

# 《数据结构》实验



## 实验一：线性结构及其应用

主讲教师：王玮

实验教师：杨扬

助 教：卜贤达、林堉欣

计算机科学与技术学院  
哈尔滨工业大学（深圳）

# 实验总体介绍

实验课程共**16**个学时，**4**次实验，**1**份实验报告  
实验课成绩占总成绩**20%**

实验 题目	一	二	三	四
学时数	4	4	4	4
实验 内容	线性结构 链表、栈与队列	树型结构	图型结构	查找排序
上课 周次	W7	W9	W11	W14
时间 (计5/6)	周五9-12	周五5-8	周四9-12	周五5-8

# 实验总体要求

- 禁止抄袭，发现抄袭，一律0分处理
- 编程语言：C（C++和其他编程语言目前OJ平台不支持）
- 实验统一在 <http://10.249.176.82:9000/> 平台上完成，平台的用户初始账号和密码均为学号，具体使用参考文档
- 每个实验将提供对应的代码模板，仅供参考
- 每次实验作业完成时间：两周
- 延迟提交期限：7天，扣除20%完成分，超期将无法提交。
- 四次实验仅挑选一次写实验报告，需在最后一次实验截止时间前提交。

# 源代码评分标准



- 对于实验的每个编程小题，通过所有测试用例才可拿到满分，没有通过则零分。对于未通过的用例实验平台会给予相应的提示。
- 建议在本地IDE（如codeblocks）编写代码，编译测试通过后再将代码复制到实验平台进行测试。
- 禁止忽略逻辑直接printf()输出结果，所有代码实验平台后台都有存储，一旦发现，本次实验0分。
- 本次实验无时间、空间复杂度限制，返回此类信息仅供参考。

# 实验报告要求及评分标准

- 四次实验选一次写报告，并于第四次实验截止日期之前在平台提交。
- 实验报告模板参见word文档
- 评分标准，采用百分制，各部分分值如下：
  - 问题分析（20分）
    - 能将原题要解决的问题转换成用计算机要解决的问题。
  - 详细设计（50分）
    - 设计思想（15分）
    - 存储结构及操作（15分）
    - 程序整体流程（20分）
  - 运行结果(10分)
    - 本地运行结果或者平台运行结果的截图。
  - 总结（20分）
    - 总结出该实验涉及到的数据结构和算法，以及遇到的问题和收获。

# 实验一 线性结构及其应用

众所周知，大学生体测的成绩一般呈现逐年下降的趋势。贴心的辅导员为了帮大家记录身体素质最好的时刻，决定统计一下大一新生的体测成绩。

## 【需求】

(1) **建立链表**：请按**头插法**将一个班的成绩存储到链表中。

注：输入的成绩是降序的，因此链表存储的成绩是升序的

(2) **反转链表**：将建立的升序链表按降序进行反转。


(3) **链表交点**：辅导员录入时，由于班级分数最低的若干同学恰好同分，她不小心将链表的尾部交叉到了一起，只保留了一部分同学。辅导员很着急，因为她想重点关注这些体测成绩吊车尾的学生，劝导他们多锻炼身体。现在辅导员请你帮她编程解决这个问题。

你的任务是编程找出两个交叉链表的第一个公共交点，你只能拿到两个链表的头结点。注：创建交叉链表的代码已给出。

# 实验内容

## 1001: 体测成绩

### 【输入示例】



```
5 6 /*两个班级的人数n1、n2*/
Student1 100 /*班级1的成绩，按降序*/
Student2 90
Student3 80
Student4 70
Student5 60
Student6 95 /*班级2的成绩，按降序*/
Student7 88
Student8 82
Student9 80
Student10 70
Student11 60
3 4 /*两个链表交叉之前的结点个数m1、m2*/
```

# 实验内容

## 1001: 体测成绩

### 【输出示例】

Store Linkedlist

```
{ID:Student5, Grade:60}->{ID:Student4, Grade:70}->{ID:Student3, Grade:80}->{ID:Student2, Grade:90}->{ID:Student1, Grade:100}  
{ID:Student11, Grade:60}->{ID:Student10, Grade:70}->{ID:Student9, Grade:75}->{ID:Student8, Grade:82}->{ID:Student7, Grade:88}->{ID:Student6, Grade:95}
```

