C++ 程式初探 II

2015暑期

C++ 程式 II – 大綱

1. 變數
2. 運算式
3. 輸出
4. 條件判斷
5. 迴圈
6. 陣列

基本變數型態

整數	位元組	浮點數	位元組	字元	位元組
short	2	float	4	char (整數)	1
int	2 (4)	double	8		
long	4 (8)	long double	8(10)		

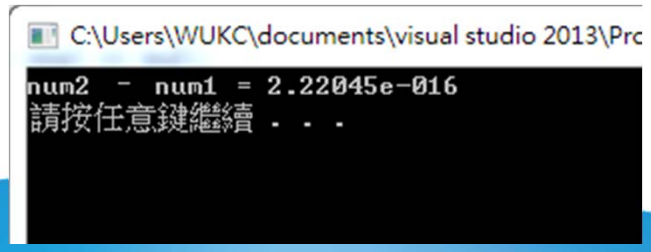
位元組	整數值域	浮點數值域	準確度
1	-128 to 127		
2	-32768 to 32767		
4	-2.1×10^{9} to 2.1×10^{9}	$\pm(1.2\times10^{-38}$ to $3.4\times10^{38})$	7
8	-9.2×10^{18} to 9.2×10^{18}	$\pm(2.2\times10^{-308}$ to $1.8\times10^{308})$	15

基本變數型態

```
#include <iostream>
using namespace std;

int main(void)
{
  int    int_var = 1;
  double float_var1 = 2.1;
  double float_var2 = 22e-1; //22×10-1 = 2.2
  char   char_var = 'a';

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

浮點數精確度 – 浮點運算誤差

```
#include <iostream>
using namespace std;

int main(void)
{
    double num1 = 0.1;
    double num2 = 2.1;
    num1 = num1 - 0.09;   //num1應該等於0.01
    num2 = num2 - 2.09;   // num2應該等於0.01

    cout << "num2 – num1 = " << num2 - num1 << endl;
    //應該顯示為 "num2 - mum1 = 0"
    system("PAUSE"); //結束
    return 0;
}
```



型態轉換

整數運算　3/2 = 1

整數指派給浮點數

result = 1.0

```
#include <iostream>
using namespace std;

int main(void)
{
    int num1 = 3;
    int num2 = 2;
    double result = num1 / num2; //result應該等於1.5
    cout << "3 / 2 = " << result << endl;
    //應該顯示為 "3 / 2 = 1.5"

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

型態轉換

```
#include <iostream>
using namespace std;

int main(void)
{
    int num1 = 3;
    int num2 = 2;
    double  result = double(num1) / num2;
    cout << "3 / 2 = " << result << endl;
    //顯示為 "3 / 2 = 1.5"

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

型態轉換 (int → double): 3 → 3.0

result =(3.0/2 → 3.0/2.0 → 1.5)

型態轉換 (顯式轉換)

(**double**) num

= (**double**)(num)

= **double** (num)

運算子(operator)

運算子	結合性	優先順序	運算子	結合性	優先順序
()		1	+	左至右	3
前置 ++			-	左至右	
前置 --			後置 ++		
*	左至右	2	後置 --		
/	左至右				
%	左至右				

整數

運算子	結合性	優先順序	運算子	結合性	優先順序
=	右至左	4	*=	右至左	4
+=	右至左		/=	右至左	
-=	右至左		%=	右至左	

練習 1

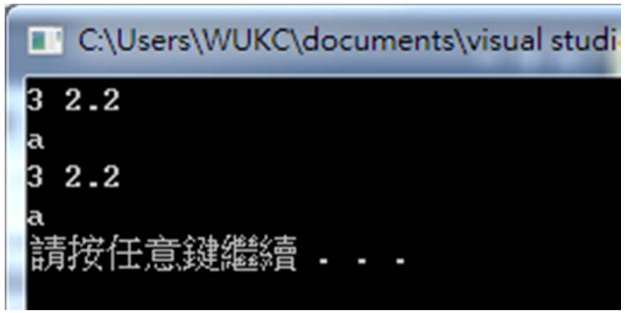
- 給定一個變數N ，判斷該數字是否為11的倍數。

1. 利用餘數運算子(%)進行倍數的判斷。
2. 測試以下數字：

112233
30800
2937
323455693
5038297
112234

輸出與輸入

