《程式語言》

試題評析

今年的考題以基本面爲主，而且只包括了物件導向、指標與遞迴三個面向，沒有考到特殊的程式語言或併行程式，對程度中等的同學而言較爲吃香。

第一題：爲物件導向的基本觀念，屬於相當基本的考題。唯與往年不同之處是明確的要考生寫出物件導向三個特性的優點，同學必須對這三個特性有深刻的了解才能拿到高分。

第二題：爲傳址呼叫的基本考題，同學只要能掌握C語言中的傳址呼叫的特性，小心作答便能寫出正確答案。

第三題：頗爲類似98年高考的題目，只要對指標運算的概念、字元整數交互運算的概念熟悉便能作答。

第四題：遞迴呼叫與活動紀錄概念對大部分的考生應該駕輕就熟，唯一不同的是今年出現了case sensitive的解釋，雖然概念很簡單，但以英文出題同學不一定能馬上理解，因此較容易出現誤解的遺憾。

第五題：物件導向程式設計的題目，只要熟悉物件的宣告與使用語法，對同學而言屬於相當好拿分數的題目。

綜觀而論，今年的程式語言題目較往年簡單許多，程度中等的同學應可拿到60-70分，程度較好的同學拿到90分以上也並非難事。

一、物件導向程式設計的內涵有繼承（inheritance）、函式多型（function polymorphism）與封裝（encapsulation）等，請分別敘述上述三個項目之要義與優點。（21分）

答：

茲將繼承（inheritance）、函式多型（function polymorphism）與封裝（encapsulation）之要義與優點整理如下表：

特性	要義	優點
繼承	若一類別繼承另一類別，則被繼承的類別稱爲父類別，繼承的類別稱爲子類別。子類別可以使用或複寫父類別的屬性及操作，也可加入新的屬性及操作。以C++爲例，宣告類別A後，可以在類別B使用public或private等關鍵字對A繼承，例如class B: public A。	可以達成程式碼共享（reuse），減少重複程式碼撰寫，加速開發、修改、降低錯誤率、維護成本。
函式多型	相同程式碼有多種執行模式，爲程序間傳遞訊息時以動態的方式決定執行的操作所產生之結果。以C++爲例，C++的父類別中以virtual保留子宣告的函數即爲虛擬函數，此種函數在子類別中重新定義之後，則呼叫這些虛擬函數的訊息在處理時，將可依照訊息的接收物件爲何來決定要執行在那個類別中定義的虛擬函數。	讓同樣的程式碼，即使經過編譯，亦能在執行時期具有多種執行模式，因此可提供很高的執行彈性。
封裝	將程式中的屬性及對屬性的操作方法包在一個語法單元中。以C++爲例，其以類別（class）爲資料抽象化的單元，可以在類別中定義變數（屬性）以及處理屬性的函數（操作）。	用類別將屬性與操作緊密結合，提高可讀性、安全性。由於將屬性與操作被包在語法單元中，故透過封裝可以只允許用特定的操作對資料進行存取，達成資訊隱藏（information hiding）保護資料。

【高分閱讀】

1. 金乃傑，《程式語言》第四回，頁36。
2. 金乃傑，《程式語言》總複習，頁37。

二、以下是C語言程式片段：（24分）

```
int sub1(int x, int *y){
```

```
        x+=5;
        *y*=2;
        return *y+x;
        }
    int sub2(int *x, int y){
        *x+=5;
        y*=2;
        return *x+y;
    }
    void main(){
        int a=3,b=5,c,d;
        c=sub1(a,&b)+sub2(&a,b);
        d=sub2(&a,b)+sub1(a,&b);
    }
```

程式執行後，a,b,c及d的值為何？

答：

a = 13

b = 20

c = 46

d = 71

解析：

a = 3 → 8 → 13

b = 5 →10 → 20

c = 18 + 28

d = 33 + 38

【高分閱讀】

1. 金乃傑，《程式語言》第二回，頁7。
2. 金乃傑，《程式語言》總複習，頁23。

三、若1個字元占用1個位元組（byte），以下程式中p的位址是OX71F21D，陣列t的起始位址是OX31C51C，則程式執行結果為何？（16分）

```
    void main(){
        char t[7]={'d','r','a','g','o','n','\0'};
        char *p;
        p=t;
        p++;
        printf("%X\n",&p);
        printf("%c\n",*p);
        (*p)++;
        printf("%c\n",t[1]);
        printf("%X\n",&t[3]);
        }
```

