

# Course04

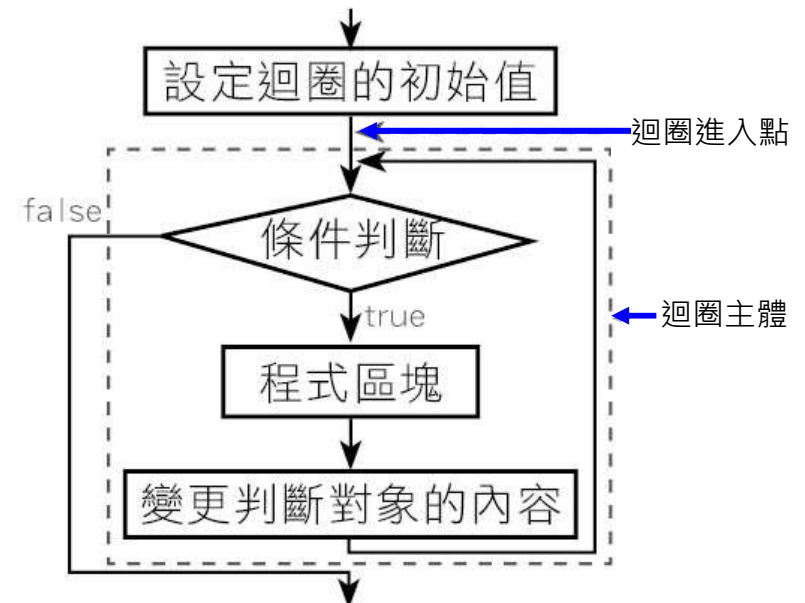
廻圈

# 迴圈-重複的實現

- 與關係、邏輯運算子搭配最密切的
  - if 敘述與能重複執行工作的敘述
- 重複執行的概念
  - 特定的工作在不斷地循環中一次又一次的被執行
  - 不斷地循環就類似繞著操場跑步一樣一圈又一圈
- 例如
  - 解  $1+2+3+\dots+10$ 。答案是 55
  - 解  $1+2+\dots+100$ 。答案是 5050
  - 用程式來實現呢？
    - $ix = 1+2+3+4+5+6+7+8+9+10;$
    - 必須利用能執行重複工作的敘述來實現

