**数据结构实验报告**

**学号：\_\_\_\_CST17037\_\_\_\_\_ 姓名：\_\_\_陈锟\_\_\_**

# 实验题目

使用链表完成学生信息管理系统

# 实验要求

(1) 理解线性表的概念

(2) 理解单链表存储结构概念和特点

(3) 掌握单链表存储结构的建立、插入、删除、查询和输出等基本操作算法

(4) 理解线性表顺序存储结构和链式存储结构的不同特点

(6) 能够根据实际应用的情况，综合考虑求解算法的时间复杂度和空间复杂度，选择合适的线性表存储结构

# 方案设计

程序分为多个模块:

性别枚举:Gender{male,female}

数据类 Date,描述数据个体的结构

单链表模板类 SinglyLinkedList,实现链表的算法模板类

学生信息表 StudentList,继承自线性表类,对线性表的拓展及功能包装，修改父类即可修改实现类型。

主函数 组织调用 StudentList类函数

# 方案实现

数据类 Date:

学号 姓名 string num,name;

性别 Gender gender ;

分数 int score ;

输出函数 printDate();格式化输出数据所有信息

顺序表模板类SinglyLinkedList:

保护型成员:

链表当前长度unsigned int length;

单链表表头 Node<T> \*head;

公有型成员:

InitList();初始化数据

~InitList();删除链表，调用setEmpty()函数依次释放所有节点

void printList();循环输出所有成员信息

void setEmpty();从表头开始依次释放所有节点内存空间，清空链表

bool modifyNodeByNum(unsigned int num, T d);将第num位节点修改为 d,修改成功返回true

void invert();逆置链表，原地逆置

bool insertByNum(unsigned int num, T d);将节点d插入到第num位,原num位元素及之后原素向后一位,成功返回true

unsigned int findListByDateNum(Date d);根据节点d信息查找节点,并返回对应编号,未找到返回 0

bool delNode(unsigned int num);删除节点num，并释放内存空间，原num位元素及之后原素向前一位,成功返回true

unsigned int getLength();获取实际长度length

学生信息表类StudentList:

私有型成员:

Date inputStudentInfo();输入学生信息,包含异常数据处理

StudentList();构造函数,调用父类构造函数创建链表

void creatStudentList();创建学生信息表,输入数据,创建学生信息表

void printStudentList();输出所有学生信息

void prinStudentListTitle();打印学生信息表表头

void modifyStudnetNode();修改某学生信息

void invertStudentList();逆置学生信息表

void insertStudentList();插入学生信息表

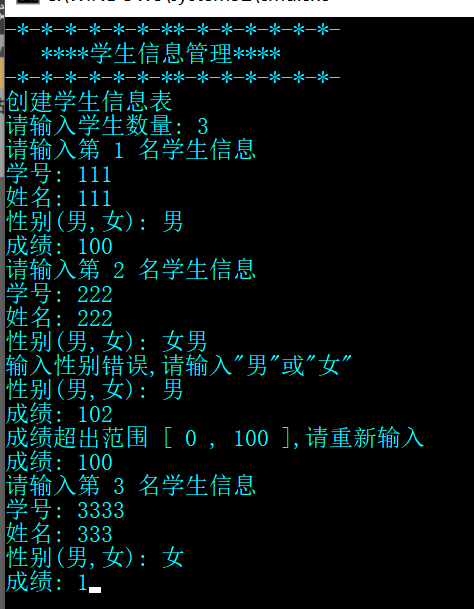
unsigned int findStudentList();查找学生信息

void delStudentList();删除学生信息

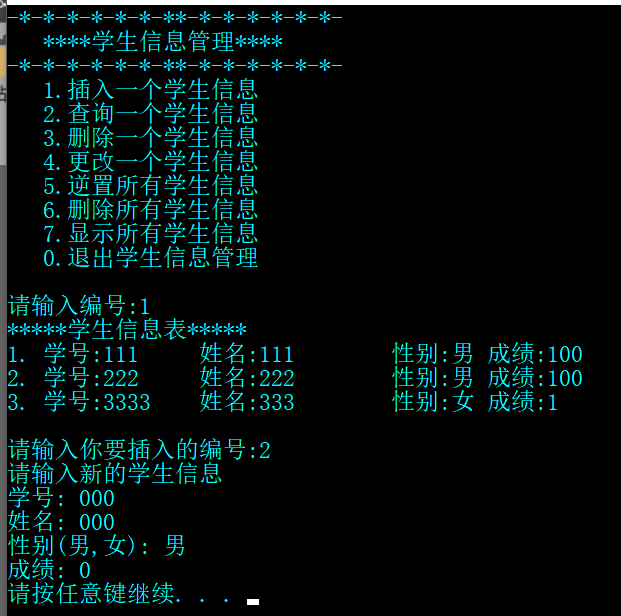
全局函数:

int inputNumber(string info);输入数字,以info提示输入数字,函数对输入数据进行异常提示处理,并返回有效数字

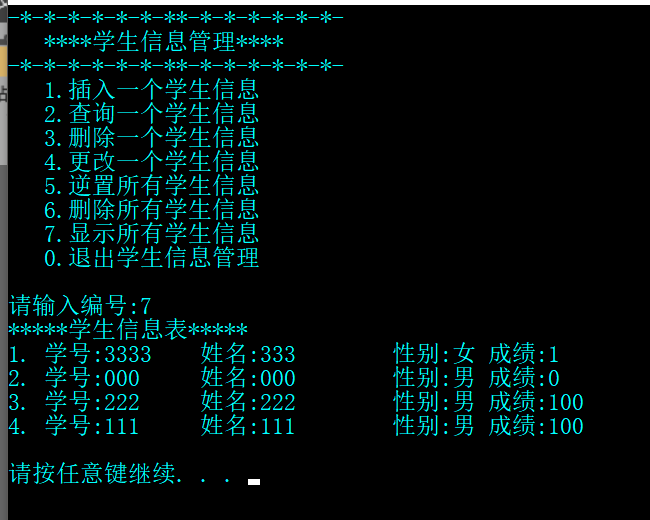
# 调试分析



插入信息



逆置信息：



时间复杂度分析:学生信息表使用单链表实现

插入与删除操作,每次只需要修改对应节点后驱节点指向,故时间复杂度为O(1);

查找与访问操作,时间复杂度为O(n);

访问后驱节点只需要一次间接寻址，时间复杂度为O(1)

访问前驱节点需要从表头开始遍历链表，依次于当前节点比对，时间复杂度为O(n)

相对于顺序表，链表优势在于插入与删除操作，因此在大量需要插入删除操作的场景适合使用链表，但是链表不支持随机访问，查找与访问操作在大问题规模下效率低。顺序表支持随机访问，但是插入与删除操作需要移动大量元素，适合频繁的访问操作。

# 实验收获

练习了面向对象程序设计,复习了链表的知识