答：

71F21D

r

s

31C51F

解析：

71F21D：指標與一般的變數一樣，進行運算改變的是裡面儲存的數值（p中存的數值是t的位址），不會改變到p原本的位址，故印出71F21D

r：使用存取運算*，可以取得p所指向的目的位址的值。因爲程式中做了p++，因此p指向t的第二個元素r。

s：先進行存取運算，因此取出r，再進行算術++運算，最後以字元型態輸出，因此印出r的下一個字母s。

31C51F：印出t的第四個元素的開頭位址。由於一個字元剛好是1 byte。因此是31C51C+3 = 31C51F。

【高分閱讀】
1. 金乃傑，《程式語言》第三回，頁4。
2. 金乃傑，《程式語言》總複習，頁8。

四、 (一)何謂case sensitive？何謂recursive call？大量的使用recursive call可能造成什麼問題？（9分）
(二)C++語言是否為case sensitive語言？C++語言是否允許recursive call？（4分）
(三)程式語言以何種資料結構維護函式呼叫的順序？每一個啟動紀錄（activation record）所記錄的兩個主要內容為何？（12分）

答：

(一)case sensitive與recursive call

1.case sensitive：指在程式語言將變數與函數名稱的大小寫視爲不同，例如變數Student與student是不同的兩個變數；函數IF()與if()也是不同的。

2.recursive call：分爲直接遞迴與間接遞迴，直接遞迴指副程式在敘述中直接呼叫該副程式；間接遞迴是副程式呼叫其他n個副程式，其他副程式再回來呼叫該副程式稱之。

3.副程式呼叫會在系統中建立啓動紀錄實體，若大量使用會使得實體過多而超過程式預設的記憶體限制，使得程式當掉。另一方面在配置記憶體時也會耗費額外的時間，因此大量使用會使程式效能低落。

(二)C++的case sensitive與recursive call

1.C++屬於case sensitive的語言，因爲在C++中變數與函數名稱的大小寫視爲不同，如if()與IF()是不同的。

2.C++允許recursive call，因爲C++使用中央堆疊來動態儲存函數的活動紀錄。

(三)中央堆疊與啓動紀錄

1.程式語言以堆疊的資料結構來維護函數的呼叫順序，這個堆疊在程式中稱爲中央堆疊（Central Stack）。當副程式呼叫時，系統會爲副程式配置記憶體空間，來儲存副程式中的區域變數與呼叫者等資訊，這個記憶體空間稱爲啓動紀錄（Active Record），該紀錄會被儲存到中央堆疊中，當副程式執行完後會將所使用者記憶體歸還系統。由於堆疊的特性是後進先出，因此越後面被呼叫到的副程式會越先被執行，如此便能維護函式呼叫的執行順序。

2.一般在副程式中，啓動紀錄有以下欄位：

(1)Return Value：儲存副程式return的結果。

(2)Return Address：執行完副程式後，接續執行的位址。

(3)Dynamic Link：指向呼叫該副程式的副程式區塊。

(4)Static Link：指向該副程式的上層副程式（供巢狀副程式使用）。

(5)Local Variables：儲存副程式的區域變數。

(6)Parameters：儲存副程式的傳入參數。

最主要的欄位一定要包括動態鏈與區域變數，因爲動態鏈可以協助副程式找到執行結束後接下來執行的程式主體；而區域變數使副程式得以運算資料。

【高分閱讀】
1. 金乃傑，《程式語言》第二回，頁20。
2. 金乃傑，《程式語言》總複習，頁26、頁28。

五、請以C++語言定義以下類別：

(一)類別名稱stu，包含一個整數id，一個整數grade（以上成員之存取層次為public）。（5分）

(二)宣告依據stu類別產生的物件陣列student；此陣列為一維陣列，有30個元素。（3分）

(三)寫出將student陣列中每個元素的grade加5的程式片段。（6分）

答：

```
3   //(一)宣告stu類別
4   class stu{
5       public:
6           int id;
7           int grade;
8   };
9
10  int main(){
11      //(二)產生student物件陣列
12      stu student[30];
13
14      //(三)每個元素的grade加5
15      for(int i = 0; i < 30; i++){
16          student[i].grade += 5;
17      }
18  }
```

【高分閱讀】

1. 金乃傑，《程式語言》第四回，頁4。
2. 金乃傑，《程式語言》總複習，頁38。