

課堂練習

對 $C^A \times C^B$ 取log

得 $A \log C + B \log C$

→ $C^A \times C^B = \exp(A \log C + B \log C)$

- 分別用C的運算式計算左右兩式
 - 令左式=Result, 右式=Result1
 - 使用 `if(Result==Result1)...` 來判斷此兩式是否相等
 - 請使用 `fabs()` 函式解決此問題

課堂練習

- 比較下列程式碼

```
double a=pow(0.1,1000)*pow(0.1,-1000);  
printf("%lf",a);
```

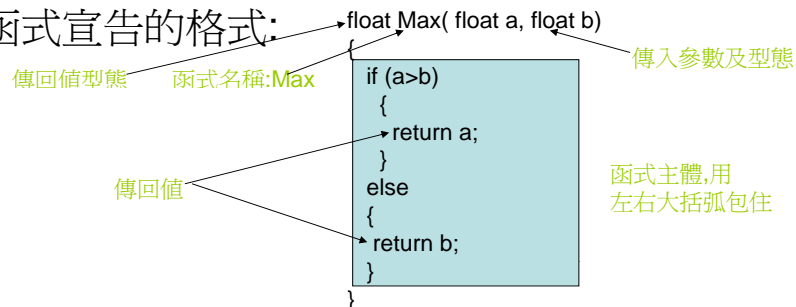
```
double a=exp(1000*log(0.1)-1000*log(0.1));  
printf("%lf",a);
```

兩者算出的答案相同嗎?

使用者定義的函式

- C++ 語言允許使用者自訂函式
 - `main()` 本身就是一個函式, 程式執行時由系統呼叫 `main()`
 - 其他的函式則經使用者的程式呼叫而執行

- 函式宣告的格式:



程式範例

```
#include <stdio.h>  
float Max( float a, float b)  
{  
    if (a>b)  
    {  
        return a;  
    }  
    else  
    {  
        return b;  
    }  
}  
  
void main()  
{  
    float a,b,c;  
    scanf("%f",&a);  
    scanf("%f",&b);  
    c=Max(a,b);  
    printf("%f和%f中較大的是%f",a,b,c);  
    printf("%f和%f中較大的是%f\n",-a,-b,Max(-a,-b));  
}
```

參見 UserDefineFunction Project

Max函式

Remark:
函式必須在呼叫前宣告

呼叫Max函式

函數偵錯

```
#include <stdio.h>
float Max( float a, float b)
{
    if (a>b)
    {
        return a;
    }
    else
    {
        return b;
    }
}

void main()
{
    float a,b,c;
    scanf("%f",&a);
    scanf("%f",&b);
    c=Max(a,b);
    printf("%f和%f中較大的是%f",a,b,c);
    printf("%f和%f中較大的是%f\n",-a,-b,Max(-a,-b));
}
```

- F7 (Next line)
 - 執行到原函數的下一行
- F8 (Into line)
 - 跳入被呼叫的函數執行

課堂演練

- 撰寫 double logab(double a, double b)
 - 傳回 $\log_a b (= \frac{\ln b}{\ln a})$
 - 呼叫該函式

Scope Rules

- Local variables
 - Declared inside body of given function
 - Available only within that function
- Can have variables with same names declared in different functions
 - Scope is local: 'that function is it's scope'
- Global variables
 - Declared 'outside' function body
 - Global to all functions in that file

範例程式(See VarScope Project)

```
int a=2; // global variable
void f ()
{
    int b=3; // Local to function f
    printf("%d",a);
}
int main(int argc, char *argv[]){
    printf("%d",b); //illegal use
    int a=3; // local to function main
    printf("%d",a);
    return 0;
}
```

隨堂演練：程式的輸出

```
int a=2;
void f(int b)
{
    double a=3;
    cout<<a+b<<endl;
}
int main(int argc, char *argv[]){
    cout<<a<<endl;
    for(int a=1;a<3;a++)
    {
        f(a);
    }
    return 0;
}
```

See VarScope1 Project

Variable Usage

- Local variables preferred
 - Maintain individual control over data
- Global variables?
 - Possible, but SELDOM-USED
 - Dangerous: no control over usage!

