电子科技大学信息与通信工程学院

**实 验 指 导 书**

（实验）课程名称 软件技术实验

**电子科技大学教务处制表**

说明

本课程实验分为三种形式：命令行式、基于成绩表的图形化式和调测实验案例式。这三种形式的实验目标一致，不同的是：

1）命令行式任务的输入一般是利用scanf从键盘读入，输出依靠printf，几乎所有的代码工作都需要实验者自行完成，需要付出更多的精力去处理输入、输出的形式。并且受此限制，初学者几乎很难完成较为复杂的目标，所学与所用之间很大的距离感可能导致学习兴趣的下降。

2）图形化式是在“基于成绩表的软件技术实验”平台上完成的任务。她有一个友好的图形化交互界面。按照MVC的设计理念，输入、输出通过图形化界面完成，形式灵活且不需要实验者投入精力，实验者可以更专注于基本数据结构和操作在复杂应用中的作用，更深刻地理解所学的基本结构和基本操作是能够直接应用的。

3）调测案例则以案例的方法，通过发现故障、解决故障，让学生有更多机会了解和接触典型的错误，同时掌握调测工具和调测方法。

学生可以在命令行式和图形化式之中选择其中的一种完成，因为实验的基本内容是一致的。调测案例则最好配合各相关基本结构的实验逐步完成，比如在实验1的预习或复习过程中完成调测案例1——都是围绕顺序表。

在选择以图形化式完成实验之前，请先按附录1的要求完成基本技能训练。这将帮助实验者更好地掌握在“基于成绩表的软件技术实验”平台上完成所需任务的方法和技巧。

提醒：

1）VC6.0下编译和调试程序，在运行VC6.0时请右键选择管理员模式运行。

目录

[实验1、 顺序表 2](#_Toc525601765)

[实验2、 链表 8](#_Toc525601766)

[实验3、 栈和队列 16](#_Toc525601767)

[实验4、 二叉树 20](#_Toc525601768)

[实验5、 检索和排序 27](#_Toc525601769)

[实验6、 复杂应用设计 35](#_Toc525601770)

[实验7、 面向多线程的缓冲池设计 49](#_Toc525601771)

[实验8、 基本数据结构程序调测 62](#_Toc525601772)

[附录1 基于成绩表的图形化软件平台使用基本技能训练 69](#_Toc525601773)

# 顺序表

**【设计目的】**

1. 掌握顺序表的创建与查询算法的设计和程序实现方法
2. 掌握顺序表的插入、按序插入、删除操作的算法设计和程序实现方法

成绩管理表一般都采用线性表的方式抽象和实现。

**【设计要求】**

一、基本题：

命令行式：

Ex1-1：

1）首先创建一个顺序表：从键盘读入一组整数（长度小于等于20），按输入顺序放入顺序表，输入以－1结束（注意－1不放到顺序表内）；将创建好的顺序表元素依次输出到屏幕上。

2）在已创建好的顺序表中插入一个元素：从键盘读入需插入的元素值和插入位置，调用插入函数完成插入操作；然后将顺序表元素依次输出到屏幕上。

3）在已创建好的顺序表中删除一个元素：从键盘读入欲删除的元素位置（序号），调用删除函数完成删除操作；然后将顺序表元素依次输出到屏幕上。

图形化式：

在table.cpp文件中实现顺序表的创建、查询、插入、按序插入和删除算法的程序模块。最终在实验者展示区展示正确的结果。

二、扩展题

Ex1-2：

（1）在已经创建的元素为整数的顺序表中，从键盘输入一个整数，按照元素值大小，将新输入的元素插入到顺序表中

注意该题中没有提供插入元素的具体位置，要根据元素值大小寻找合适的位置。

（2）输入一组数，建立顺序表，包括多个负数，编写算法删除其中所有的负数

本题的特点是优化：怎样在一轮循环中删除，而不是每删除一个，都把后续元素搬移一次。

三、选作题

Ex1-3：

输入一组数，建立顺序表，编写算法将整个表的元素全部反序存放——即第一个元素放到最后……。

本题的特点在优化，怎样在原表上，直接实现反序——即不额外使用一张新表。

**【设计原理】**

1. 顺序表结构和算法

顺序表是在计算机内存中以数组的形式保存的线性表，是指用一组地址连续的存储单元依次存储数据元素的线性结构。线性表采用顺序存储的方式存储就称之为顺序表。顺序表是将表中的结点依次存放在计算机内存中一组地址连续的存储单元中。

顺序表的存储特点是：只要确定了起始位置，表中任一元素的地址都通过下列公式得到：LOC（ai）=LOC（a1）+（i-1）\*L 1≤i≤n 其中，L是元素占用[存储单元](http://baike.baidu.com/view/1223079.htm)的长度。

顺序表操作的关键点是元素的搬移过程和方向。

插入操作需要从后向前逐个将元素向后移动。稍不留心就会犯以下错误：

**for( j = i; j < table.length; j++){**

**table.data[ j+1 ] = table.data[ j ];**

**}**

**【图形化式设计步骤】**

注意：在table.cpp文件中完成本部分实验。

1. 设计并实现顺序表创建算法

【设计】

1）Function name: init\_table()

Descriptions : 初始化顺序表

顺序表利用数组作为基础，其特点是需要事先获得全部元素空间，因此本函数的主要功能就是向系统申请足够的空间作为顺序表的存储空间。涉及的系统函数为：malloc()。此外，良好的习惯是将空间内的各项数据进行适当的初始化，比如清零。

2）Function name: add\_table()

Descriptions : 将指定元素放入到顺序表的末尾

【调测步骤】

点击****按钮进行调测，直到实验者展示区内容与例程展示区内容相同。

1. 设计并实现顺序表查询算法

【设计】

Function name: get\_table()

Descriptions :查询顺序表中第i个元素

Input:

table\_t \* table; 顺序表指针

int index; 查询位置，即第i个元素

Output:

element\_t \* elem; 元素域指针，用来存放被查询到的元素内容，

注意，需要将元素全部内容拷贝到该指针所记录的空间中，即，使用memcpy()。

【调测步骤】

填写需要查询的元素位置信息，点击“查询”按钮：



查询算法的输出结果会显示在控制区，检查查询结果的正确性。



1. 设计并实现顺序表插入操作

【设计】

1）Function name: insert\_table()

Descriptions : 将指定元素插入到顺序表的指定位置之前

Input:

table\_t \* table; 顺序表指针

element\_t data; 待放入的元素

int location; 插入位置，语义是：第X个元素前，，当location大于链表元素总数时，该元素将插入到表尾。

Output:

table\_t \* table; 插入新元素后的顺序表指针

2）Function name: insert\_table\_by\_order()

Descriptions : 将指定元素按照学号从小到大顺序插入到顺序表中

Input:

table\_t \* table; 顺序表指针

element\_t data; 待放入的元素

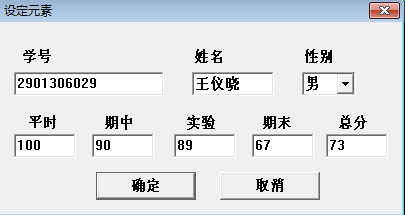
Output:

table\_t \* table; 插入新元素后的顺序表指针

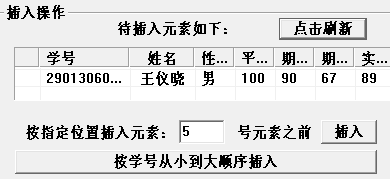
【调测步骤】

1）随机生成待插入元素

点击“点击刷新按钮，弹出窗口中将显示随机生成的一个数据元素，此时可以修改该元素的值以满足某些特殊测试的需要。点击“确定”后，该元素将作为插入函数的输入参数



2）填写插入的位置要求，点击插入按钮



检查插入结果，修改程序直到获得正确结果

3）再随机生成一个插入元素

4）点击“按学号从小到大顺序插入”按钮，调测按序插入算法：insert\_table\_by\_order()。

1. 设计并实现顺序表删除操作

【设计】

1）Function name: delete\_table()

Descriptions : 删除顺序表中指定姓名作为关键字的元素

Input:

table\_t \* table; 顺序表指针

char \* name; 以该姓名为关键字的元素将被删除

Output:

table\_t \* table; 删除指定元素后的顺序表指针

2）Function name: delete\_table\_below()

Descriptions : 删除顺序表中总分小于某个指定值的所有元素，本算法的特点是希望一趟能在顺序表中删除多个元素

Input:

table\_t \* table; 顺序表指针

int x; 删除范围，即被删除的元素总分小于这个值

Output:

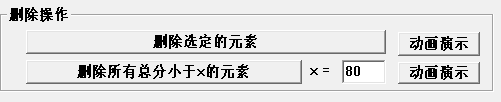
table\_t \* table; 删除元素后的顺序表指针

【调测步骤】

1. 在数据展示区选择某个表项准备删除

在表项上点击，即表示被选中

1. 点击“删除选定的元素“按钮



1. 检查输出结果，并调测使之正确
2. 设定x值的范围后，点击“删除所有总分小于x的元素”按钮，触发delete\_table\_below（）函数执行，调测使其运算的结果正确

**【思考题】**

1. 顺序表的插入和删除操作为什么要大量搬移元素？
2. 要想顺序表的操作不过界，应该采用什么方法来防止过界

# 链表

**【设计目的】**

1. 掌握链表的创建与遍历查询算法的设计和程序实现方法
2. 掌握链表的插入、按序插入、删除操作的算法设计和程序实现方法

**【设计要求】**

一、基本题

【命令行式】

Ex2-1:

1）首先创建一个单链表：从键盘读入五个整数，按输入顺序形成单链表。将创建好的链表元素依次输出到屏幕上。

2）在已创建好的链表中插入一个元素：从键盘读入元素值和插入位置，调用插入函数完成插入操作。然后将链表元素依次输出到屏幕上。

3）在已创建好的链表中删除一个元素：从键盘读入欲删除的元素位置（序号），调用删除函数完成删除操作。然后将链表元素依次输出到屏幕上。

【图形化式】

在link.cpp文件中实现链表的创建、查询、插入、按序插入和删除算法的程序模块。最终在实验者展示区展示正确的结果。

二、扩展题

Ex2-2:

1）创建一个单链表，其数据元素为整数，从键盘输入，输入0结束（注意0不放到链表内）；

2）从键盘任意输入一个整数，在单链表中查询该数，如果单链表中已经存在这个数，就调用删除函数，删除该元素所在结点，并将单链表在删除前后的数据元素依次输出到屏幕上；

如果单链表中不存在这个数，就调用插入函数，将这个数插入到单链表尾，并将单链表在插入前后的数据元素依次输出到屏幕上。

3）在已经创建的元素为整数的链表中，从键盘输入一个整数，按照元素值大小，将新输入的元素插入到链表中。

三、选做题

Ex2-3:

1）删除单链表中全部的负数

2）创建一个双向链表，按照冒泡排序的思路对这个双向链表进行排序，打印排序结果。注意，本算法在交换元素时是将链点整个交换而不是将链点中的元素值交换。

**【设计原理】**

1. 链表结构和算法

链表是一种物理[存储单元](http://baike.baidu.com/view/1223079.htm)上非连续、非顺序的[存储结构](http://baike.baidu.com/view/2820182.htm)，[数据元素](http://baike.baidu.com/view/38785.htm)的逻辑顺序是通过链表中的[指针](http://baike.baidu.com/view/159417.htm)链接次序实现的。链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。每个结点包括两个部分：一个是存储[数据元素](http://baike.baidu.com/view/38785.htm)的数据域，另一个是存储下一个结点地址的[指针](http://baike.baidu.com/view/159417.htm)域。

线性表的链式存储表示的特点是用一组任意的存储单元存储线性表的数据元素（这组存储单元可以是连续的，也可以是不连续的）。因此，为了表示每个数据元素与其直接后继数据元素之间的逻辑关系，除了存储其本身的信息之外，还需存储一个指示其直接后继的信息（即直接后继的存储位置）。由这两部分信息组成一个"结点"（如概述旁的图所示），表示线性表中一个数据元素。

链表，优点在于对存储空间的充分利用，插入和删除元素的操作不需要搬移元素，可以节约大量时间。其最主要的缺点就是要找一个元素，必须要从头开始找起，十分麻烦。

如何描述我们在链表上“逐个去找”的动作？

1）先找到链表首结点的地址

2）通过“地址”，找到链点

3）在链点中找到后继元素的“地址”

4）记录这个地址，回到2

while( counter < i -1 && p != NULL){

counter = counter + 1;

p = p->next;

}

链表的插入和删除操作主要通过修改链点指针完成。在操作中要特别数据操作的顺序，例如如果先断开链表关系，再接入新链点时，就会导致链表上的元素“丢失”，例如下面的程序段：

p->next = new\_node;

new\_node->next = p->next;

从链表上取下的元素应该输出给调用者或者释放其空间，总之不能仅仅将其从链表上“取下”。

**【图形化式设计步骤】**

注意，在link.cpp文件中完成本部分实验。

1. 设计并实现链表创建和查询操作

**【设计】**

1）Function name: init\_link()

Descriptions :初始化链表，链表的特点是利用内存离散空间存放各个元素，因此不必事先为链表准备足够大的内存空间，只是在需要插入新元素时，才申请必要的链点空间。因此在初始化链表时，只需要为链表申请少量结构所需的空间即可。关键是要将其中的头指针设置为空

Input: NONE

Output: NONE

return: 类型：link\_t \*，返回链表的结构指针

2）unction name: add\_link()

Descriptions : 将指定元素放入到链表的末尾

Input:

link\_t \* link; 链表结构指针

element\_t data; 待放入的元素

Output:

link\_t \* link; 添加新元素后的链表指针

return: 类型：int; 为-1表示放入失败，一般是因为申请不到链点空间，为0表示正确放入

3）Function name: get\_link()

Descriptions : 查询链表，查询链表中第i个元素

Input:

link\_t \* link; 顺序表指针

int index; 查询位置，即第i个元素

Output:

element\_t \* elem; 元素域指针，用来存放被查询到的元素内容，

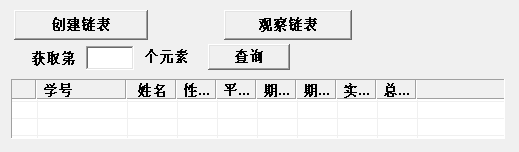
（注意，需要将元素全部内容拷贝到该指针所记录的空间中，即，使用memcpy()）

return: 类型：int，返回查询是否成功，为0表示找到指定元素，为-1表示没有找到，一般是因为index指示的位置超出了顺序表的范围

**【调测步骤】**

1. 切换控制区

点击导航区链表/创建



1. 点击“创建链表”按钮

此时将触发init\_link()执行，成功后，实验平台将逐个将成绩表中的元素通过add\_link（）放入到链表尾部。插入完毕后，实验平台将对比例程结果与实验者的结果对比，帮助实验者排错。

1. 观察链表形态，检错、排错

点击“观察链表”，将出现以链接方式展示的链表，点击指针变量，就能进一步打开下一个链点。帮助实验者体验链表的组织结构和特征。



图中左侧是例程链表，右侧是学生的链表，点击head和next就能逐个打开后继链点，每个指针上还标记了指针记录的地址。错误的内容将用黄色标记，帮助程序员发现错误。

1. 查询元素

填写待查询的元素位置后，点击“查询”按钮，实验平台会触发get\_link()，查询到的结果将显示到控制区。

1. 设计并实现链表插入操作

**【设计】**

1）Function name: insert\_link()

