7

Case Study of Functions (part 2)

How to write the “Recursive Function”

- **First step :**設定邊界條件
- **Second step :** 自我呼叫

- **Verify recursive function**
  - **Given corner case (error answer)**
  - **Given normal case (infinite loop)**

3

Example 1 : GCD(1/3)

Problem : 輸入兩個正整數, 利用**recursive function** 來求得最大公因數**(GCD)**

Hint : 使用輾轉相除法

4

Example 1 : GCD(2/3)

Idea : 輾轉相除法

1.將輸入之兩個正整數定義爲**m, n**

2.判定**n** 是否等於零**(**因爲**n**爲除數**)**

n==0 ➔ m爲最大公因數

n!=0 ➔ Go to step 3.

3.將兩數相除得到餘數**res=m % n,**將**m=n, n=res;**

4. Go to step2

5

Example 1 : GCD(3/3)

```
int main()
{
   int m = 0;
   int n = 0;
   cout << "輸入兩數：";
   cin >> m >> n;
   cout << "GCD: " << gcd(m, n) << endl;
   return 0;
}
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
int gcd(int m, int n)
{
   if(n == 0)   //餘數爲零 : 這是停止條件
      return m;
   else
   {
      int res=0; // m=n ; n=餘數
      res=m % n;
      return gcd(n, res); //自己呼叫自己
   }
}
```

6

Example 2 : Hanoi Tower(1/3)

- **Problem :** 有三根柱子**S,T,D**和**n**個環，**n**是任意正整數。**n**個環的直徑都不相等。開始時，柱**S**上已套好**n**個環，直徑大的在下，直徑小的在上，由小而大，柱**T**和柱**D**是空的。現在要把套在柱**S**的**n**個環，搬套到柱**D**上，但是有兩個條件**:**
  ⑴每一次只能從一條柱子的頂端拿走一個環套到另一條柱子的頂端。
  ⑵一環套到一柱頂端時，此環直徑必須小於柱頂環之直徑。
  Hint :
  1.將**S**柱中前**n-1**個環搬到**T**柱中暫置**(**呼叫自己）
  2.將**S**柱中剩下最大的環搬到**D**柱中。
  3.將**T**柱中的**n-1**個環搬到**D**柱中**(**呼叫自己）

Example 2 : Hanoi Tower(2/3)

- **Idea：**

判定**n** 是否等於**1(**剩下最大的那一片**)**

n==1 ➔ 將n那一片從**S**移到**D**

n!=1 ➔ 1.將**n-1**片從**S**移到**T (n**最後會遞減至**1)**

➔ 2.將**n-1**片從**T**移到**D**

Example 2 : Hanoi Tower(3/3)

```
int main()
{
   int n=3;
   hanoi(n,'S','T','D');
   return 0;
}

void hanoi(int n, char S , char T, char D)
 {
   if(n == 1)
   {
            printf("Move sheet %d from %c to %c\n", n, S, D);
   }
   else
   {
            hanoi(n-1, S, D, T); //將n-1片從S移到T
            printf("Move sheet %d from %c to %c\n", n, S, D);
            hanoi(n-1, T, S, D); //將n-1片從T到D
   }
}
```

9

Example 3 : Compound Interest(1/3)

- **Problem :** 我有一筆錢**m**要存到銀行，銀行月利率是複利率**r,** 那**n**個月後我有多少錢在銀行。
- **Hint :**
  - **m1=m0*(1+r) → n=1**
  - **m2=m1*(1+r)=(m0*(1+r))*(1+r) → n=2**
  - **m3=m2*(1+r)=((m0*(1+r))*(1+r))*(1+r)→n=3**

10

Example 3 : Compound Interest(2/3)

- **Idea :**

1.設定每個月的利率爲變數**r**， 最初本金爲**m**，共要計算月數爲**n**

2.如果計算次數**n==0**則傳回本金**m**

n!=0 → m=m*(1+r);

Example 3 : Compound Interest(3/3)

```
double CompoundI(int n, double rate, double money){
  if(n==0)
        return money;
  else
        return CompoundI(n-1, rate, money*(1+rate));
}

int main(){
  double money=0, rate=0; //m 是本金, rate 是利率
  int n=0; //複利次數
  cout << "Money : "  << endl; cin >> money ;
  cout << "Rate : "  << endl; cin >> rate ;
  cout << "Month : "  << endl; cin >> n ;
  cout <<  " Money: " << CompoundI(n, rate, money) << endl;
  return 0;
}
```

Example 4 : Compound Interest (loop)

```
double CompoundI(int n, double rate, double money)
{
  for (int idx=0; idx < n; ++idx)
  {
        money=money*(1+rate);
  }
  return money;
}
```

遞迴vs.迴圈

	遞迴	迴圈
控制結構	選擇結構	重複結構
重複方法	重複呼叫	for,do/while,while
終止檢定	到達基本條件	迴圈條件失敗
無窮迴圈	記憶體耗盡	程式無法停止
額外代價	高	無
程式難易	易	難

Summery

- **Recursive Function**
  - **1.** 決定何時要終止遞迴
  - **2.** 列出正確的計算公式，並且自行呼叫
- **Try to practice the difference of “loop method” and “recursive function”.**