```
#include <iostream>
using namespace std;

int main(void)
{
    int    int_var = 3;
    double float_var = 2.2;
    char   char_var = 'a';
    //多行
    cout << int_var;
    cout << " ";
    cout << float_var << endl;
    cout << char_var << endl;

    //單行
    cout << int_var << " " << float_var << endl << char_var << endl ;

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



輸出與輸入

```
#include <iostream>
using namespace std;

int main(void)
{
    int num1 = 3;
    int num2 = 2;

    cout << "3 / 2 = \t" << num1 / num2 << "\n";
    system("PAUSE");
    return 0;
}
```

跳脫字元
cout<<“\t”; //tab
cout<<“\n”; //換行
cout<<“\\”; //顯示\
cout<<“\'”; //'
cout<<“\””; //"

輸出與輸入

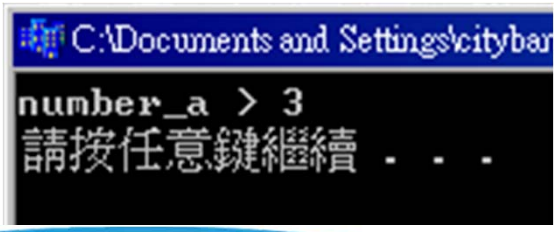
輸出函數	運算子	變數名稱	;
cin	>>	var	;

```
#include <iostream>
using namespace std;

int main(void)
{
    double num1, num2;
    cin >> num1;
    cin >> num2;

    cout << num1 << " / " << num2 << " =\t" << num1 / num2 ;
    cout << endl;
    system("PAUSE");
    return 0;
}
```

條件判斷

條件判斷運算子(1)：

```
int number_a = 6;
number_a>3;    // true   [大於]
number_a>=3;   // true   [大於等於]
number_a<6;    // false  [小於]
number_a<=6;   // true   [小於等於]
number_a==6;   // true   [等於]
number_a!=6;   // false  [不等於]
```



條件判斷

條件判斷運算子(2)：

運算子	意義
&&	and
||	or
!	not

a && b		
a	b	a&&b
true	true	true
true	false	false
false	true	false
false	false	false

a || b		
a	b	a||b
true	true	true
true	false	true
false	true	true
false	false	false

條件判斷

1. if條件判斷

```
if(判斷式){
    敘述...
}
```

2. if-else條件判斷

```
if(判斷式){
    敘述...
} else {
    敘述...
}
```

3. if-else if條件判斷

```
if(判斷式1){
    敘述...
} else if(判斷式2){
    敘述...
}
```

4. if-else if條件判斷

```
if(判斷式1){
    敘述...
} else if(判斷式2){
    敘述...
} else if(判斷式3){
    敘述...
}
... ...
else {
    敘述...
}
```

練習 2

設計一個程式，使用者可輸入一個介於0-100的數字，
當輸入數字介於 0-69 之間時，輸出D；
當輸入數字介於70-79之間時，輸出C；
當輸入數字介於80-89之間時，輸出B；
當輸入數字介於90-100之間時，輸出A。

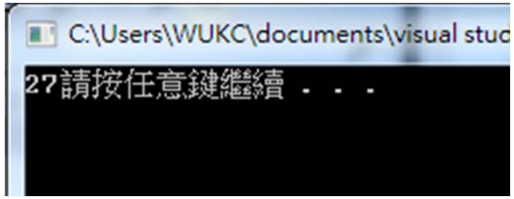
1.只使用if配合 && 完成上述程式。
2.使用if-else if 完成上述程式。

for 迴圈

```
for(運算式; 判斷式; 運算式) {
        敘述式...
}
```

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;

    for (int i = 1; i <= 3; i++) {
        num = num * 3;
    }
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```

