

《程式語言》

試題評析

今年程式語言的考題，較特別的是五題中有兩題與程式設計有關，包括第一題以C++來設計鏈結串列(linked list)時，如何透過三種不同的程式設計技巧，來解決C++類別中私有 (private) 區資料的存取限制；以及第三題要求以遞迴及非遞迴的方式來設計二元搜尋(binary search)副程式，並比較遞迴及非遞迴語言之差異，屬於較單純的程式設計問題。這兩題可能要花較多的時間來回答。第二題考捷徑計算與非捷徑計算運算子之差異和優缺點，第四題考運算式文法的明確性(ambiguity)、運算子優先順序及結合性等問題，第五題則考程式語言中程式可靠度之評核指標，都不是很難的問題。因此，若能妥善運用時間，先掌握較有把握的題目回答，便能獲得好的分數。整體而言，今年的考題對一般的考生而言，應可拿到45~55分，成績較好者可得55~70分。

一、以C++語言為例，為了保有最佳的資料隱藏性 (Information hiding)，一個鏈結 (Linked list) 資料結構常被以下列兩個級 (Class) 來定義：(二十分)

```
class listnode{
    private:
        char data[32]; //for node data
        listnode *pointer;
};
class listhead{
    public:
        // some list manipulation operations
    private:
        listnode *head;
};
```

(一)但上列的程式並無法通過編譯器 (Compiler) 之編譯，理由何在？

(二)舉出三種不同之改進方法，使上列之程式可以在保持同樣以該兩個級以及資料隱藏所定義，又能順利完成鏈結資料結構所需之功能，並說明改進之理由。(沒說明理由者不予計分)

答：

1.以這兩個類別所定義的鏈結串列，其標頭指標記錄在listhead類別個體中，而串列中的節點則記錄在listnode類別之個體中。由於listnode類別中，data及pointer兩欄位皆宣告在private區，都是不允許外界存取的，這個限制將使得在listhead類別之public區所定義的串列處理運算 (list manipulation operations)，對於串列節點內容 (即data和pointer) 之存取都變成不合法 (然而這些存取是必須的)，因此這個程式將無法通過編譯器之編譯。

2.三種改進方法如下：

(1)將listhead類別宣告為listnode類別之伙伴 (friend)，亦即將listnode類別宣告如下：

```
class listnode {
    private:
        char data [32];
        listnode *pointer;
        friend listhead;
};
```

如此在listhead類別中即可隨意存取listnode類別中的data和pointer，因為宣告listhead為listnode之伙伴時，在listhead中可具有與listnode相同的存取權利。

(2)將data和pointer改宣告在listnode類別之protected區，並且將listhead宣告為listnode之衍生類別 (即子類別)，程式修改如下：

```
class listnode {
    protected:
        char data [32];
        listnode *pointer;
};
class listhead: private listnode {
    public:
        void print_list (void) {
            listnode *ptr = head;
            while (ptr != NULL) {
                printf ("%s", ptr -> data);
            }
        }
};
```


優點：包含side effect的布林運算式之執行不會發生問題。

缺點：A.較浪費運算時間。

B.程式設計上較為複雜。

- 2.(1)ADA語言中，and與or為非捷徑運算之運算子，而and then與or else則為捷徑運算之運算子。
- (2)C語言的&&與||皆為捷徑運算之運算子。
- (3)PASCAL的and與or皆為非捷徑運算之運算子。

三、如果在一整數陣列data[20]中存放有按大小次序排列的整數，另有一整數x：(二十分)

- (一)試以反覆(Iteration)之技術，設計二分查尋(Binary search)次程式(Subprogram)，查尋x是否存在data陣列中。
- (二)試以遞迴(Recursion)之技術，設計該二分查尋次程式。
- (三)指出有支援遞迴程式設計之程式語言與不支援遞迴程式設計之程式語言，最主要的不同何在？