Reverse Linkedlist

```
{ID:Student1, Grade:100}->{ID:Student2, Grade:90}->{ID:Student3, Grade:80}->{ID:Student4, Grade:70}->{ID:Student5, Grade:60}  
{ID:Student6, Grade:95}->{ID:Student7, Grade:88}->{ID:Student8, Grade:82}->{ID:Student9, Grade:75}->{ID:Student10, Grade:70}->{ID:Student11, Grade:60}
```

Cross node

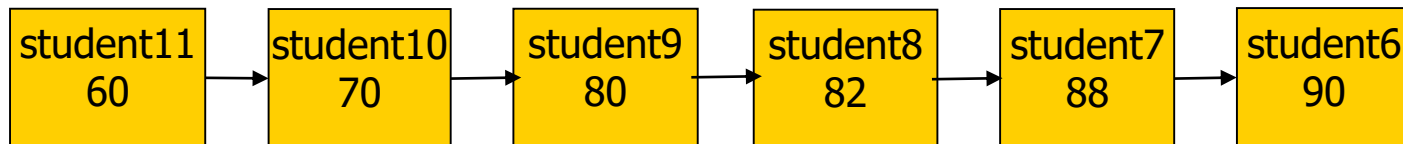
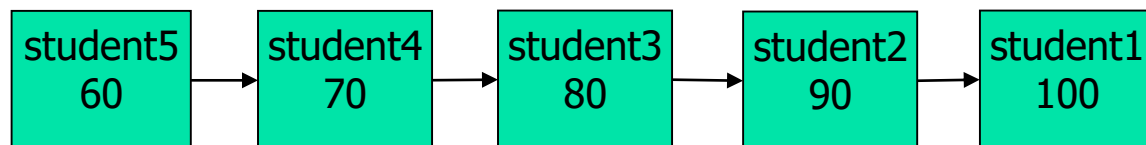
```
{ID:Student4, Grade:70}
```



