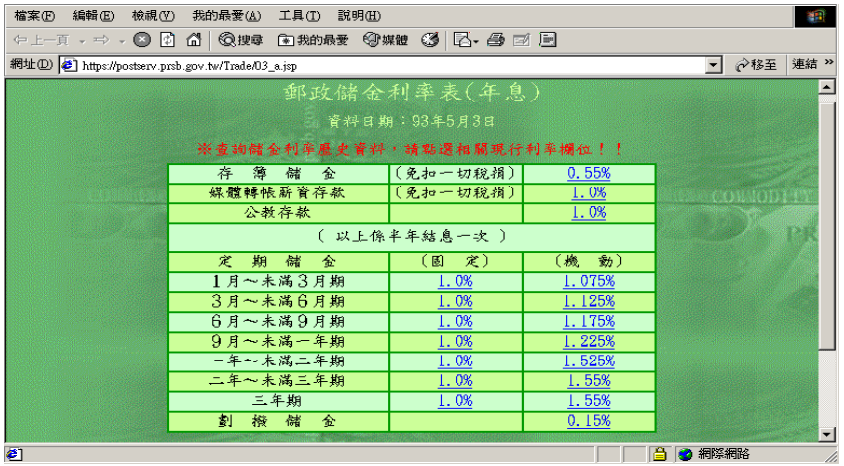
陣列結構

一維陣列
實例演練:不同天期的定存的未來值的計算
二維陣列及更高維度的陣列
實例演練:使用零息利率計算含息債券的價格
實例演練:使用拔靴法(Bootstrap method) 計算零息利率
實例演練:遠期利率的推算
實例演練:遠期利率鎖定
矩陣計算
實例演練:計算投資組合的變異度

考慮利率資料的變數宣告



變數宣告:
float IntFix1, IntFix3, …., IntFloat1, IntFloat3, ……
共需宣告14個變數,十分不便

使用陣列來儲存資料

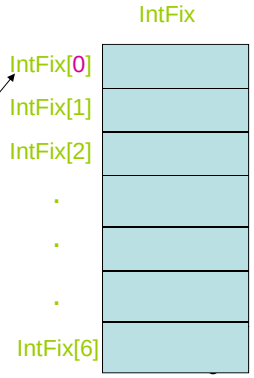
- 陣列
  - 在記憶體中佔一塊連續的儲存空間
  - 每一筆資料的資料型態都相同
  - 範例

```
float IntFix[7];
```

資料型態　陣列名稱　陣列大小

從編號0至6,共7筆

取代宣告 IntFix1, IntFix2,..

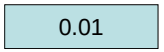


陣列資料的使用

變數宣告:
float a;
float IntFix[7];

參見 Array Project

a

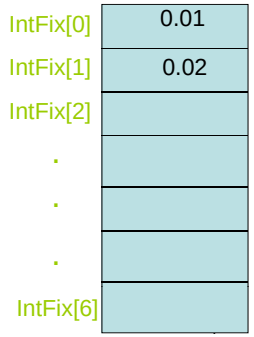


資料存取:
a=0.01;
IntFix[0]=a;
IntFix[1]=a+IntFix[0];

資料的輸入輸出
scanf("%f",&IntFix[2]);
printf("%f",IntFix[2]);

Don't write: IntFix[7]=>不存在

IntFix



課堂演練

- 使用for loop計算 IntFix[0]~IntFix[2]的和
- 使用for loop計算 IntFix[0]~IntFix[3]的和
  - 輸出的結果是亂數
  - IntFix[3]的值未經初始化,其值爲亂數

利率的輸入及定存的FV的計算

- 假定期初存1元,計算定存期滿的金額

```
for(int i=0;i<7;i=i+1)
{
 scanf("%f",&IntFix[i]);
}
a=1*(1+IntFix[0]/12);
printf("一個月定存FV:%f\n",a);
a=1*(1+IntFix[1]/4);
printf("三個月定存FV:%f\n",a);
a=1*(1+IntFix[2]/2);
printf("六個月定存FV:%f\n",a);
a=1*(1+IntFix[4]/2)*(1+IntFix[4]/2);
printf("一年定存FV:%f\n",a);
```