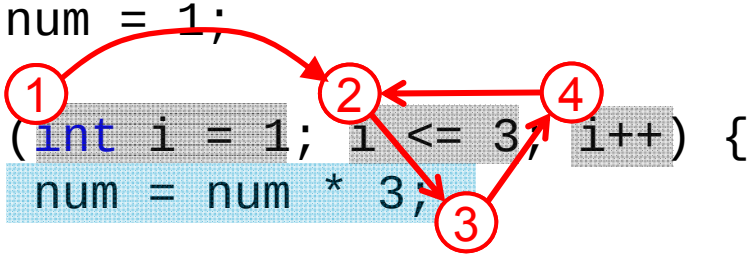
num	i
27	4



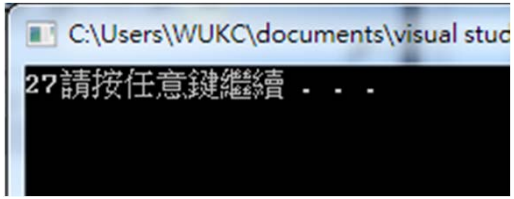
for 迴圈

```
for(運算式; 判斷式; 運算式) {
        敘述式...
}
```

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;

    for (int i = 1; i <= 3; i++) {
        num = num * 3;
    }
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```



num	i
27	4

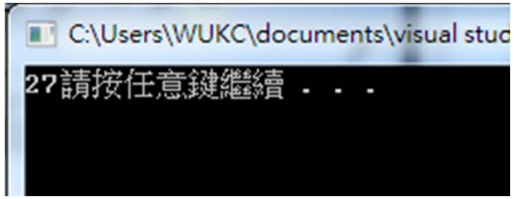


while 迴圈

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    while(i <= 3) {
        num = num * 3;
        i++;
    }
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```

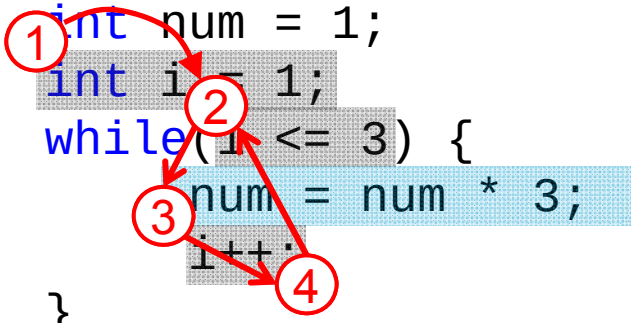
```
while(判斷式) {
        敘述式...
}
```

num	i
27	4



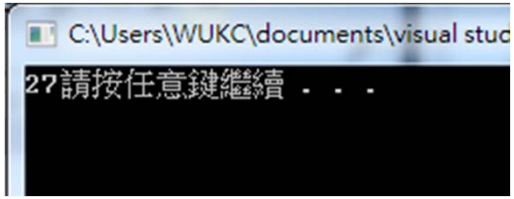
while 迴圈

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    while(i <= 3) {
        num = num * 3;
        i++;
    }
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```




```
while(判斷式) {
        敘述式...
}
```

num	i
27	4

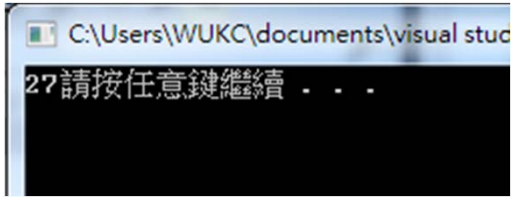


do-while 迴圈

```
do{
        敘述式…
} while(判斷式) ;
```

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    do {
        num = num * 3;
        i++;
    } while(i <= 3);
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```