Black-Scholes Formula

- Black-Scholes Formula的買權評價公式可

表示如下： $SN(d_1) - Xe^{-rT}N(d_2)$

$$d_1 = \frac{\ln(S/X) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$

$$d_2 = d_1 - \sigma\sqrt{T}$$

- 其中**N(*)**表標準常態分配的累積機率函數
- 需要用到的函式
 - 使用者自訂: **N(*)**
 - 系統提供: **log()**, **sqrt()**, **exp()**

程式範例

參見BSCall Project

Cumulative Normal Distribution

```
double Standard_Normal_Distribution(double d)
{
    int flag=0; //Flag =1 if d<0
    if(d<0)
    {
        flag=1;
        d=fabs(d);
    }
    double rr=0.2316419;
    double a1=0.31938153;
    double a2=-0.356563782;
    double a3=1.781477937;
    double a4=-1.821255978;
    double a5=1.330274429;
    double k=1/(1+d*rr);
    double PI=3.14159265359;
    double value=1-exp(d*d/(-2))*
    (a1*k+a2*pow(k,2)
    +a3*pow(k,3)+a4*pow(k,4)
    +a5*pow(k,5))/sqrt(2*PI);
    if(flag) return 1-value;
    else return value;
}
```

main function

```
float S,r,T,X,V;
printf("輸入股價:");
scanf("%f",&S);
printf("輸入利率:");
scanf("%f",&r);
printf("輸入到期日:");
scanf("%f",&T);
printf("輸入履約價:");
scanf("%f",&X);
printf("輸入波動率:");
scanf("%f",&V);
double b=1/exp(r*T);
double d1=(log(S/(X*b))+V*V*T/2)/(V*sqrt(T));
double d2=d1-V*sqrt(T);
double Value=S*
Standard_Normal_Distribution(d1)
-X*b*Standard_Normal_Distribution(d2);
printf("Call value=%f",Value);
```

課堂練習

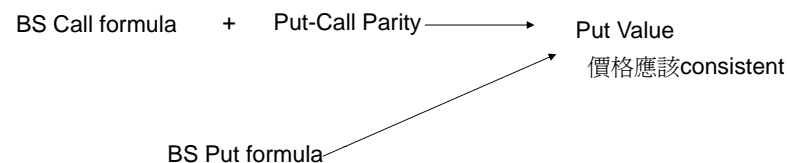
- 仿照上述程式,計算賣權的價值
 - 不要修改買權的程式碼,在 **put-call parity**會用到
- Remark:賣權價值: $Xe^{-rT}N(-d_2) - SN(-d_1)$

使用Put-Call Parity評價賣權

- 考慮買權和賣權到期日的報酬:
 - 買權: $(S(T) - X)^+$ 賣權: $(X - S(T))^+$
- 如果買一個買權,賣一個賣權,到期日報酬
 - $(S(T) - X)^+ - (X - S(T))^+ = S(T) - X$
 - 其報酬相當於期初借 Xe^{-rT} ,並購買標的物
- 在無套利的假定之下,假定C,P為買權和賣權的價格,可得 $C - P = S - Xe^{-rT}$

課堂練習

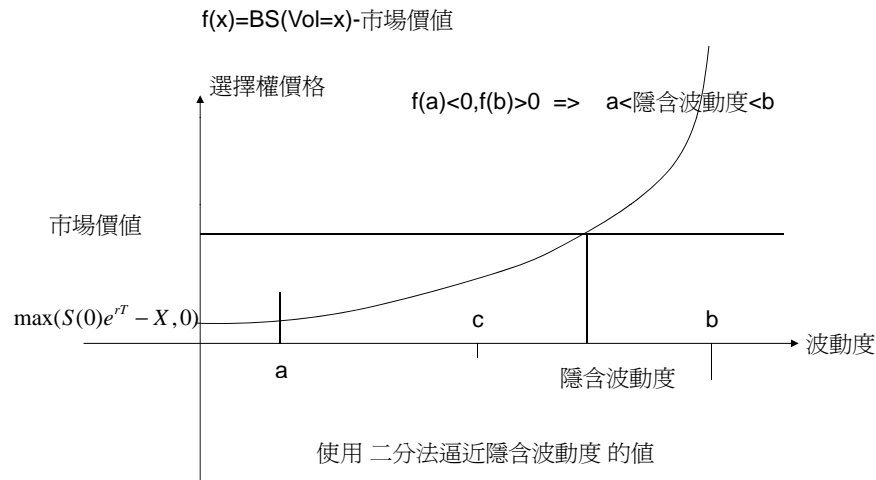
- 使用put-call parity來求算賣權的價值
- 比較使用Black-Scholes Formula和put-call parity計算出的賣權的價值
- 程式驗證:
 - 如何擔保程式碼以及概念推導過程正確