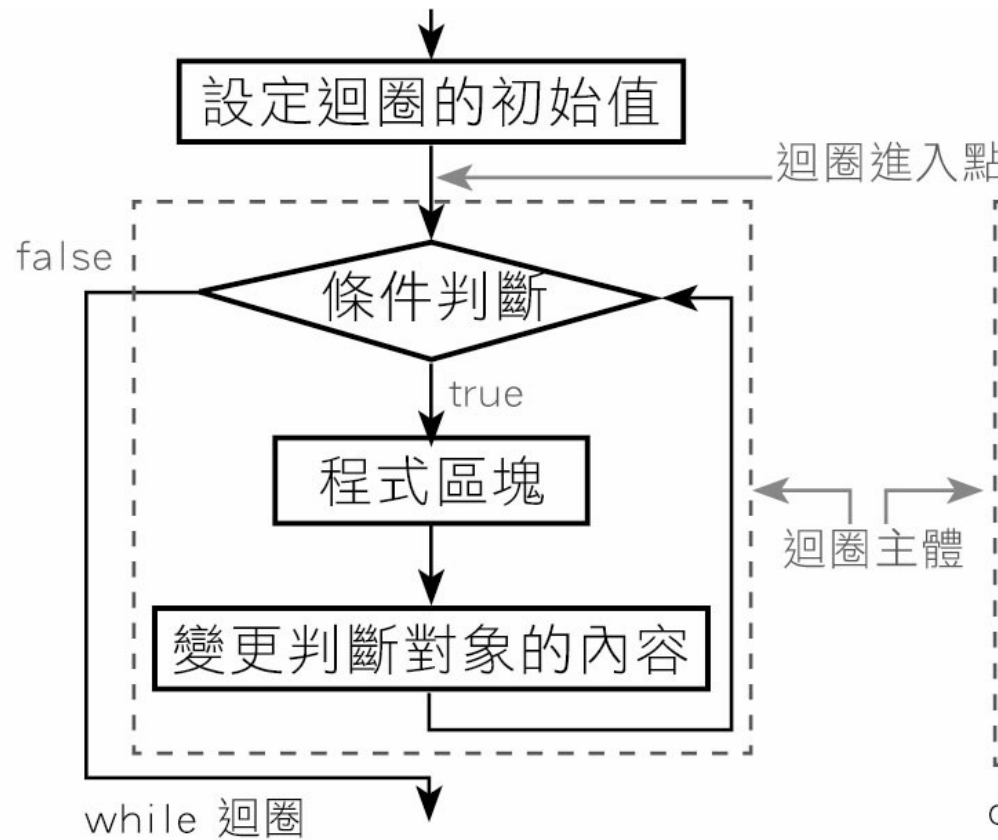
# 迴圈的概念與基本架構

- 迴圈的基本架構包含
  - 設定條件判斷對象的初始值
    - 以迴圈初(始)值稱之
  - 迴圈進入點
  - 迴圈主體
  - 控制迴圈執行的條件
  - 改變條件判斷對象的內容
    - 以變更判斷對象的內容稱之
    - 使迴圈的條件會改變成為False

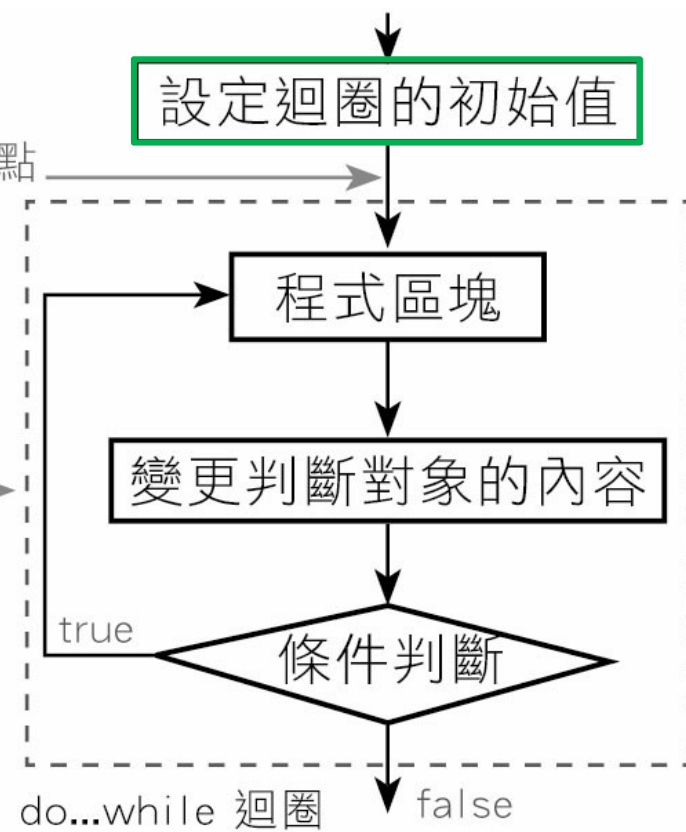


- 此五個元素可構成兩種不同的迴圈結構

- 先判斷再執行



- 先執行再判斷



# while – 先判斷(符合)再執行

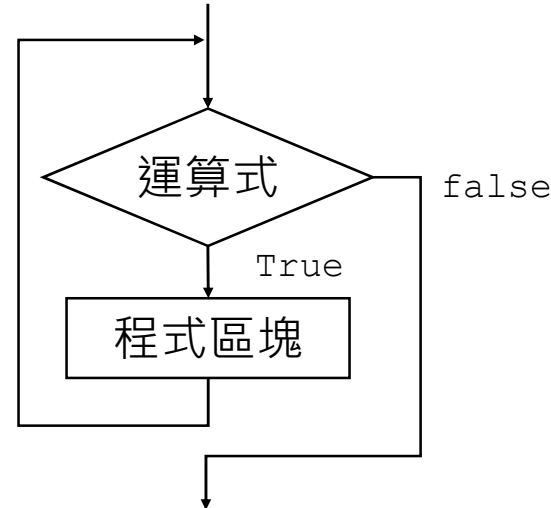
- while 敘述的基本型

```
while (運算式/true)  
{
```

