控制迴圈

選擇執行架構 if
簡易的Range note 的計算
巢狀架構
現值的計算(考慮客戶的信用風險加碼)
重複執行架構 for, while
實務演練:債券價格和存續期間的計算
實務演練:使用二分法求IRR,和 yield spread

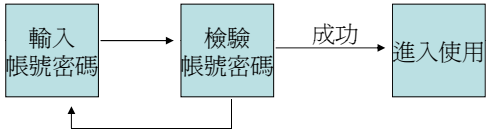
程式流程控制

- 一般而言,程式會循序執行

```
printf("The first program in your C++ \n");
printf("and quantitative finance class");
```

先執行
後執行

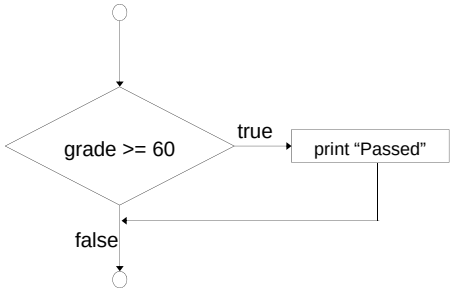
- 程式必須根據不同的狀況,執行不同程式碼
  - Ex:



- 程式流程控制結構
  - 選擇執行結構: if .. else .. ; switch
  - 重複執行結構: for, while, do while

if 控制結構

- Ex:



程式碼:見IfStructure project

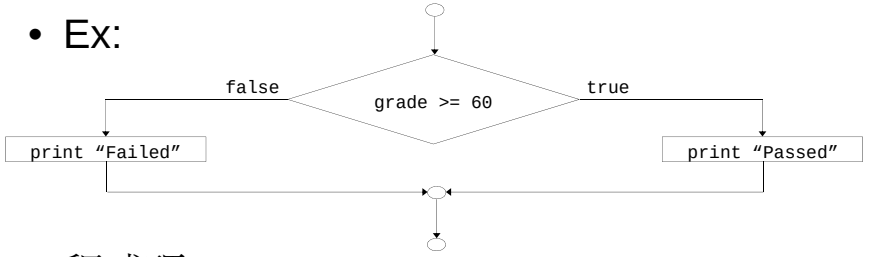
```
if ( grade >= 60 )
  {printf( "Passed\n" );}
```

- 條件若爲真(grade>=60)=> 則印出 Passed
- 條件句的撰寫:
  - grade>60; grade<=60; grade==60

單一等號代表將grade設成60分

if ..else控制結構

- Ex:



- 程式碼:

```
if ( grade >= 60 )
  {
   printf( "Passed\n" );
  }
 else
  {
    printf(“Failed\n”);
  }
```

if ..else控制結構 (左右括弧的省略)

見 IfStructure project

```
if ( grade >= 60 )
  {
   printf( "Passed\n" );
  }
 else
  {
    printf("Failed\n");
  }
```

當區塊中只有一行敘述時,左右括弧可省略

```
if ( grade >= 60 )
  printf( "Passed\n" );
 else
   printf("Failed\n");
```

```
if ( grade >= 60 )
  {
   printf( "Passed\n" );
   printf( "Passed\n" );
  }
 else
  {
    printf("Failed\n");
  }
```

左右括弧不可省略

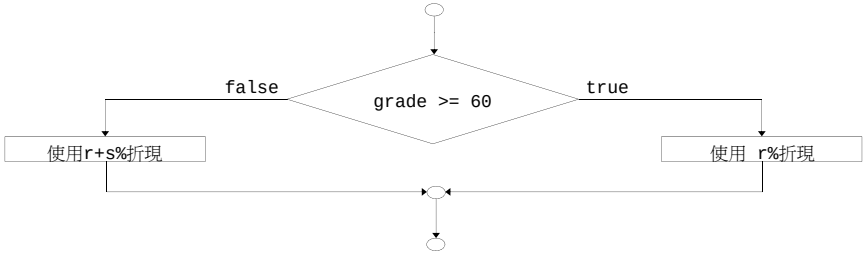
```
if ( grade >= 60 )
   printf( "Passed\n" );
   printf( "Passed\n" );
 else
  {
    printf("Failed\n");
  }
```

程式無法通過complier

左右括弧所包含的敘述在C的compiler中視爲完整一行敘述

折現值計算

- 假定利率r%,計算客戶A於一年後支付d元,兩年後支付e元的現值,假定客戶信用評等用grade表示,當grade<60,需考慮信用風險貼水s%.