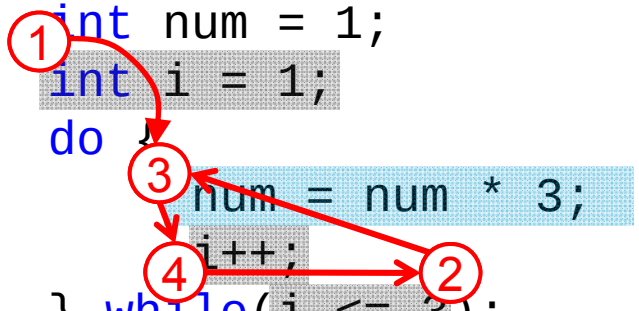
num	i
27	4



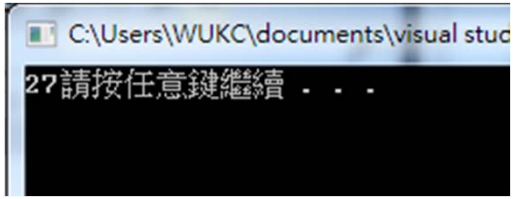
do-while 迴圈

```
do{
        敘述式…
} while(判斷式) ;
```

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    do {
        num = num * 3;
        i++;
    } while(i <= 3);
    cout << num;

    system("PAUSE");
    return 0; //回傳0，程式結束
}
```



num	i
27	4



while vs. do-while

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    while(i <= 0) {
        num = num * 3;
        i++;
    }
    cout << num;

    system("PAUSE");
    re
}
```

式結束

```
#include <iostream>
using namespace std;

//主函式
int main(void)
{
    int num = 1;
    int i = 1;
    do {
        num = num * 3;
        i++;
    } while(i <= 0);
    cout << num;

    system("PAUSE");
}
```

回傳0，程式結束

練習 3

- 考慮以下的演算法：
  1. 輸入 n
  2. 印出 n
  3. 如果 n = 1 結束
  4. 如果 n 是奇數 那麼 n=3*n+1
  5. 否則 n=n/2
  6. GOTO 2
- 例如輸入 22, 得到cycle length為 16的數列：

  22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1
- 輸入任意整數，求cycle length。
- 請測試右列數字：8, 9, 17

陣列

```
#include <iostream>
using namespace std;

int main(void)
{
    int array_a[10];
    int array_b[] = { 1, 2, 3 };
    const int array_size = 10;
    int array_c[array_size];

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

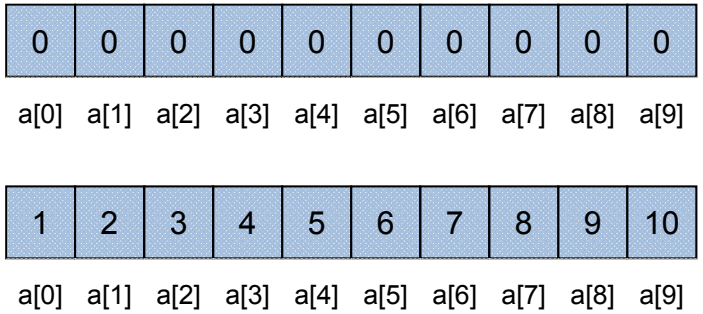
陣列型別	陣列	陣列大小	;
int	array	[10]	

陣列

```
#include <iostream>
using namespace std;

int main(void)
{
   int a[10] = { 0 }; //設定初始值為0
   for (int i = 0; i<10; i++) {
      a[i] = i + 1;
   }
   system("PAUSE");
   return 0;
}
```

陣列型別 陣列 陣列大小 ;

