遞迴函數

18
CHAPTER

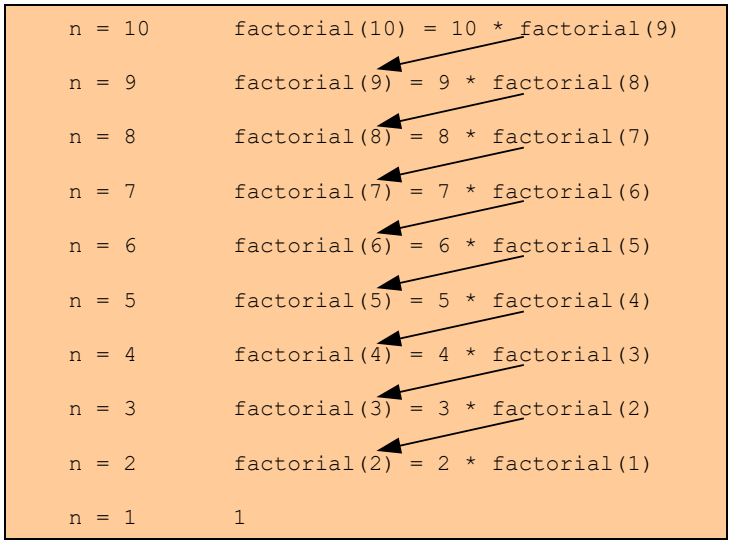
18.1 一般函數遞迴

遞迴函數呼叫（Recursive Function Calls）就是函數中包含一個呼叫自己的敘述。它也可能出現在二函數之間，如函數 1 呼叫函數 2，而函數 2 再呼叫函數 1 而形成遞迴呼叫。

18.1.1 階乘函數

下面範例是遞迴呼叫函數 n*factorial(n-1)，當 n>0 時傳回 n*factorial(n-1)，而 factorial(n-1) 又呼叫 factorial 函數並傳遞 n-1 值。

```
long factorial(int n)
{
   if(n>1)
      return n * factorial(n-1);
   else
      return 1;
}
```



如上圖，假設 n=10，所以呼叫 factorial(10) 傳回 10*factorial(9)，接著呼叫 factorial(9) 傳回 9*factorial(8)，factorial(8) 傳回 8*factorial(7)，factorial(7) 傳回 7*factorial(6)，factorial(6) 傳回 6*factorial(5)，factorial(5) 傳回 5*factorial(4)，factorial(4) 傳回 4*factorial(3)，factorial(3) 傳回 3*factorial(2)，factorial(2) 傳回 2*factorial(1)，factorial(1) 則傳回 1 並結束遞迴呼叫。

n 值	遞迴運算值
10	10 * factorial(9)
9	10 * 9 * factorial(8)
8	10 * 9 * 8 * factorial(7)
7	10 * 9 * 8 * 7 * factorial(6)
6	10 * 9 * 8 * 7 * 6 * factorial(5)
5	10 * 9 * 8 * 7 * 6 * 5 * factorial(4)
4	10 * 9 * 8 * 7 * 6 * 5 * 4 * factorial(3)
3	10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * factorial(2)
2	10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * factorial(1)
1	10 * 9 * 8 * 7 * 6 * 5 * 4 * 3 * 2 * 1

程式 18-01：計算階乘的遞迴函數

```
1.  //儲存檔名：d:\C++18\C1801.cpp
2.  #include <iostream>
3.  #include <iomanip>
4.  using namespace std;
5.
6.  long factorial(int);
7.
8.  int main(int argc, char** argv)
9.  {
10.    int count = 1;                              //while 迴圈初值
11.    cout << "計數\t" << setw(8) << "階乘\n"; //輸出字串
12.    do {
13.       cout << setw(3) << count << '\t';    //輸出計數值
14.       cout << setw(7) << factorial(count) << endl;    //輸出階乘
15.    } while(++count<=10);
16.    return 0;
17. }
18.
19. long factorial(int n)
20. {
21.    if(n>1)
22.       return n * factorial(n-1);
23.    else
24.       return 1;
25. }
```