程式碼

見PV project

```
#include<stdio.h>
int main()
{
    int grade;
        float d,e,f,r,s;
        scanf("%d",&grade);
        scanf("%f",&d);
        scanf("%f",&e);
        scanf("%f",&r);
        scanf("%f",&s);
        if(grade>=60)
        {
           f=d/(1+r);
           f=f+e/((1+r)*(1+r));
        }
        else
        {
           f=d/(1+r+s);
           f=f+e/((1+r+s)*(1+r+s));
        }
        printf("Present value=%f",f);

        return 0;
}
```

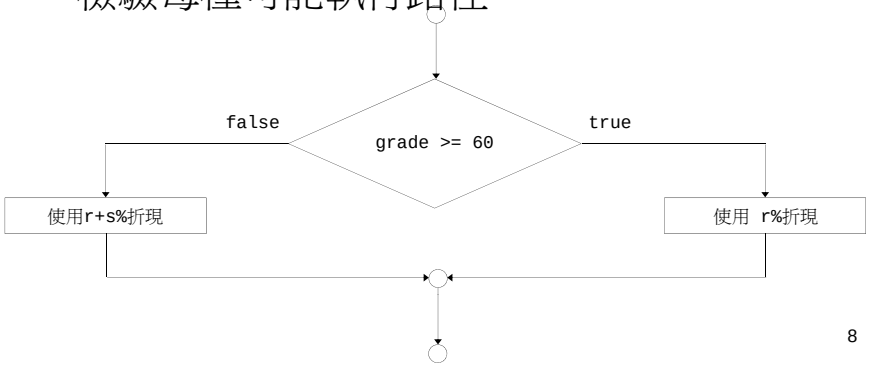
參數輸入

使用r%折現

使用r+s%折現

如何檢驗程式正確性

- 帶入 special case:
  - Ex: d=5, e=105, r=0.05 →PV=100
- 檢驗每種可能執行路徑



課堂演練
A very simple range note

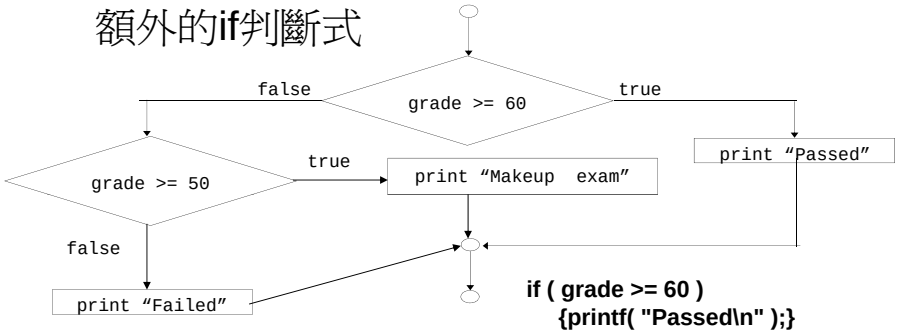
- 開啓新的project
- 輸入 rate1, rate2, rate3;
- 當 rate1, rate2, rate3 這三個數中,有兩個數在0.01以上,則Payoff=(大於0.01的個數)/2*100.5;
- 否則 Payoff=70

Hint:

```
double  a=(rate1>0.01)+(rate2>0.01)…
 if(…)
  {
    ….
  }
else
 {
   Payoff=70;
 }
```


巢狀式 if

- 如果有超過兩種以上的狀況要考慮,則需要額外的if判斷式



見 NestIf project

```
if ( grade >= 60 )
   {printf( "Passed\n" );}
 else
   {
    if(grade>=50)
     {printf(“Makeup exam\n”);}
    else
     {printf(“Failed\n”);}
   }
```


else if 敘述及Dangling else

```
if ( grade >= 60 )
   printf( "Passed\n" );
  else if(grades>=50)
     printf(“Makeup exam\n”);
   else
    printf(“Failed\n”);
```

=

```
if ( grade >= 60 )
   {printf( "Passed\n" );}
 else
   {
    if(grades>=50)
     {printf(“Makeup exam\n”);}
    else
     {printf(“Failed\n”);}
   }
```

```
if ( grade >=60)
        if (grade >=80)
                printf("Good job\n");
else
                printf("???");
```

else 歸最近的if所有

課堂練習

- 使用上述巢狀架構
- 更改上述折現值計算的問題 (PV project)
  - grade>=60 ->使用 (r+s)% 折現
  - 60>grade>=50 ->使用 (r+2s)% 折現
  - grade<50 ->使用 (r+3s)% 折現