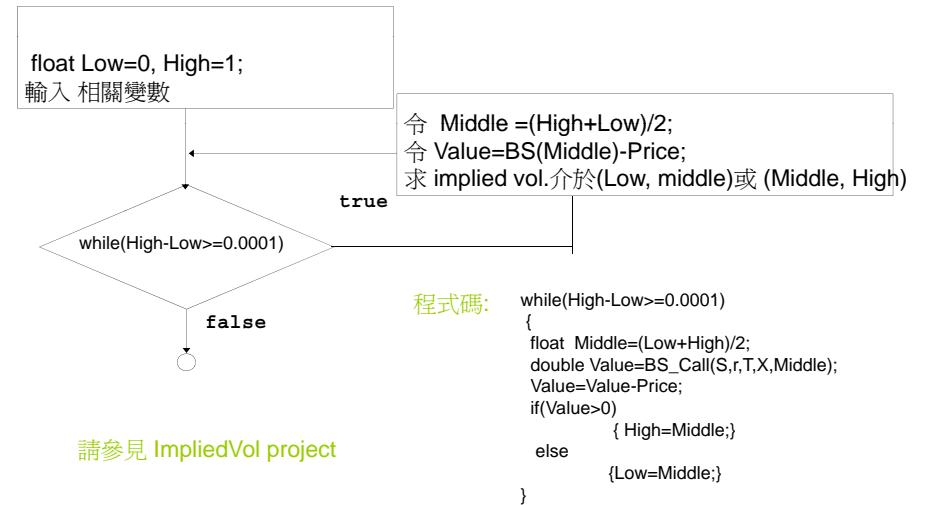
隱含波動度(Implied Volatility)

- 無法直接觀察股票的波動度(Volatility)
- 欲估計波動度
 - 歷史波動度(Historical volatility)
 - 不一定能夠忠實表達未來真實的波動度
 - 隱含波動度
 - 用市場上觀察到的選擇權價格,代入Black-Scholes Formula反求
 - 可使用二分法(Bisection method)估計

波動度和選擇權價格關係



程式結構



課堂練習: 利用賣權價格逆推隱含波動度

- 修改上述程式, 用賣權價格倒推隱含波動度
- Hint:
 - 撰寫一函式 `BS_Put()`, 用來求算賣權價格
 - 修改主程式

```

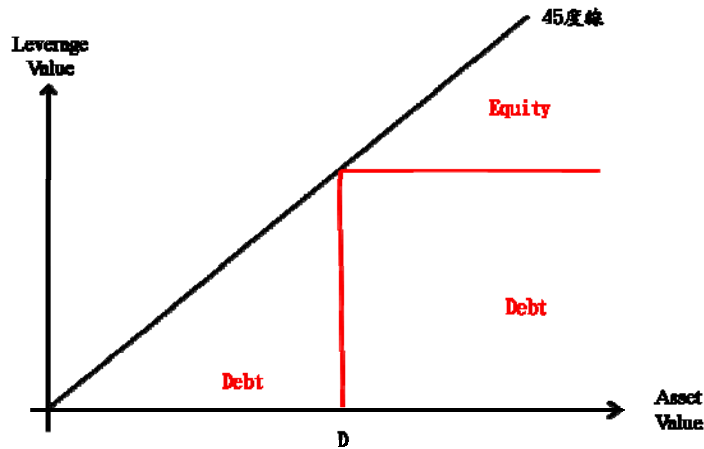
while(High-Low >= 0.0001)
{
    float Middle=(Low+High)/2;
    double Value=BS_Put(S,r,T,X,Middle);
    Value=Value-Price;
    if(Value > 0)
        { High=Middle;}
    else
        {Low=Middle;}
}
    
```

