110 年公務人員特種考試身心障礙人員考試試題

考　試　別：身心障礙人員考試
等　　別：三等考試
類　　科：資訊處理
科　　目：程式語言

一、在程式語言中有所謂的早期繫結（early binding）和晚期繫結（late binding），請詳述這兩個繫結的差異性和使用時機點，並以 C++程式語言說明如何達成晚期繫結的功能。（20 分）

【擬答】：
早期繫結：是指程式在編譯時某些名稱或物件已經與某個方法建立關聯或綁定。

晚期繫結：程式在執行時才將名稱或物件與方法建立關聯或綁定。

差別：程式於 runtime 對於變數型別的處理方式不同。早期繫結物件可以讓編譯器配置記憶體，在程式執行之前執行最佳化，產生更有效率的程式。但是晚期繫結的程式執行彈性比較高。C++屬於早期繫結，而 Java 則屬於晚期繫結程式語言。

C++為了提高運行效率，預設是使用早期繫結，在編譯時就會決定是哪個物件來執行函式或方法而將其綁定在一起。C++可以使用透過虛擬函數，在函式宣告加上 virtual，讓函式轉變成晚期繫結。請參考下面的函數宣告：

virtual void show(){}

二、函式之間的呼叫，其參數的傳送可分為那幾種？請詳述之；並以任何一種程式語言撰寫兩個整數對調的情形，並加以說明最後處理的結果。需註明使用的程式語言。（20 分）

【擬答】：
以 C 語言來說，參數的傳遞可分為傳值(pass by value)以及傳址(pass by address)這兩種。請參考下面的程式與解說。

pass by value：參數的傳遞是傳值。下面程式裡函數參數的交換不會更改到源頭的值。

```
1  void swap(int a, int b){
2      int tmp = a;
3      a = b;
4      b = tmp;
5  }
6  void main(){
7      int m = 1, n = 3;
8      swap(m, n);
9      printf("%d %d", m, n);
10 }
```

程式輸出：1　3

pass by address：參數的傳遞是變數的位址。在函數內將此位址源頭的值取出，交換的是源頭變數的值，所以結果會交換。

```
1   void swap(int *a, int *b){
2       int tmp = *a;
3       *a = *b;
4       *b = tmp;
5   }
6   void main(){
7       int m = 1, n = 3;
8       swap(&m, &n);
9       printf("%d %d", m, n);
10  }
```

程式輸出：3　1

三、程式語言中有一重要的主題是指標（pointer）。何謂指標？其功用為何？ int a = 100；並詳述⑴int *ptr = &a；⑵int *fn(int a)；⑶int(*fn)(int a)； ⑷int *arr[3]；⑸int(*arr)[3]。（30 分）

【擬答】：

指標是一個變數，此變數儲存的不是一個數值，而是另外一個變數的記憶體位址。指標變數可以用來儲存變數的記憶體位址，也可以透過此位址取出源頭的值來使用。

㈠宣告指標變數 ptr，其值為整數變數 a 的記憶體位址。

㈡宣告函數 fn，函式回傳值的資料型態是一個指標，指標源頭是整數。

㈢宣告一個函式指標，指向一個回傳 int 的函式。

㈣宣告一個可存放 3 個指標變數的陣列，指標源頭是整數。

㈤宣告一個指向陣列的指標 arr，此陣列可存放 3 個元素。

四、程式語言中有一重要的主題是指標和參考（reference），它們當用來撰寫鏈結串列或樹狀結構的問題。我們將它用來處理鏈結串列，假設有一單向鏈結串列（singly linked list）之節點中有三個項目，分別是 id（整數型態）、score（浮點數型態）以及 next（指標或參考型態），如下圖所示：

id	score	next

今有一節點名為 ptr，試分別撰寫將此 ptr 節點加入於已含有多個節點的單向鏈結串列之尾端，以及刪除鏈結串列尾端節點的片段程式。可以任何程式語言撰寫之，但請註明使用的程式語言。（30 分）

【擬答】：

C/C++的結構定義：

```
struct node {
    int id;
    float score;
    struct node *next;
};
struct node *head = NULL, *ptr, *current, *prev;
```

新增 ptr 到單向鏈結串列之尾端：

```
ptr = (struct node*)malloc(sizeof(struct node));
current = head -> next;

while (current -> next != NULL) {
    current = current -> next ;
}
current -> next = ptr;
```

刪除鏈結串列尾端節點：

```
current = head -> next;
while (current -> next != NULL) {
    prev = current;
    current = current -> next;
}
prev -> next = NULL;
free(current);
```