Descriptions : 将指定元素插入到链表的指定位置之前

Input:

link\_t \* link; 链表结构指针

element\_t data; 待放入的元素

int location; 插入位置，语义是：第X个元素前，当location大于链表元素总数时，该元素将插入到表尾。

Output:

link\_t \* link; 插入新元素后的链表结构指针

return: 类型：int; 为-1表示插入失败，一般是因为没有申请到空间而无法生成链点为0表示正确插入。

2）Function name: insert\_link\_by\_order()

Descriptions : 将指定元素按照学号从小到大顺序插入到链表中

Input:

link\_t \* link; 链表指针

element\_t data; 待放入的元素

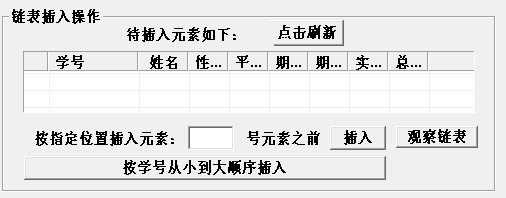
Output:

link\_t \* link; 插入新元素后的链表指针

return: 类型：int; 为-1表示插入失败，一般是因为没有申请到空间而无法生成链点,为0表示正确插入

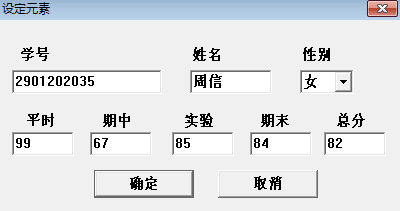
**【调测步骤】**

1. 切换控制区到链表插入



1. 随机生成一个新元素

点击“点击刷新按钮”



1. 按指定位置插入

填写插入位置后，点击“插入”按钮，此时将触发insert\_link（）,可以通过“观察链表”来查找出错的原因。

1. 按序插入

点击“按学号从小到大顺序插入”按钮，此时触发insert\_link\_by\_order()

1. 设计并实现链表删除操作

**【设计】**

1）Function name: delete\_link()

Descriptions : 删除链表中指定姓名作为关键字的元素

Input:

link\_t \* link; 链表指针

char \* name; 以该姓名为关键字的元素将被删除，其链点空间也需要被释放

Output:

link\_t \* link; 删除指定元素后的链表指针

return: 类型：int; 为-1表示删除失败，一般是因为在链表中没有找到指定元素，为0表示正确删除

2）Function name: delete\_link\_below()

Descriptions : 删除链表中总分小于某个指定值的所有元素，本算法的特点是希望一趟能在链表中删除多个元素

Input:

link\_t \* link; 链表指针

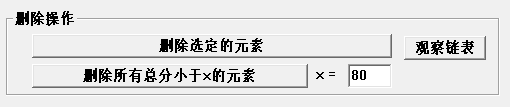
int x; 删除范围，即被删除的元素总分小于这个值

Output:

link\_t \* link; 删除元素后的链表结构指针

return: 类型：void;

**【调测步骤】**



1. 在数据展示区选择表项
2. 点击“删除选定的元素”按钮，触发delete\_link（）
3. 填写x值，点击“删除所有总分小于x的元素”，触发delete\_link\_below();

**【思考题】**

1. 链表的按序插入算法中，为什么不能从链表尾部开始逐个访问
2. 从链表上取下的元素，应该如何处理？

# 栈和队列

**【设计目的】**

1）掌握顺序栈和链栈的创建与入栈、出栈算法的设计和程序实现方法

2）掌握循环队列的入队、出队操作的算法设计和程序实现方法

**【设计要求】**

一、基本题

Ex3-1:【链栈】

1）链栈结点类型定义为：

typedef struct node

{

int data;

struct node \*next;

}node\_type;

2）编写进栈函数push

3）编写出栈函数pop

4）编写main函数，首先建立一空链栈；

调用进栈函数，将从键盘输入的数据元素逐个进栈，输入0结束；显示进栈后的数据元素；

调用两次出栈函数，显示出栈后的数据元素。

Ex3-2:【循环队列】

1）循环队列类型定义为：

#define N 20

typedef struct

{ int data[N];

int front, rear;

}queue\_type;

2）编写循环队列出队函数dequeue

3）编写循环队列入队函数enqueue

4）编写函数：void aa(queue\_type \*q)；

其功能为，调用出队函数把队列q中的元素一一出队，如果是负数直接抛弃；如果是正数，则调用入队函数，插入到q的队尾。

5）编写main函数，首先建立一个队列，其中的数据元素为：{2, 3, -4, 6, -5, 8, -9, 7, -10, 20}；然后调用aa函数，并将aa函数调用前后队列的数据元素分别输出到屏幕上。

二、扩展题

Ex3-3:【栈】

1）两个栈共用一个数组空间，它们的栈底分别在数组两端，栈顶相向而行。编写入栈和出栈函数，实现两个栈元素分别的（但共用）入栈和出栈。

2）main中函数完成以下测试：

a、能否在共用空间上实现两个独立的栈：即能否向两个栈分别输入元素；能否分别从两个栈取出元素，每个栈取出的元素的顺序符合各自栈的特点

b、能否在共用空间用满时，及时制止新的入栈行为。

例如：

假设数组大小为6，main函数实现以下动作，向栈1接连入栈4个元素后，向栈2入栈2个元素致栈满，再向栈2输入一个元素，将报错。接着从栈1出栈1个元素，再向栈2入栈，就会成功。最后，两个栈分别出空，观察输出顺序是否满足栈的特点。

提示：

struct dhStack{

int top1;

int top2;

int data[MAXNUM];

};

int pop( struct dhStack \* s, int which); which用来指明操作哪个栈

void push(struct dhStack \* s, int which, int x);

Ex3-4:【队列】

1）实现一种扩展的循环队列，使得全部的数组空间都能使用，基本思路是当传统循环队列放满时：即

(rear+1)%MAXNUM ==front 为真时，

可以再入队一个元素，接着rear = (rear+1)%MAXNUM后就会与front相等，此时将另外一个变量flag设置为1，表示此时的rear==front不是为空，而是满。否则flag为0时，如果出现rear==front，则表示队列为空。

2）main()函数实现以下测试：

a、能否实现全部“装满”，即装入元素个数为MAXNUM

b、能否按照循环队列那样绕着存放空间循环存放。

c、能否在装满后，拒绝再装。

d、能否在装满后，不会变成“空”的——即可以还可正常出队。

e、能否在全部出空后，不会变成“满”的——即可还可正常入队。

例如：

假设循环队列最大空间为5，main()函数实现以下动作，接连成功入队5个元素，入队第6个元素时，报错。接着出队3个元素，入队3个元素，均成功。再入队1个，报错。继续连续成功出队6个元素，出队第7个时报错。最后，再成功入队2个元素。

提示：

struct ExtQueue{

int front;

int rear;

int flag;

int data[MAXNUM];

};

三、选做题

Ex3-5:

利用栈实现算术四则运算，即录入一个包含加减乘除运算的多项式，计算出结果。暂不考虑“括号”。

本题算法难点在如何利用栈，驱动运算过程。

本题程序难点在从输入的一个形如“3+4×16-4/3”的字符串中依次提取出各个数字和运算符。

ex3-6

计算并输出二项展开式（a+b）n的各项系数，即求一个杨辉三角形的最下面一层所有元素。

本题算法难点在利用杨辉三角形计算原理，不断利用队列在上一层元素的基础上，求出下一层元素。

**【设计原理】**

1. 顺序栈

栈是限定尽在表尾进行插入或者删除操作的线性表。因此，对栈来说，表尾端有其特殊含义，称为栈顶，相应地，表头端称为栈底。不含元素的空表称为空栈。栈又称为后进先出的线性表。

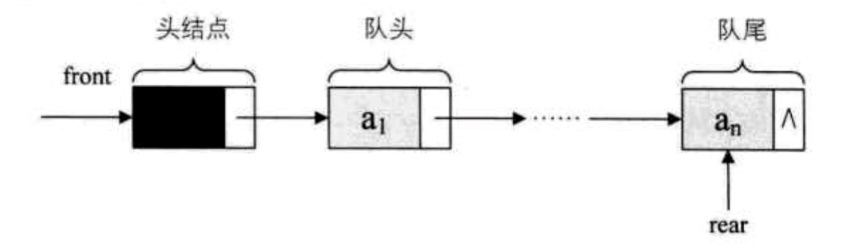
顺序栈，即栈的顺序存储结构是利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素，同时附设指针top只是栈顶元素在顺序栈中的位置。这里以top=-1表示空栈。

1. 链栈

链栈可以通过单链表的方式来实现，使用链式栈的优点在于它能够克服用数组实现的顺序栈空间利用率不高的特点，但是需要为每个栈元素分配额外的指针空间用来存放指针域。

1. 链队列

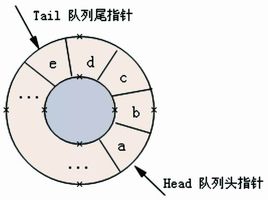
将对头指针front指向链队列的头结点，队尾指针rear指向终端结点。



空队列时，头指针front和尾指针rear都指向头结点。

1. 循环队列

为充分利用向量空间，克服"[假溢出](https://baike.baidu.com/item/%E5%81%87%E6%BA%A2%E5%87%BA)"现象的方法是：将向量空间想象为一个首尾相接的圆环，并称这种向量为循环向量。存储在其中的队列称为循环队列（Circular Queue）。这种循环队列可以以[单链表](https://baike.baidu.com/item/%E5%8D%95%E9%93%BE%E8%A1%A8/3228368)的方式来在实际编程应用中来实现。



**【设计步骤】**

按设计要求逐个完成设计任务即可**:**

1. 链栈
2. 循环队列
3. 共享栈
4. 扩展循环队列

**【思考题】**

1. 如果链队列的尾链点的next指针指向表首，形成的循环链表，是否能称为循环队列，为什么？
2. 假设循环队列空间size为N，已知front和rear，循环队列的元素个数是多少？

# 二叉树

**【设计目的】**

1. 掌握二叉树的结构特点
2. 掌握二叉排序树的创建算法和程序实现
3. 掌握二叉树的遍历算法

**【设计要求】**

一、基本题

【命令行式】

Ex4-1:

1）二叉树结点类型定义为：

typedef struct bnode

{

int data;

struct bnode \*lchild, \*rchild;

}bnode\_type;

2）编写二叉树的创建函数，可以是排序二叉树的创建思路（见教材），或者以先序遍历为框架。

3）编写中序遍历函数；

4）编写后序遍历函数；

5）编写先序遍历函数；

Ex4-2:

编写main()函数，先调用create函数，建立一颗二叉排序树；然后分别调用中序、后序、先序遍历函数，将二叉树的先序、中序和后序遍历序列输出到屏幕上。

【图形化式】

在tree.cpp文件中实现二叉排序树的创建（含插入）、遍历算法的程序模块。最终在实验者展示区展示正确的结果。

二、扩展题

Ex4-3:

1）实现二叉排序树的插入函数

2）main函数中，输入一组无序数，调用二叉排序树插入算法，将元素放入二叉树中

3）中序遍历这颗二叉树，得到排序顺序。

4）编写函数求一颗二叉树的深度

5）求一颗二叉排序树的排序的反序结果，即将二叉排序树每个节点的左右子树交换，然后中序遍历之。

三、选作题

Ex4-4:

1）输入一组数（权值），编写建立哈夫曼树的函数

2）以先序遍历为框架，输出每个权值对应的二进制编码

**【设计原理】**

1. 二叉树

二叉树是每个节点最多有两个子树的有序树。通常子树被称作“左子树”（left subtree）和“右子树”（right subtree）。二叉树常被用于实现二叉查找树和二叉堆。二叉树的每个结点至多只有二棵子树(不存在出度大于2的结点)，二叉树的子树有左右之分，次序不能颠倒。二叉树的第i层至多有2的 i -1次方个结点；深度为k的二叉树至多有2^(k) -1个结点。

二叉树很像一株倒悬着的树，从树根到大分枝、小分枝、直到叶子把数据联系起来，这种数据结构就叫做树结构，简称树。树中每个分叉点称为结点，起始结点称为树根，任意两个结点间的连接关系称为树枝，结点下面不再有分枝称为树叶。结点的前趋结点称为该结点的"双亲"，结点的后趋结点称为该结点的"子女"或"孩子"，同一结点的"子女"之间互称"兄弟"。

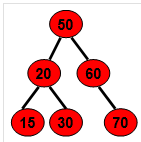
1. 二叉排序树

二叉排序树（Binary Sort Tree）又称二叉查找树（Binary Search Tree），亦称二叉搜索树。 它或者是一棵空树；或者是具有下列性质的二叉树：

（1）若左子树不空，则左子树上所有结点的值均小于它的根结点的值；

（2）若右子树不空，则右子树上所有结点的值均大于它的根结点的值；

（3）左、右子树也分别为二叉排序树。



二叉排序树是一种动态树表。其特点是：树的结构通常不是一次生成的，而是在查找过程中，当树中不存在关键字等于给定值的节点时再进行插入。新插入的结点一定是一个新添加的叶子节点，并且是查找不成功时查找路径上访问的最后一个结点的左孩子或右孩子结点。

**【图片式设计步骤】**

注意：在tree.cpp文件中完成本部分实验。

1. 设计并实现二叉排序树创建算法

【设计】

1）Function name: init\_tree()

Descriptions : 初始化二叉树

二叉树的是非线性结构，大多数情况下采用链接存储，采用动态申请内存的方式在需要的时候为链点申请空间。二叉树结构中最重要的是根节点指针，因此在初始化时，除了要申请二叉树结构空间外，还应将其中的根节点指针设置为空

return: 类型：tree\_t \*，返回链表的结构指针

2）Function name: insert\_BS\_tree()

Descriptions : 向二叉排序树按递增排序规则插入一个元素，以元素中的数据项——总分作为排序码。

Input:

tree\_t \* tree; 二叉树指针

element data; 新元素

Output: NONE

return: 类型：int，0，表示成功插入；-1，表示插入失败，一般是因为没有足够的内存。

3）Function name: get\_element\_tree()

Descriptions : 在二叉树中，查询指定学号的元素，获得该元素的链点指针，以备进一步访问所需查询算法需要对二叉树进行遍历，可采用先序（根）遍历算法的框架,

Input:

tnode\_t \* root; 二叉树指针

double stuID; 学号

Output: NONE

return: 类型：tnode\_t \*; 返回具有指定学号的链点指针，如果没找到则返回值为NULL

【调测步骤】

1. 创建二叉树

点击导航区二叉树/创建二叉排序树

在控制区点击创建排序二叉树



将首先触发init\_tree（），初始化树结构，该树根节点为空。

实验平台弹出二叉树的展示窗口。如右图所示

实验者可以选择逐个输入或连续输入的方式，由“逻辑控制模块”从成绩表中逐个取出元素，触发insert\_BS\_tree（）函数，将输入的元素按照二叉排序树的规则插入到二叉树中。“逐个输入”每点击一次就输入一个元素。“连续输入”，则逻辑控制模块会连续触发insert\_BS\_tree。两种方式对于insert\_BS\_tree的功能要求没有影响，仅仅是实验者的观察方式不同。