利率資料輸入到陣列中,使用控制變數 i決定資料擺放位置

計算一個月定存

計算一年定存,每半年付息一次

課堂練習

- 補足上述程式未完的部分
  - 九個月定存
  - 兩年期定存
  - 三年期定存
  - Remark: 使用for loop計算利息累乘的值

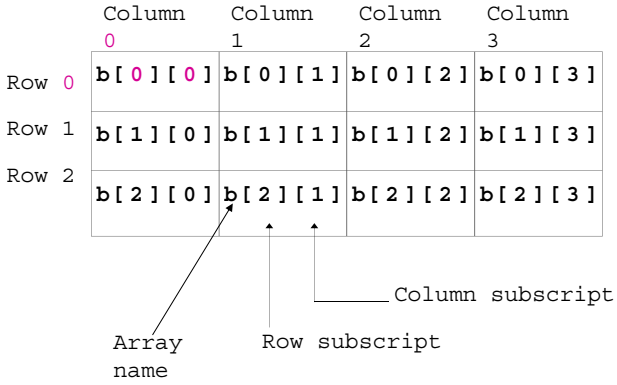
Ex: $\left(1+\frac{\text{IntFix[6]}}{2}\right)^{6}$

程式碼:

```
a=1;
for(int j=1;j<=6;j=j+1)
{
 a=a*(1+IntFix[6]/2);
}
```


二維陣列

- 假定一浮點數陣列有3列(rows),每列有4行(columns)
  - 陣列宣告: float b[3][4];
  - 共有3*4=12筆資料, 示意圖如下:



二維陣列資料的使用

變數宣告:

float b[3][4];

資料存取:

a=0.01;

b[0][0]=a;

b[0][1]=a+b[0][0];

資料的輸入輸出

scanf("%f",&b[0][2]);

printf("%f",b[0][2]);

Don't write: b[3][0] or b[0][4]

	Column 0	Column 1	Column 2	Column 3
Row 0	0.01	0.02		
Row 1				
Row 2				

高維度的陣列

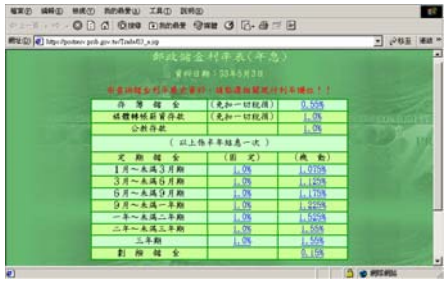
- 三維陣列:
  - Ex: int c[3][4][5];
  - 共有 3*4*5=60 個元素
  - 資料存取:

    c[l][m][n]= 1;

    a= c[l][m][n];

    l= 0~2
    m=0~3
    n=0~4
- 四維陣列:
  - Ex: float d[3][4][5][6];
  - 更高維度可依此類推

使用巢狀迴圈輸入利率資料



資料中包含固定利率和機動利率,
各有7筆資料:
1月,3月,6月,9月,1年,2年,3年
可使用二維陣列儲存
Int[2][7];
其中 Int[0][*] 儲存固定利率
Int[1][*] 儲存機動利率

陣列如下張投影片所示

程式碼

利率輸入

列印利率

```
float Int[2][7];
for(int l=0;l<7;l++)
{
 for(int m=0;m<2;m++)
 {
  scanf("%f",&Int[m][l]);
 }
}
printf("固定利率\n");
for(int k=0;k<7;k++)
{
 printf("%f\n", Int[0][k]);
}
printf("機動利率\n");
for(int p=0;p<7;p++)
{
 printf("%f\n", Int[1][p]);
}
```