# 实验内容

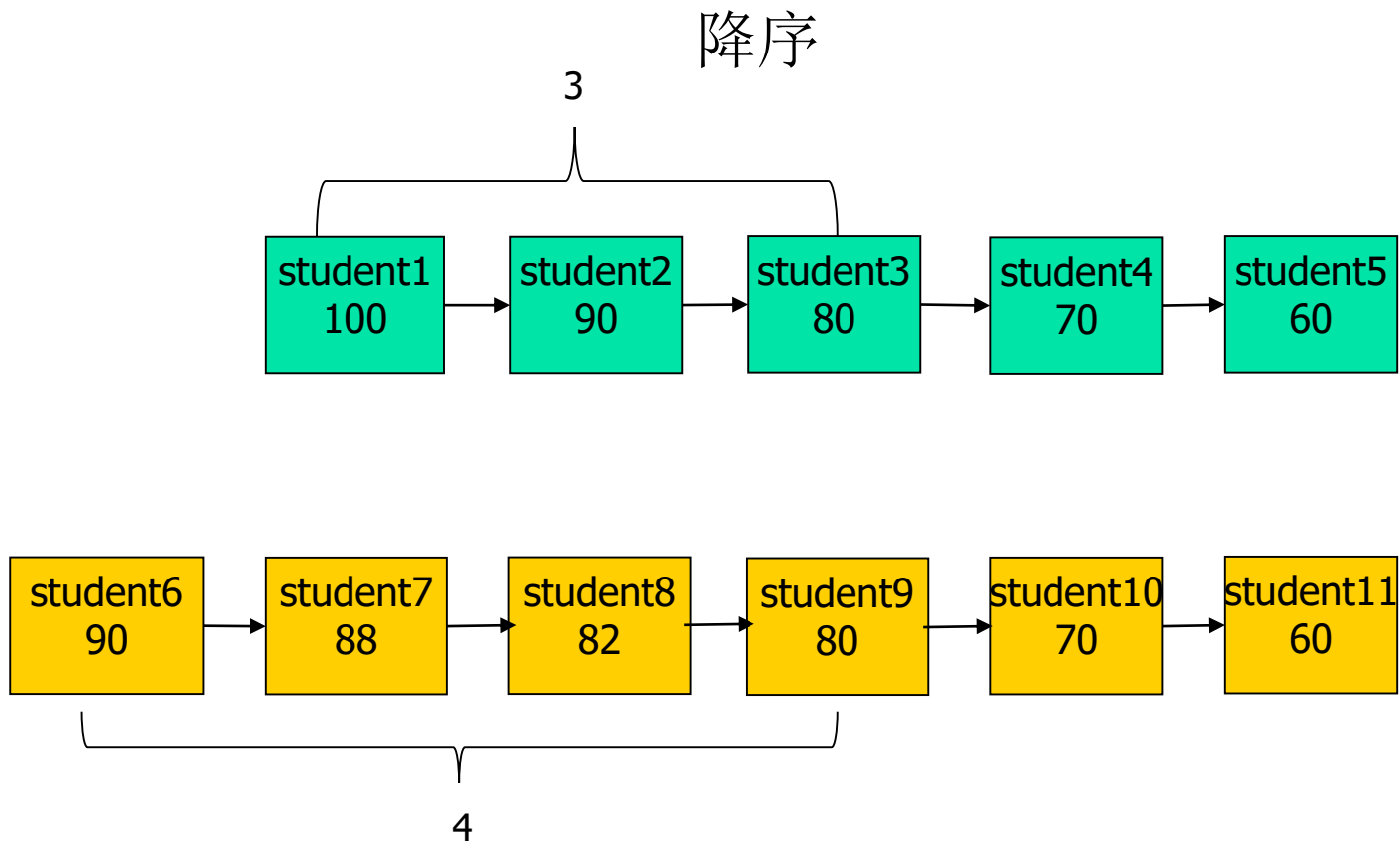
## (1) :建立链表

头插法，升序



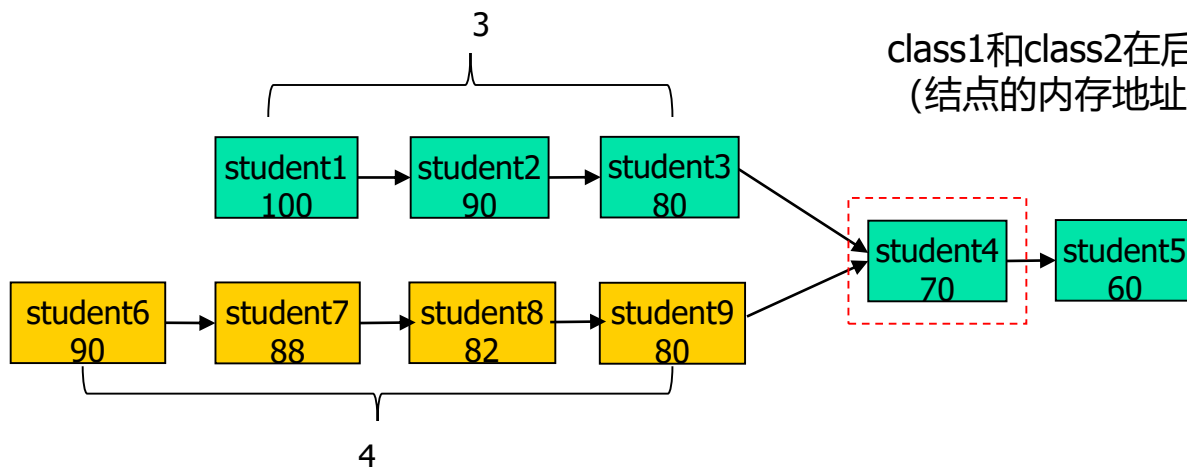
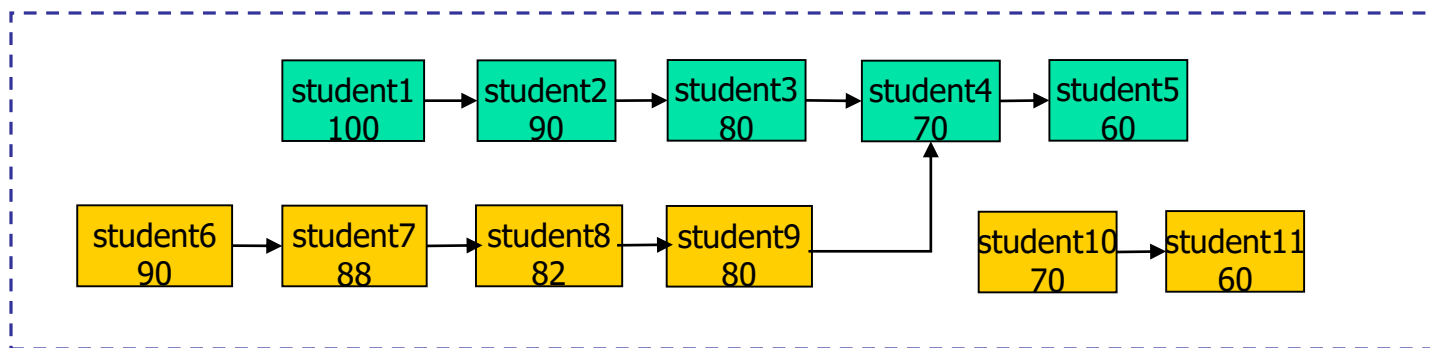
# 实验内容

