《程式語言》

試題評析	今年程式語言的考題大多是簡單的題型，並沒有艱深的問題，部分試題屬於程式設計概要的範圍，因此並不難答。第一題考程式設計問題，要求分別撰寫遞迴副程式及非遞迴副程式來計算 Fibnacci 數，但題目中 Fibnacci 數的定義有誤，考生要能分辨並加以改正，才能正確作答；第二題考程式語言的評鑑準則，以及就所熟悉的一種語言進行評鑑，這一題算是稍較難答的；第三題要求考生舉例說明物件導向語言的三大特徵：資料抽象化、繼承及動態繫結，這一題需要以物件導向語言設計一個範例程式來做說明，要拿到高分需費一番功夫才行；第四題考的是各種演算法複雜度的分辨，並依照由小到大排列出來，只要觀念正確，這一題可以最快答出來；第五題考傳值呼叫及傳位址呼叫這兩種參數傳遞法的使用時機，並舉例說明，這也是容易作答的一題。總括而言，今年的試題對一般的考生而言應有 60-70 分的機會，程度較好者可望拿到 80-90 分的好成績。

一、請以 BASIC, PASCAL, FORTRAN, C, JAVA, COBOL 中任一種語言寫一程式來計算 Fibonacci 數，其定義如下：

$$f(n) = f(n+1) + f(n+2), \quad n > 1;$$
$$= 1, \quad n = 1, \text{ 或 } 0$$

(一)請使用遞迴副程式（recursive call，即副程式呼叫本身）。(10 分)

(二)不使用遞迴副程式。(10 分)

【擬答】

本題 Fibnacci 數之定義有誤，正確之定義應如下：

$$f(n) = \begin{cases} f(n-1) + f(n-2), & n>1 \\ 1, & n=1 \text{ 或 } 0 \end{cases}$$

以 C 語言撰寫遞迴副程式 Fib1 及非遞迴副程式 Fib2，並進行測試如下

```
#include <stdio.h>
int Fib1(int n);
int Fib2(int n);

void main(void)
{
     int   n;
     printf("Enter an integer n: "); scanf("%d", &n);
     printf("Recursive Fib1(%d) = %d\n", n, Fib1(n));
     printf("NonRecursive Fib2(%d) = %d\n", n, Fib2(n));
}

int Fib1(int n)
{
     if ( n == 1 || n == 0)
          return 1;
     else
          return Fib1(n-1) + Fib1(n-2);
}

int Fib2(int n)
{
     int   i, Fib, n1 = 1, n2 = 1;
     if ( n == 1 || n == 0 )
          return 1;
     else
     {
```

```
        for ( i = 2; i <= n; i++ ) {
            Fib = n1 + n2;
            n2 = n1;
            n1 = Fib;
        }
        return Fib;
    }
}
```

執行結果：

```
Enter an integer n: 8
Recursive Fib1(8) = 34
NonRecursive Fib2(8) = 34
```

二、(一)試說明評鑑一種程式語言優劣之準則。(10 分)
(二)試以上述準則評鑑一種你（妳）所知之程式語言。(10 分)

【擬答】

一、 評鑑一程式語言優劣之準則如下：

(一)高可讀性(readability)：容易閱讀與容易了解之特性。

(二)高可寫性(writability)：容易建立程式之特性。

(三)高可靠性(reliability)：語言的設計可讓使用者不易犯下錯誤，即使犯錯也很容易找出來之特性。

(四)低成本(cost)：程式語言的成本包括

1.學習成本。
2.撰寫程式之成本。
3.編譯成本。
4.執行成本。
5.編譯程式之建構成本。
6.低可靠性之成本。
7.維護程式之成本。

二、 述準則評鑑 PASCAL 語言如下：

(一)此語言具有嚴謹的語法及結構化程式設計能力，也有豐富的資料型態，故具有很高的可讀性。

(二)因不支援資料抽象化，運算式的表達能力亦不高，致使其可寫性欠佳。

(三)PASCAL 屬於接近強型態(strongly-typed)之語言，除了可變記錄之外，型態檢查能力佳，但因爲提供例外處理功能，故可靠性只能算上可。

(四)在成本方面，PASCAL 語言因爲簡單，通常做爲初學者之學習語言，故學習成本很低，執行成本及編譯程式之建構成本亦很低；但因可寫性欠佳，撰寫程式之成本較高；另外因 PASCAL 程式無法個別編譯，其編譯成本亦較高；而低可靠性之成本和維護程式之成本皆稍高。