在二叉排序树的展示窗口内，同样也安排了例程结果与实验者算法结果的对比，用黄色标识出错误的元素，提醒实验者检错、排错。



1. 在排序二叉树上查询

在控制区，填入待查的学生学号，点击“查询按钮”，将触发get\_element\_tree,执行遍历操作，找到指定学号的元素。

查询后的结果将被填写在控制区，以供检查。

1. 设计并实现二叉排序树的遍历算法

中序遍历二叉树将能得到递增排序后的结果。

【设计】

注意，遍历算法中“访问节点”的动作通过将该元素显示到实验者展示区来体现，这时需要调用ShowElement(node->data)来完成

1）Function name: inorder\_tree()

Descriptions : 中序（根）遍历一棵二叉树,在访问节点内容时，调用ui.h中的ShowElement()来将元素内容显示在窗口内以供检查

Input:

tnode\_t \* root; 二叉树指针

2）Function name: preorder\_tree()

Descriptions : 先序（根）遍历一棵二叉树,在访问节点内容时，调用ui.h中的ShowElement()来将元素内容显示在窗口内以供检查

Input:

tnode\_t \* root; 二叉树指针

3）Descriptions : 后序（根）遍历一棵二叉树,在访问节点内容时，调用ui.h中的ShowElement()来将元素内容显示在窗口内以供检查

Input:

tnode\_t \* root; 二叉树指针

Output: NONE

return: 类型：void

实验平台向实验者提供了先序、中序和后序算法的实现和调测平台。

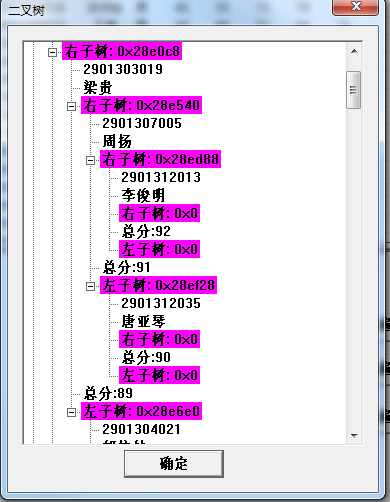
【调测步骤】

1）打开控制区

点击导航区二叉树/遍历。

2）测试不同的遍历算法

控制区内出现遍历算法的选择。点击不同的遍历按钮，就会触发相应的遍历函数执行。执行前，实验平台会弹出二叉树结构视图，供实验者观察：



点击确定后，遍历的结果将显示到实验者展示区。并与例程结果做对比。

**【思考题】**

1. 怎样理解二叉树的递归算法也是一种循环？
2. 二叉排序树为什么被称为排序树？

3、二叉过滤树在构造逻辑表达式时，利用了二叉树的什么性质？

# 检索和排序

**【设计目的】**

1. 掌握顺序检索、二分检索和哈希检索的算法设计和程序实现
2. 掌握简单插入排序、简单选择排序、冒泡排序和快速排序的算法设计和程序实现。

**【设计要求】**

一、基本题

【命令行式】

Ex5-1:

1）编写顺序查找函数

2）编写二分查找函数

3）在主程序中输入关键字（43和5），分别调用两种查找函数，输出结果。

Ex5-2:

1）一个班有30位同学，安排装进一个有30个元素的数组，以姓名作为关键字进行哈希存储，具体方法如下：将姓名字符串中的每个字节按ASCII码（中文也支持 的哦）加起来，除以30，取得的余数作为元素存放位置（数组下标）。冲突解决采用线性探查法。

2）输入少于30个学生姓名，按Hash方式存入表中。

3）验证能够按Hash方式找到表中学生，不在表中将提示错误

Ex5-3:【排序】

1）编写简单选择法函数

2）编写直接插入法函数

3）编写冒泡法排序函数

4）编写快速排序函数

5）在主程序中输入一组数据元素，如513，87，512，61，908，170，897，275，653，462，分别调用三种排序函数，输出每趟排序结果。

【图形化式】

在实验平台的search.cpp文件中编写顺序检索、二分检索和hash检索算法模块，实现在成绩表中对指定表项的检索。

在实验平台的sort.cpp文件中编写简单插入、简单选择、冒泡和快速排序算法模块，实现对成绩表项按“总分”递增进行排序。

二、选作题

Ex5-4：

1）两个有序的顺序表，共用1个空间，存放时两个表是连续的，即表A先存放，接着是表B，编写合并函数，合并成一个有序的顺序表，在原表上完成。

2）编写归并排序函数，先以每个元素为子表，两两归并——调用合并函数，然后继续把新表两两合并，直到最后合并成一个大表，完成归并排序

3）main函数中输入一组无序数，调用归并排序函数，进行排序，并显示排序结果。

**【设计原理】**

1. 顺序检索

顺序检索是最简单的检索方式。按照线性表结构从前向后，逐个比对。

1. 二分检索

二分检索又称折半检索，优点是比较次数少，检索速度快，平均性能好；其缺点是要求待查表为有序表。因此，折半检索方法适用于不经常变动而检索频繁的有序列表。

首先，假设表中元素是按升序排列，将表中间位置记录的关键字与检索关键字比较，如果两者相等，则检索成功；否则利用中间位置记录将表分成前、后两个子表，如果中间位置记录的关键字大于检索关键字，则进一步检索前一子表，否则进一步检索后一子表。重复以上过程，直到找到满足条件的记录，使检索成功，或直到子表不存在为止，此时检索不成功。

1. hash检索

根据关键字，进行某种运算，确定元素存储位置的检索方式。

1）哈希检索的关键因素之一是哈希算法，给定一个输入值，算出一个输出值，这个输出值就作为元素的存储位置，一般是元素的数组下标。

常用的哈希算法有以下几类

截段法：从关键字中截取一段

例：关键字为学号，在管理时可从关键字中截取后三位作为元素在表格中的存放位置。

除法：将关键字除以表格长度，取其余数。

乘法：

分段迭加法：

2）哈希检索关键之二是元素的存储必须按照哈希算法完成

3）哈希检索关键之三是对“冲突”的处理，当哈希结果相同时，必须考虑如何解决存储位置的冲突问题。

常用的冲突解决方法如下：

Hi（k） ＝ （H（K）＋ di）mod m

i：第i个冲突的元素

i ＝ 1 to n

（1）di ＝ 1, 2, 3, …

（2）di ＝ 12, －12, 22 ，－22

（3）di ＝ 伪随机序列

1. 简单插入排序

基本思想：从表取一个元素，按照排序关系插入到新的排序表中。

若直接在原表中进行：将表分为已排序子表和未排序子表。每次从未排序子表中取出表头元素按排序关系插入到已排序子表中，当未排序子表为空时，排序完成。

1. 简单选择排序

基本思想：从无序表中依次选择出最小的元素，插入在新的排序表尾

若直接在原有的表中进行排序，将表分为已排序子表和未排序子表。每次从未排序子表中选出最小的元素，与未排序子表的表首元素交换，同时未排序子表表首成为排序子表表尾，当未排序子表为空时，排序完成。

1. 冒泡排序

重复地走访过要排序的数列，一次比较两个元素，如果他们的顺序错误就把他们交换过来。走访数列的工作是重复地进行直到没有再需要交换，也就是说该数列已经排序完成。

这个算法的名字由来是因为越大的元素会经由交换慢慢“浮”到数列的顶端，故名。简单而言就是逐个交换次序不当的相邻表项，多趟扫描后得到排序表。

冒泡排序算法的运作如下：（从后往前）

1）比较相邻的元素。如果第一个比第二个大，就交换他们两个。

2）对每一对相邻元素作同样的工作，从开始第一对到结尾的最后一对。在这一点，最后的元素应该会是最大的数。

3）针对所有的元素重复以上的步骤，除了最后一个。

4）持续每次对越来越少的元素重复上面的步骤，直到没有任何一对数字需要比较。

1. 快速排序

从表中任意选取一个元素，把它放于它在排序表中应该放置的位置。算法过程如下：

(1)选取表首元素

(2)将该元素放在表中这样的位置：在它之前的元素都比它小，在它之后的元素都比它大。

(3)以该元素为界限，两端的子表不一定有序

(4)将快速算法分别作用于两端的子表，不断进行下去，直到最小的子表是一个元素的子表。

**【图形化式设计步骤】**

1. 设计并实现顺序检索算法

【设计】

1）Function name: sequence\_search()

Descriptions : 顺序检索，在顺序表中找到指定元素后，将元素值取出，放置到指定内存中，可能需要使用memcpy()

Input:

table\_t \* table; 顺序表指针

double stuNum; 待查询元素的关键字——学号

Output:

element\_t \* elem; 用于存放指定元素内容的内存指针

return: 类型：int;待查元素在顺序表中的序号（从0开始），-1查询失败

【调测步骤】

1. 切换控制区

点击导航区检索/顺序检索，切换控制区内容



1. 填入查找所需的关键字——学号，点击“查询”按钮，将触发binarry\_search（）函数执行。查询结果将显示在控制区内。
2. 设计并实现二分检索算法

1）Function name: binaray\_search()

Descriptions : 二分检索

在顺序表中找到指定元素后，将元素值取出，放置到指定内存中，可能需要使用memcpy()

Input:

table\_t \* table; 顺序表指针

double stuNum; 待查询元素的关键字——学号

Output:

element\_t \* elem; 用于存放指定元素内容的内存指针

return: 类型：int;待查元素在顺序表中的序号（从0开始），-1查询失败

【调测步骤】

1. 切换控制区

点击导航区检索/顺序检索，切换控制区内容

1. 填入查找所需的关键字——学号，点击“查询”按钮，将触发sequence\_search（）函数执行。查询结果将显示在控制区内。
2. 设计并实现hash检索算法

1）Function name: hash()

Descriptions : 计算关键字的哈希值，即按照设定的Hash公式计算出关键字经过Hash运算后的结果本算法设定的Hash公式仿照Daniel J.Bernstein 教授设计的DJB算法，以学生姓名的字符串为输入关键字完成计算，然后再采用截断法，将结果的最后五位(二进制)输出，即获得0~31之间的Hash结果。

DJB算法：

设输入为a(1)，a(2)，……a(n)这样的字节流(字符串)，

hash(0) = 5381;

for i = 1 to n

hash(i) = (hash(i-1)\*32 + hash(i-1)) + a(i)

result = hash(n) & 0x1f;

Input:

char \* stu\_name; 学生姓名字符串

Output:

return: 类型：int; Hash运算结果, 0 ~ 31

2）Function name: hash\_search()

Descriptions : hash检索, 根据关键字计算出的hash值对hash表进行检索。hash表中，学号为0表示该位置没有填写有效的内容冲突解决方法为线性探查法。

Input:

table\_t \* table; 顺序表指针

char \* stu\_name; 待查询元素的关键字——姓名

Output:

element\_t \* elem; 用于存放指定元素内容的内存指针

return: 类型：int;待查元素在顺序表中的序号（从0开始），-1查询失败

3）Function name: hash\_insert()

Descriptions : hash插入, 根据关键字计算出的hash值，将元素插入到hash表中。冲突解决方法为线性探查法

Input:

table\_t \* table; 顺序表指针

element\_t elem; 新元素

Output:

return: 类型：int ; 插入结果,0表示成功，-1表示失败。

【调测步骤】

1. 切换控制区

点击导航区检索/顺序检索，切换控制区内容



1. 点击初始化hash表，将触发hash\_insert（）执行，实验平台会按照成绩表原有顺序逐个元素地调用hash\_insert函数，创建hash存储表
2. 填入查找所需的关键字——学号，点击“查询”按钮，将触发hash\_search（）函数执行。查询结果将显示在控制区内。
3. 设计并实现简单插入排序算法（操作方式同上）
4. 设计并实现简单选择排序算法（操作方式同上）
5. 设计并实现冒泡排序算法（操作方式同上）
6. 设计并实现快速排序算法（操作方式同上）

**【思考题】**

1. 如果利用二叉排序树进行检索，其效率是否与二分检索一样，为什么，制约二叉排序树的因素是什么？
2. hash检索的效率是否受到元素进入存储空间的顺序有关，为什么？
3. 冒泡排序是否是稳定的算法，为什么？

# 复杂应用设计

**【设计目的】**

1）掌握结合应用需求，设计数据结构、核心算法的方法

2）掌握设计和改进基本数据结构和操作，解决复杂问题的方法

本部分共分为两个复杂应用。

（一）设计一个可以对线性表内容按设定复杂规则进行过滤的过滤器

（二）基于费波拉契数列对目标集合进行分组，获得最佳方案。

本部分实验均在图形化的软件实验平台上完成，实验者也可以根据实验要求自行设计基于命令行的程序，只需要做好相关的输入、输出设计。

**【设计一、要求】**

在filter\_tree.cpp文件中实现基于二叉树的过滤规则集结构操作算法及数据过滤算法。最终在实验者展示区展示正确的结果。

**【设计一、原理】**

1. 基于二叉树的过滤规则公式

表面上过滤规则公式可以看作是规则项的线性集合，但实际上它是一个二叉树结构。如：假设有A～G，7个过滤规则，形成的过滤公式如下：

***P = A&(B&(C|D)|(G&E|F)***

过滤规则公式的运算符有“与”运算和“或”运算。与加减乘除的运算规则相比，逻辑运算两个符号之间有优先级，“与”优先于“或”运算，而且当与运算结果为假时，还要看或运算的结果是否为真。这就类似于二叉树遍历时，遍历完左子树，再遍历右子树。

于是过滤规则式可以通过一颗二叉树来表达，上式可以通过下图表示。值得注意的是，表达式有可能形成图状结构，比如同一条规则出现在多处，此时需要我们通过化简过滤式，尽量让规则项只出现一次，实在不行时，可用两个规则项，内容一样的方法来表达。总之，形成一棵二叉树的形态。

and

or

or

and

and

or

这颗二叉树的每个节点代表了一条过滤规则，节点之间的左子树关系描述了逻辑运算的“与”关系，右子树描述了“或”关系。

针对一个具体的过滤对象，一个规则树的过滤结果如果为真，基于，根的结果为真且左子树结果为真；或者根结果为假，右子树的结果为真。一个规则树的过滤结果为假，则基于，根的结果为真，且左子树结果为假，且右子树结果为假，或者根结果为假且右子树的结果为假。

过滤算法将以先序遍历为框架，分别根据根节点、左子树，右子树的过滤结果，决定本树的过滤结果。具体的算法过程为：

1. 计算根节点过滤结果
2. 如果根结果为真，获得左子树过滤结果
3. 如果左子树过滤结果为真，则本树的过滤结果为真，结束
4. 如果左子树过滤结果为假，获得右子树过滤结果
5. 如果右子树过滤结果为真，则本树过滤结果为真，结束
6. 如果右子树过滤结果为假，则本树的过滤结果为假，结束
7. 如果根结果为假，则获得右子树的过滤结果
8. 如果右子树过滤结果为假，则本树过滤结果为假
9. 如果右子树过滤结果为真，则本树过滤结果为真。结束

**【设计一、步骤】**

注意：在filter\_tree.cpp中完成本部分实验

设计并实现二叉过滤规则树的插入、删除和过滤算法。其中，插入和删除算法用于建立二叉过滤规则树，过滤算法是遍历整个输入线性表，对每个元素进行过滤，把不能满足规则的元素删除。

1）Function name: insert\_filter\_tree()

Descriptions : 向过滤规则树插入一个元素，插入在parent\_id节点的type（0-左子或1-右子上），如果parent\_id为0，则本节点作为整个树的根，而且type指示的是原来的树根做为本节点的左子或者右子。否则type指示的是本节点作为父节点的左子或右子。当本节点作为父节点的左子树插入时，父节点原来的左子树将作为本节点的左子树当本节点作为父节点的右子树插入时，父节点原来的右子树将作为本节点的右子树

