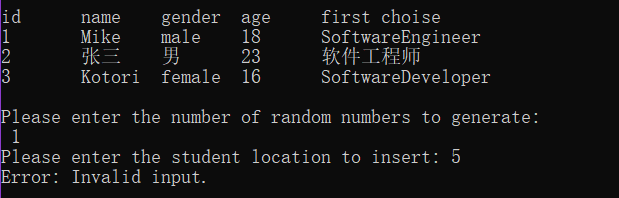


### 4.3.3 插入位置不存在

**测试用例：**链表里有三条记录，向链表的第五个位置插入结点

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

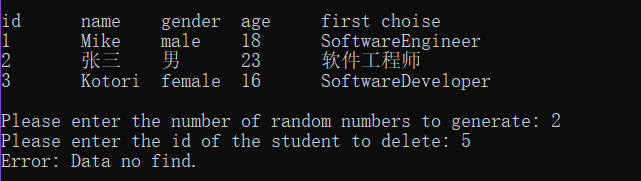


### 4.3.4 删除考号不存在

**测试用例：**要删除的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

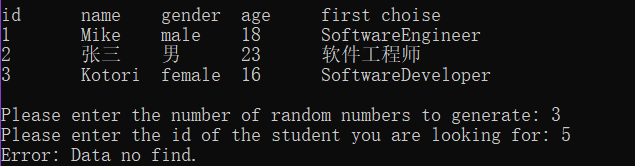


### 4.3.5 查找考号不存在

**测试用例：**要查找的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**



### 4.3.6 修改考号不存在

**测试用例：**要修改的考号不存在

**预期结果：**程序给出错误信息，程序正常运行不崩溃。

**实验结果：**