Homework:

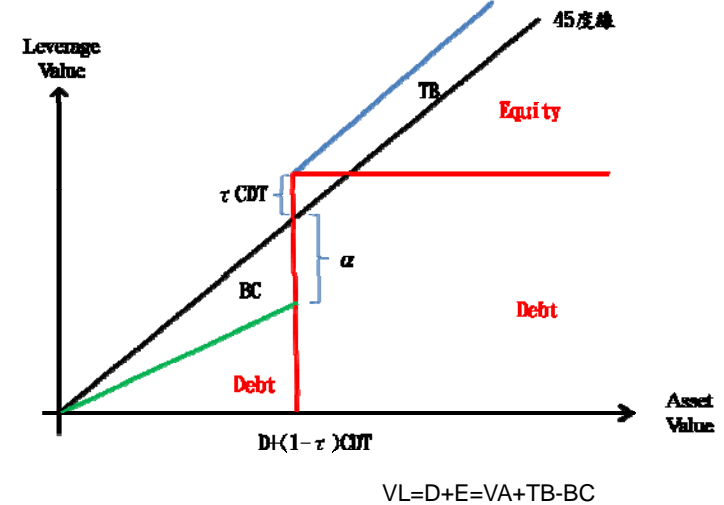
The Structural Credit Risk Model

- 使用選擇權評價模型估算公司的信用風險
- 假定公司資產為服從對數常態分配的標的物
- Equity can be treated as a call option on the firm's asset
- Debt value can be viewed as a contingent claim on the firm value
- See the next figure

Merton's Structural Credit Risk Model



Consider Tax Benefit (TB) and Bankruptcy Cost (BC)



障礙選擇權

- 選擇權的價格常會受標的物的價格路徑而影響,令障礙值=H

– 下出局(Down-and-out) 買權:

$$\text{payoff} = \begin{cases} (S(T)-X)^+ & \text{if } S(t) > H \quad \forall t \in (0, T) \\ 0 & \text{if } \exists t \in (0, T) S(t) \leq H \end{cases}$$

– 下入局(Down-and-in) 買權:

$$\text{payoff} = \begin{cases} 0 & \text{if } S(t) > H \quad \forall t \in (0, T) \\ (S(T)-X)^+ & \text{if } \exists t \in (0, T) S(t) \leq H \end{cases}$$

– In-out parity:

下入局買權+ 下出局買權= 標準買權

課堂演練: 呼叫函數計算 下入局買權的價值

- 下入局買權的價值 請參見 ExoticOption project

$$S(H/S)^{2\lambda} N(y) - Xe^{-rT} (H/S)^{2\lambda-2} N(y - \sigma\sqrt{T})$$

$$\lambda = \frac{r + \sigma^2/2}{\sigma^2} \quad y = \frac{\ln(H^2/(SX))}{\sigma\sqrt{T}} + \lambda\sigma\sqrt{T}$$

prototype

```
double DownAndInCall (double S, double T, double X, double H, double r, double q, double Sigma)
{
    double OptionValue;
    double lambda = (r-q+0.5*Sigma*Sigma)/(Sigma*Sigma);
    double y = log((H*H)/(S*X))/(Sigma*sqrt(T)) + lambda*Sigma*sqrt(T);
    OptionValue = S * pow((H/S), 2*lambda) * Standard_Normal_Distribution(y) / exp(q*T) -
    X * pow((H/S), (2*lambda-2)) * Standard_Normal_Distribution(y - Sigma*sqrt(T)) / exp(r*T);
    return OptionValue;
}
```