Input:

filter\_tree\_t \* tree; 过滤规则树指针

int my\_id, 本节点id，即待插入节点的id

Parms rules, 过滤规则，作为过滤规则树节点主要的元素域

int parent\_id, 指示应该在哪个父节点后插入，0表示本节点将作为根

int type, 指示本节点是作为父节点的左子树插入还是右子树插入，0表示左子树，1表示右子树

Output: NONE

return: 类型：int，0，表示成功插入；-1，表示插入失败，一般是因为没有足够的内存，-2表示找不到指定的父节点

2）Function name: delete\_filter\_tree()

Descriptions : 按过滤规则树的方式删除指定id的元素，当本节点是树的根时，原左子树将作为新的树根，原右子树将作为新根的最右子树，即从根开始一直向右，到达尽头时插入,如果本节点没有左子，则直接将本节点右子作为根，结束。

当本节点作为父节点左子删除时，本节点原来的左子将作为父节点的左子树,本节点原来的右子树将作为本节点原左子树的最右子树，如果本节点没有左子，则直接将本节点右子作为父节点的左子，结束。

当本节点作为父节点右子删除时，本节点原来的左子将作为父节点的右子树,本节点原来的右子树将作为本节点原左子树的最右子树，如果本节点没有左子，则直接将本节点右子作为父节点的右子，结束。

Input:

filter\_tree\_t \* tree; 过滤规则树指针

int id;待删除节点的id

Output: NONE

return: 类型：int, 0表示正确删除，-1表示删除失败，因为没找到指定节点。

3）Function name: do\_filter()

Descriptions : 使用过滤规则树种节点的rules对输入的data进行过滤检查根据过滤规则树的原理，过滤算法将以先根遍历为框架，具有递归特点：首先检查树根条件，如果根判决结果（以下简称结果）为真，就继续检查其左子树（与条件），如果右子树查得的结果为真，则直接返回结果为真如果根结果为假或者根为真且继续检查左子树为假，才继续检查根的右子树（因为如果左子树为真则在上一部就直接返回了），如果右子树结果为真，则返回结果真，结束，如果右子树结果为假，则返回结果也是假，结束。如果根的左子或右子为空，则跳过相应步骤，以根的结果来决定下一步。如根的结果为假，又没有右子树，则最终结果为假。如果根结果为真，没有左子树，则结果直接为真，并结束。

Input:

filter\_tnode\_t \* root; 过滤规则树指针

element\_t data; 用于过滤检查的数据元素

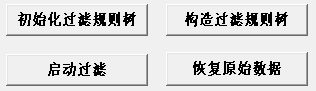
Output: NONE

return: 类型：bool 过滤检查结果，0表示结果为假，1表示结果为真

【设计一、调测步骤】

1. 切换控制区

点击实验软件左侧导航区的：复杂应用/设计过滤器，控制区切换为：



1. 初始化过滤规则树

点击初始化过滤树按钮，此时init\_filter\_tree()函数会被触发

1. 构造过滤规则树

点击构造过滤规则树，出现过滤规则树的构造界面。点击界面上的“读取过滤脚本”按钮，选择filter.script文件，可以读取已经存盘的脚本。



图中，每条过滤规则都有一个唯一的ID号以作区分。脚本中已经有三条规则，分别是:

按照过滤树的二叉遍历规则，将得到过滤式为：

含义为：获得总分大于80和总分小于80且期中成绩也小于80的成绩项。

点击“And”按钮和“Or”按钮，将分别在指定的过滤规则（）后添加一个“与”规则或者“或”规则，即添加一个在指定的二叉树节点后添加一个左孩子或右孩子。

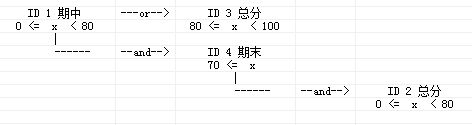
请在ID1后添加一条与规则：期末成绩>70。

这将意味着过滤式变为：

点击“And”按钮，将弹出过滤规则设定窗口，根据要求设定“期末成绩>70”的规则。



点击“OK”按钮后，insert\_filter\_tree（）函数将被触发执行。插入后的结果应为：



1. 启动过滤并检查结果

点击“启动过滤”按钮后，do\_filter（）函数将被触发，遍历过滤树，根据遍历结果输出过滤结果。

实验平台成绩过滤表中的每条数据逐条过滤，并在数据展示区显示过滤结果为“真”的表项，换句话说，就是过滤结果为“假”的表项都被过滤掉了。实验平台将通过对比例程过滤结果和实验者过滤结果，帮助实验者检错和排错。

1. 自拟规则树，并执行过滤

**【设计二、要求】**

全班有几十个学生，现在要开展一项关于合作工作的心理学研究，对学生进行任意分组，观察他们合作的情况。一个基本的原则是，成绩接近的学生更容易开展合作，而成绩相差过大，则成绩较差者往往依赖成绩高的人

假设某个项目个人独立完成的评价值为1，两位成绩差小于等于3的学生合作，每个人的评价值将提高为1.5，而成绩差大于3的学生合作，评价值将减低为0.8，之后，随着成绩差越高，每个人的评价值将随之降低，其计算公式为0.8\*(100-成绩差)/100。

如果只考虑独立完成和两人小组的任意组合情况，按成绩排序后，利用斐波拉切数列的爬楼梯思路来简化解题过程，请计算，共有多少种组合？请在这些组合中，找出合作性评价值最高的组合。

1）请编写程序计算在不考虑成绩因素的情况下，进行1人或2人的随意组合，共有多少种分组方案？

2）结合爬楼梯的斐波拉切数列求解方法进行优化和简化，其组合有多少种分组方案？

3）编写程序求解所有分组方案中，合作评价值总和最高的方案，输出该方案的分组情况和合作评价总和。

**【设计二、原理】**

1. 斐波拉契数列

斐波那契数列（Fibonacci sequence），又称黄金分割数列、因数学家列昂纳多·斐波那契（Leonardoda Fibonacci）以兔子繁殖为例子而引入，故又称为“兔子数列”，指的是这样一个数列：

1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233，377，610，987，1597，2584，4181，6765，10946，17711，28657，46368........

在数学上，斐波纳契数列以如下被以递推的方法定义：F(1)=1，F(2)=1, F(n)=F(n-1)+F(n-2)（n>=3，n∈N\*）；

数列的通项公式为：

F(1) = 1;

F(2) = 1;

当n趋向于无穷大时，前一项与后一项的比值越来越逼近黄金分割0.618。

在现代物理、准晶体结构、化学等领域，斐波纳契数列都有直接的应用，为此，美国数学会从1963年起出版了以《斐波纳契数列季刊》为名的一份数学杂志，用于专门刊载这方面的研究成果。

1. 列式求解人数为n时，组员人数为1或2，有多少种组合。

n为1时，f(1) = 1;

n为2时，f(2) = 2;

n>2,分解为从n中任选2m人，组成m组；

2i个数组成的数列，两两任意组合形成新的数列，组合方案的数量为：

排列的总个数为2i!,2i表示两个数形成的小组组内没有顺序，共有i组；i!表示组间没有顺序上的区别：

所以：

求和公式可以采用循环语句迭代计算出结果。

1. 应用斐波拉契数列实现优化

本问题表面上是所有人按照1或2人形成小组的所有组合情况，实际上因为要求小组成员的成绩越接近越能得到高的合作评价值，没有必要在整个数列的范围内进行小组组合，如果将成绩表中的学生按成绩进行排序，按序将相邻的人员进行分组,可以优化求解过程。小组人数可由1或者2人构成。分组方案类似爬楼梯问题：小明爬一段楼梯，一次爬1阶或2阶，共有多少种方法爬上n阶楼梯。分组组合的数量则形成斐波拉契数列。

证明：用n表示分到第n组，an表示分到第n组时，总共有多少种分配方案。则：

n=1：1 —— a1=1种

n=2：11 ，2——a2=2种

n=3：111, 12，21——a3=3种

n=4：1111，112,121,211,22——a4=5种

n=5:11111,1112,1121,1211,2111,122,212,221,——a5=8种

……

f（n）= f（n-1）+f（n-2），f（1）=1，f（0）=1，n>1

f（n）= {1,1,2,3,5,8,13,21，……}；

利用斐波拉契递推数列能够求出按序排列的相邻组合的个数。利用通项式或递推方法，编写程序求解时都需要采用循环语句迭代求解。

1. 利用深度优先搜索穷举所有方案

求解最大合作评价总量的方法，可采用穷举法求解所有可能的组合，然后比较得到最佳的方案，是常用的方法之一。问题是如何实现对所有可能的方案的穷举？

按照斐波拉契数列的思路，从f(x)开始，下一步的选择总是两种：1或者2。这样，执行方案的每一步就构成了一棵二叉树，二叉树的左子树可以看作是下一步采用单人组，右子树看作是采用双人组。这样一步一步深入下去，直到将所有的人都分配完毕。

于是，利用二叉树遍历方法就能进行穷举，当然，二叉树的遍历方法也可以被看作是一种深度优先搜索方法。

**【设计二、步骤】**

注意：本部分实验在application.cpp文件中完成。

算法设计并实现的基本步骤为：

1. 数据结构设计：根据应用要求，选取适当的基本数据结构并进行合理扩充。
2. 关键算法设计：组合、改进数据结构基本操作，实现应用核心功能
3. 设计解决应用问题所需其他软件模块，如输入、输出等，或利用实验软件平台，实现问题的输入和输入。
4. 实现全部设计，并调测解决问题，获得结果

根据本设计要求，至少应设计三个算法及相关的数据结构：

1. 计算n个元素的集合分为多个小组，每组1～2人，有多少种组合方法。

Function name: plans\_total\_app()

Descriptions : 计算总分组方案数量，按照1或2人组成小组，计算总数为n时，总的分组方案数量

Input: int n ;人数总数

Output:

return: double 总分组方案数量算法流程图：

temp =

sum += temp; i ++;

2i<n

sum = 1;i=1;

结束

注：建议编写求阶乘的子函数，以帮助简化代码。

1. 计算斐波拉契序列值

将元素按成绩排列后，只考虑邻近元素之间的组合情况，将大大优化本题目的分组要求，根据费波拉契数列，计算优化后的组合方案数量。

Function name: plans\_feb\_app()

Descriptions : 计算费波拉契优化的分组方案数量

按照1或2人组成小组，计算总数为n时，按序排列的分组方案数量

Input: int n ,总人数

Output:

return: double 总分组方案数

算法流程图

sum = x+y;

i ++;

i<=n

x = 1;y=1;i =3;sum = 0

结束

x = y;

y = sum;

1. 使用递归方式获得最大的合作评价值总和分组方案。

每次分组时，可以选择1人或者2人形成小组，因此可以将分组过程视作一棵二叉树，左子树为1人组，右子树为2人组，当树的带权路径和为n时，分组过程结束，接下来可以后退一步选择另一种分配方案，直到将所有的分配方案都尝试过一次，从而比较计算出具有最佳合作值的方案。

提示：本算法需要一个一维数组记录分组方案。

算法采用先序遍历法框架进行递归求解。

Function name: best\_plans\_app()

Descriptions : 寻找费波拉契优化后的最佳分组方案

按照1或2人组成小组，按序排列的分组方案中合作评价总值最大的方案

Input: table\_t \* table，输入的待分组的线性表。

Output: table\_t \* table; 分组后的组号可以利用table元素中的reserve成员，不同分组分组号不同。

return: double 合作评价值总和

也可以采用基于栈的非递归算法，算法的流程设计如下：

图片包含 文字, 地图

已生成极高可信度的说明

1. 学生成绩表的输入、分组方案输出

从文件或键盘输出学生成绩表，将最佳的分组方案输出。

如果利用基于成绩表的《软件技术基础实验》则可以利用平台得到输入表单，按格式准备好输出表单，即可显示，并与标准答案对比。

注意：平台中输入的成绩表是无序的，在基于费波拉契数列进行优化前，需要先按成绩排序。

【设计二、调测步骤】

1. 切换控制区

点击软件导航区：复杂应用/利用费波拉契数列寻找最佳分组方案，切换导航区为：

图片包含 屏幕截图

已生成极高可信度的说明

1. 计算总分组方案数量

1）可以在输入框中更改总人数，更改以后，点击一下上方的表格，就可以发现表中元素数量按照输入的总人数进行了调整。

2）点击“计算总分组方案数量”按钮，完成总数量计算

计算完成后的总数量会在右侧结果中显示，并给出评判：



可以看到总人数为40人是，这几乎就是一个天文数字。

1. 计算优化后的方案数量

点击“计算费波拉契数列优化分组方案数量”按钮，完成计算

计算完成后的总数量会在右侧结果中显示，并给出评判：

图片包含 屏幕截图

已生成极高可信度的说明

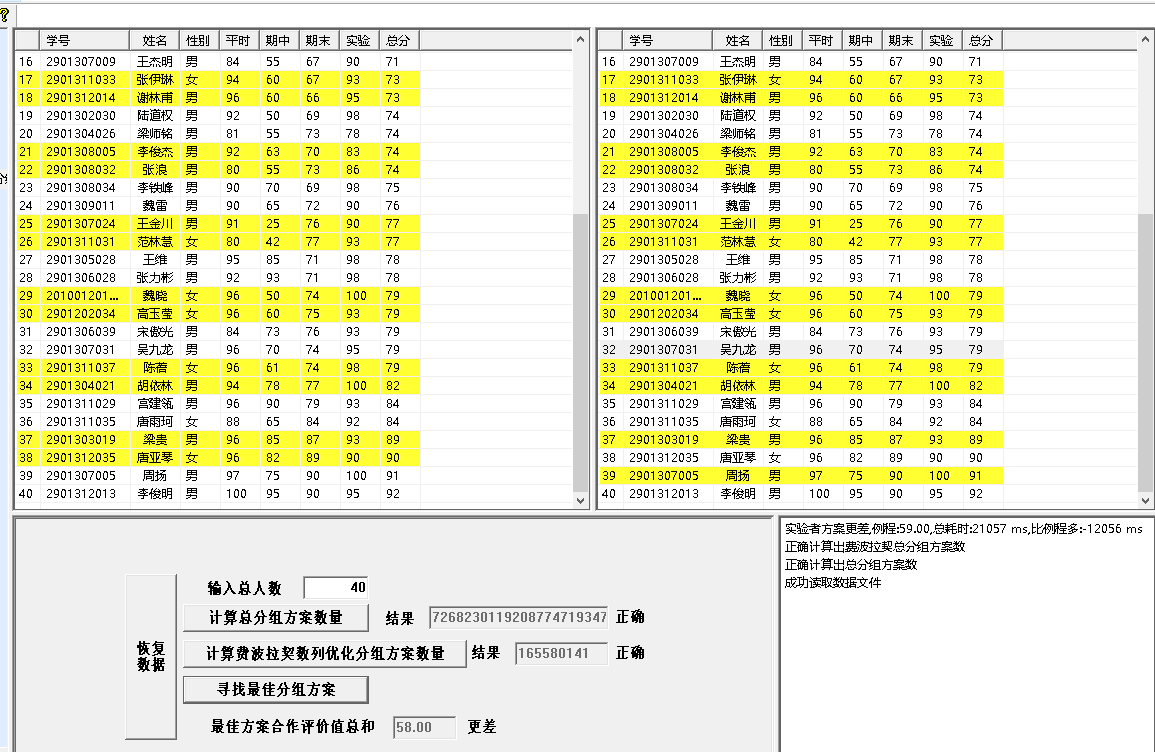
1. 寻找最佳分组方案

点击“寻找最佳分组方案按钮”，完成方案寻找和合作评价值计算

如果是40人总数，寻找需要花相当长时间。

完成计算后，数据区会对比例程和实验者寻找出来方案。

例如下图：



实验虽然比例程少消耗12秒时间，但是寻找出来的方案比例程差。注意，根据题目要求，合作评价综合是越大越好。

# 面向多线程的缓冲池设计

**【设计目的】**

1. 了解和体验多线程同时操作成绩表会出现什么问题
2. 观察并分析演示单线程和多线程的效率区别
3. 掌握利用信号量进行多线程同步的方法
4. 设计能支持多个设备与用户线程操作的缓冲区