## (2) :反转链表



# 实验内容

## (3) :链表交点



class1和class2在后一部分完全重合  
(结点的内存地址相同)

# 实验要求

- 使用链表实现上述需求，参考代码中是**非空头结点**；
- 编程语言：C（C++和其他编程语言目前平台不支持）
- 建表时要求成绩升序存储；
- 测试用例包含两个班级的成绩，每个班级的成绩存在一个链表中；
- main函数及部分函数已给出，请不要改动，你只需完成其他函数；
- 寻找链表交点时，请**不要使用分数进行判断**，且尽可能使用较少的比较次数；
- 请到实验平台<http://10.249.176.82:9000/>完成题目1001。

# 栈和队列的应用

## 1002: 用数组实现栈的基本操作

**Push** 将元素 x 压入栈顶

**Pop** 移除并返回栈顶元素

**GetTop** 返回栈顶元素

**StackEmpty** 如果栈是空的，返回 true；  
否则，返回 false

操作数	对应操作
-1	Exit（已实现）
0	Push
1	Pop
2	GetTop
3	StackEmpty

# 实验内容

## (1) :Push 将元素压入栈顶

### 【输入示例】

0 7 0 1 2 3 4 5 6 /\*入栈7次, 数据为0、1、2、3、4、5、6\*/

### 【输出示例】

Stack: 6 5 4 3 2 1 0 /\*栈的全部元素\*/

操作数	对应操作
-1	Exit (已实现)
0	Push
1	Pop
2	GetTop
3	StackEmpty

- 单次输入一行为一个独立的指令，一个或几个整型参数，用空格隔开。
- 第一个参数是指令，0-3对应上述四种操作，-1表示退出程序。
- 第二个参数是操作次数。
- 后续参数是数据。

# 实验内容

## (2) :Pop 获取并移除栈顶元素

### 【输入示例】

```
1 2    /*将栈顶元素移除并获取*/
```

### 【输出示例】

```
Pop: 6    /*依次弹出栈顶元素6、5*/  
Stack: 5 4 3 2 1 0  
Pop: 5  
Stack: 4 3 2 1 0
```

弹出失败

```
Pop failed    /*失败了几次就会打印几次该信息*/
```

操作数	对应操作
-1	Exit（已实现）
0	Push
1	Pop
2	GetTop
3	StackEmpty

# 实验内容

## (3) :GetTop 获取栈顶元素

### 【输入示例】

```
2  /*获取栈顶元素，而不弹出*/
```

### 【输出示例】

```
GetTop: 4  
Stack: 4 3 2 1 0
```

弹出失败

```
GetTop failed
```

操作数	对应操作
-1	Exit（已实现）
0	Push
1	Pop
2	GetTop
3	StackEmpty



# 实验内容

## (4) :StackEmpty 判断栈是否为空

### 【输入示例】

```
3    /*判断栈是否为空*/
```

### 【输出示例】

```
The Stack is Empty    /*空*/
```

```
The Stack is not Empty    /*非空*/
```

```
Stack: 4 3 2 1 0
```

操作数	对应操作
-1	Exit（已实现）
0	Push
1	Pop
2	GetTop
3	StackEmpty

# 实验要求



- 基于数组实现上述需求;
- 编程语言: C (C++和其他编程语言目前平台不支持)
- main函数及部分函数已给出, 请不要改动, 你只需完成其他函数。  
你也可以不使用模板代码, 但请注意代码规范以及输入输出格式;
- 请到实验平台<http://10.249.176.82:9000/>完成题目1002。