```
int array [10]
```



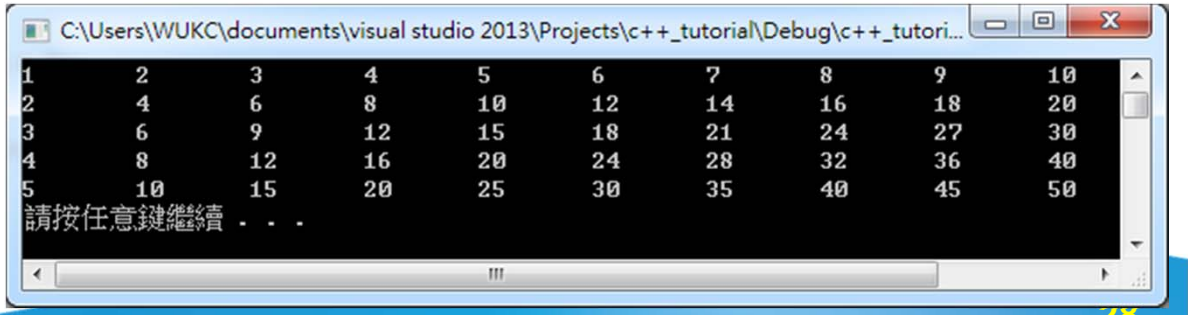
2維陣列

陣列型別	陣列	第一維大小	第二維大小；
int	array	[10]	[11] ;

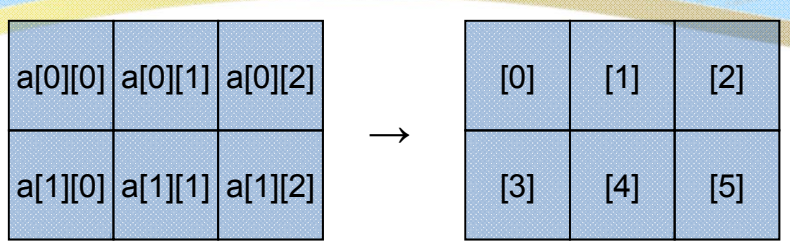
```
#include <iostream>
using namespace std;

int main(void)
{
        int array_a[2][3]; //設定大小
        int array_b[2][3] = { { 1, 2, 3 },
                              { 4, 5, 6 } }; //初始化
        int array_c[][3] = { { 1, 2, 3 },
                             { 4, 5, 6 } }; //初始化設定大小
        int array_d[2][3] = { 0 }; //初值為0
        const int row = 2;
        const int col = 3;
        int array_e[row][col];

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

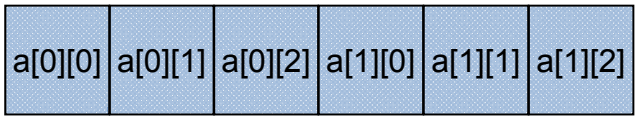
2維陣列

```
#include <iostream >
using namespace std;
int main(void)
{
        const int row = 5;
        const int col = 10;
        int array[row][col];
        for (int i = 0; i < row; i++)
                for (int j = 0; j < col; j++)
                        array[i][j] = (i + 1) * (j + 1);
        for (int i = 0; i < row; i++) {
                for (int j = 0; j < col; j++)
                        cout << array[i][j] << "\t";
                cout << endl;
        }
        system("PAUSE");
        return 0;
}
```



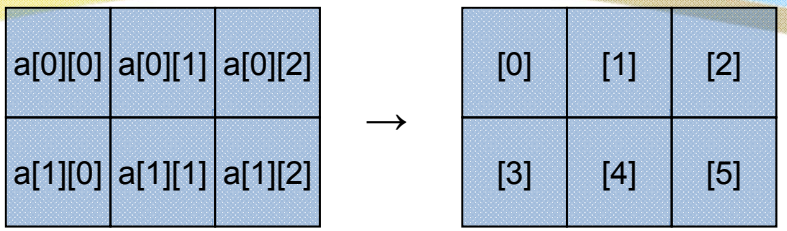
2維陣列



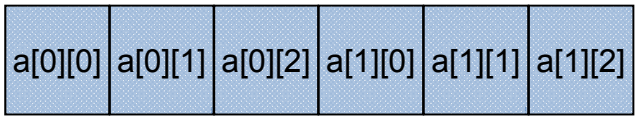



```
#include <iostream>
using namespace std;
//記憶體配置
int main(void)
{
        int a[2][3];
        for (int i = 0; i<2; i++)
                for (int j = 0; j<3; j++)
                        a[i][j] = i * 3 + j;
        for (int i = 0; i<2; i++){
                for (int j = 0; j<3; j++)
                        cout << a[i][j] << " ";
                cout << endl;
        }
        system("PAUSE");
        return 0;
}
```