程式輸出

```
計數          階乘
  1             1
  2             2
  3             6
  4            24
  5           120
  6           720
  7          5040
  8         40320
  9        362880
 10       3628800
```

18.1.2 gcd 函數

下面範例是遞迴呼叫函數 gcd(a, b, n-1)，當 a%n==0 && b%n==0 為 false 時，呼叫 gcd(a, b, n-1)，而當 a%n==0 && b%n==0 為 true 時，傳回 n 值且結束遞迴呼叫。

```
int gcd(int a, int b, int n)
{
   if(a%n==0 && b%n==0)                              //二數除以 n 皆等於 0
      return n;
   else
      return gcd(a, b, n-1);
}
```

假設 a=9, b=6, n=a，當呼叫 gcd(9,6,9) 因為 9%9==0 但 6%9!=0，所以遞迴呼叫 gcd(9,6,8)，同理 9%8!=0 且 6%8!=0 接著遞迴呼叫 gcd(9,6,7)，而 9%7!=0 且 6%7!=0 再遞迴呼叫 gcd(9,6,6)，9%6!=0 且 6%6==0 再遞迴呼叫 gcd(9,6,5)，…以此類推，最後呼叫 gcd(9,6,3) 時 9%3==0 且 6%3==0 則傳回 3 並結束遞迴呼叫。

n 值	a%n	b%n	遞迴運算值
9	9%9 == 0	6%9 != 0	gcd(9, 6, 9-1)
8	9%8 != 0	6%8 != 0	gcd(9, 6, 8-1)
7	9%7 != 0	6%7 != 0	gcd(9, 6, 7-1)
6	9%6 != 0	6%6 == 0	gcd(9, 6, 6-1)
5	9%5 != 0	6%5 != 0	gcd(9, 6, 5-1)
4	9%4 != 0	6%4 != 0	gcd(9, 6, 4-1)
3	9%3 == 0	6%3 == 0	3

程式 18-02：求二數 GCD 的遞迴函數

```
1.  //儲存檔名：d:\C++18\C1802.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  int gcd(int, int, int);
6.
7.  int main(int argc, char** argv)
8.  {
9.     int a, b, n;                                   //宣告變數
10.    cout << "請輸入二個整數 (a b)：";               //輸出訊息
11.    cin >> a >> b;                                 //輸入 a, b 二數
12.    n = a;
13.    cout << "GCD = " << gcd(a, b, n) << endl;          //輸出 GCD
14.    return 0;
15. }
16.
17. int gcd(int a, int b, int n)
```

```
18.  {
19.     if(a%n==0 && b%n==0)                    //二數除以 n 皆等於 0
20.        return n;
21.     else
22.        return gcd(a, b, n-1);
23.  }
```

程式輸出

```
請輸入二個整數 (a b)：9 6 Enter
GCD = 3
```

程式輸出

```
請輸入二個整數 (a b)：305 135 Enter
GCD = 5
```

18.1.3 lcm 函數

下面範例是遞迴呼叫函數 lcm(a, b, n+a)，起初 n=a 當 n%b!=0 時，遞迴呼叫 lcm(a, b, n+a)，而當 n%b==0 時，傳回 n 值並結束遞迴呼叫。

```
int lcm(int a, int b, int n)
{
   if(n%b==0)                                  //n 除以 b 等於 0
      return n;
   else
      return lcm(a, b, n+a);
}
```

假設 a=8, b=10, n=a，當呼叫 lcm(8,10,8) 因為 8%10!=0，所以遞迴呼叫 lcm(8,10,16)，同理 16%10!=0 接著遞迴呼叫 lcm(8,10,24)，而 24%10!=0 再遞迴呼叫 lcm(8,10,32)，而 32%10!=0 再遞迴呼叫 lcm(8,10,40)，最後 40%10==0 則傳回 40 並結束遞迴呼叫。

n 值	n%b	遞迴運算值
8	8%10 != 0	lcm(8, 10, 8+8)
16	16%10 != 0	lcm(8, 10, 16+8)
24	24%10 != 0	lcm(8, 10, 24+8)
32	32%10 != 0	lcm(8, 10, 32+8)
40	40%10 == 0	40