程式區塊

```
}
```

這裡不能有分號



- 執行規則
  - 判斷 while 後面運算式
    - 當運算為 **true** 時，就會執行大括弧 {} 所涵蓋程式區塊
    - 區塊執行完畢後，會回到 while 後面的運算式，再次進行判斷
  - 直到 while 運算式的運算結果為 **false** 時，
    - 程式的執行才會跳離 while，執行 while 所涵蓋範圍之後的後續敘述

- $1+2+3+\dots+100$ 
  - 這樣的循環過程，習慣上以迴圈來代表它
  - 想法
    - 先宣告一個變數，讓它一開始為 1
    - 讓它在 **while** 迴圈裡面每次加 1 (++)
      - 最後一定會加到 100
      - 當它加到超過 100 時( $\geq 100$ )，就結束 **while** 迴圈

# 程式

- 試寫一程式，利用while迴圈，執行 $1+2+\dots+100$ 
  - 輸出:
    - 總和(sum):  $1+2+\dots+100$
  - 思考
    - `sum = 0;`  
`sum += 1;`  
`sum += 2;`  
`...`  
`sum += 99;`  
`sum += 100;`
    - 寫100行
    - 或是使用迴圈，讓i來控制

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    int i, sum;
```

```
    i=1;
```

```
    sum=0;
```

迴圈變數的初始值

```
    while (i<=100) {
```

```
        sum += i;
```

```
        i++;
```

當while迴圈當中

條件滿足 $i \leq 100$ 時，則繼續執行

當條件 $> 1000$ 則結束迴圈

```
    }
```

```
    printf("1+2+...+100 = %d\n", sum);
```

```
    system("PAUSE");
```

```
    return 0;
```

```
}
```

1+2+...+100 = 5050

請按任意鍵繼續 . . .



# 程式邏輯1

- 計算1到100之間所有奇數的和
  - 輸出
    - 1到100間的所有奇數之和
  - 思考
    - $1+3+5+\dots+99$
    - 重覆運算，使用迴圈
    - 迴圈控制
      - 初值為1
      - 終值為99
      - 每次增值為2

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, sum;
    i=1;
    sum=0;
    while (i<=99) {
        sum += i;
        i+=2;
    }
    printf("1+3+...+99 = %d\n", sum);
    system("PAUSE");
    return 0;
}
```

1+3+...+99 = 2500

請按任意鍵繼續 . . .

# 程式邏輯2

- 試用一**while**迴圈，分別計算1到100當中所有奇數及偶數的和
  - 輸出
    - 奇數總和:  $1+3+\dots+99$
    - 偶數總和:  $2+4+\dots+100$
  - 思考
    - 使用一個迴圈，就必須在迴圈當中使用判斷
      - $\text{if}(i\%2==0)$  ... 偶數
      - $\text{if}(i\%2==1)$  ... 奇數
    - $i$ 必須要從1累加到100

```

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, oddSum, evenSum;
    i=1;
    oddSum =0;
    evenSum=0;
    while(i<=100) {
        if(i%2==0)
            evenSum += i;
        else
            oddSum += i;
        i++;
    }
    printf("1+3+...+99 = %d\n", oddSum);
    printf("2+4+...+100= %d\n", evenSum);
    system("PAUSE");
    return 0;
}

```

1+3+...+99 = 2500  
 2+4+...+100= 2550  
 請按任意鍵繼續 . . .

# 程式邏輯3

- 由使用者輸入一個大於1的整數，使用while計算 $1+2+\dots+n$ 之結果
- 輸入
  - $n$ ， $n>1$
- 輸出
  - 總和:  $1+2+\dots+n$
- 思考
  - 輸入檢查  $n>1$ ，滿足才執行
  - $i\leq n$ 為迴圈執行條件