2維陣列




```
#include <iostream>
using namespace std;

//記憶體配置
int main(void)
{
        int array_a[2][3] = { { 1, 2, 3 },
                              { 4, 5, 6 } }; //初始化
        int array_b[2][3] = { 1, 2, 3,
                              4, 5, 6 }; //初始化
        system("PAUSE");
        return 0;
}
```



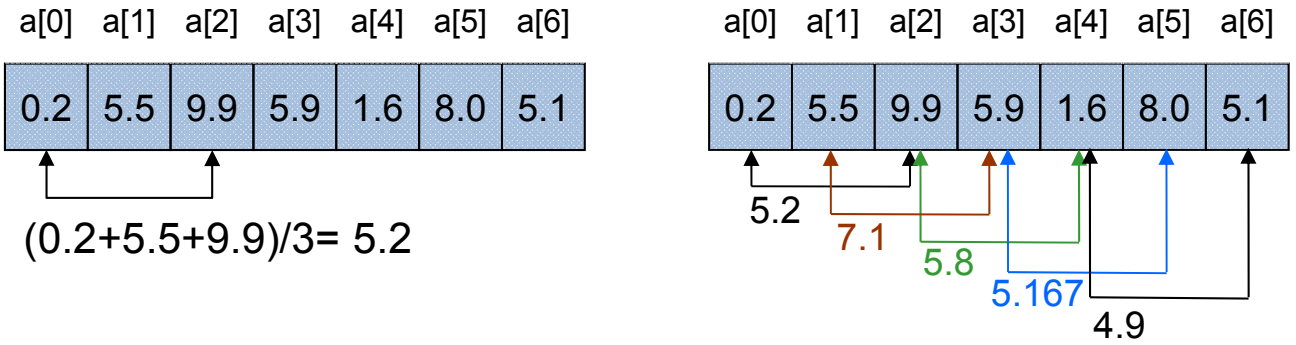
多維陣列

```
#include <iostream >
using namespace std;

int main(void)
{
        int array_a[2][3][4]; //設定大小
        //初始化
        int array_b[2][3][2] = { { {  0,  1 },
                                   {  2,  3 },
                                   {  4,  5 } },
                                 { {  6,  7 },
                                   {  8,  9 },
                                   { 10, 11 } } };
        int array_c[2][3][2] = {  0,  1,  2,  3,  4,  5,
                                  6,  7,  8,  9, 10, 11 };

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

練習 4 – 移動平均法

```
double a[7] = {0.2, 5.5, 9.9, 5.9, 1.6, 8.0, 5.1};
```

利用移動平均法，計算每三個數字為一期的平均數。



1. 已知平均期數為3期。
2. 可彈性輸入需平均之期數：
   - 要求使用者輸入一個介於2到7的數字，
     依據輸入的數字計算移動平均。

產生隨機亂數 (generating pseudo-random numbers) (1/5)

傳統C之亂數產生器：

- srand(time(0)); // 設定亂數種子(include <ctime>)
- srand(unsigned int); // 隨機亂數種子
- double x = double(rand())/RAND_MAX; // 產生[0,1]的浮點數亂數
- int y = rand()%n // 產生0~(n-1)的整數亂數

```
#include <iostream>
#include <ctime>
using namespace std;
int main()
{
        srand(time(0)); //初始化亂數種子
        for (int i = 0; i < 20; i++) {
                cout << rand() % 5 << " "; // 產生0-4的整數亂數
        }
        system("PAUSE");
        return 0;
}
```

產生隨機亂數 (generating pseudo-random numbers) (2/5)

visual studio 的 <random>：

- random_device是一種「不確定隨機數生成器 (non-deterministic random number generator) 」，但若環境沒有提供實現不確定隨機數生成器的條件，則無法使用。
- mt19937 以梅森旋轉法(mersenne twister)產生亂數。
- mt19937適合搭配機率分配使用，常用的分配包含：
  - uniform_real_distribution<type> dist(min, max);
  - normal_distribution<type> dist(mean, sigma);
  - student_t_distribution<type> dist(N);
  - poisson_distribution<type> dist(mean);

產生隨機亂數 (generating pseudo-random numbers) (3/5)