三、請以任何一程式語言舉例並說明物件導向程式語言（object-oriented programming languages）之資料抽象化（data abstraction）、繼承（inheritance）及動態繫結（dynamic binding）性質。(20 分)

【擬答】

以 C++語言爲例，考慮下列三個類別之定義

```
class shape{
            private:
                int   x, y, color;
            public:
                void set_xy(int x1, int y1) {
                    x = x1; y = y1;
                }
                int get_x(void) { return x; }
                int get_y(void) { return y; }
                virtual void draw() { … } //draw a point
};
class circle: public shape {
```

```
            private:
                int   radius;
            public:
                virtual void draw() { … } //draw a circle
};
class rectangle: public shape{
            private:
                int   length, width;
            public:
                virtual void draw() { … } //draw a rectangle
};
```

一、資料抽象化：C++語言以類別做爲資料抽象化之語法單元，類別中可包藏資料和程序之定義；例如 shape 類別中定義了 x、y、color 等三個資料，以及 set_xy()、get_x()、get_y()、draw()等四個程序。

二、繼承：C++語言中將一個類別定義爲另一類別之子類別時，子類別可繼承其父（超）類別中所定義的資料和程序；例如 circle 類別與 rectangle 類別皆定義爲 shape 之子類別，因此 circle 類別與 rectangle 類別皆可繼承 shape 類別中所定義的資料和程序。

三、動態繫結：物件導向語言的動態繫結是指以動態方式來決定程序呼叫（經由訊息傳遞）與程序定義之間的繫結，這是一種讓相同程式碼具有多種執行模式的多模(polymorphism)功能。C++語言是以虛擬函數(virtual function)來提供動態繫結功能，例如上述三個類別中的 draw()程序即爲一虛擬函數，當傳送訊息要執行 draw()程序時，可依據訊息的接收者爲何來決定要執行那一個類別中的 draw()，考慮下列程式碼：

```
shape   shape_obj, *ptr;
circle   circle_obj;
rectangle   rectangle_obj;
…
if ( a > b ) ptr = &circle_obj;
if ( a = b ) ptr = &shape_obj;
if ( a < b ) ptr = &rectangle_obj;
ptr->draw();
```

其中 ptr->draw()爲一個訊息，要傳送給 ptr 所指的物件（可能爲 circle_obj 或 shape_obj 或 rectangle_obj），且 ptr->draw()所要呼叫執行的程序，將是 ptr 所指物件的類別中之 draw()，亦即此訊息將動態地繫結某一類別中的 draw()虛擬函數。

四、請由小至大列出下述函數之複雜度：

(一)100000, nlogn, n^2, $(3/2)^n$, loglogn。(10 分)

(二)lognlogn, n^2logn, $2^{lgnlglgn}$, 3^{lgn}, $n2^{lglgn}$。(10 分)

【擬答】

一、00000 < loglogn < nlogn < n^2 < $(3/2)^n$

二、ognlogn < n^2logn < $n2^{lglgn}$ < 3^{lgn} < $2^{lgnlglgn}$

五、試分別舉例並說明參數傳遞方法中以值呼叫（call by value）及以址呼叫（call by reference or call by address）之較佳使用時機。(20 分)

【擬答】

一、傳值呼叫(call by value)較佳的使用時機是當呼叫程式只需將實際參數值傳給被呼叫程式，而不需從被呼叫程式中取得執行結果之情況。例如要在 PASCAL 程式中撰寫一個可接受參數 n 並畫一條 n 個'*'號組成之直線的副程式 drawline，便適合採用傳值呼叫來傳遞參數，如下之程式碼：

```
procedure drawline(n : integer);
var i : integer;
begin
for i := 1 to n do write('*');
writeln
end;
```

當呼叫 drawline(10)可畫一條 10 個'*'號組成之直線，而呼叫 drawline(50)可畫一條 250 個'*'號組成之直線。

二、傳位址呼叫(call by address) 較佳的使用時機是當呼叫程式不只要將實際參數值傳給被呼叫程式，也需要從被呼叫程式中取得執行結果之情況。例如要在 PASCAL 程式中撰寫一個可將兩個變數交換的副程式 swap，便適合採用傳位址呼叫來傳遞參數，如下之程式碼：

```
procedure swap(var a : integer; var b : integer);
var temp : integer;
bagin
temp := a;
a := b;
b := temp;
end;
```

則執行下列程式碼：

```
x := 5; y := 8;
swap(x, y);
```

可將實際參數 x 與 y 的值交換，得到 x = 8，而 y = 5。