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[])
{
    int i, n, sum;
    i=1;
    sum = 0;
    printf("input a number: ");
    scanf("%d", &n);
    if(n<=1)
        printf("input error.\n");
    else {
        while (i<=n) {
            sum+=i;
            i++;
        }
        printf("The sum is %d\n", sum);
    }
    system("PAUSE");
    return 0;
}
```

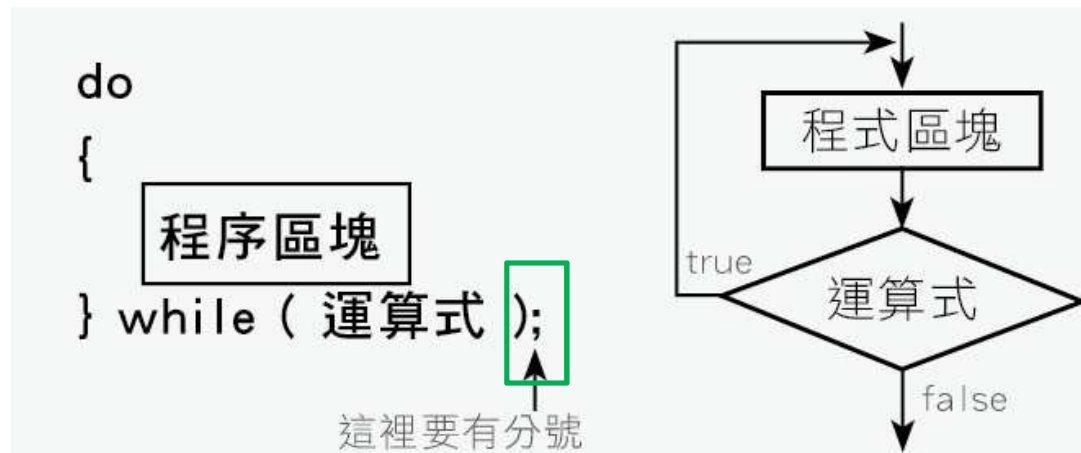
```
input a number: 100
The sum is 5050
請按任意鍵繼續 . . .
```

```
input a number: 1
input error.
請按任意鍵繼續 . . .
```

錯誤檢查/控制機制

# do...while – 先執行再判斷(符合)再執行

- 語法與相對應的流程圖



- 語法說明
  - do 就是迴圈進入點
  - 先執行迴圈中的程式區塊，然後才進行 while 後面運算式的判斷
  - 當運算式為 true 時，會回到迴圈的開頭處(do)執行
  - 當運算式為 false 時，會離開迴圈
- do...while 必須在 while 的後面加上分號

# 程式

- 試用一do-while迴圈，分別計算1到100當中所有奇數及偶數的和
  - 輸出
    - 奇數總和:  $1+3+\dots+99$
    - 偶數總和:  $2+4+\dots+100$
  - 思考
    - 使用一個迴圈，就必須在迴圈當中使用判斷
      - $\text{if}(i\%2==0)$  ... 偶數
      - $\text{if}(i\%2==1)$  ... 奇數
    - $i$ 必須要從1累加到100