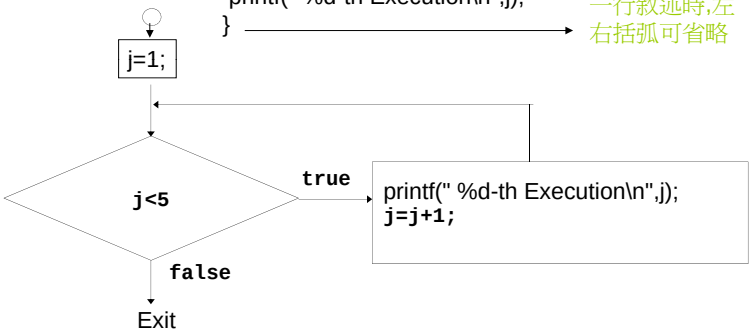
for 控制結構

- 透過for的結構,程式的片段可重複執行固定次數

檢視:BondValue project

```
for(int j=1;j<5;j=j+1)
{
 printf(" %d-th Execution\n",j);
}
```

當區塊中只有一行敘述時,左右括弧可省略



巢狀式的for 迴圈結構

```
for(int j=1;j<5;j=j+1)
{
  for(int z=1;z<j;z++)
   {
     printf(" %d-th Execution\n",j);
   }
}
```

- 迴圈中可再加入迴圈

變數j和z的改變狀況

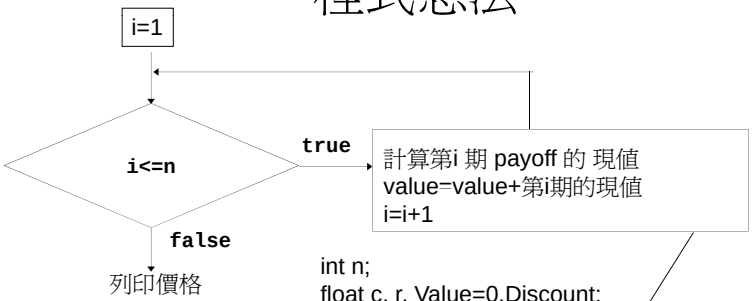
j=1	X
j=2	z=1
j=3	z=1
	z=2
j=4	z=1
	z=2
	z=3

for 控制結構

- 考慮債券價格的計算
  - 假定單期利率爲r
  - 每一期支付coupon c,共付n期
  - 到期日還本100元

債券價格P= $c\times(1+r)^{-1}+c\times(1+r)^{-2}+...+c\times(1+r)^{-n}+100\times(1+r)^{-n}$

程式想法




```
int n;
float c, r, Value=0,Discount;
scanf("%d",&n);
scanf("%f",&c);
scanf("%f",&r);
for(int i=1;i<=n;i=i+1)
{
}
printf("BondValue=%f",Value);
```

計算第i 次 payoff的 現值

- i<n 現值= $(1+r)^{-i} \times c$
- i=n 現值= $(1+r)^{-n} \times (c+100)$
- 用for計算 $(1+r)^{-i}$

計算第i 次 payoff的 現值

計算$(1+r)^{-i}$

考慮最後一期本金折現

```
Discount=1;
for(int j=1;j<=i;j++)
{
 Discount=Discount/(1+r);
}
Value=Value+Discount*c;
if(i==n)
{
 Value=Value+Discount*100;
}
```


完整程式碼(包含巢狀結構)

```
#include <stdio.h>
int main()
{
        int n;
        float c, r, Value=0,Discount;
       scanf("%d",&n);
        scanf("%f",&c);
        scanf("%f",&r);
       for(int i=1;i<=n;i=i+1)
        {
         Discount=1;
         for(int j=1;j<=i;j++)
         {
          Discount=Discount/(1+r);
         }
        Value=Value+Discount*c;
         if(i==n)
         {
          Value=Value+Discount*100;
         }
        }
   printf("BondValue=%f",Value);

    return 0;
}
```

第i 次 payoff的 現值

Value爲前i次payoff現值

課堂練習
債券價格的特性

- 當票面利率=折現率
  - 債券價值=
- 票面利率上升(下降)
  - 債券價值 …
- 折現利率上升(下降)
  - 債券價值 …
- 每一期的債息=票面利率*面值
  - 驗證程式的答案是否滿足這些特性