**【设计要求】**

观察实验平台上单线程与多线程并发时的执行效率。观察多线程不同步时的出错的执行结果，并分析出错的原因，掌握利用信号量进行多线程同步的方法。

在可并发时长为2000ms，表操作时长为10ms下，多线程并发不出错的执行效率比单线程提高40%。

实现能够支持多个设备线程和用户线程同步完成输入、输出操作的缓冲池。缓冲池中至少有100个缓冲区，支持的设备线程和用户线程数量不少于10个，能够实现平台中模拟的设备线程和用户线程之间的同步操作，并完成他们提交的全部输入和输出任务。

本部分实验均在软件实验平台上完成。

**【设计原理】**

1. 多任务场景

假设成绩管理表被放在了一台网络服务器上，现在有多个客户同时要访问这个成绩表，有的要插入新的成绩，有的要删除某条信息。

实验平台利用套接字接口实现了多个任务同时服务场景。首先，实验平台利用指令序列生成器，生成了一系列成绩表操作指令。这些指令的类型和内容是随机生成的。指令序列生成器将这些命令同时发送到实验平台的服务端口上，服务器如果为每个命令创建一个线程来应对，就形成了多任务并发运行的场景。



1. 单线程效率与多线程比较

当服务端收到命令后，一般会有两种选择，一种是单线程逐条处理收到的命令，另一种是每条命令创建一个线程，执行这个命令。

由于计算机运行程序的速度非常快，凭实验者的肉眼将很难观察到程序以单线程或多线程在效率上的区别。

实验平台为了让实验者体验程序执行的过程，将线程执行任务的时间放大到人能够分辨的程度，并尝试定量计算效率。被放大的时间主要有两种：**成绩表操作时间和不操作表完成其他任务的时间**。都采用用Sleep（）函数进行模拟。



“成绩表操作时间”模拟是在操作一个成绩表的过程中发生，如从成绩表取出元素、搬移元素、放置元素等。比如搬移一个元素位置的真实运行时间可能只需要几十个ns，我们在真实的表操作完成后，加入Sleep（50）延时，将搬移元素的时间消耗延长到50ms，表操作的每个动作，实验者基本上就能观察到了。

有些时候线程还需要完成一些与访问成绩表无关的操作，这些操作通过“可并发时长”来延长消耗的时间。

在实验平台上，我们可以观察和体验到并发执行的效率比单线程的效率高很多。这是因为并发执行的线程，在很多时候是并发推进的，交替使用CPU，尤其在多个线程同时进入可并发的任务环节中时，几乎可以认为他们的推进也是同时的，而不是线性的。

1. 多线程同步控制

本部分实验代码在Thread.cpp中

多个插入或删除操作同时发生，将会导致灾难性的后果，这一点我们在实验平台的执行结果对比上可以观察到。要解决这个错误，必须实现线程同步。即避免两个以上的线程同时操作成绩表，同时又保证它们可以同时运行“非表操作”——可同步的操作。

多线程的同步是在信号量同步理论指导下，利用windows系统各种同步操作中的一种来实现线程同步。

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//小提示：Windows同步操作

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

CriticalSection:

CRITICAL\_SECTION g\_csA;

InitializeCriticalSection(&g\_csA);

EnterCriticalSection(&g\_csA);

LeaveCriticalSection(&g\_csA);

DeleteCriticalSection(&g\_csA);

互斥对象:

HANDLE hMutex;

hMutex=CreateMutex(NULL,TRUE,LPCTSTR("tickets"));

WaitForSingleObject(hMutex,INFINITE);

ReleaseMutex(hMutex);

CloseHandle(hMutex);

事件对象

HANDLE g\_hEvent;

g\_hEvent=CreateEvent(NULL,FALSE,FALSE,LPCTSTR("tickets"));

OpenEvent(),打开一个事件——用的少

ResetEvent(g\_hEvent);

SetEvent(g\_hEvent);回置事件

WaitForSingleObject(g\_hEvent,IFINITE)

WaitForMultipleObjects()

信号量

CreateSemaphore() 创建一个信号量

OpenSemaphore() 打开一个信号量

ReleaseSemaphore() 释放信号量

WaitForSingleObject() 等待信号量

HANDLE global\_Semephore;

global\_Semephore= CreateSemaphore(NULL, 1, 1, NULL);

WaitForSingleObject(global\_Semephore, INFINITE);

ReleaseSemaphore(global\_Semephore, 1, NULL);

1)互斥量与临界区的作用非常相似，但互斥量是可以命名的，也就是说它可以跨越进程使用。所以创建互斥量需要的资源更多，所以如果只为了在进程内部是用的话使用临界区会带来速度上的优势并能够减少资源占用量。因为互斥量是跨进程的互斥量一旦被创建，就可以通过名字打开它。

2）互斥量（Mutex），信号灯（Semaphore），事件（Event）都可以被跨越进程使用来进行同步数据操作，而其他的对象与数据同步操作无关，但对于进程和线程来讲，如果进程和线程在运行状态则为无信号状态，在退出后为有信号状态。所以可以使用WaitForSingleObject来等待进程和线程退出。

3）通过互斥量可以指定资源被独占的方式使用，但如果有下面一种情况通过互斥量就无法处理，比如现在一位用户购买了一份三个并发访问许可的数据库系统，可以根据用户购买的访问许可数量来决定有多少个线程/进程能同时进行数据库操作，这时候如果利用互斥量就没有办法完成这个要求，信号灯对象可以说是一种资源计数器。

1. 多线程非同步与多线程同步的效率

实现多线程同步后，程序的执行结果不再出现偏差。但是效率会有所下降，毕竟表操作之间是互斥的，不是并发的。可并发执行的时间越长，同步并发的效率就越高，而且还不会出错。

1. 缓冲池设计

缓冲池是一种复杂的结构，多用来支持设备与用户程序之间的同步操作，以提高系统效率，此时假设成绩表是一个缓冲池，有两类程序利用这个缓冲池完成数据的交换：设备进程和用户进程。

emptyq

inq

outq

收容输入

提取输出

提取输入

收容输出

设备

用户程序

数据到达

取走数据

发送数据

取走发送

缓冲池一般包括三条队列：

emptyq：空白缓冲区队列；

inq：设备输入的数据，供用户程序读取的缓冲块队列；

outq：用户程序输出的数据，供设备读取并输出的缓冲块队列。

操作缓冲池的进程一般分为两种：设备驱动进程和用户进程，缓冲池除提供设备与用户程序之间的速率匹配外，还实现他们之间的同步。

缓冲池的操作一般分为四类：

由用户程序操作的“提取输入”和“收容输出”

由设备程序操作的“收容输入”和“提出输出”

设计时，可以将缓冲池视为双向的生产者和消费者，建立程序之间的同步关系。emptyq队列提供空白消息块，对于inq队列，用户程序是消费者，设备程序是生产者；对于outq队列，设备是消费者，用户是生产者。

用户线程分为用户发送线程和用户接收线程两类

发送线程按照：生产数据，取空白缓冲块，填充缓冲块，收容输出到缓冲池，这样的流程往复进行。

接收线程按照：提取输入缓冲池数据，消化数据，放回空白缓冲块的流程往复进行。

设备线程也分为设备输入线程和设备输出线程两类

设备输入线程按照：等待设备输入指示，取空白缓冲块，填充数据，收容输入到缓冲池。

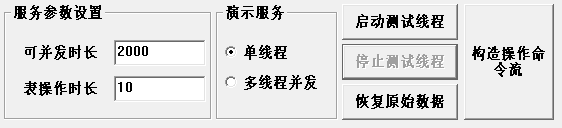
设备输出线程按照：提取输出缓冲池数据，取空白缓冲块，复制数据，归还空白缓冲块，通过设备发送数据。

**【设计步骤】**

1. 测试单线程与多线程的执行效率

点击导航区：多线程同步/访问顺序表

1. 选择演示服务为单线程，可以先设置线程可并发消耗时长为1000ms，表操作时长为500ms，观察执行情况，在具体测试中，可以缩小10倍，缩短实验时间。



“可并发时长”是模拟线程生成新数据、或处理已得到的数据时，互不影响可以并发执行的过程。

“表操作时长”是模拟在共享表中插入、删除元素的过程，这个过程是互斥的，不可并发。

1. 启动测试线程以等待命令流

点击“启动测试线程”按钮，将测试线程启动起来，接连来构造好命令流以后，测试线程将按照命令流向各执行下达响应的执行命令。

1. 构造命令流

点击“构造操作命令流”按钮，在弹出的命令流生成器窗口中点击“刷新”按钮，将自动随机生成N条指令。



1. 启动服务，观察现象

点击“执行”按钮，我们将观察到例程数据展示区以动画的方式演示单线程逐个完成指令的过程，在实验者数据展示区中以动画方式演示多线程并发完成指令的过程。

请记录各工作的开始时间和结束时间。对比哪种方式的效率高。

1. 观察多线程非同步的执行结果

重复上一步骤，注意观察实验者展示区内的操作结果是否有错，从观察到的执行过程分析出错的原因。

本部分实验在Thread.cpp文件中，可以看到提供的并发执行线程——child\_thread（）的代码中并没有同步控制指令——需要实验者增加，本步骤先观察不进行互斥同步控制的执行效果。

1. 设计多线程的同步操作，并观察执行结果

在Thread.cpp文件中，增加信号量，在表操作前后都基于信号量增加互斥同步控制，完成设计。

**【设计】**

Function name: child\_thread()

Descriptions : 具体执行命令的子线程，每个子线程执行一条操作命令，在执行插入或删除命令的前后，增加同步操作指令。

Input:

char \*buf; 需要解析的命令报文

Output: NONE

**【调测步骤】**

同上一步骤，测试并调整最终的结果为无错。

1. 测试多线程同步操作与非同步操作的效率

记录多线程同步下，从开始工作到结束最后一条指令的工作之间的延时，与非同步及单线程方式对比。

可以调整“可并发时长”和“表操作时长”的比例关系，多做几次对比测试。

1. 设计缓冲池，启动多设备和用户线程测试。

注意：本部分实验在bufferpool.cpp文件中完成。

需要设计具有三条队列的缓冲池，三个队列采用数据结构中的链队列。队列可以考虑采用链队列，取放方便。但是要按照队列的要求设计，避免出现后进而先出的现象，违背了缓冲区的基本FIFO原则。

1. 完成链队列设计
2. 完成具有三条链队列的缓冲池设计

建议缓冲池结构如下：

typedef struct buff\_pool\_t{

queue\_t empty;

queue\_t in;

queue\_t out;

HANDLE smEmptyMutex;

HANDLE smInMutex;

HANDLE smOutMutex;

HANDLE smEmpty;

HANDLE smIn;

HANDLE smOut;

}buff\_pool\_t;

empty、in、out分别对应空白缓冲队列，系统输入队列和系统输出队列。考虑到缓冲池操作的互斥性，为每条队列设计一个互斥型信号量：smEmptyMutex、smInMutex、smOutMutex。设备线程和用户线程之间还需要进行合作，按照合作特点，设计了三个私有信号量：smEmpty、smIn、smOut；smEmpty代表空白缓冲块数量，smIn代表输入队列缓冲块个数，smOut代表输出队列缓冲块个数，典型的生产者、消费者机制。

以下是需要完成的缓冲池基本操作接口函数：

init\_pool()：初始化缓冲池，为缓冲池配备足够的空白缓冲块，并初始化信号量

collect\_empty()：向缓冲池中放入一个空白缓冲块

extract\_empty();从缓冲池取出一个空白缓冲块

extract\_out();提取输出，设备程序从缓冲池输出队列上取得一个缓冲块用于系统输出

extract\_in()：提取输入，用户程序从缓冲池输入队列上取得一个缓冲块用于处理

collect\_out()：收入输出，用户程序向缓冲池输出队列上放入一个数据缓冲块，用于输出

collect\_in()：收容输入，设备程序从设备得到输入数据后，向缓冲池输入队列放入一个缓冲块。

get\_pool\_length()：获取缓冲池指定队列的长度，用于界面动态显示当前状态。

is\_pool\_empty()：检查缓冲池中是否有待处理的缓冲块，包括输入和输出队列。用于总任务结束的判断。

除了上述接口函数外，实验者可能还需要设计一些内部逻辑控制和工具性的子函数。

(1）\*\* Function name: init\_pool()

\*\* Descriptions : 对缓冲池初始化

\*\* Input:

\*\* int size; 缓冲池中空白缓冲块的数量。

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：int; 为-1表示创建失败，为0表示创建成功，

注意对信号量的初始化。

(2）\*\* Function name: collect\_empty();

\*\* Descriptions : 将空白块放回缓冲池

\*\* Input:

\*\* node\_t \* node; 空白块

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：int; 为-1表示放入失败，为0表示放入成功，

(3）\*\* Function name: extract\_empty();

\*\* Descriptions : 从缓冲池取出空白块

\*\* Input:

\*\*

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：node\_t \*; 提出的空白缓冲块，为NULL表示提取失败，，

(4）\*\* Function name: extract\_out();

\*\* Descriptions : 提取输出，设备从缓冲池提取——输出

\*\* Input:

\*\*

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：node\_t \*; 提出的数据缓冲块，为NULL表示提取失败，，

(5）\*\* Function name: extract\_in();

\*\* Descriptions : 提取输入，用户从缓冲池提取——输入

\*\* Input:

\*\*

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：node\_t \*; 提出的数据缓冲块，为NULL表示提取失败，，

(6）\*\* Function name: collect\_in();

\*\* Descriptions : 收容输入，设备向缓冲池放入——输入

\*\* Input:

\*\* node\_t \* node; 输入的数据

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：int; 为-1表示放入失败，为0表示创建成功，

(7）\*\* Function name: collect\_out();

\*\* Descriptions : 收容输出，用户向缓冲池放入——输出

\*\* Input:

\*\* node\_t \* node; 输出的数据

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：node\_t \*; 提出的数据缓冲块，为NULL表示提取失败，，

(8）\*\* Function name: is\_pool\_empty();

\*\* Descriptions : 检查缓冲池内是否还有待处理数据

\*\* Input:

\*\* Output:

\*\* return: 类型：int ;为1，表示in和out队列均为空，为0表示两个队列至少有一个不空

（9）\*\* Function name: get\_pool\_length()

\*\* Descriptions : 获得缓冲队列长度

\*\* Input:

\*\* int type; 队列类型。POOLID\_EMPTY,POOLID\_IN,POOLID\_OUT

\*\* Output:

\*\* 全局缓冲池

\*\* return: 类型：int; 指定队列的长度

1. 完成同步控制

在bufferpool.cpp文件中相应的：设备输入——device\_in\_thread、设备输出——device\_out\_thread，用户输入——cpu\_in\_thread、用户输出——cpu\_out\_thread线程中，根据注释语句里的提示，增加同步控制。

