# 实验一　线性表、栈、队列及其应用（1.5~2.5 次上机）

## 【上机时间】

第 1 次，第 2 次

## 【实验目的】

深入理解线性表、栈和队列的特性，领会它们各自的应用背景。熟练掌握它们在不同存储结构、不同的约定中，其基本操作的实现方法与差异。体会以下几点（注意你所做的约定）：

1、线性表：顺序表（插入、删除）、链表（查找、插入、删除）

2、栈：顺序栈（栈空/栈满条件，入栈/出栈）、链栈（栈空条件，入栈/出栈）；

3、队列：链队列（队空条件，入队/出队）、顺序队列/循环顺序队列（队空/队满条件，入队/出队）；

## 【实验内容】

本次实验共四个题目，可任选其中的一题或多题。

**1.　链表操作**

1）从一个给定的数据集合中随机、可重复地选取数据元素，并通过尾插法构建一个双向链表，然后设计算法删除该链表中的重复元素。

2）构建一个单链表和一个无头节点的循环链表，将其链接成一个有环的链表，然后设计算法确定环的入口点，并计算环的长度。

**2.　算术表达式求值的演示**

教材 P95 案例 3.3 中的描述。

**案例 3.3：表达式求值。**

【案例分析】

任何一个表达式都是由操作数（operand）、运算符（operator）和界限符（delimiter）组成的，统称它们为单词。一般地，操作数既可以是常数，也可以是被说明为变量或常量的标识符；运算符可以分为算术运算符、关系运算符和逻辑运算符 3 类；基本界限符有左右括号和表达式结束符等。为了叙述的简洁，在此仅讨论简单算术表达式的求值问题，这种表达式只含加、减、乘、除 4 种运算符。读者不难将它推广到更一般的表达式上。

下面把运算符和界限符统称为算符。

我们知道，算术四则运算遵循以下 3 条规则：

（1）先乘除，后加减；

（2）从左算到右；

（3）先括号内，后括号外。

根据上述 3 条运算规则，在运算的每一步中，任意两个相继出现的算符$\theta_1$和$\theta_2$之间的优先关系，至多是下面 3 种关系之一：

$\theta_1 < \theta_2$　$\theta_1$的优先权低于$\theta_2$

$\theta_1 = \theta_2$　$\theta_1$的优先权等于$\theta_2$

$\theta_1 > \theta_2$　$\theta_1$的优先权高于$\theta_2$

表 3.1 定义了算符之间的这种优先关系。

表 3.1　　算符间的优先关系

| $\theta_1$ \ $\theta_2$ | + | − | * | / | ( | ) | # |
|---|---|---|---|---|---|---|---|
| + | > | > | < | < | < | > | > |
| − | > | > | < | < | < | > | > |
| * | > | > | > | > | < | > | > |
| / | > | > | > | > | < | > | > |
| ( | < | < | < | < | < | = | |
| ) | > | > | > | > | | > | > |
| # | < | < | < | < | < | | = |

由规则（1），先进行乘除运算，后进行加减运算，所以有“+” < “*”；“+” < “/”；“*” > “+”；“/” > “+”等。

由规则（2），运算遵循左结合性，当两个运算符相同时，先出现的运算符优先级高，所以有“+” > “+”；“−” > “−”；“*” > “*”；“/” > “/”。

由规则（3），括号内的优先级高，+、−、*和/为$\theta_1$时的优先性均低于“(”但高于“)”。

表中的“(” = “)”表示当左右括号相遇时，括号内的运算已经完成。为了便于实现，假设每个表达式均以“#”开始，以“#”结束。所以“#” = “#”表示整个表达式求值完毕。“)”与“(”、“#”与“)”以及“(”与“#”之间无优先关系，这是因为表达式中不允许它们相继出现，一旦遇到这种情况，则可以认为出现了语法错误。在下面的讨论中，我们暂假定所输入的表达式不会出现语法错误。

【案例实现】

为实现算符优先算法，可以使用两个工作栈，一个称做 OPTR，用以寄存运算符；另一个称作 OPND，用以寄存操作数或运算结果。



**3.　N-皇后问题**

假设有一 N×N 的棋盘和 N 个皇后，请为这 N 个皇后进行布局使得这 N 个皇后互不攻击(即任意两个皇后不在同一行、同一列、同一对角线上。

**要求：**

1) 输入 N，输出 N 个皇后互不攻击的布局；

2) 要求用非递归方法来解决 N-皇后问题，即自己设置栈来处理。

**4. 背包问题**

假设有一个能装入总体积为 T 的背包和 n 件体积分别为 $w_1$ , $w_2$ , … , $w_n$ 的物品，能否从 n 件物品中挑选若干件恰好装满背包，即使 $w_1 + w_2 + \ldots + w_n = T$，要求找出所有满足上述条件的解。

例如：当 T=10，各件物品的体积{1，8，4，3，5，2}时，可找到下列 4 组解：

（1，4，3，2）

（1，4，5）

（8，2）

（3，5，2）。

**提示**：可利用回溯法的设计思想来解决背包问题。

首先将物品排成一列，然后顺序选取物品装入背包，假设已选取了前 i 件物品之后背包还没有装满，则继续选取第 i+1 件物品，若该件物品“太大”不能装入，则弃之而继续选取下一件，直至背包装满为止。但如果在剩余的物品中找不到合适的物品以填满背包，则说明“刚刚”装入背包的那件物品“不合适”，应将它取出“弃之一边”，继续再从“它之后”的物品中选取，如此重复，直至求得满足条件的解，或者无解。

## 【实验要求】

1. 要求所编写的程序应采用 c 或 c++语言；
2. 必须带命令行参数；
3. 必须通过命令行参数指定输入、输出文件的文件名，练习对文件的操作。

## 【检查期限】

1. 上机内容检查时间：第 2~3 次上机时，以第 3 次上机为截止时间(10.29)；