課堂練習
Macaulay Duration

- 使用每一段現金流的現值作爲權重,計算平均的還本時間
- 以含息債券 爲例(F: 票面價值)

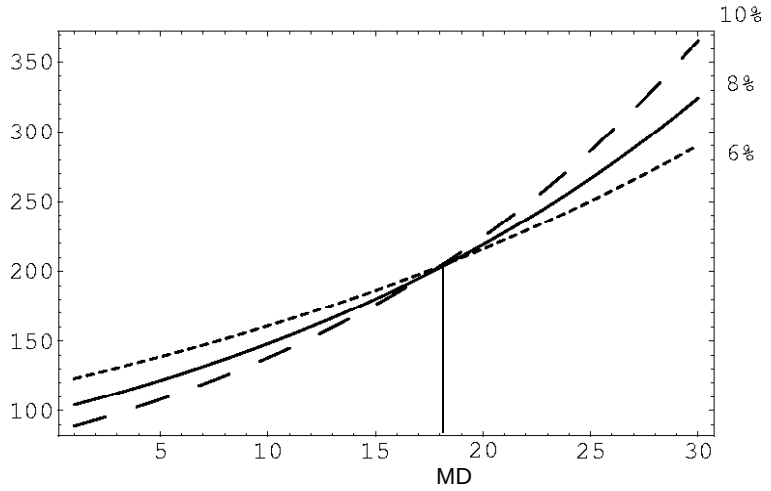
$$\mathrm{MD} = \frac{1}{P}\left(\sum_{i=1}^{n} \frac{ic}{(1+r)^i} + \frac{nF}{(1+r)^n}\right) = -(1+r)\frac{\partial P/P}{\partial r}$$

彈性

- 零息債卷的 MD=n

課堂練習
Macaulay Duration的特性

- 當持有債券時,持有的價格會隨著市場利率(r)的波動而改變
  - Ex: r下降,債券價格上升
  - Ex: r上升,債券價格下降
- 假定債券的債息可用r再投資
  - 可得下述圖形
  - r上升時,持有債券價格下降,但債息的再投資所得上升
  - r下降時,持有債券價格上升,但債息的再投資所得下降



Proof for Macaulay Duration

- Let $FV \equiv P(1+r)^m$, where P is the PV of the portfolio. Now,

$$\frac{\partial FV}{\partial r} = m(1+r)^{m-1}P + (1+r)^m \frac{\partial P}{\partial r}$$

- Imposing $\frac{\partial FV}{\partial r} = 0$ leads to

$$m = -(1+r)\frac{\partial P / P}{\partial r}$$

- The MD is equal to the horizon *m*.

課堂練習
Macaulay Duration的特性

- 在平均還本時間時,手中的債券價格和債息的再投資價格不隨利率變動而變動
  - r上升時,持有債券價格下降,但債息的再投資所得上升抵銷損失的部份
  - r下降時,持有債券價格上升,但債息的再投資所得下降抵銷盈餘的部份

課堂練習
應用:Immunization

- 意義:對於利率的波動造成的風險免疫
- 假定未來有一筆債務,在時間m時須支付B元
- 市場上未必有時間m到期的零息債券
- 購買一債券組合,其在時間m的Future value=B, MD=m

課堂練習

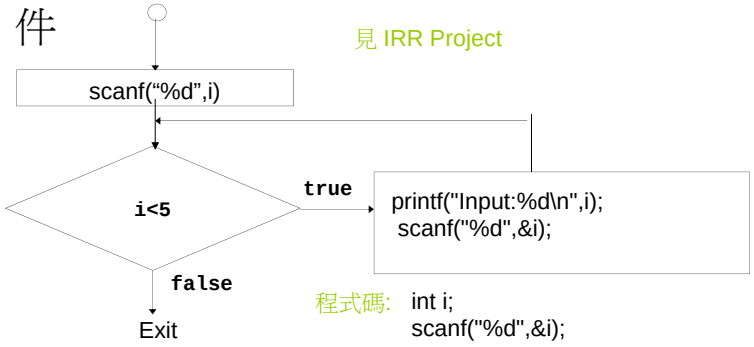
- Duration的計算

$$D = \frac{1}{P}\left(\sum_{i=1}^{n}\frac{ic}{(1+r)^i} + \frac{nF}{(1+r)^n}\right)$$