* 信号量操作小技巧

*同步控制中，等待信号量的一种简单的控制方法是“无限期”等待，即一直等到事件发生时为止。如，申请一个空白缓冲块：*

*ret = WaitForSingleObject(buffPool.smEmpty, INFINITE);*

*当任务完成或者主线程关闭时，会在主线程里关闭信号量，如：*

*CloseHandle(buffPool.smEmpty)*

*此时等待信号量的线程的阻塞状态会被唤醒，并得到一个WAIT\_ABANDONED返回值，因为此时的真实事件并没有发生，所以线程不能继续执行既定方案，而是进入到关闭状态，比如跳出线程主循环，从而结束线程。所以后续处理应该是：*

*if(ret != WAIT\_OBJECT\_0)*

*break;*

**【调测步骤】**

1. 切换控制区

点击左侧导航区：多线程同步/设计缓冲池

图片包含 屏幕截图

已生成极高可信度的说明

时间参数中：用户产生数据、用户处理输入数据、设备输入数据、设备输出数据耗时分别对应于四个线程的数据处理过程，这些过程是可以并发的。

缓冲池操作耗时模拟了缓冲池的插入、删除操作过程，是互斥的。

开始时可以将时长设得长一些，以便观察。具体测试时可以设置得较短。

1. 初始化缓冲池

点击“初始化”按钮，执行初始化。

1. 启动多个设备线程、用户线程开始执行任务

点击“执行”按钮，观察多个线程的执行过程。

图片包含 屏幕截图

已生成极高可信度的说明

1. 取得结果，分析。

任务完成后，观察总的输入与输出数量之间是否匹配，各线程执行任务的数量，数量不等是正常的，正说明各线程的并发执行特点。

记录总的时间消耗，与例程比较，以改进同步控制。





**【思考题】**

1. 为什么非同步多线程的执行会出错？
2. 你在实验中选择的是那种同步方法？使用方法是？
3. 实验中的单线程多事件，多线程非合作，多线程合作同步，三种模式中，哪种效率最高，为什么？哪种结果是正确的，为什么？

# 基本数据结构程序调测

**【实验目的】**

1. 掌握程序调试的一般方法，包括单步跟踪、断点调试等
2. 熟悉顺序表、链表、栈、队列、二叉树操作中典型的程序编写错误、影响和解决方法

**【实验要求】**

本实验提供了数十个基本数据结构的典型错误案例，各案例的代码非常类似，简单的做法可以通过对比各案例代码，发现错误，但是这不是设计这些案例的初衷，这些案例主要帮助学习者快速建立一个可以调试的环境，真正去体验和学习如何通过调试定位问题，分析问题和解决问题。

所以，即使能够通过阅读代码发现问题，也希望学习者能真正进入调试态去观察错误的具体表现，包括记录具体的调试现象和数据，记录不是目的，目的是通过记录来促使学习者真正进入调试状态。

指导书针对调试案例，提供一些建议的输入顺序和数据，便于错误重现，帮助学习者迅速发现错误。学习者也可以根据自己的需要设定输入顺序和数据。

**【实验原理】**

1、什么是调试？

采用跟踪程序执行的方法，将一个语法正确的程序的执行结果，调整为符合设计要求。

对于初学者，只有通过调试，才能感受到语句的力量，才会真正爱上“编程”。

编程就是与计算机的对话，人类总是保持着对外界事物的好奇心，总想与之对话。所以不爱“编程”是不可能的，关键是，怎样去爱……

2、调试的基本内容

观察程序的执行：重点关注判断语句、循环语句

观察变量的变化：重点关注赋值、函数参数传递

都是为了验证程序实际执行是否符合设想，进一步考量设想是否满足要求。

3、调试的方法

1）观察程序的执行

单步跟踪：单步执行语句，子程序只算一条语句。

进入函数：进入被调用函数，继续单步调试。

执行到函数外：不再单步调试每条语句，而是运行到函数的调用位置

断点调试：设置断点，执行到断点，程序挂起

* 观察要点：提示现在将要执行哪条语句了

2）观察变量变化

变量监视窗口，鼠标移到变量上，直接检视内存（指针或设法获得变量地址）

* 重点观察：进入子函数时的输入变量值，逻辑功能中重要的变量，全局变量，循环语句有关的变量。

4、调试的过程

1）先根据设想预测——大胆假设：

下一步应该执行到哪里

下一步执行后，变量应该出现什么值

2）执行跟踪——小心求证

3）根据结果，反推原因

5、注意事项

1）不要进入标准库函数调试

即不要在库函数上执行“进入子函数”命令，否则只能退出重新开始

2）scanf语句有阻塞状态，必须进入程序窗口，输入数据后，才能继续调试。

**【实验步骤】**

Case：1-1：

本案例包括两个错误：

1. 本案例先逐个输入顺序表的元素（以-1结束），建立顺序表，建议输入：2,4,6,8,10，-1
2. 显示顺序表
3. 要求输入元素x=9，
4. 输入位置l=2 （该元素将应被插入到第二个元素之前）
5. 观察顺序表内容有什么问题
6. 调试分析问题

解决以上错误后

1. 本案例先逐个输入顺序表的元素（以-1结束），建立顺序表，建议输入：2,4,6,8,10，-1
2. 显示顺序表
3. 要求输入元素x=9，
4. 输入位置l=2
5. 显示顺序表内容（该元素将应被插入到第二个元素之前）
6. 输入要求删除的元素位置：2
7. 观察顺序表内容有什么问题
8. 调试分析问题

Case：1-2：

1. 程序自行逐个输入5个元素后，输入-1结束，如2,4,6,8,

提示：

1. 观察链表内容有什么错误，
2. 通过调试分析问题

提示，注意数组最大MAXLENGTH 为 4,观察length的值怎么就会突然变成10，然后又变成11了，可以通过调试态下的view/debug windows/memory窗口，输入table的内存地址，观察其中的变化，分析原因。

Case：1-3：

1. 本案例先自动生成一个具有6个元素的顺序表
2. 显示顺序表
3. 输入元素x=5
4. 输入插入位置l=2
5. 观察顺序表内容有什么问题
6. 调试分析问题，

Case：1-4：

本案例要求删除一个顺序表中所有负数元素。

1. 先逐个输入元素，最好里面有多个负数，注意-1表示结束输入且-1将不会放入表中，如3，-2，-4,5,6，-3,8，-1
2. 显示顺序表
3. 执行删除全部负数
4. 观察顺序表内容有什么错误
5. 调试定位错误位置，分析问题

Case：1-5：

1. 本案例先自动生成一个具有5个链点的顺序表
2. 显示菜单，选择1，插入
3. 输入元素x=11
4. 输入位置l=1
5. 观察main函数中顺序表内容有什么错误
6. 调试分析问题

Case：2-1：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=9，
4. 输入位置l=3
5. 观察链表内容有什么问题
6. 调试分析问题

Case：2-2：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=3
4. 输入位置l=1
5. 观察链表有什么问题
6. 调试分析问题

类似问题：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择2，删除，
3. 输入位置1
4. 观察链表有什么问题
5. 调试分析问题

Case：2-3：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入，本案例是按元素大小插入
3. 输入元素x=5
4. 观察链表有什么问题
5. 调试分析问题

Case：2-4：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=11
4. 输入位置l=7，大于链表长度，实际是希望插入到链表尾
5. 观察报告什么错误
6. 调试定位错误位置，分析问题

Case：2-5-1：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=11
4. 输入位置l=1
5. 观察main函数中链表内容有什么错误
6. 调试分析问题

Case：2-5-2：

本案例是学生编写的代码，学习者可以对比一下编写风格，本案例有两处错误

错误1：

1. 本案例是学生编写的代码，学习者可以对比一下编写风格。
2. 显示菜单，选择1，插入
3. 输入元素2
4. 输入元素3
5. 观察main函数中链表内容有什么错误
6. 调试分析问题

错误2：

1）在解决第一个错误，能正确创建链表后，在显示链表内容时，遇到错误。

Case：2-6：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=11
4. 输入位置l=2
5. 观察报告什么错误
6. 调试定位错误位置，分析问题

Case：2-7：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入元素x=11
4. 输入位置l=2
5. 观察报告什么错误
6. 调试定位错误位置，分析问题

Case：2-8：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择2，删除
3. 输入位置2
4. 关闭程序
5. 观察调试窗口会不会报错
6. 请阅读代码寻找问题

提示：

1）显示菜单，选择3，这段测试代码会不断向系统申请内存，而不释放，可以观察到内存也有被耗尽的时候

Case：2-9：

1. 本案例先自动生成一个具有5个链点的链表
2. 显示菜单，选择1，插入
3. 输入x=5，
4. 输入位置l=3
5. 显示菜单，选择2，删除
6. 输入3，即删除刚才插入的那个链点
7. 观察程序会报什么错
8. 通过调试定位什么语句出错，报告什么错误
9. 百度查询错误原因——提示：内存使用超界？
10. 根据原因，分析问题，阅读程序找出错误语句。

Case：2-10：

1. 本案例先自动生成一个具有5个链点的循环链表
2. 显示菜单，选择1，
3. 输入元素x=1，
4. 输入位置l=6，表示希望插入到表尾
5. 观察链表内容有什么错
6. 通过调试分析错误原因

Case：3-1：

1. 本案例先自动生成一个具有6个元素的栈：1、2、3、4、5、6，栈底为1
2. 输入元素x=9，
3. 观察栈的内容有什么问题
4. 调试分析问题

Case：3-2：

本案例目标是输入一个整数的算术四则算式后，能够按四则运算规则优先计算乘除法，算出结果，暂时不支持括号符。案例利用一个栈存储算式中的整数，另一个栈存储运算符，设计思路是，如果遇到乘除运算符，就优先运算，否则，将算式中的整数和运算符依次入栈，最后完成全部的计算。

算式的输入最好在一行内输入完成，以‘#’号或者回车作为结束

1. 先完成一些简单的测试：

输入算式：4+3\*5，结果？

输入算式：4+3\*5-6，结果？

输入算式：4+3\*5-6+2，结果？

1. 上面那个算式出错了？调试跟踪程序执行，找到出错的位置。可以先通过3）的测试现象分析问题可能出在哪里，然后用设置断点的方式，加快调试过程。

引导问题：通过3）的测试，判断，程序在读入算式内容方面有没有问题？程序是否能实现乘除法优先加减法？

1. 分析问题。提示：其实本案例的思路用队列模型更容易。

Case：3-3：

1. 本案例先自动生成一个具有8个空间的循环队列，队首在a[3]，队尾在a[7]
2. 显示菜单，
3. 选择1，输入10，观察现象
4. 调试分析问题

Case：3-4：

1. 本案例先自动生成一个具有8个空间的循环队列，队首在a[3]，队尾在a[7]
2. 显示菜单
3. 选择入队，输入10，观察现象
4. 选择入队，输入20，观察现象
5. 通过调试分析问题，为什么front变成了20？

有同学在改错时提出，只需要将队列的定义修改为 elemtype data[MAXNUM + 1 ];

这个案例仍然是个正确的循环队列，请问你怎么认为，实测情况如何？

Case：3-5：

本案例目标是在队列中间插入一个元素，难点包括如何确定插入位置，以及如何完成元素搬移以放入新元素，尤其是在首尾相接时搬移元素。

1. 本案例先自动生成一个具有8个空间，队首在a[3]，队尾在a[7]的循环队列
2. 显示菜单，选择3，插入10，插入在3后，观察有没有问题
3. 调试分析问题
4. 把

cir\_queue s = {{1,2,3,4,5,6,7,8},3,7};一行改为：

cir\_queue s = {{1,2,3,4,5,6,7,8},6,3};

即队首在a[5]，队尾在a[3];

1. 显示菜单，选择3，插入10，插入在3后，观察有没有问题
2. 调试分析问题
3. 把

cir\_queue s = {{1,2,3,4,5,6,7,8},3,7};一行改为：

cir\_queue s = {{1,2,3,4,5,6,7,8},5,2};

即队首在a[6]，队尾在a[3];

1. 显示菜单，选择3，插入10，插入在1后，观察有没有问题
2. 调试分析问题
3. 关于案例的实现方法，你有没有更简洁的方法？比如只编写一个搬移元素的循环？

# 附录1 基于成绩表的图形化软件平台使用基本技能训练

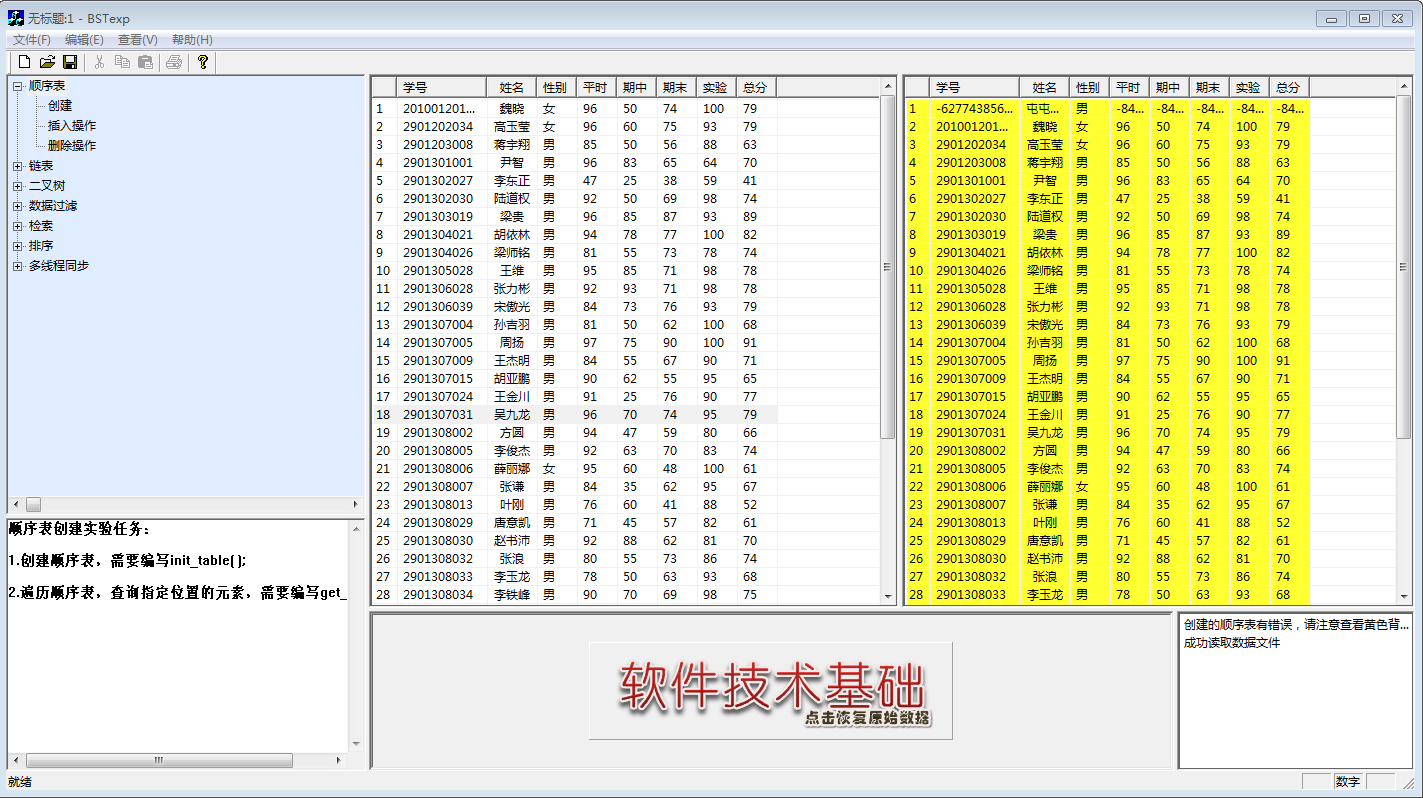
**【实验目的】**

1. 熟悉实验平台的功能
2. 掌握实验平台的基本使用方法
3. 掌握在实验平台上编写数据结构算法函数的方法

**【实验原理】**

1. 简介“基于成绩表管理的软件技术基础平台”的设计思路
2. 搭建友好的交互环境

软件技术基础课程数据结构部分的上机作业传统上是基于命令行方式，这种方式采用scanf和printf等函数实现数据的输入和输出，容易上手，但是交互方式比较原始，真正掌握并不容易。在实际教学中，程序设计的初学者往往把大量时间浪费在如何通过命令行式正确实现输入和输出上，浪费在对回车符的处理，格式化字串的使用等细节问题中，没有把宝贵的时间和精力投入到对核心算法的研讨上，这是很可惜的。利用本设计平台的图形化界面，实验者可以在程序的调测过程中获得友好的交互方式，方便地获得输入数据和进行输出。