答：

1.以FORTRAN撰寫非遞迴副程式如下：

```
SUBROUTINE BINARYSEARCH (DATA, N, X, I)
  INTEGER DATA (N), X
  INTEGER LOW, HIGH, MIDDLE
  LOW = 1
  HIGH = N
100 IF (LOW, LE HIGH) THEN
  MIDDLE = (LOW + HIGH) / 2
  IF (DATA (MIDDLE) 50X) THEN
    I = MIDDLE
    RETURN
  END IF
  IF (DATA (MIDDLE) LT. X) LOW = MIDDLE + 1
  IF (DATA (MIDDLE) GT. X) HIGH = MIDDLE - 1
  GOTO 100
```

```
ELSE
  I = 0
  RETURN
ENDIF
END
```

此副程式可從N個資料的DATA陣列中搜尋是否有X存在，若有則傳回出現X的陣列註標I，若沒有則令I值為0。

2.以C語言撰寫遞迴副程式如下(若傳回值I不為0表示Data(I) = X，若傳回0表示找不到)：

```
int Binary_Search (int data [], int x, int low,
                  int nigh)
{
  int middle;
  if (low <= high) {
    middle = (low + high) / 2;
    if (data [middle] == X) return (middle);
    if (data [middle] < X)
      return (Binary_Search (data, X, middle + 1, high));
    if (data [middle] > X)
      return (binary_Search (data, X, low, middle - 1));
  }
  else
    return (0);
}
```

(三)最主要的不同在於支援遞迴程式設計的語言中，實作副程式所需的活動記錄為動態配置，而不支援遞迴程式設計的語言中，其副程式之活動記錄可為靜態配置。

四、就下列巴納記法(BNF, Backus-Naur notation)之文法(Grammar):(二十分)

```
<assign> → <id> := <expr>
<id> → A|B|C
<expr> → <expr> + <expr> | <expr> * <expr> | (<expr>) | <id>
```

- (一)如果以該文法定義吾人常用的包含加與乘兩運算子之算術表式(Arithmetic expression)，指出其三大缺點。
- (二)寫出改進所指出缺點後之新文法。

答：

1.此文法的三大缺點如下：

- (1)此文法為不明確文法(ambiguous grammar)。
- (2)此文法無法判斷加與乘兩運算子之優先次序。
- (3)此文法無法決定加與乘兩運算子之結合性。

2. 為改進上述三個缺點，可將文法改寫如下：

```

<assign>  <id> := <expr>
<id>      A | B | C
<expr>    <expr> + <term>
           | <term>
<term>    <term> * <factor>
           | <factor>
<factor>  (<expr>)
           | <id>

```

如此可得一明確文法 (unambiguous grammar)，且乘號“*”之優先順序高於加號“+”，同時“*”與“+”皆具有左結合性。

五、舉出並說明四種針對程式語言提供程式可靠度 (Reliability) 之評核指標。(二十分)

答：

程式語言提供程式可靠度的評核指標如下

(一) 型態檢查 (type checking)

1. 在編譯時期或執行時期檢查程式中是否有資料型態上的錯誤稱之。
2. 有提供型態檢查者其程式可靠度較高。
3. 編譯時期型態檢查所發現的錯誤，其修改成本又低於執行時期型態檢查之錯誤修改成本。

(二) 例外處理 (exception handling)：

1. 當系統偵測到程式執行時由硬體或軟體所引發的不正常事件或錯誤時，可自動進行處理，再恢復程式執行的能力稱之。
2. 有提供例外處理功能者其可靠度較高。

(三) 別名之限制 (aliasing)

1. 允許以兩種或多種名稱來存取同一塊記憶體稱為別名。
2. 程式語言若有限制別名之使用可提高程式之可靠度。

(四) 可讀性 (readability) 與可寫性 (writability)

1. 可寫性是指容易建立程式 (即以較自然方式撰寫程式) 之特性，可寫性愈高愈容易修正程式之錯誤。
2. 可讀性是指容易閱讀和容易了解的特性，程式的可讀性將愈容易撰寫也愈容易修改。