- 利用for loop同時求算 $\frac{1}{P}$ 和 $\sum_{i=1}^{n}\frac{ic}{(1+r)^i} + \frac{nF}{(1+r)^n}$
- 相乘即爲答案

while 控制結構

- 程式的片段可重複執行,直到滿足某預設條件



見 IRR Project

程式碼:

```
int i;
scanf("%d",&i);
while(i<5)
{
 printf("Input:%d\n",i);
 scanf("%d",&i);
}
```

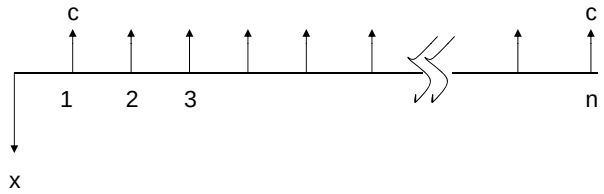

“break” and “continue”

- 兩者皆爲C++在控制結構中的指令
  - break: 跳出迴圈到下一行
  - continue:回到迴圈的第一行繼續執行

```
int i;
 scanf("%d",&i);
 while(i<5)
 {
  printf("Input:%d\n",i);
  scanf("%d",&i);
  if(i==3) break;
  continue;
  printf("這行不會被執行");    這行不會被執行
}
```

while 控制結構

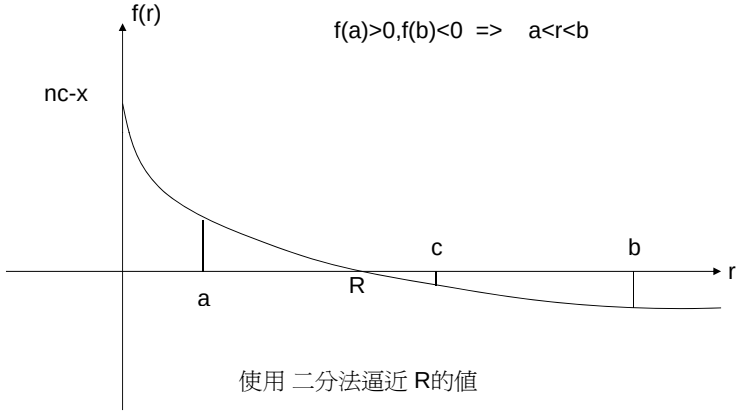
- 考慮IRR (internal rate of return)的計算
  - 假定期初投資x元,每期可回收c元,共回收n期,求投資報酬率R



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

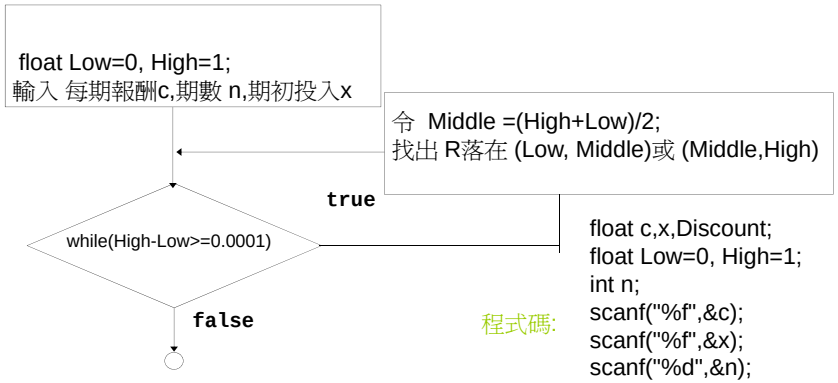
勘根定理及 二分法 (Bisection Method)

- 令 $f(r) = c\times(1+r)^{-1} + c\times(1+r)^{-2} + ... + c\times(1+r)^{-n} - x$
- 本問題相當於解f(r)=0的根



使用 二分法逼近 R的值

使用 while 建構 二分法



程式碼:

```
float c,x,Discount;
float Low=0, High=1;
int n;
scanf("%f",&c);
scanf("%f",&x);
scanf("%d",&n);
while(High-Low>=0.0001)
{
                    
}
printf("Yield rate=%f",High);
```


用Bisection method縮小根的範圍

- 已知 $f(r) = c\times(1+r)^{-1} + c\times(1+r)^{-2} + ... + c\times(1+r)^{-n} - x$
  - f(r)<0 → r >R
  - f(r)>0 → r<R
- 令 Middle=(High+Low)/2
  - 將根的範圍從(Low,High)縮減到
    - (Low,Middle)
    - (Middle,High)