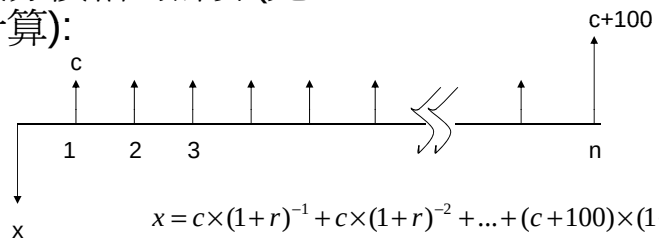

	0 (一月)	1 (三月)	2 (六月)	3 (九月)	4 (一年)	5 (兩年)	6 (三年)
0 固定							
1 機動							

課堂演練

- 利用上述程式,使用for loop計算不同天期的固定利率和機動利率的利差(interest rate spread)

殖利率 和 零息利率

- 債券價格的計算(見 Lecture 2 BondValue 計算):

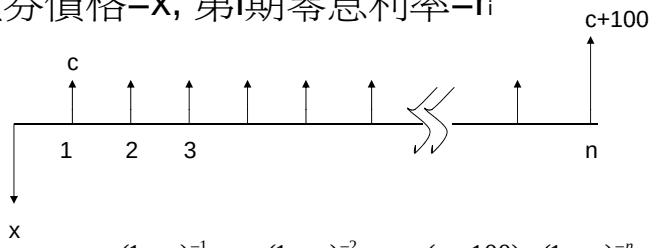


$$x = c\times(1+r)^{-1} + c\times(1+r)^{-2} + ... + (c+100)\times(1+r)^{-n}$$

- 不同的時間現金流用相同的殖利率折現
  - 無法計算任意現金流的折現值
- n年 零息利率: 可視爲n年後到期的零息債券的殖利率

使用 零息利率計算債券價格

- 債券價格=x, 第i期零息利率=r_i



$$x = c\times(1+r_1)^{-1} + c\times(1+r_2)^{-2} + ... + (c+100)\times(1+r_n)^{-n}$$

- 將含息債券拆解成n個零息債券,分別使用不同的零息利率折現
- 程式範例:考慮一個5期的債券,每期支付c元,債券價格可用下列程式計算


```
#include<stdio.h>
int main()
{
  float ZeroRate[5];
  float C;
  scanf("%f",&C);
  printf("輸入Zero rate:\n");
  for(int i=0;i<5;i=i+1)
  {
   scanf("%f",&ZeroRate[i]);
  }
  float BondValue=0;
  for(int k=0;k<5;k=k+1)
  {
   float Discount=1;
   for(int j=0;j<=k;j=j+1)
   {
    Discount=Discount/(1+ZeroRate[k]);
   }
   BondValue=BondValue+C*Discount;
   if(k==4)
   {BondValue=BondValue+100*Discount;}
  }
  printf("債券價格=%f", BondValue);

return 0;
}
```

見ZeroRate project

輸入zero rate

計算第i+1次payoff的PV

計算 $(1+r_{i+1})^{-(i+1)}$

課堂演練

- 假定該債券每期的債息不是定值,須由外界輸入
  - 宣告陣列 coupon[5]
  - 使用for loop輸入每一期的債息
  - 更改債券價格的計算程式

使用殖利率 求算零息利率

- 零息利率易於用來求算未來現金流的現值
- 然而市場上無法直接觀察到零息利率曲線
- 利用市場上交易的公債或利率交換,可觀察到 殖利率曲線
- 使用拔靴法(Bootstrap method)來求算 零息利率