```
// random_device 範例，但因速度慢，通常只做種子使用
#include <iostream>
#include <random>
using namespace std;
int main(void) {
    random_device rd;
    cout << "min = " << rd.min() << endl;
    cout << "max = " << rd.max() << endl;
       for(int i=0; i<10; i++) {
              cout<<rd()<<endl;
       }
       system("PAUSE");
    return 0;
}
```

產生隨機亂數 (generating pseudo-random numbers) (4/5)

```
// MTd亂數產生器 範例
#include <iostream>
#include <random>
#include <ctime>
using namespace std;
int main(void) {
        random_device rd;
        mt19937 rng(rd()); //mt19937 rng(time(0));
        uniform_real<double> ugen(0.0, 1.0);
        for (int i = 0; i<10; i++) {
                cout << ugen(rng) << endl;
        }
        system("PAUSE");
        return 0;
}
```

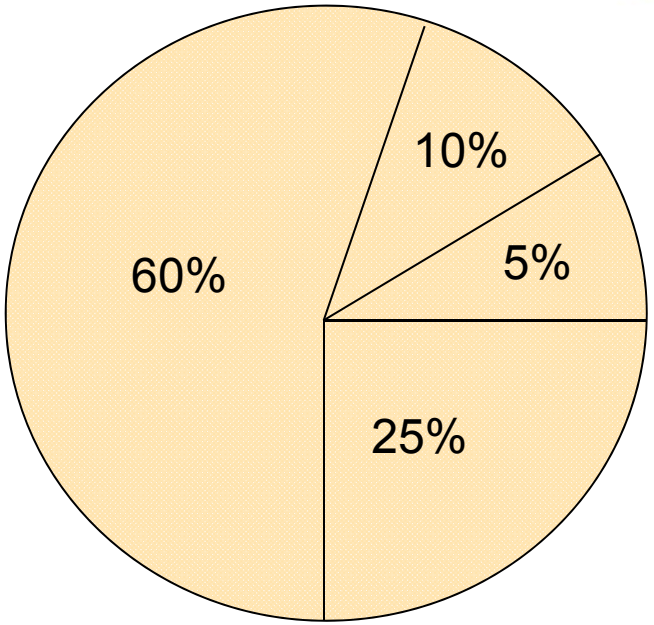
產生隨機亂數 (generating pseudo-random numbers) (5/5)

```
// MTd亂數產生器，使用「函數綁定」之範例
#include <iostream>
#include <random>
#include <ctime>
#include <functional>
using namespace std;
int main(void) {
        random_device rd;
        mt19937 rng(rd());
        uniform_real<double> u(0.0, 1.0);
        function<double(void)> ugen = bind(u, rng);
        for (int i = 0; i<10; i++) {
                cout << ugen() << endl;
        }
        system("PAUSE");
        return 0;
}
[[
```

Remark: 網路上有採用以 variate_generator 綁定(bind)「亂數產生器」與「隨機分配」的語法，但已在VS 2013被移除。

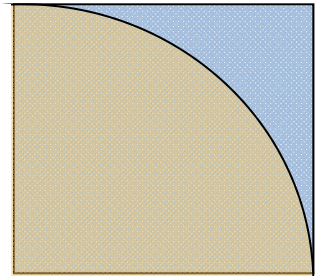
作業1

- 模擬射飛鏢的過程，標靶面積如圖所示，顯示射到每個面積區塊的飛鏢數量。



作業2

- 右圖有一個半徑為R=1的扇形與邊長L=1的正方形所構成的圖形，利用機率的方式隨機產生1000個介於[0 1]之間的數字代表所選的座標，請問有多少個點落入在藍色面積內?



L=R=1