程式 18-03：求二數 LCM 的遞迴函數

```
1.   //儲存檔名：d:\C++18\C1803.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   int lcm(int, int, int);
6.
7.   int main(int argc, char** argv)
8.   {
9.      int a, b, n;                                  //宣告變數
10.     cout << "請輸入二個整數 (a b)：";              //輸出訊息
11.     cin >> a >> b;                                //輸入 a, b 二數
12.     n = a;
13.     cout << "LCM = " << lcm(a, b, n) << endl;          //輸出 LCM
14.     return 0;
15.  }
16.
17.  int lcm(int a, int b, int n)
18.  {
19.     if(n%b==0)                                    //n 除以 b 等於 0
20.        return n;
21.     else
22.        return lcm(a, b, n+a);
23.  }
```

程式輸出

```
請輸入二個整數 (a b)：8 10 Enter
LCM = 40
```

程式輸出

```
請輸入二個整數 (a b)：46 115 Enter
LCM = 230
```

18.2 巢狀遞迴函數

巢狀遞迴函數（Nested Recursive Function）就是在遞迴函數中呼叫另一個遞迴函數。如果將一個遞迴函數視為一個迴圈，則巢狀遞迴函數可視為巢狀迴圈。

18.2.1 選擇排序

下面範例是第 7 章使用巢狀迴圈執行選擇排序。

```
for (i = 0; i < MAX-1; i++)                    //排序外迴圈
{
   mindex = i;                                 //mindex=最小值索引
   minimum = number[i];                        //minimum=最小值
   for (j = i+1; j < MAX; j++)                 //排序內迴圈
   {
      if (number[j] < minimum                  //若 number[j]<最小值
      {
         minimum = number[j];                  //minimum=新最小值
         mindex = j;                           //mindex=新最小值索引
      }
   }
   number[mindex] = number[i];                 //number[最小值索引]=較大值
   number[i] = minimum;                        //number[i]=最小值
}
```

下面範例是將上面範例的外迴圈改成 outerSort 函數，並且當計數值 i<MAX 時利用遞迴呼叫 outerSort 函數達成外迴圈的功能。

```
void sort(int number[], int i)
{
   int j, minimum, mindex;                     //宣告整數變數
   if(i < MAX) {
      mindex = i;                              //mindex=最小值索引
      minimum = number[i];                     //minimum=最小值
      for (j = i+1; j < MAX; j++)              //排序迴圈
      {
         if (number[j] < minimum)              //若 number[j]<最小值
         {
            minimum = number[j];               //minimum=新最小值
            mindex = j;                        //mindex=新最小值索引
         }
      }
      number[mindex] = number[i];              //number[最小值索引]=較大值
      number[i] = minimum;                     //number[i]=最小值
      sort(number, i+1);                       //遞迴呼叫
   }
}
```

下面範例再將上面範例的內迴圈改成 innerSort 函數，並且當計數值 j<MAX 時利用遞迴呼叫 innerSort 函數達成內迴圈的功能。因為 outerSort 函數與 innerSort 函數都要使用 minimum 與 mindex 變數，所以必須將此二變數定義成公用變數。

```
int minimum, mindex;                          //宣告整數公用變數
void outerSort(int number[], int i)
{
   int j;                                     //宣告整數區域變數
   if(i < MAX) {
      mindex = i;                             //mindex=最小值索引
      minimum = number[i];                    //minimum=最小值
      innerSort(number, i+1);
      number[mindex] = number[i];             //number[最小值索引]=較大值
      number[i] = minimum;                    //number[i]=最小值
      outerSort(number, i+1);                 //遞迴呼叫
   }
}

void innerSort(int number[], int j)
{
   if(j < MAX) {
      if (number[j] < minimum)                //若 number[j]<最小值
      {
         minimum = number[j];                 //minimum=新最小值
         mindex = j;                          //mindex=新最小值索引
      }
      innerSort(number, j+1);
   }
}
```