implementation

S: 股票價格, T: 距到期日的時間長度, X: 履約價格, H: 障礙價格, r: 無風險利率, q: 股利率, Sigma: 股價波動度

課堂演練: 呼叫函數計算 下出局買權的價值

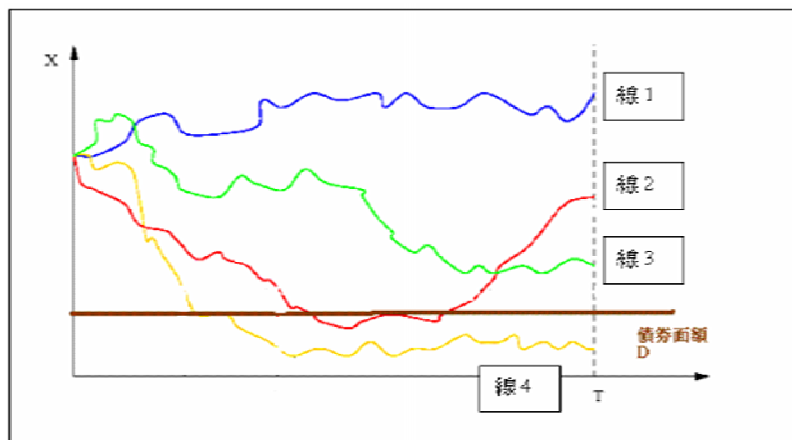
- 呼叫DownAndInCall () 計算買權價值
 - $S=100$
 - $X=100$
 - $T=1$
 - $r=0.1$
 - $\text{Sigma}=0.3$
 - $H=90$
 - $q=0$

課堂練習

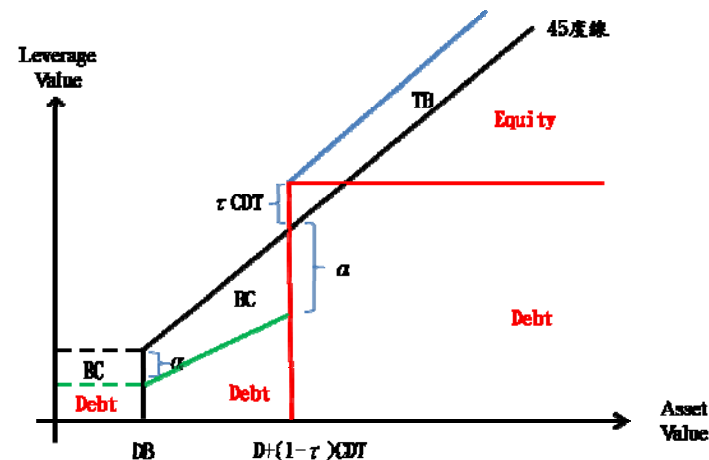
- 使用 in-out-parity 和 BS formula 求 下出局買權的價值
 - 下出局買權 + 下出局買權 = 標準買權
- 撰寫函式 DownAndOutCall()
 - `double DownAndOutCall(double S, double T, double X, double H, double r, double q, double Sigma)`
 - 呼叫DownAndInCall () 和 BS_Call() 求下出局買權和標準買權的價值

Modeling the Credit Risk with Barrier Options

Equity can be viewed as a down-and-out call option under the "first-passage" model.



Homework: Evaluate the Equity/ Debt Value under the Black&Cox Model



評價新奇選擇權的函數

- 二維常態累積機率函數
- 遠期生效選擇權
- 複合選擇權
- 回顧型選擇權
- 後定選擇權
- 幾何亞式選擇權

二維常態累積機率函數

- `double Two_Dimension_Normal(double a, double b, double rho)`
- 假定兩個二維常態隨機變數 x, y , 相關係數 $=rho$, 本函數計算 $x \leq a, y \leq b$ 的累積機率
- Ex: `Two_Dimension_Normal(0,0,0)=0.25`

//Check C++財務程式設計 5-2.3常態分配累積密度函式(p5-19)

遠期生效選擇權

- 假設未來約定的時間點為 t_1 ，到期日為 T

$$C = \max\{[S(T) - S(t_1)], 0\}$$

```
double ForwardStartCall  
(double S, double T_1, double T, double r, double q, double Sigma)
```

Example:
`ForwardStartCall(100, 0.25, 1, 0.1, 0, 0.3)`

複合選擇權

- 以選擇權為標的物的選擇權，投資人在期初支付此選擇權的權利金，取得在未來某一時點以某一個價格購買(或賣出)另一個選擇權的權利。
 - 買權的買權(call on call)
 - 賣權的買權(call on put)
 - 買權的賣權(put on call)
 - 賣權的賣權(put on put)

