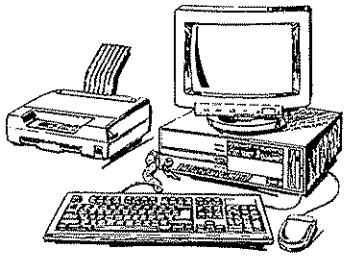


4 程式設計

重點整理

一、前序（preorder），中序（inorder），後序（postorder）的轉換
二、何謂recurrive
三、何謂time complexity
四、時間複雜度的排列
- ❖ $O(1)<O(\log n)<O(n)<O(n \log n)<O(n^2)<O(2^n)$

五、何謂link list
六、何謂binary tree，binary search tree，234 tree
七、何謂stack，queue
八、各種sort的方式
- ❖ insertion sort，bubble sort，quick sort，merge sort，heap sort，radix sort

九、何謂演算法？演算法的四個條件為何
十、call by value，call by address，call by name三者之比較
十一、全域變數和區域變數之比較
十二、何謂ADT（抽象資料型態）
十三、何謂物件導向程式語言

4.1 資料結構與演算法

㈠遞迴的觀念與實際運算。

㈡一些演算法的時間複雜度要記熟，尤其是排序（sort）的比較。

㈢陣列位址的計算。

㈣堆疊與佇列的原理。

㈤前序、中序、後序在堆疊中動作的情形。

㈥搜尋最常考的是二元搜尋法。

㈦樹的一些名詞，如level、depth、AVL等。

㈧二元樹的追蹤：

1. 中序（LDR）：左中右（inorder）。
2. 後序（LRD）：左右中（postorder）。
3. 前序（DLR）：中左右（preorder）。

㈨給定前、中序或後、中序求二元樹。

㈩圖（graph）是相當熱門的考題，要會求最短距離、DFS、BFS、MST、臨界路徑、相鄰矩陣等問題。

㈪資料結構的內容相當繁多，本書只針對一些較常出現的考題作重點複習，讀者可以參考資料結構的聖經－霍洛維茲（Horowitz）大師等人所著的「Data Structure」，而且有使用PASCAL及C語言所寫的版本。

・題型1・前序、中序、後序表示式與轉換

以下何者爲A*(B + C/(D − E))的後序式（postfix expression）？
(A)DE − C/B + A*　(B)ABCDE − / + *　(C)DE − CB/ + A*
(D)BCDE − / + A*。

(B)

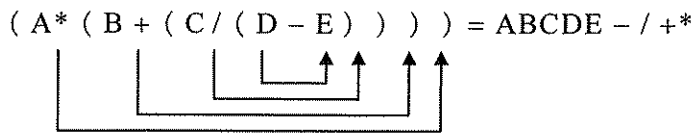
1. 中序表示式化成前序或後序表示式時，步驟如下：

Step 1：將運算優先順序以括弧來表示。

Step 2：⑴化成前序：將運算子移到左邊最接近的括弧上，由最裡面的運算子開始移，不可重疊。

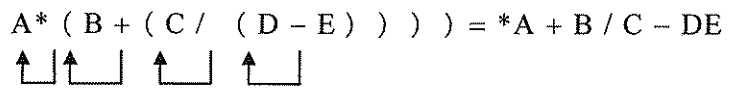
⑵化成後序：方法同上，只是將運算子移到最接近的右邊括弧。

2. A*(B + C/(D – E))：先括上括弧以示運算優先順序，



故選(B)。

3. 此題若化成前序表示法則爲：



・類題1.1・後序表示式計算

後序運算式（postfix expression）“235 × 27 – / + 63 × +”中的運算元（operand）皆爲個位數，而運算子（operator）皆爲二元運算子，則其運算結果爲：　(A)10　(B)15　(C)17　(D)23。

(C)

Step 1：先將後序式轉換成中序式，逐一加括弧

Step 2：計算中序式之值：

235 × 27 – / + 63 × +

→2(3 × 5)(2 – 7)/ + (6 × 3) +

→2(3 × 5)/(2 – 7) + (6 × 3) +

→2 + (3 + 5)/(2 – 7) + (6 × 3)

故2 + (3 + 5)/(2 – 7) + (6 × 3) = 17。

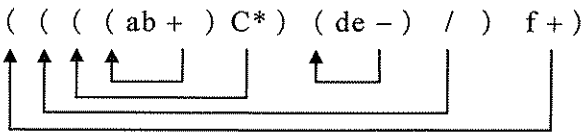
・類題1.2・

將後置配法（postfix notation）運算式ab＋c*de－/f＋ 轉換成前置配法（prefix notation）： (A) ＋ab*c/－de＋f (B) ＋ab*c－/de＋f (C) ＋/*＋abc－def (D) ＋/*＋ab－def。

(C)

1. 後序→前序：先將運算關係用括弧表示，再將運算子移到最靠近的左邊括弧。

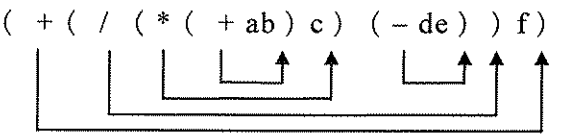
例：



→＋/*＋abc－def。

2. 前序→後序：先用括弧表示運算關係，再將運算子移到最靠近的右邊括弧。

例：



→ab＋c*de－f＋。

・題型2・二元樹追蹤

關於算術運算式表示法，下列敘述何者不正確？ (A)(A＋B)/C－D*E的前序式（prefix）爲－/＋ABC*DE (B)(A＋B)/C－D*E的後序式（postfix）爲AB＋C/DE*－ (C)中序式（infix）比後序式利於翻譯程式（assembler, compiler, ……）處理 (D)後序式算術運算式的執行通常會利用堆疊（stack）資料結構。