方法簡介

- 考慮一個二期的模型,假定有兩個債券 B1,B2到期日分別爲第一和第二期. 債券殖利率分別爲 y1,y2,債券價格如下

$$B1=\frac{c+F}{(1+y1)} \qquad B2=\frac{c}{(1+y2)}+\frac{c+F}{(1+y2)^2}$$

- 考慮在第一期到期的零息債券,其零息利率 Z1=y1
- 考慮在第二期到期的零息債券,其零息利率 Z2計算如下:
  - B2可拆解成兩個零息債券,
    - 一個在第一期到期,面值爲c =>用 Z1折現
    - 另一個在第二期到期,面值爲c+F =>用Z2折現
  - 可得以下算式: $B2=\frac{c}{(1+Z1)}+\frac{c+F}{(1+Z2)^2}$ 因Z1已知,可求出Z2

$$Z2=\sqrt{\frac{c+F}{B2-\frac{c}{(1+Z1)}}}-1$$

n期的零息利率計算

- 假定Z1, Z2, Z3,… Zn-1 皆已知
- Zn滿足以下式子:

$$Bn=\frac{c}{(1+Z1)}+\frac{c}{(1+Z2)^2}+\ldots+\frac{c+F}{(1+Zn)^n}$$

- 移項可得 $(1+Zn)^n=\frac{c+F}{Bn-\frac{c}{(1+Z1)}-\frac{c}{(1+Z2)^2}-\ldots-\frac{c}{(1+Zn-1)^{n-1}}}$
- 下一程式計算一個五期的zero curve

Zn求算

$$(1+Zn)^n=\frac{c+F}{Bn-\frac{c}{(1+Z1)}-\frac{c}{(1+Z2)^2}-\ldots-\frac{c}{(1+Zn-1)^{n-1}}}$$

- 經移項 $Zn=\sqrt[n]{\frac{c+F}{Bn-\frac{c}{(1+Z1)}-\frac{c}{(1+Z2)^2}-\ldots-\frac{c}{(1+Zn-1)^{n-1}}}}-1$

- 在這裡引用了pow(x,y) 這個函數: $pow(x,y)=x^y$
  - 需加入 #include <math.h>
- ZeroRate[i]=pow((C+100)/BondValue,1.0/(i+1))-1;
- C語言提供許多好用的函數,可化簡程式
- 下一章 將會介紹函數的使用

```
float ZeroRate[5];
float Yield[5];
float C;
scanf("%f",&C);
for(i=0;i<5;i=i+1)
{
 printf("輸入Yield rate %d:",i+1);
 scanf("%f",&Yield[i]);
}
ZeroRate[0]=Yield[0];
for(i=1;i<=4;i++)
{
 float BondValue=0;
 for(j=0;j<=i;j=j+1)
 {
        float Discount=1;
        for(int k=0;k<=j;k++)
        {
         Discount=Discount/(1+Yield[i]);

        }
  BondValue=BondValue+Discount*C;
        if(j==i)
        {
         BondValue=BondValue+Discount*100;
        }
 }
 for(j=0;j<i;j=j+1)
 {
  float PV=C;
        for(int k=0;k<=j;k++)
        {
         PV=PV/(1+ZeroRate[j]);
        }
  BondValue=BondValue-PV;
 }
 ZeroRate[i]=pow((C+100)/BondValue,1.0/(i+1))-1
}
for(i=0;i<=4;i++)
{
 printf("第%d期zero rate=%f\n",i,ZeroRate[i]);
}
```

程式宣告

輸入殖利率

第一期Zero rate=Yield

計算第i+1期zero rate

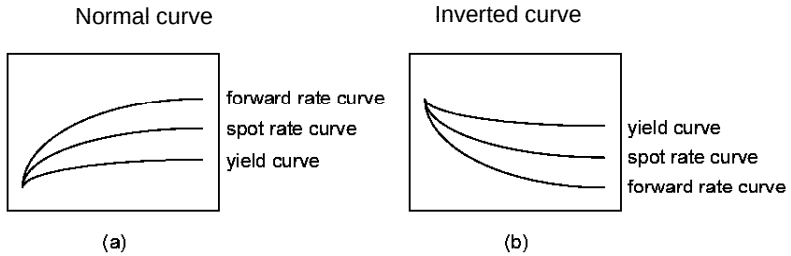
計算債券價格Bi+1

列印zero rate

見 ZeroCurve project

殖利率曲線, 零息利率曲線的關係

- 如果殖利率曲線爲一 Normal curve / Flat curve / Invert curve ,則零息利率…



課堂作業

- 使用pow(x,y)這個函式,化簡上述程式
- Remark:

```
for(int k=0;k<=j;k++)
        {
         Discount=Discount/(1+Yield[i]);
        }
```