複合選擇權

- 在T1時，持有人可以用X1來(買/賣)一選擇權
 - 該選擇權在T2之時，可選擇用X2買/賣標的資產。

```
double CallOnCall  
(double S, double T_1, double T_2, double X_1, double X_2, double r,  
double q, double Sigma)  
double CallOnPut(...)  
double PutOnCall(...)  
double PutOnPut(...)
```

課堂練習

- 計算複合選擇權的價值
 - 標的物
 - S=100, Vol.=0.3, q=0
 - 買權:
 - 半年後可用10元的價格購買賣權
 - 賣權
 - 再隔半年後可用100元的價格出售標的物

回顧型選擇權

- 允許投資人在一段特定期間內賣在此間內股價的最高價(賣權)，或是此期間內的股價最低價買入(買權)

$$\max(S - S_{\min}, 0)$$

↑
從選擇權開始日到到期日會出現的最低價格

```
double LookbackCall  
(double S, double T, double Smin, double r, double q, double Sigma)
```

↑
從選擇權開始日到評價日會出現的最低價格

後定選擇權

- 允許購買者能夠改變其心意，讓持有人可以在到期日前某一特定時間點t，在決定該商品是買權還是賣權
- 後定選擇權通常用於未來某段時間內，可能發生某些重大事件而影響標的物的價格，譬如公司合併、總統大選、國際事件等等。

```
double ChooserOption  
(double S, double X, double T, double t, double r, double q, double Sigma)
```

幾何亞式選擇權

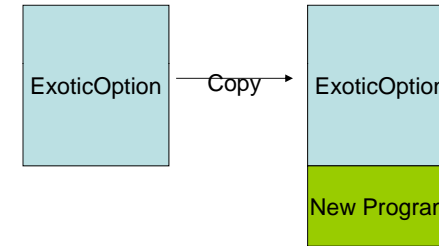
- 到期價值，取決於過去股價的平均

$$C = \max(\bar{S} - X, 0) \quad \bar{S} = \sqrt[n]{S_1 \cdot S_2 \cdots S_n}$$

```
double GeometricCall(double S, double T, double X, double r, double Sigma)
```