```
#include <stdio.h>
#include <stdlib.h>
```

```
int main(int argc, char *argv[])
```

```
{
```

```
    int i, oddSum, evenSum;
```

```
    i=1;
```

```
    oddSum =0;
```

```
    evenSum=0;
```

```
    do {
```

```
        if (i%2==0)
```

```
            evenSum += i;
```

```
        else
```

```
            oddSum += i;
```

```
        i++;
```

```
    }while (i<=100);
```

```
    printf("1+3+...+99 = %d\n", oddSum);
```

```
    printf("2+4+...+100= %d\n", evenSum);
```

```
    system("PAUSE");
```

```
    return 0;
```

```
}
```

1+3+...+99 = 2500

2+4+...+100= 2550

請按任意鍵繼續 . . .

迴圈變數的初始值

當do-while迴圈當中  
條件滿足i<=100時，則繼續執行  
當條件>100則結束迴圈

# 不確定迴圈與計數迴圈

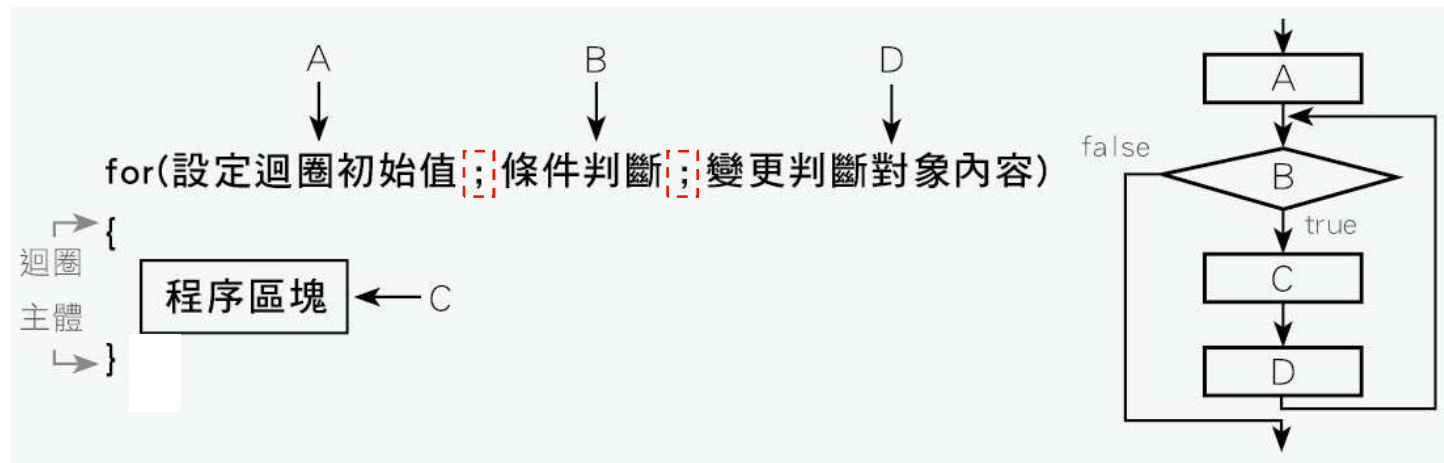
- 根據迴圈的執行與停止的方式，迴圈可分成兩種
  - 計數迴圈(Counting Loops)
  - 不確定迴圈(Indefinite Loops)
- 不確定迴圈
  - 變更判斷對像的內容中一定要包含迴圈的終止條件
- 計數迴圈三項原則
  - 計數器一定要設定初始值→迴圈的初始值
  - 計數器必須與某些限制條件進行比較→迴圈的終止條件
  - 計數器必須在每次的迴圈執行中，改變計數器的內容

# 程式

- 由使用者持續輸入一個大於1的整數，計算 $1+2+\dots+n$ 之結果，若使用者輸入錯誤，則必須重新輸入直到正確為止。若輸入-1，則表示中止程式
- 輸入
  - $n$ ， $n > 1$
  - $n$ ，-1，結束
- 輸出
  - 總和:  $1+2+\dots+n$
- 思考
  - 輸入檢查  $n > 1$ ，直到正確為止
    - 若偵測到-1則代表程式結束。
  - $i \leq n$ 為迴圈執行條件
  - 使用兩個迴圈：可以使用do-while+while或是兩個do-while

# for

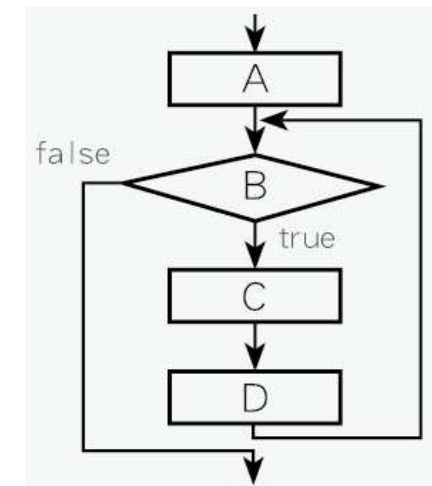
- **for** 是將迴圈執行的「設定迴圈的初始值」、「條件判斷」與「變更判斷對象的內容」三要素是寫在同一行
- **For** 敘述的語法與相對應流程圖如下



- 說明
  - **for** 後的小括弧內**依序放置**的內容分別
    - 設定迴圈的初始值
    - 條件判斷
    - 變更判斷對象的內容
  - **一定要以分號來區隔，剛好只有兩個分號，不能多也不能少**

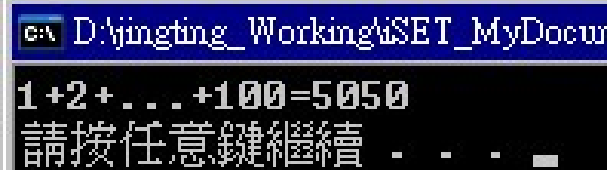
# 程式

- 試寫一程式，利用for迴圈，執行 $1+2+\dots+100$
- 思考方式：
  - 兩個變數：加總的變數 `sum`與迴圈的計數變數 `i`
  - A 區塊：設定迴圈初始值 — 就是  $i = 1$
  - B 區塊：條件判斷 —  $i \leq 100$
  - C 區塊：計算加總 —  $sum += i$
  - D 區塊：對 `i` 進行遞增 — 也就是  $i = i + 1$  或是  $i++$



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, sum;
    sum = 0;
    for(i=1; i<=100; i++) {
        sum += i;
    }
    printf("1+2+...+100=%d\n", sum);
    system("PAUSE");
    return 0;
}
```



