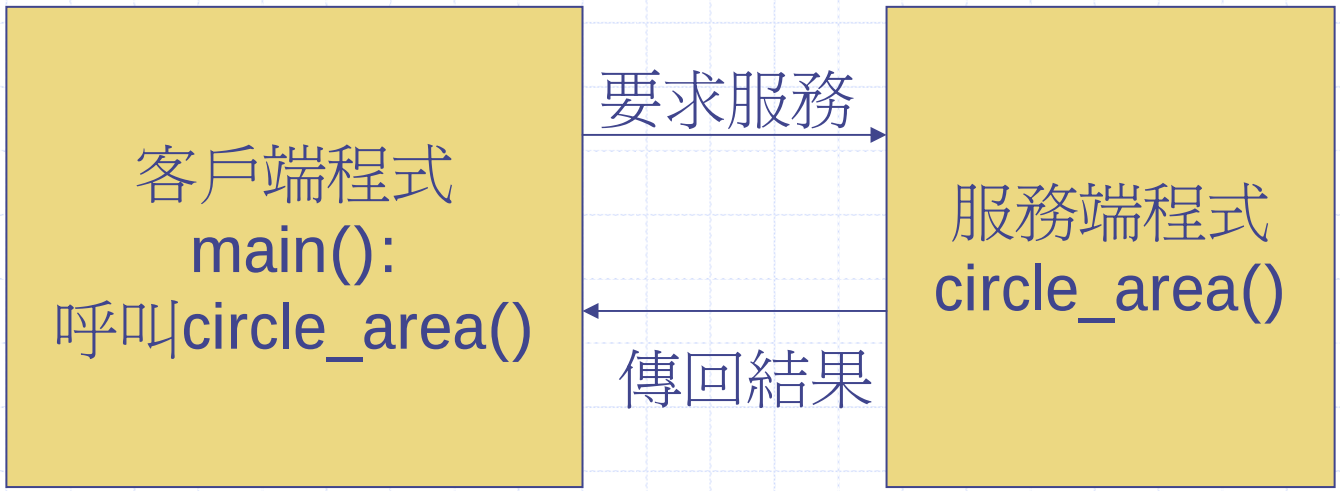
第十一章 (下篇) 例外處理

service-side: throw Exceptions

client-side: try-and-catch

客戶端與服務端程式



錯誤(例外,非正常)狀況發生時雙方的權責??

例外處理(Exception Handling)

```
double circle_area(int r) {
    if (r<0) {return –1 ;}
    return 3.14*r*r;
}
void main() {
    double r ;  cin >> r ;
    double a = circle_area(r) ;
    if (a == -1) { /*錯誤處理 */ }
    else cout << a << endl ; }
}
```

例外處理

```
double circle_area(double r, bool& suc) {
    if (r<0) {cout << “r<0”; suc = false ;}
    suc = true ; return 3.14*r*r;
}
void main() {
    double r ;  bool suc;       cin >> r ;
    double a = circle_area(r, suc) ;
    if (suc == false) { /*錯誤處理 */ }
    else cout << a << endl ; }
}
```

例外處理

```
double circle_area(double r, int& error_code) {
        if (r <0) error_code = 1 ;
        if (r==0) error_code = 2 ;
        error_code = 0 ;          return 3.14*r*r;
}
void main() {
    double r ;     int error_code;  cin >> r ;
    double a = circle_area(r, error_code) ;
    if (error_code == 0)  cout << a << endl ;
    else if (error_code == 1) { /*錯誤處理 */ }
    else if ......
}
```

例外處理

```
double circle_area(double r) {
    if (r<0) { cout << “circle(): r<0”;  exit(1) ; }
    return 3.14*r*r;
}
void main() {
    double r ;  cin >> r ;
    cout <<  circle_area(r) <<endl ;
}
```

缺點

- 錯誤發生時，服務端程式通知客戶端程式的方式千奇百怪，使用者窮於應付
- 服務端程式有時也參與錯誤處理，權責不清
- 縱使權責清楚，客戶端程式也將因處理錯誤而使可讀性大大降低