例程

成绩表展示

实验者

成绩表展示

状态信息

控制栏

帮助信息

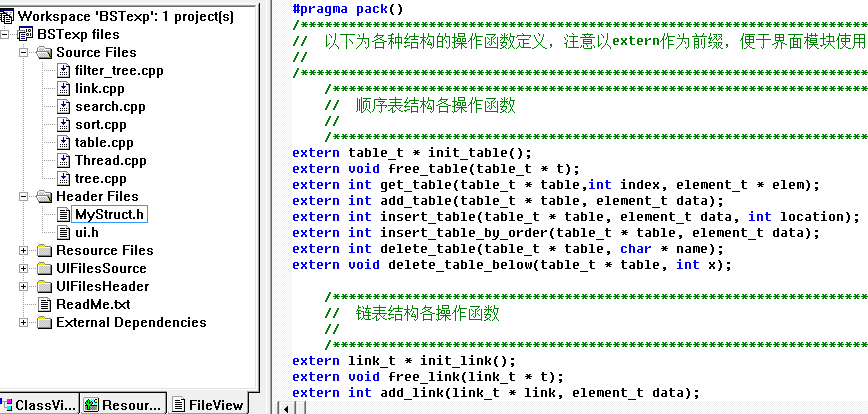
导航栏

在图形化界面上，左侧导航栏目可以选择完成指定功能的测试，可以阅读实验帮助信息，观察输入、输出结果，进行各类控制等，如下图所示：



1. 模块化设计

数据结构算法设计中已经开始引入模块化思想，但程序设计的初学者很难搭建出一个合理的模块化的架构，实验平台采用MVC架构设计，提供了一个合理的模块化设计环境。



如上图所示，整个设计任务根据不同的数据结构类型分为不同的模块。在每个模块中，实验者需要在设定的模块结构中，完成相应的数据结构操作算法。

1. 事件驱动

事件驱动的软件模式是现代程序的主流模式之一，尤其对于通信专业的学生而言，事件驱动是通信软件的基本特征。传统的命令行方式下的程序，不具备事件驱动的框架，也不可能靠学生自行完成这种框架，因此学生只得到了单一线性的程序结构的设计和变成训练。

本实验平台采用图形化界面，方便灵活的交互方式是依靠事件驱动机制来提供的。因此学生在实验中也能体验到事件驱动程序模式的基本特征。

1. 应用导向

传统的命令行式界面的程序已经很少见到了，实验者很难认同这样的程序有什么实际的作用，他们常常会对编写这样的程序究竟有什么意义产生疑问，从而动摇了学习的信心。



软件平台采用图形化界面设计，与实验者平时所使用的软件相似，自己编写的算法模块能直接应用到这样一个实用化的项目中，理论与实践的距离如此之近，从而提高了他们的学习兴趣。

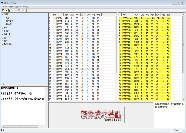
实验平台选材于学生们比较熟悉的“学习成绩管理表”，实验者对结构的内容和作用比较熟悉，与仅仅将一些抽象的数字和符号作为对象的任务相比，实验者肯定会对前者执行效果的实用性更容易认同，有了认同就有了深入学习和实践的动力。

1. 基于成绩表管理的软件技术基础平台结构。

本设计平台采用了MVC架构，以图形化交互界面代替了软件基础课程传统的命令行式界面，同时又不会引入过多、过难的技术挡在学生和数据结构算法设计之间。

本实验软件平台从逻辑上可分为四个部分：界面处理、逻辑控制、实验者算法模块和例程算法模块。

界面处理



逻辑控制

实验者算法

例程算法

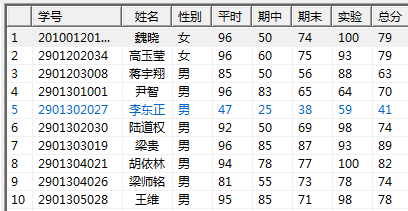
数据结构算法

数据接口

界面处理模块负责维护各种图形化界面的交互动作，以适当形式获得输入和完成输出。

逻辑控制模块负责解读界面的输入指令和参数，转变为相应数据结构的操作请求——包括函数名和参数表，调度对应的数据结构算法，完成所要求的处理。处理的结果又通过逻辑控制模块转变为可在图形化界面显示的形式，通过界面处理模块以恰当的形态显示到图形化界面的适当位置。

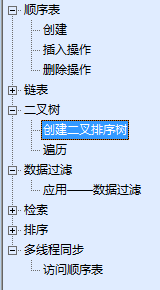
如在控制栏目中点击“创建顺序表模块”按钮后，界面处理模块将点击事件传递给逻辑控制模块，逻辑控制模块将事件解析为顺序表创建命令后，准备好输入参数，调用数据结构算法模块中的顺序表创建函数，完成相应功能。逻辑控制模块将创建好的结果，要求界面处理模块以指定方式进行显示

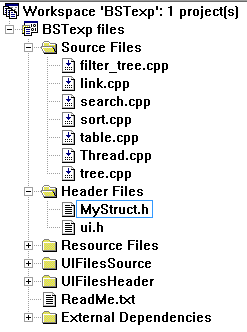
 

数据结构算法模块包含支持“成绩管理表”各项功能的众多核心算法。当然这些算法都是数据结构的各种操作。为了帮助实验者做出正确的设计，平台还以库文件的形式提供了例程，一方面实验平台可通过对比例程结果和实验者的算法结果，帮助实验者排错；另一方面也通过让实验者观察实际结果的方法，帮助学生明确最终的结果，帮助他们完成算法设计。

1. 平台的可扩展性

本软件搭建了一个友好的实验平台，采用模块化设计，可以根据需要在导航栏中增加新的功能，方法简单，在代码中增加一个新控制单元不需要修改已有的框架和代码，只需要在代码中增加新的部分即可。

**【实验步骤】**

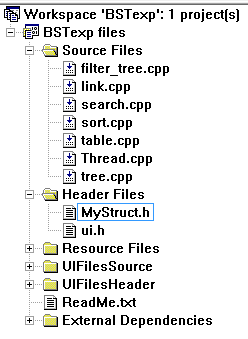
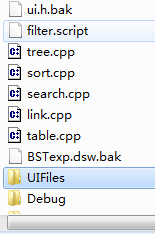
1. 认识实验平台
   1. 软件开发环境

软件设计平台提供BSTexp工程，工程中包含诸多源代码文件，采用模块化架构以后，实验者只需要按照实验要求，完成相应的数据结构算法程序，相关文件列表说明如下：

表 1重要的文件列表

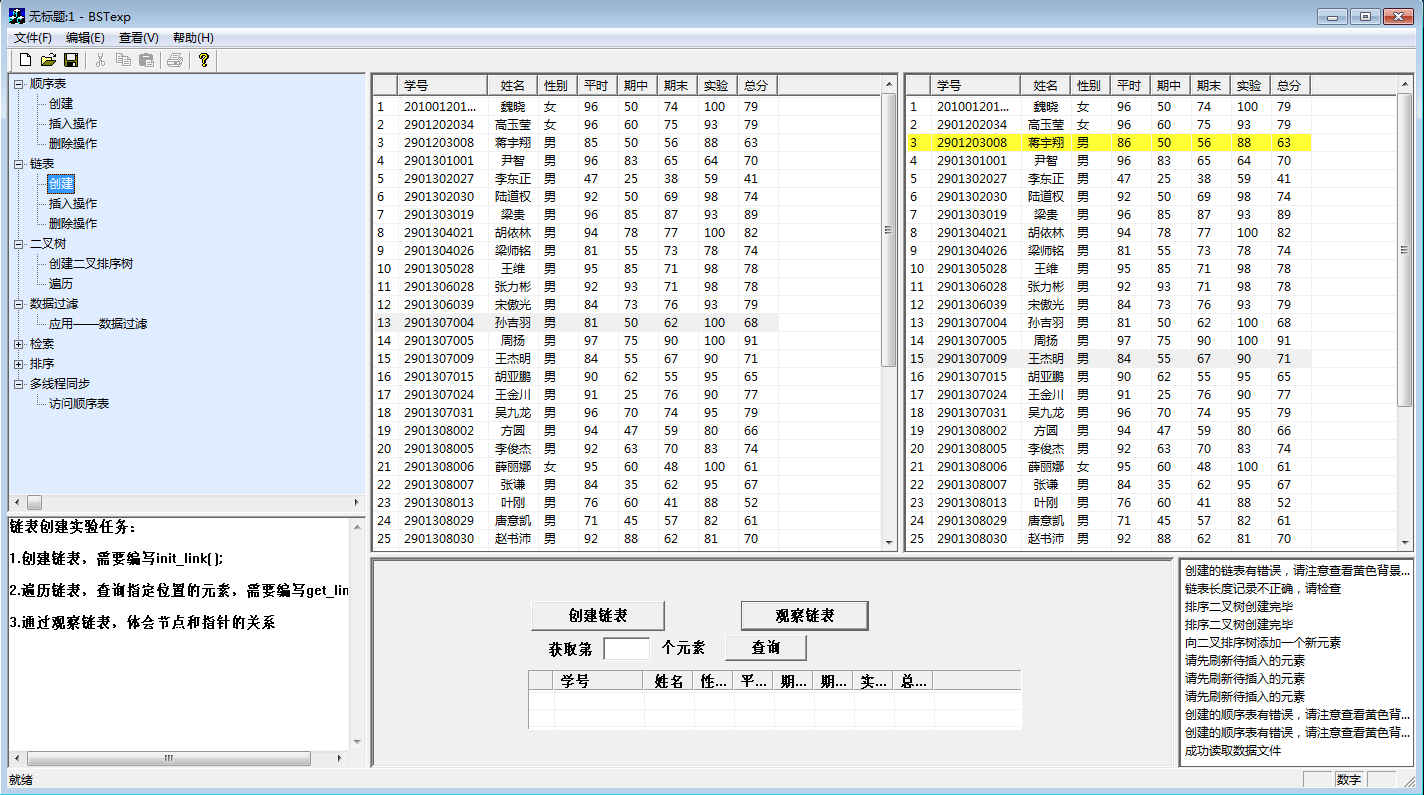
|  |  |  |
| --- | --- | --- |
| 序号 | 文件名 | 实验任务内容或功能 |
| 1 | table.cpp | 顺序表结构的创建、插入、删除算法 |
| 2 | link.cpp | 链表结构的创建、插入、删除算法 |
| 3 | tree.cpp | 二叉排序树结构创建、插入、删除算法 |
| 4 | filter\_tree.cpp | 基于二叉树结构的过滤规则算法 |
| 5 | search.cpp | 检索算法：包括顺序检索、二分检索和hash检索 |
| 6 | sort.cpp | 排序算法：包括简单插入、简单选择、冒泡、快速 |
| 7 | thread.cpp | 多线程同步操作实验 |
| 8 | MyStruct.h | 包含“逻辑控制”模块与“实验者算法”模块的数据元素定义、数据结构定义和相关操作函数接口的定义。实验者必须掌握和遵守这些定义，才能实现与实验平台其他模块的合作。 |
| 9 | ui.h | 主要包含实验平台部分界面输出的接口函数定义。必要时，实验者可以使用其中的某些函数来辅助核心算法的功能实现，如获得系统时间等。 |
| 10 | 数据.txt | 记录成绩管理表内的所有数据 |

除上表列出的重要文件外，工程中还有一些与界面特效处理有关的文件，对这部分文件，实验者不需要了解其内容和进行任何修改，它们位于源代码包的UIFiles目录下，在工程中位于UIFilesSource和UIFilesHeader目录内

* 1. 实验平台界面及主要功能

实验平台界面如下图所示：



状态信息

导航栏

帮助信息

控制栏

例程

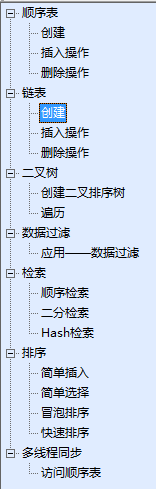
成绩表展示

实验者

成绩表展示

实验平台界面分为6个区域：导航、帮助、例程成绩表展示、实验者成绩表展示、控制和状态信息区域。每个区域都可以根据需要动态调整大小。

1. 导航区

导航栏将顺序表、链表、二叉树、检索、排序以及多线程同步等需要实验者在平台完成的设计任务，以树形结构组织起来，便于实验者选择和切换。

实验者只需要在“任务树”的子任务上单击，就能切换任务，包括：

显示该任务的帮助

切换控制区内容

展示区显示相关数据

状态区显示该任务的完成状态

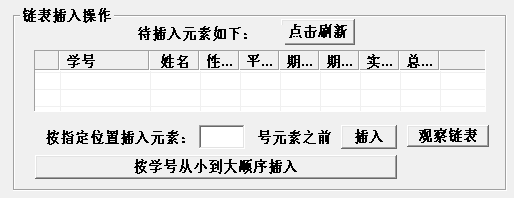
1. 控制区

不同的任务有不同的控制区。控制区内可以输入新的数据和触发指定功能的执行。

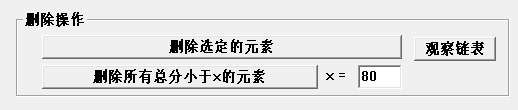
在一级任务项上点击，控制区出现欢迎页面，点击欢迎页面的文字，平台将把展示区的数据恢复到初始状态。这对于实验者切换实验环节，恢复现场十分有用。



在二级任务项上点击，控制区将配合出现相应的控制界面。如在链表结构的“插入操作”上点击，控制区将出现如下控制内容。



在“删除操作”上点击，控制区内容则切换为：



控制区内的各项功能将在后续实验中根据任务要求分别介绍。

1. 帮助区

由导航区内容触发，显示针对该任务的任务要求和帮助信息。

1. 例程数据展示区

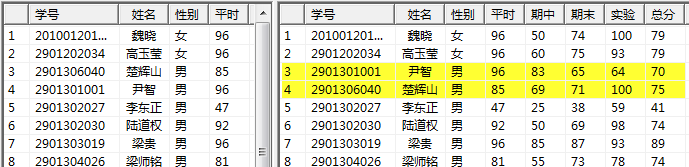
例程数据展示区以表格的形式展示相应操作完成前的成绩表内容和操作完成后正确的结果。在有些任务环节中，这个区域还以表项动态移动的方式展示某算法的正常执行过程，帮助实验者找到算法编写思路。