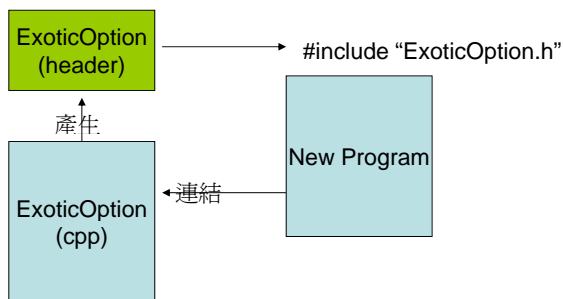
自建函式庫及函式庫的引用 Intuition

- ExoticOption project提供許多有用的函式
- 欲引用函式必須先定義



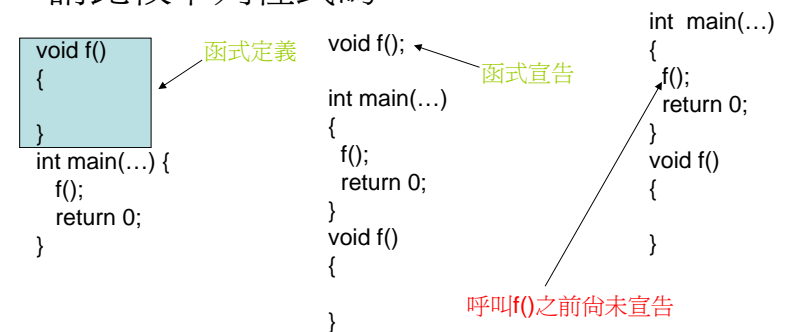
- 浪費空間而且降低程式的可讀性

自建函式庫及函式庫的引用 Intuition



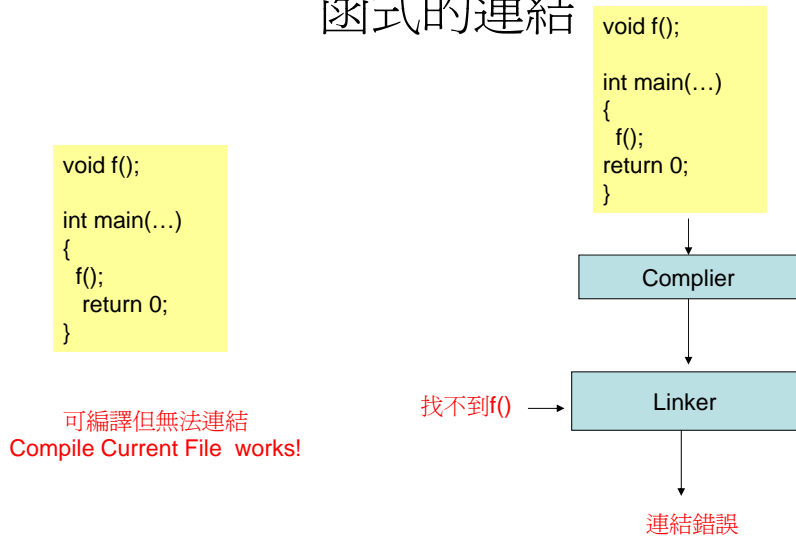
自建函式庫及函式庫的引用 函式的宣告和定義

- 請比較下列程式碼 見 FunDeclaration project

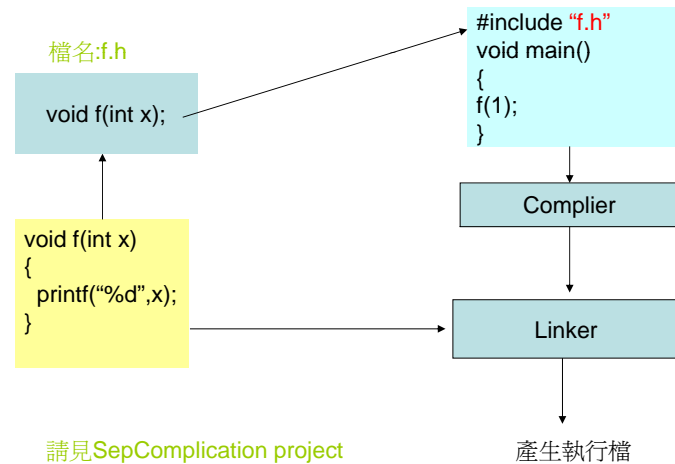


程式可正確編譯

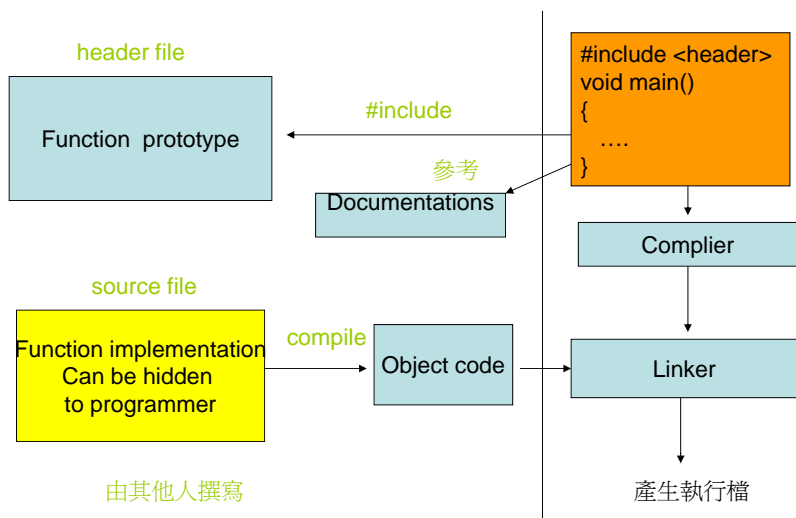
自建函式庫及函式庫的引用 函式的連結



自建函式庫及函式庫的引用 建立函式庫並引用



引用他人撰寫的函式庫



課堂演練

- 請開啓WriteHeader project
 - WriteHeader.cpp 檔包含常態累積密度函數
 - 撰寫header file
- 另開啓main.cpp
 - include 該 header file
 - 撰寫 Black Scholes買權評價公式
 - 請參見 BSCall project