可讀性降低

```
// 完全無例外處理
void main() {
    double r, w, h, ta=0 ; cin>>r>>w>>h ;
    ta = circle_area(r) + rectangle_area(w,h);
    cout << “result=“ << ta <<endl;
}
```

→ 加上例外處理的結果

可讀性大幅降低

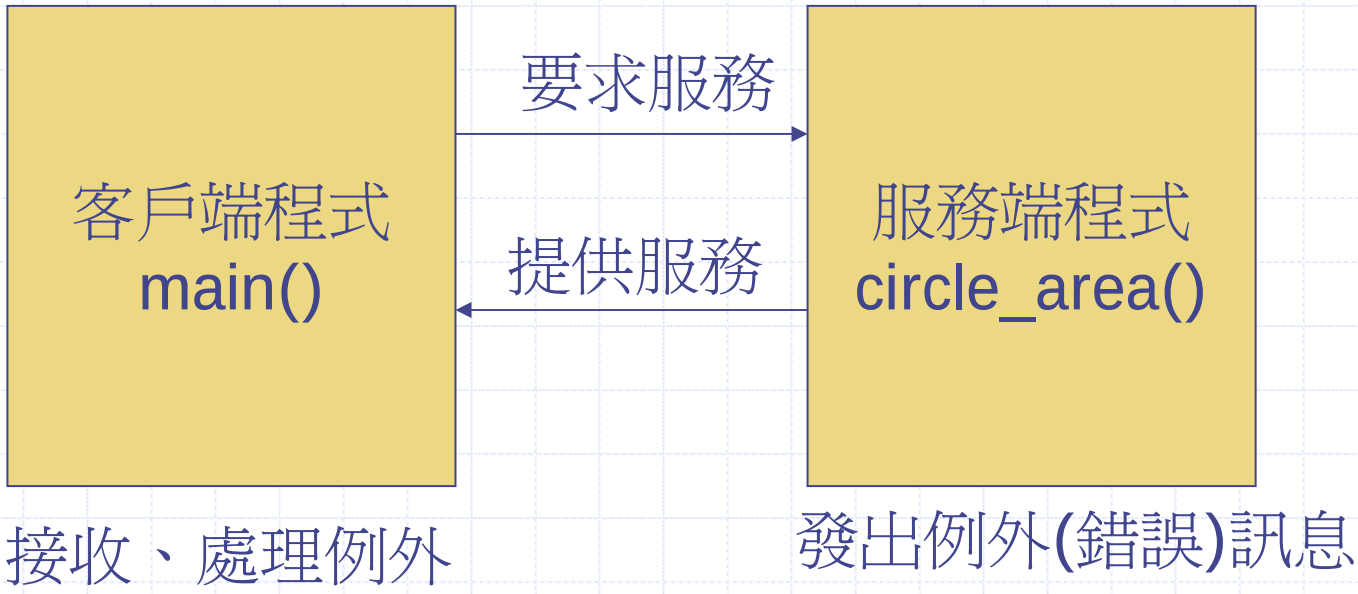
```
void main() {
    double r, w, h, ta=0 ; bool suc ; cin >> r>>w>>h;
    double a = circle_area(r, suc) ;
    if ( suc) { ta += a; }
    else {cout << “error”; }
    a = rectangle_area(w, h) ;
    if (a == -1) { ....}
    else if (a == -2) {...}
    else if {ta += a; }
    cout << “result=“ << ta <<endl ;
}
```

結論

- 平心而論，完全沒有錯誤處理的程式不論在發展時期或執行時期都將造成困擾。

- 錯誤處理既是必要的，是否有明確而一致的方式?
  - C++, Java
    - 客戶端: try-catch，服務端: throw

例外發生時的權責



C++例外處理: 使用前

```
double circle_area(double r, int& error_code) {
    if (r<0) error_code = -1 ;
    error_code = 0 ;
    return r*r*3.14 ;
}
void main() {
    double r ;     cin >> r ;  int error_code ;
    double a = circle_area(r, error_code) ;
    if (error_code == -1)
          cout << “error, code=“ << x <<endl ;
    else
          cout << a <<endl ;
    cout << “End of Program” <<endl ;
}
```

throw, try-and-catch

```
double circle_area(double r) throw (int) {
    if (r<0) { throw –1; }
    return r*r*3.14 ;
}
void main() {
    double r,a ;   cin >> r ;
    try {
        a = circle_area(r) ;
        cout << a <<endl ;
    } catch (int x) {
        cout << “error, code=“ << x <<endl ;
    }
    cout << “End of Program” <<endl ;
}
```