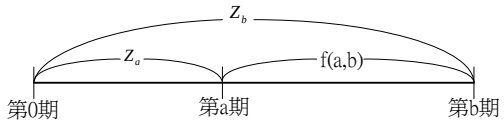
- 化簡成

```
Discount=1/pow(1+Yield[i],j+1);
```

- #include <math.h>

使用零息利率計算遠期利率

- 今天市場的零息利率期限結構，推論未來某段期間隱含的零息利率。



$$(1+Z_b)^b = (1+Z_a)^a(1+f(a,b))^{b-a}$$

$$f(a,b) = \sqrt[b-a]{\frac{(1+Z_b)^b}{(1+Z_a)^a}} - 1$$

使用二維陣列儲存**遠期利率**

行編號 列編號	0	1	2	3	4	5
0	f(0,0)	f(0,1)	f(0,2)	f(0,3)	f(0,4)	f(0,5)
1	X	f(1,1)	f(1,2)	f(1,3)	f(1,4)	f(1,5)
2	X	X	f(2,2)	f(2,3)	f(2,4)	f(2,5)
3	X	X	X	f(3,3)	f(3,4)	f(3,5)
4	X	X	X	X	f(4,4)	f(4,5)
5	X	X	X	X	X	f(5,5)


```
for(int i=0;i<5;i=i+1)
{
 printf("輸入第 %d期的零息利率:",i+1);
 scanf("%lf",&ZeroRate[i]);
}
```

輸入零息利率

```
for(i=0;i<5;i=i+1)
{
 Forward[0][i+1]=ZeroRate[i];
}
for(i=0;i<=5;i=i+1)
{
 Forward[i][i]=0;
}
```

ForwardRate[0,i+1]=ZeroRate

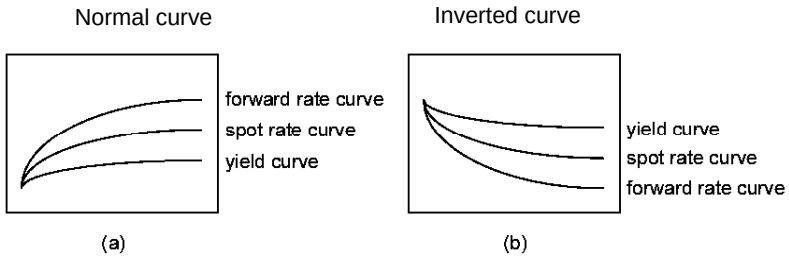
ForwardRate[i,i]=0

```
for(int a=1;a<5;a=a+1)//計算遠期利率
{
 for(int b=a+1;b<=5;b=b+1)
 {
 Forward[a][b]=pow(pow(1+Forward[0][b],b)/pow(1+Forward[0][a],a),1.0/(b-a))-1;
 }
}
```

$$f(a,b)=\sqrt[b-a]{\frac{(1+Z_b)^b}{(1+Z_a)^a}}-1$$

見ForwardRate project

殖利率曲線, 零息利率曲線, 遠期利率曲線關係



$$(1+Z_b)^b=(1+Z_a)^a(1+f(a,b))^{b-a}$$

當Zb>Za=> f(a,b)>Zb>Za

使用 上述程式驗證零息利率和遠期利率關係

課堂演練 遠期利率的鎖定

- f(i,j)不等於Z(i,j)
- 如何確保在時間i時,可用f(i,j)的利率投資
  - 于時間0時作如下規劃:
  - 買一個單位時間j到期的零息債券
  - 賣 $\frac{(1+Zi)^i}{(1+Zj)^j}$ 于時間i到期的零息債券
  - 今日的現金流量

$$\frac{(1+Zi)^i}{(1+Zj)^j}\times\frac{1}{(1+Zi)^i}-1\times\frac{1}{(1+Zj)^j}=0$$