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

用計算債券的公式計算

```
float  Middle=(Low+High)/2;
float Value=0;
for(int i=1;i<=n;i=i+1)
{
 Discount=1;
 for(int j=1;j<=i;j++)
 {
          Discount=Discount/(1+Middle)
 }
  Value=Value+Discount*c;
}
Value=Value-x;
if(Value>0)
          { Low=Middle;}
 else
          {High=Middle;}
```

縮小根的範圍

計算 IRR (完整程式碼)

用while控制根的範圍

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

計算 $(1+r)^{-i}$

縮小根的範圍

```
float c,x,Discount;
float Low=0, High=1;
int n;
scanf("%f",&c);
scanf("%f",&x);
scanf("%d",&n);
while(High-Low>=0.0001)
{
 float  Middle=(Low+High)/2;
 float Value=0;
 for(int i=1;i<=n;i=i+1)
 {
  Discount=1;
  for(int j=1;j<=i;j++)
  {
          Discount=Discount/(1+Middle);
  }
   Value=Value+Discount*c;
 }
 Value=Value-x;
 if(Value>0)
          { Low=Middle;}
  else
          {High=Middle;}
}
printf("Yield rate=%f",High);
```


課堂練習: *Yield Spread*

- 投資計畫的期望報酬，通常與風險成正相關
- 期望報酬超過無風險報酬稱爲風險貼水
- yield spread，指的是根據一家公司的債券市場價格，反推出該債券的風險貼水。

$$\text{債券價格} = \sum_{i=i}^{n} \frac{c}{(1+R_f+S)^i} + \frac{100}{(1+R_f+S)^n}$$

c：債息
R_f：無風險利率
n：期數
S：yield spread

給定債息,無風險利率,期數,債券價格,求yield spread

Variable Scope Rules
Block Scope

- Declare data inside compound statement
  - Called a ‘block’
  - Has ‘block-scope’
- Loop blocks:

```
for (int ctr=0;ctr<10;ctr++)
{
   int sum+=ctr;
}
```

  - Variable sum has scope in loop body block only
- Note: all function definitions are blocks!
  - This provides local ‘function-scope’

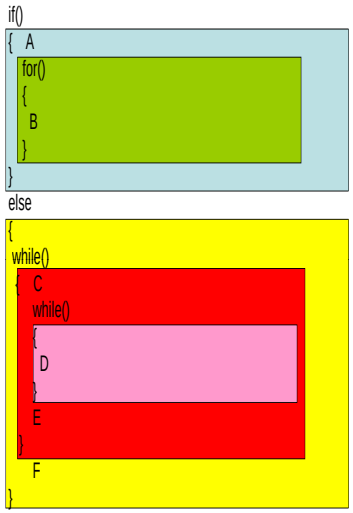
Nested Scope

- Variables with the same name can be declared in multiple blocks
- Legal; scope is ‘block-scope’
  - No ambiguity
  - Each name is distinct within it’s scope

```
int a=1;
while (a>0)
{
  double a=0;
  cout<<a;   → a refers to?
}
```

See VariableScope Project

```
int Z;
printf("請輸入Z");
 scanf("%d",&Z);
 if(Z)
 {
    int a=1;
    for(;Z>0;Z=Z-1)
     {
      int b=a+1;
                    a=b;
    }
  printf("a=%d",a);
  //printf("b=%d",b);  // 不合法引用
}
 else
 {
  //printf("a=%d",a);  // 不合法引用
  int a=1;
    while(Z==0)
  {
     int a=2;
       while(Z==0)
      {
        Z=Z-a;
      }
      printf("Z=%d\n",Z);
  }
  Z=Z-a;
  printf("Z=%d\n",Z);
}
```



更多相關細節會在Lecture 4 提到

ISO Standard

- Many years ago, you will see code like this:
  - for(int i =0; i<10; i++){

    ……

    }

    for(i = 0; i<30; i++){

    ……

    }

ISO Standard

- But since the new ISO Standard for C++ (ISO C++ 03) had launched, this way for declaration was no longer support.
- So, you have to rewrite your code like:
  - int i;

    for(i=0;i<10;i++){

    …….

    }

    for(i=0; i<30; i++){

    ……..

    }

ISO Standard

- Or

for(int i=0; i<10; i++){

……..

}

for(int k=0;k<30;k++){

…..

}

The two writing styles mentioned above are all legal with the new ISO Standard.