程式 18-04：選擇排序遞迴函數

```
1.  //檔案名稱：d:\C++18\C1804.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  void innerSort(int [], int);
6.  void outerSort(int [], int);
7.  const int MAX = 5;                          //MAX = 陣列最大範圍
8.  int minimum, mindex;                        //定義公用變數
9.
10. int main(int argc, char** argv)
11. {
12.    int i;                                   //宣告整數變數
13.    int number[MAX] = {57, 19, 33, 92, 6}; //宣告一維陣列
14.
```

```
15.     cout << "排序前：";                    //顯示排序前資料
16.     for(i = 0; i < MAX; i++)
17.        cout << number[i] << '\0';
18.     cout << "\n 排序後：";                  //顯示排序後資料
19.     outerSort(number, 0);
20.     for (i = 0; i < MAX; i++)
21.        cout << number[i] << '\0';
22.     cout << endl;
23.     return 0;
24.  }
25.
26.  void outerSort(int number[], int i)
27.  {
28.     if(i < MAX) {
29.        mindex = i;                          //mindex=最小值索引
30.        minimum = number[i];                 //minimum=最小值
31.        innerSort(number, i+1);              //呼叫 innerSort
32.        number[mindex] = number[i];          //number[最小值索引]=較大值
33.        number[i] = minimum;                 //number[i]=最小值
34.        outerSort(number, i+1);              //遞回呼叫 outerSort
35.     }
36.  }
37.
38.  void innerSort(int number[], int j)
39.  {
40.     if(j < MAX) {
41.        if(number[j] < minimum) {              //若 number[j]<最小值
42.           minimum = number[j];                //minimum=新最小值
43.           mindex = j;                         //mindex=新最小值索引
44.        }
45.        innerSort(number, j+1);                //遞回呼叫 innerSort
46.     }
47.  }
```

程式輸出

```
排序前：57 19 33 92 6
排序後：6 19 33 57 92
```

18.2.2 二分搜尋

下面範例是第 7 章使用 do-while 迴圈執行二分搜尋。

```
do                                             //搜尋迴圈
{
   if ((low + high) % 2 > 0.5)                 //計算搜尋位置
      middle = (low + high) / 2 + 1;
   else
      middle = (low + high) / 2;
```

```
      if (search < number[middle])              //計算搜尋上限
         high = middle - 1;
      else if (search > number[middle])         //計算搜尋下限
         low = middle + 1;
      else if (search == number[middle])        //找到相符數值
         break;
   } while (low <= high);
   if (low > high)                              //顯示搜尋結果
      cout << "\n找不到數值：" << search << endl;
   else
      cout << "\n找到數值：" << search << endl;
}
```

下面範例是將上面範例的 do-while 圈改成 search 函數，並將 while 的條件 (low <= high)改為 if 的條件。

```
void search(int number[], int low, int high, int nav)
{
   int middle;                                  //宣告整數變數
   if (low <= high) {
      if ((low + high) % 2 > 0.5)               //計算搜尋位置
         middle = (low + high) / 2 + 1;
      else
         middle = (low + high) / 2;

      if (nav < number[middle])                 //計算搜尋上限
         search(number, low, middle-1, nav);    //遞迴呼叫
      else if (nav > number[middle])            //計算搜尋下限
         search(number, middle+1, high, nav);   //遞迴呼叫
      else if (nav == number[middle])           //找到相符數值
         cout << "\n找到數值：" << nav << endl;
   } else {
      cout << "\n找不到數值：" << nav << endl;
   }
}
```