# 实验内容

## 1003: 用两个栈实现队列的基本操作

**EnQueue** 将元素  $x$  推到队列的末尾

**DeQueue** 从队列的开头移除并返回元素

**GetHead** 返回队列开头的元素

**QueueEmpty** 如果队列为空，返回 true；否则，返回 false

并实现辅助接口 **QueueToArray** 将队列中元素按照从头到尾的顺序写到数组中

**思考：**怎么使用两个栈能使**连续的EnQueue/DeQueue**操作效率最高？

操作数	对应操作
-1	Exit（已实现）
4	EnQueue
5	DeQueue
6	GetHead
7	QueueEmpty

# 实验内容

## (1) :EnQueue 将元素插入到队列的尾

### 【输入示例】

4 7 0 1 2 3 4 5 6 /\*入队7次，数据为0、1、2、3、4、5、6\*/

### 【输出示例】

Queue: 0 1 2 3 4 5 6 /\*队列的全部元素（队头到队尾，用空格隔开）\*/

操作数	对应操作
-1	Exit（已实现）
4	EnQueue
5	DeQueue
6	GetHead
7	QueueEmpty

- 单次输入一行为一个独立的指令，一个或几个整型参数，用空格隔开。
- 第一个参数是指令，4-7对应上述四种操作，-1表示退出程序。
- 第二个参数是操作次数。
- 后续参数是数据。

# 实验内容

## (2) :DeQueue获取并移除队列头元素

### 【输入示例】

```
5 3  /*弹出队头元素并获取*/
```

### 【输出示例】

```
DeQueue: 0  /*弹出的队头元素和队列的剩下元素（队头到队尾，用空格隔开）*/  
Queue: 1 2 3 4 5 6  
DeQueue: 1  
Queue: 2 3 4 5 6  
DeQueue: 2  
Queue: 3 4 5 6
```

弹出失败

```
DeQueue failed  /*失败了几次就会打印几次该信息*/
```

操作数	对应操作
-1	Exit（已实现）
4	EnQueue
5	DeQueue
6	GetHead
7	QueueEmpty

# 实验内容

## (3) :GetHead 获取队列开头的元素

### 【输入示例】

```
6  /*获取队头元素*/
```

### 【输出示例】

```
GetHead: 3  /*队头元素和队列的剩下元素（队头到队尾，用空格隔开）*/  
Queue: 3 4 5 6
```

获取失败

```
GetHead failed
```

操作数	对应操作
-1	Exit（已实现）
4	EnQueue
5	DeQueue
6	GetHead
7	QueueEmpty

# 实验内容

## (4) :QueueEmpty判断队列是否为空

### 【输入示例】

```
7  /*判断队列是否为空*/
```

### 【输出示例】

```
The Queue is Empty  /*队列空*/
```

```
The Queue is not Empty  /*队列非空*/
```

```
Queue: 3 4 5 6
```

操作数	对应操作
-1	Exit (已实现)
4	EnQueue
5	DeQueue
6	GetHead
7	QueueEmpty

# 实验要求



- 可基于题目1002数组实现的栈来实现本题的队列。也可通过其他方法实现的栈来实现本题的队列。
- 编程语言：C（C++和其他编程语言目前平台不支持）
- main函数及部分函数已给出，请不要改动，你只需完成其他函数。你也可以不使用模板代码，但请注意代码规范以及输入输出格式。
- 请到实验平台<http://10.249.176.82:9000/>完成题目1003。



# 实验一评分标准

---

- **总分100分**

- 体测成绩 (40分)
- 栈的基本操作 (30分)
- 队列的基本操作 (30分)



请同学们开始实验