D:\yingting\_Working\SET\_MyDocu  
1+2+...+100=5050  
請按任意鍵繼續 . . .

```

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, sum;
    sum = 0;
    for (i=1; i<=100; i++) {
        sum += i;
    }
    printf("1+2+...+100=%d\n", sum);
    system("PAUSE");
    return 0;
}

```

```

#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, sum;
    i=1;
    sum=0;
    while (i<=100) {
        sum += i;
        i++;
    }
    printf("1+2+...+100 = %d\n", sum);
    system("PAUSE");
    return 0;
}

```

針對計數型迴圈，for較有效率

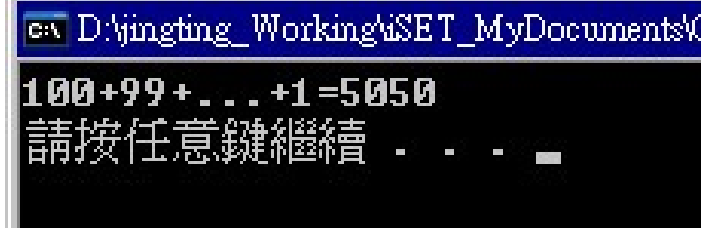
# 反過來的邏輯

- 試寫一程式，利用for迴圈，執行 $100+99+\dots+1$
- 思考
  - 遞減
    - 初值為 100，每次遞減 1
    - 使用`i--`



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int i, sum;
    sum=0;
    for (i=100; i>=0; i--)
        sum += i;
    printf("100+99+...+1=%d\n", sum);
    system("PAUSE");
    return 0;
}
```



```
c:\ D:\jingting_Working\iSET_MyDocuments\
100+99+...+1=5050
請按任意鍵繼續 . . .
```

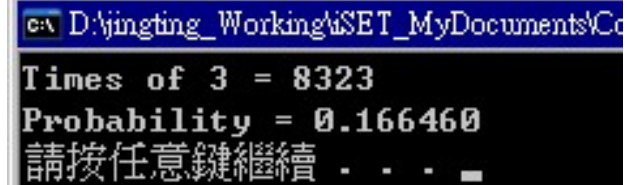
# 程式

- 利用亂數模擬擲骰子 50000 次，然後計算 3 出現的機率
- 思考
  - 變數 i 從0到 49999，每次都產生一個亂數
    - `i=0;i<50000;i++`
  - 只要 `rand()%6+1==3` 成立，就表示目前骰到 3
  - 最後再將出現的次數除上 50000 就是出現的機率
- 亂數
  - `#include <time.h>`
  - `srand((unsigned)time(NULL));` // 設定亂數種子
  - `rand()%6+1` – 取1~6之間的亂數
    - `rand() % 100` – 取0~99之間的亂數
    - `rand() % 100+1` – 取1~100之間的亂數

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc, char *argv[])
{
    int i, cnt;
    srand( (unsigned) time (NULL) );
    cnt=0;
    for(i=0;i<50000;i++) {
        if( (rand()%6+1)==3) {
            cnt++;
        }
    }
    printf("Times of 3 = %d\n",cnt);
    printf("Probability = %f\n", (float)cnt/50000);

    system("PAUSE");
    return 0;
}
```



c:\ D:\jingting\_Working\SET\_MyDocuments\C  
Times of 3 = 8323  
Probability = 0.166460  
請按任意鍵繼續 . . .

# 巢狀迴圈 – 迴圈中包含迴圈

- 簡單的範例瞭解巢狀迴圈的執行結果

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    int i, j;

    for( i = 1 ; i <= 4 ; i++ )
    {
        printf("i = %d, ",i);
        for( j = 1 ; j <= 3 ; j++ )
            printf("j=%d ",j);
        printf("\n");
    }
    system("pause");
    return(0);
}
```

i = 1, j=1 j=2 j=3  
i = 2, j=1 j=2 j=3  
i = 3, j=1 j=2 j=3  
i = 4, j=1 j=2 j=3