程式 18-05：二分搜尋的遞迴函數

```
1.  //檔案名稱：d:\C++18\C1805.cpp
2.  #include <iostream>
3.  using namespace std;
4.
5.  void innerSort(int [], int);
6.  void outerSort(int [], int);
7.  void search(int [], int, int, int);
8.  const int MAX = 5;                          //MAX = 陣列最大範圍
9.  int minimum, mindex;                        //定義公用變數
10.
11. int main(int argc, char** argv)
```

```
12. {
13.    int i, navigate;                          //宣告整數變數
14.    int number[MAX] = {57, 19, 33, 92, 6}; //宣告整數陣列
15.
16.    cout << "排序前：";                        //顯示排序前之值
17.    for (i = 0; i < MAX; i++)
18.       cout << number[i] << '\0';
19.
20.    cout << "\n 請輸入要搜尋數值：";
21.    cin >> navigate;                          //輸入搜尋值
22.    outerSort(number, 0);
23.
24.    cout << "\n 排序後：";                      //顯示排序後之值
25.    for (i = 0; i < MAX; i++)
26.       cout << number[i] << '\0';
27.    search(number, 0, MAX-1, navigate);
28.    return 0;
29. }
30.
31. void outerSort(int number[], int i)
32. {
33.    if(i < MAX) {
34.       mindex = i;                          //mindex=最小值索引
35.       minimum = number[i];                 //minimum=最小值
36.       innerSort(number, i+1);              //呼叫 innerSort
37.       number[mindex] = number[i];          //number[最小值索引]=較大值
38.       number[i] = minimum;                 //number[i]=最小值
39.       outerSort(number, i+1);              //遞回呼叫 outerSort
40.    }
41. }
42.
43. void innerSort(int number[], int j)
44. {
45.    if(j < MAX) {
46.       if(number[j] < minimum) {            //若 number[j]<最小值
47.          minimum = number[j];              //minimum=新最小值
48.          mindex = j;                       //mindex=新最小值索引
49.       }
50.       innerSort(number, j+1);              //遞回呼叫 innerSort
51.    }
52. }
53.
54. void search(int number[], int low, int high, int nav)
55. {
56.    int middle;                             //宣告整數變數
57.    if (low <= high) {
58.       if ((low + high) % 2 > 0.5)          //計算搜尋位置
59.          middle = (low + high) / 2 + 1;
60.       else
61.          middle = (low + high) / 2;
62.
```

```
63.        if (nav < number[middle])                //計算搜尋上限
64.           search(number, low, middle-1, nav);   //遞回呼叫
65.        else if (nav > number[middle])           //計算搜尋下限
66.           search(number, middle+1, high, nav); //遞回呼叫
67.        else if (nav == number[middle])          //找到相符數值
68.           cout << "\n找到數值：" << nav << endl;
69.     } else {
70.        cout << "\n找不到數值：" << nav << endl;
71.     }
72.  }
```

程式輸出

```
排序前：57 19 33 92 6
請輸入要搜尋數值：33 Enter

排序後：6 19 33 57 92
找到數值：33
```

18.3 類別函數遞迴

遞迴成員函數（Recursive Member Function）就是在類別中包含遞迴函數成員。一般獨立的函數可以呼叫函數自己，而類別中的成員函數也可以呼叫自己形成遞迴成員函數。

18.3.1 遞迴成員函數

下面範例是在 Division 類別中，定義一個 quotient 遞迴函數與 remainder 遞迴函數。quotient 遞迴函數是以 a-b 的次數 q 而得到 a 除以 b 的商 q，而 remainder 遞迴函數則是以 a-b 後若 a<b 則 a 等於 a 除以 b 的餘數 r。

```
class Division
{
   int a, b, q;
public:
   Division(int a1, int b1)
   {
      a = a1;
      b = b1;
      q = 0;
   }
   int quotient()
   {
      if(a>=b) {                                    //a 仍大於等於 b
```

```
         a -= b;                                  //a = a - b
         q++;
         return quotient();                       //遞迴呼叫
      } else
         return q;                                //傳回商數
   }
   int remainder()
   {
      if(a>=b) {                                  //a 仍大於等於 b
         a -= b;                                  //a = a - b
         return remainder();                      //遞迴呼叫
      } else
         return a;                                //傳回餘數
   }
};
```