課堂演練　遠期利率的鎖定

- 時間j時可拿到$1
- 時間i時需支付$ $\frac{(1+Zi)^i}{(1+Zj)^j}$
- 利率計算=> $\frac{(1+Zi)^i}{(1+Zj)^j}\times(1+Z)^{j-i}=1$

$$Z=F(i,j)$$

計算購買一單位的時間j到期的零息債券,須賣出多少單位時間i到期的零息債券,
才能鎖住 遠期利率 f(i,j),其中0<i<j<=5,並將結果存在
二維陣列LockStrategy[i,j]中

LockStratrgy陣列

行編號 列編號	0	1	2	3	4	5
0	X	X	X	X	X	X
1	X	X	$\frac{(1+Z_1)^1}{(1+Z_2)^2}$	$\frac{(1+Z_1)^1}{(1+Z_3)^3}$	$\frac{(1+Z_1)^1}{(1+Z_4)^4}$	$\frac{(1+Z_1)^1}{(1+Z_5)^5}$
2	X	X	X	$\frac{(1+Z_2)^2}{(1+Z_3)^3}$	$\frac{(1+Z_2)^2}{(1+Z_4)^4}$	$\frac{(1+Z_2)^2}{(1+Z_5)^5}$
3	X	X	X	X	$\frac{(1+Z_3)^3}{(1+Z_4)^4}$	$\frac{(1+Z_3)^3}{(1+Z_5)^5}$
4	X	X	X	X	X	$\frac{(1+Z_4)^4}{(1+Z_5)^5}$
5	X	X	X	X	X	X

使用二維陣列處理矩陣的運算
矩陣的輸入

開啓Matrix Project

外層迴圈:決定row index

```
printf("Input Matrix A\n");
for(int i=0;i<MATRIX_SIZE;i++)
{
  for(int j=0;j<MATRIX_SIZE;j++)
      {
        printf("A[%d,%d]=",i,j);
        scanf("%lf",&A[i][j]);
      }
}
```

內層迴圈:決定column index

矩陣A

A11	A12	A13
A21	A22	A23
A31	A32	A33

i的值的變化	j的值的變化	輸入的值存於陣列A的位置
0	0	A[0][0]
0	1	A[0][1]
0	2	A[0][2]
0	3(跳出內迴圈)	X
1	0	A[1][0]
1	1	A[1][1]
1	2	A[1][2]
1	3(跳出內迴圈)	X
2	0	A[2][0]
2	1	A[2][1]
2	2	A[2][2]
2	3 (跳出內迴圈)	X
3(跳出外迴圈)	X	X

使用二維陣列處理矩陣的運算
矩陣的加法

- 矩陣的加法 (以3*3的矩陣爲例)

A11	A12	A13
A21	A22	A23
A31	A32	A33

+

B11	B12	B13
B21	B22	B23
B31	B32	B33

=

A11+B11	A12+B12	A13+B13
A21+B21	A22+B22	A23+B23
A31+B31	A32+B32	A33+B33

使用巢狀迴圈處理

使用二維陣列處理矩陣的運算
矩陣的加法

- 使用巢狀迴圈處理

```
for(i=0;i<MATRIX_SIZE;i++)
{
  for( j=0;j<MATRIX_SIZE;j++)
        {
          C[i][j]=A[i][j]+B[i][j];
        }
}
```

矩陣的大小

外層迴圈:決定row index

內層迴圈:決定column index

```
for(i=0;i<MATRIX_SIZE;i++)
 {
   for( j=0;j<MATRIX_SIZE;j++)
          {
            printf("%lf\t",C[i][j]);
          }
          printf("\n");
 }
```