这个区域称之为“例程”，是因为本平台所有的任务的算法都已经在例程模块中实现了，这个区域展示的算法执行后的结果总是正确的，可以用来对比实验者算法执行的结果，进行检错和排错。

1. 实验者数据展示区

本区域将展示实验者的算法执行结果。实验者可以直观地观察到自己的算法的运行结果。

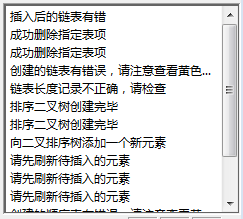
此外，实验平台还将对比实验者算法执行的结果与例程结果，将结果中不同的内容用颜色标注出来，帮助实验者检错和排错。



黄色高亮显示与例程结果不同的地方

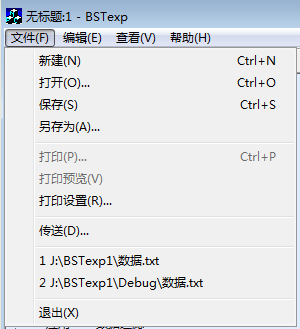
1. 状态区

显示算法执行状态。如有必要，实验者也可以通过UI接口函数AddReport()向状态区输出一行提示信息。



* 1. 请实验者通过导航栏切换各种控制，熟悉实验软件各项功能
  2. 菜单项功能

实验平台菜单项的主要功能是选择并读取成绩表文件内容，以及将处理后的结果保存到文件中。



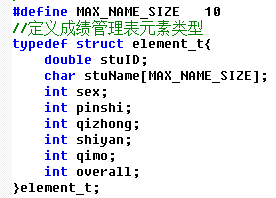
点击文件，选择打开，就可以选择平台实验所需的数据文件了。

* 1. 请实验者通过通过菜单选择数据文件，熟悉软件的数据读取功能。

1. 在实验平台编写代码
   1. 数据元素、数据结构基本定义

根据学生成绩管理表的功能需求，实验平台已经完成定义数据元素和数据结构的定义。

1）成绩管理表数据元素定义

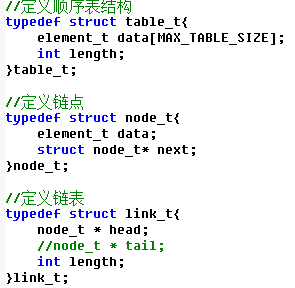


与教材的标准定义相比，采用了相同的技术路线——typedef，基本结构相似，内容针对实际需要有所丰富，包含了一条成绩表项的所有数据项，元素各数据项（结构成员）说明如下表所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 类型 | 名称 | 作用 |
| 1 | double | stuID | 学生学号，往往是一串号码，本实验平台为便于对学号的排序操作，将学号定义为double类型。  有些时候学号中也带有字符，此时应将学号定义为字符串型。相应的学号比对动作也需要调整为字符串比对操作。 |
| 2 | char | stuName[MAX\_NAME\_SIZE] | 学生姓名，字符串，40字节空间，因为中文学生的姓名一般不超过4个字/8个字节。少数名族姓名经过简化后，一般也不超过10个字。 |
| 3 | int | sex | 性别，采用整数标识，为1表示男性，为0表示女性 |
| 4 | int | pinshi | 平时成绩 |
| 5 | int | qizhong | 期中成绩 |
| 6 | int | shiyan | 实验成绩 |
| 7 | int | qimo | 期末考试成绩 |
| 8 | int | overall | 总评成绩 |

在实验设计过程中，元素内部细致的定义对数据结构的操作算法没有实质影响，但对最终的显示效果有影响，需要针对实际需求进行设计。

2）结构定义



平台采用了与教材的经典定义几乎完全相同的方式和内容，包括架构和关键成员的名称。一方面，通过这种方式缩短理论与实践的距离。另一方面，教材中的经典定义来源于对客观实际的抽象，本身也适用于各种实践场合。

这些定义作用于各数据结构的接口函数中，实验者可以根据自己的需要修改模块内部的定义，但是在接口函数的输入、输出上，必须掌握和遵守这些定义。

如：顺序表的插入操作接口函数：

*int insert\_table(table\_t \* table, element\_t data, int location);*

输入的table和data变量，输出的table变量，均满足MyStruct.h中的定义。

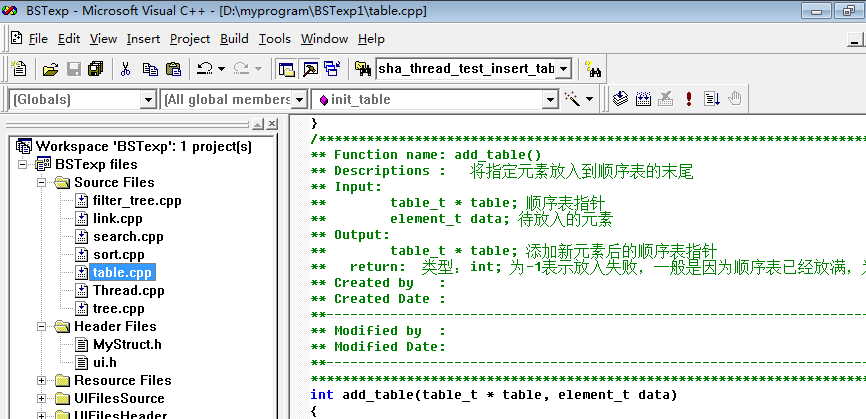
* 1. 编写任务所要求算法

在实验平台的系列实验任务中，每项任务都有针对不同的源文件，有各自不同的要求，实验者需要完成相应源文件内的相应函数的编写和调测。

前文中已经给出了各源文件对应的实验任务。本次技能训练以编写**顺序表算法程序**为例，引导实验者掌握在实验平台上编写算法的方法。

1. 打开BSTexp工程

在资源管理器中双击，使用VC6.0打开BSTexp.dsw文件。

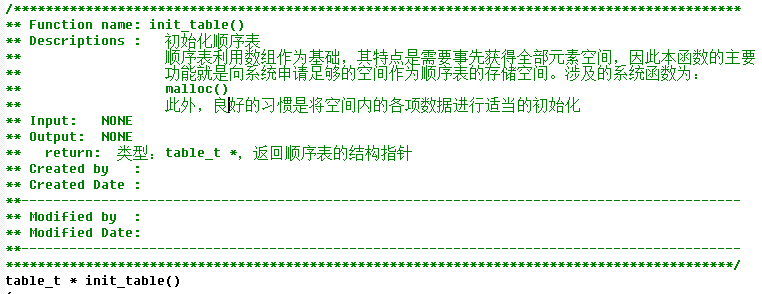


1. 打开源程序文件

在文件视图（FileView）中双击选择打开table.cpp文件。

1. 观察文件内容，找到任务要求对应的接口函数

假如本次任务要求实现顺序表的初始化操作和插入操作，我们首先需要在table.cpp文件中找到init\_table()函数和add\_table()函数。仔细阅读函数说明。包括功能说明、输入参数和输出参数要求等。（软件开发的工程训练）



函数功能说明中各字段内容

|  |  |  |
| --- | --- | --- |
| 序号 | 字段名 | 内容及作用 |
| 1 | Function name | 函数名称 |
| 2 | Descriptions | 函数功能描述。实验者应仔细阅读，这部分不仅仅描述了函数应该实现的基本功能，有时还描述了建议的技术路线和注意事项。 |
| 3 | Input | 描述了输入参数的变量名、类型和功能 |
| 4 | Output | 描述了除函数返回值外的输出参数的变量名、类型和功能。  有时输出参数与输入参数可能是同一个，但在描述中将其作为输入、输出参数时的内容和功能分别描述。在C语言语法中，在函数参数表中有时很难确定某个参数的输入、输出作用。因此在“函数功能说明”中清楚地指出每个参数的实际作用是十分必要的。 |
| 5 | return | 函数返回值，标准的输出参数。描述其类型和作用。 |
| 6 | Created by  Created Date | 第一次编写这个函数的作者和时间，是软件工程管理中版本追溯的关键依据。  同时软件知识产权法也保护每个代码作者的署名权。 |
| 7 | Modified by  Modified Date | 修改算法的作者和修改时间  如果有多次比较成型的修改，则为每次修订增加一个条项。 |

请在init\_table函数和add\_table函数的说明中添加实验者姓名和日期。

1. 添加算法代码，实现相应功能

为init\_table添加如下代码：

*table\_t \* init\_table()*

*{*

*table\_t \* t;*

*t->length = 0;*

*return t;*

*}*

为add\_table()添加以下代码

*int add\_table(table\_t \* table, element\_t data)*

*{*

*table->length ++;*

*table->data[table->length] = data;*

*return 0;*

*}*

编译，解决语法错误

1. 利用实验平台调测算法程序

由于实验平台采用了事件驱动机制，调测程序的方法应采用“断点调试”，在编译完成算法模块后，需要在特定语句处设置断点，才能进入调试状态进行检错和排错。

1. 观察提示信息、相关动画等，拟定算法设计思路

为帮助实验者思考和拟定算法的设计思路，实验平台不仅为每个函数给出了说明和提示，还针对部分典型算法设计了动画展示算法的正确执行过程，实验者可以通过观测，体验算法的运行过程，找出思路。

如顺序表的“插入操作”任务中，点击控制区的动画演示按钮，

点击动画演示



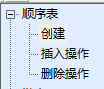
实验平台会弹出插入算法的动态执行过程演示窗口。实验者可以在这个窗口中观察到算法执行的全部过程



图中，高亮条逐个向上移动，提醒实验者算法的执行过程和特征。

1. 执行程序

在导航区点开顺序表，点击“创建”

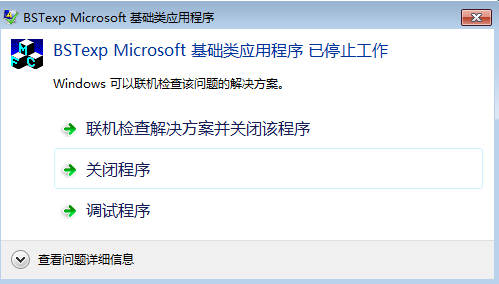


控制区就会显示顺序表创建任务的控制界面



此时，我们观察到实验者展示区是空的，因为此时尚没有创建成绩管理表的顺序存储结构。

点击创建顺序表按钮



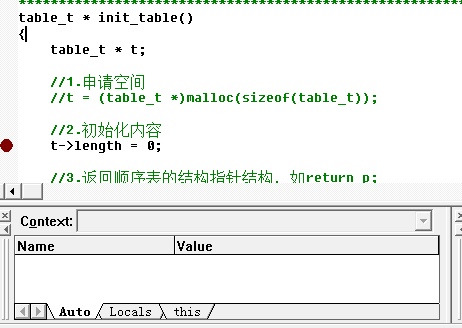
程序提示出错。（在不同系统下，如WinXP，报错的信息可能不同），这样就需要我们利用调试工具来检查错误。

1. 设置断点，体验事件驱动式的程序框架

关闭出错的程序，在init\_table中t->length = 0；一句设置断点：



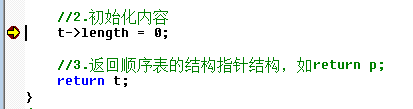
按F5进入调试状态。



断点位置

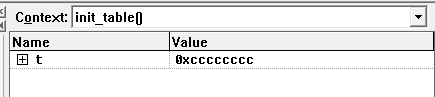
1. 调试程序

在程序界面上点击“创建顺序表按钮”，就会触发init\_table算法被调度执行。我们可以观察到程序在调试状态下，停止在所设置的断点处（黄色箭头出现）。

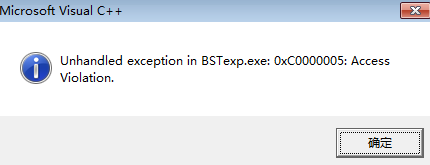


黄色箭头表示程序将要执行的语句

此时观察变量窗口中执行变量t的值为0xcccccccc，这个值不是一个有效的指针，或者说t没有记录一片有效的内存起始位置。



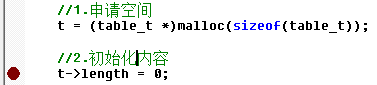
按F10，执行t->length = 0；



程序报错，这个错误便是操作非法指针的结果。

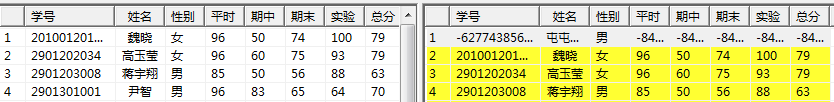
1. 修正算法

顺序表创建需要生成一个有效的空间，我们将按照函数说明中的提示信息，利用malloc来“生成”——向系统申请一片有效空间。



1. 利用结果对比来检错

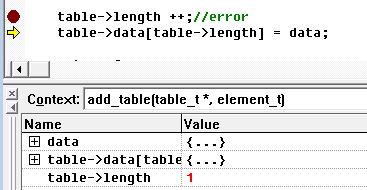
重新编译，执行程序。



此时，我们发现“创建顺序表”后的执行结果都是黄色，表示这些表项的内容与所要求的正确结果不同，我们还将特别关注到第一个表项的内容完全不正确。

在add\_table()内设置断点。点击“创建顺序表”按钮后，先触发init\_table的执行，接着便逐个触发add\_table函数，要求将输入的元素插入到顺序表尾部。

我们观察第一个被触发的add\_table中，元素实际被放置在了下标为1的位置，而不是理论上下标为0的位置。

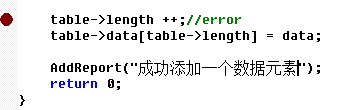


下标不应该为1

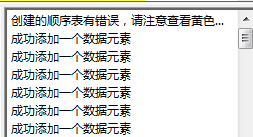
1. 利用状态区，显示可用信息

基本上我们利用调试工具和对比工具，能够检测出几乎所有的程序错误，当程序员需要时可以在算法流程中输出一些提示信息。

请在add\_table中增加AddReport(),向状态栏输出一行信息。



程序执行了创建顺序表以后，将在状态区有以下显示：



此外，还可以通过ShowElement（），向实验这展示区的最末尾一行显示1个元素。

**【思考题】**

1. 如果在测试中需要将平台内成绩表的数据重置为初始状态，应该怎么做？

答：

点击导航区1级任务条，在控制区的“软件技术基础”上点击，就能恢复初始状态的数据

1. 如果需要在模块中使用全局变量，请问该变量的声明应该放在什么位置？

答：按照C语言格式要求，在所有的函数外面，尽量在cpp文件的头部申明全局变量。

1. table.cpp中的insert\_table函数将因为什么事件而被触发执行？

答：

在顺序表的插入操作控制区中，点击插入按钮，就会触发insert\_table

**实验一 XXXX**

1. 实验目的和任务

（该实验项目要求学生掌握哪些原理和方法，提高哪方面的能力，要求学生完成哪些任务）

1. 实验预习与思考

（本节课相关预备知识，要求学生在课前自主学习解决的问题，某些实验若存在有安全隐患、在预习时就要提示实验时如何注意人员或仪器的安全要点）

1. 实验原理
2. 实验内容

（若实验中使用了某种专用的设备或电路板，应在介绍实验内容之前较系统地介绍专用设备或电路板的组成情况和使用方法）

1. 实验报告要求
2. 实验仪器设备
3. 思考题