程式 18-06：除法運算的遞迴函數

```
1.   //儲存檔名：d:\C++18\C1806.cpp
2.   #include <iostream>
3.   using namespace std;
4.
5.   class Division
6.   {
7.      int a, b, q;
8.   public:
9.      Division(int a1, int b1)
10.     {
11.        a = a1;
12.        b = b1;
13.        q = 0;
14.     }
15.     int quotient()
16.     {
17.        if(a>=b) {                            //a 仍大於等於 b
18.           a -= b;                            //a = a - b
19.           q++;
20.           return quotient();                 //遞迴呼叫
21.        } else
22.           return q;                          //傳回商數
23.     }
24.     int remainder()
25.     {
26.        if(a>=b) {                            //a 仍大於等於 b
27.           a -= b;                            //a = a - b
28.           return remainder();                //遞迴呼叫
29.        } else
30.           return a;                          //傳回餘數
31.     }
32.  };
33.
```

```
34.  int main(int argc, char** argv)
35.  {
36.     int a, b;                                    //宣告變數
37.     cout << "請輸入二個整數 (a b)：";             //輸出訊息
38.     cin >> a >> b;                               //輸入 a, b 二數
39.     Division div(a, b);
40.
41.     cout << a << " / " << b << " = " << div.quotient(); //輸出商
42.     cout << " R " << div.remainder() << endl;          //輸出餘數
43.     return 0;
44.  }
```

程式輸出

```
請輸入二個整數 (a b)：99 23 Enter
99 / 23 = 4 R 7
```

當然如果只是計算整數或浮點數資料的商或餘數，可以直接使用除法運算符號（/）與餘數運算符號（%）。但若是執行 40 位數的除法運算，計算 40 位數的商與餘數，則無法使用除法運算符號（/）與餘數運算符號（%），而必須使用減法運算來求得商與餘數。

18.4 習題

選擇題

1. 函數中包含一個呼叫自己的敘述稱為_____。

   a) 自我函數呼叫 b)遞迴函數呼叫 c) 循環函數呼叫 d) 迴圈函數呼叫

2. 在遞迴函數中呼叫另一個遞迴函數稱為_____。

   a) 交叉遞迴函數 b) 相互遞迴函數 c) 巢狀遞迴函數 d) 循環遞迴函數

3. _____就是在類別中包含遞迴函數成員。

   a) 遞迴成員函數 b) 類別成員函數 c) 迴圈成員函數 d) 巢狀函數成員

4. 一般獨立的函數可以呼叫函數自己，而類別中的成員函數也可以呼叫自己形成_____。

   a) 遞迴成員函數 b) 類別成員函數 c) 迴圈成員函數 d) 巢狀函數成員

實作題

1. 寫一 C++ 程式，以呼叫遞迴函數的方法求三數的 GCD。
   a) 定義一個 gcd 遞迴函數，接受三個整數參數，然後傳回三個參數的最大公約數（GCD）給呼叫敘述。
   b) 寫一個驅動程式，從鍵盤輸入三個整數並存入整數變數中，呼叫 gcd 函數並傳遞此三個整數，最後輸出從 gcd 函數傳回的最大公約數。

2. 寫一 C++ 程式，以呼叫遞迴函數的方法計算一維陣列元素值的總和。
   a) 定義一個 sum 遞迴函數，接受一個整數陣列參數，然後將陣列所有元素資料加總，最後傳回總和給呼叫敘述。
   b) 寫一個驅動程式，定義並起始一維整數陣列，呼叫 sum 函數並傳遞陣列參數給 sum 函數，最後輸出 sum 函數傳回的總和。

3. 寫一 C++ 程式，以巢狀遞迴函數的方法執行選擇排序。
   a) 定義一個 outerSort 遞迴函數範本與一個 innerSort 遞迴函數範本，取代選擇排序的外迴圈與內迴圈，此 outerSort 與 innerSort 函數範本可以接受並排列任何 C++ 內建型態的資料。
   b) 寫一個驅動程式，定義並起始一維整數陣列與一維字串陣列，呼叫 outerSort 函數並傳遞整數陣列參數，然後輸出排列後的整數陣列資料。再呼叫 outerSort 函數並傳遞字串陣列參數，然後輸出排列後的字串陣列資料。

4. 寫一 C++ 程式，以呼叫遞迴函數的方法列出九九乘法表。
   a) 定義一個 mulTable 遞迴函數，輸出九九乘法表。
   b) 寫一個驅動程式，呼叫 mulTable 函數。