矩陣的輸出

換行

課堂演練
常數乘上矩陣

X

A11	A12	A13
A21	A22	A23
A31	A32	A33

=

X*A11	X*A12	X*A13
X*A21	X*A22	X*A23
X*A31	X*A32	X*A33

- 輸入常數X,
- 將計算結果放入矩陣C
- 將結果輸出

使用二維陣列處理矩陣的運算
矩陣的乘法

- 矩陣的乘法 (以3*3的矩陣爲例)

A11	A12	A13
A21	A22	A23
A31	A32	A33

×

B11	B12	B13
B21	B22	B23
B31	B32	B33

=

A11*B11+A12*B21+A13*B31	A11*B12+A12*B22+A13*B32	A11*B13+A12*B23+A13*B33
A21*B11+A22*B21+A23*B31	A21*B12+A22*B22+A23*B32	A21*B13+A22*B23+A23*B33
A31*B11+A32*B21+A33*B31	A31*B12+A32*B22+A33*B32	A31*B13+A32*B23+A33*B33

使用三重巢狀迴圈處理

使用二維陣列處理矩陣的運算
矩陣的乘法

外層迴圈:決定row index

內層迴圈:決定column index

```
for(i=0;i<MATRIX_SIZE;i++)
{
  for( j=0;j<MATRIX_SIZE;j++)
        {
          C[i][j]=0;
          for( k=0;k<MATRIX_SIZE;k++)
          {
                  C[i][j]+=A[i][k]*B[k][j];
          }
        }
}
```

Ex: C23=A21*B13+A22*B23+A23*B33

使用二維陣列處理矩陣的運算
向量和矩陣相乘

D1	D2	D3

×

B11	B12	B13
B21	B22	B23
B31	B32	B33

=

D1*B11+D2*B21+D3*B31	D1*B12+D2*B22+D3*B32	D1*B13+D2*B23+D3*B33

使用二維陣列處理矩陣的運算
向量和矩陣相乘

```
double D[MATRIX_SIZE],E[MATRIX_SIZE];
 printf("輸入向量D");
 for(i=0;i<MATRIX_SIZE;i++)
 {
   scanf("%lf",&D[i]);
 }

 for(i=0;i<MATRIX_SIZE;i++)
 {
   E[i]=0;
          for( j=0;j<MATRIX_SIZE;j++)
          {
           E[i]+=D[j]*B[j][i];
          }
 }
```

計算E[i]

E[2]= D1*B12+D2*B22+D3*B32

課堂演練:向量內積

- 使用迴圈計算向量D,E 內積

D1	D2	D3

•

E1	E2	E3

=

D1*E1+D2*E2+D3*E3

課堂演練:使用共變異數矩陣計算投資組合的變異度

- 假定一個投資組合由數個資產構成
- 每個資產的價格有不同的變異度
  - 股票: 高風險
  - 債券基金: 低風險
- 兩個資產之間的變異可能會有連動
  - Ex: 美股和台股的連動
- 可用共變異數矩陣表示
- 常用來處理VaR的問題

使用共變異數矩陣計算投資組合的變異度

- 矩陣的特性: (以三個資產爲例)

S=

	資產一	資產二	資產三
資產一	S11	S12	S13
資產二	S21	S22	S23
資產三	S31	S32	S33

  - Sij= Cov(第i個資產報酬率,第j個資產報酬率)
  - 矩陣對稱: Sij=Sji
  - 半正定性:對任意向量x $xSx^T \geq 0$

X可用來代表投資組合在不同標的物上投資的數量

X=

資產一投資數量	資產二投資數量	資產三投資數量

課堂演練:計算投資組合的變異度

- 開啓 PortfolioVar project
- 程式已處理好向量x和矩陣S的輸入
- 計算投資組合的變異度
  - 計算 xS ,得新向量y
  - 計算向量y和x內積→資產報酬率的變異度
- 95%信心水準下的VaR ($-1.645\times\sqrt{\text{變異度}}$)