負責丟出例外訊息

負責接收及處理例外

正常狀況下的執行次序

```
1 double circle_area(double r) throw (int) {
2     if (r<0) { throw –1; }
3     return r*r*3.14 ;
4 }
5 void main() {
6     double r,a ;cin >> r ; // 5
7     try {
8         a = circle_area(r) ;
9         cout << a <<endl ;
10    } catch (int x) {
11        cout << “error, code=“ << x <<endl ;
12    }
13    cout << “End of Program” <<endl ;
14 }
```

5, 6, 7, 8, 1, 2, 3, 9, 13

例外發生時的執行次序

```
1 double circle_area(double r) throw (int) {
2     if (r<0) { throw –1; }
3     return r*r*3.14 ;
4 }
5 void main() {
6     double r, a ; cin >> r ; // -5, 故意讓circle_area產生例外
7     try {
8         a = circle_area(r) ; //發生例外狀況
9         cout << a <<endl ;
10    } catch (int x) {
11        cout << “error, code=“ << x <<endl ;
12    }
13    cout << “End of Program” <<endl ;
14 }
```

5, 6, 7, 8, 1, 2, 10, 11, 12 ,13

EX: 例外發生時以下二程式的輸出有何不同

```
void main() {
    double r,a ;  cin >> r ;
    try {
        a = circle_area(r) ;
        cout << a <<endl ;
    } catch (int x) {
        cout << x <<endl ;
    }
    cout << "End of Program"
    <<endl ;
}
```

```
void main() {
    double r,a ;  cin >> r ;
    try {
        a = circle_area(r) ;
    } catch (int x) {
        cout <<  x <<endl ;
    }
    cout << a <<endl ;
    cout << "End of Program"
    <<endl ;

}
```

討論

- try block中至少該放入甚麼?
  - 需要接收例外的程式碼

```
void main() {
        double r ; cin >> r ;
        double a1 = circle_area(r) ; //可能產生例外
        double w, h ; cin >> w >> h ;
        double a2 = rectangle_area(w,h);//可能產生例外
        cout << a1+a2<<endl ;
}
```

討論(續)

```
void main() {
  double r,a1,a2 ; cin >> r ;
  try {
     a1 = circle_area(r) ;
  }catch (int x) {
       // … 錯誤處理
  }
  double w, h ;
  cin >> w >> h ;
  try {
     a2 = rectangle_area(w,h);
  }catch(int y) {
       // … 錯誤處理
  }
  cout << a1+a2<<endl ;
}
```

```
void main() {
  try {
     double r,a1,a2 ; cin >> r ;
     a1 = circle_area(r) ;
     double w, h ;
     cin >> w >> h ;
     a2= rectangle_area(w,h);
      cout << a1+a2<<endl ;
  }catch(int x) {
            // … 錯誤處理
  }
}
```

EX: 例外型態判別-使用錯誤碼

```
double circle_area(double r)
    throw (int)
{
    if (r<0) { throw –1; }
    return r*r*3.14 ;
}
double rectangle_area(int w, int
    h) throw (int) {
    if (w<0 && h<0) throw –2 ;
    if (w<0 || h < 0) throw –3 ;
    }
    return w*h ;
}
```

```
void main() {
    double r, w, h, a = 0 ;
    try {
        cin >> r >> w >> h ;
        a = circle_area(r) +
            rectangle_area(w,h);
    } catch (int x) {
        switch(x) {
          case –1: ....
          case –2: ....
        }
    }
    cout << “area=“ << endl ;
}
```

EX:例外型態判別-使用資料型態

- 探討各種例外發生時程式執行的次序

```
double circle_area(double r)
    throw (int)
{
    if (r<0) { throw –1; }
    return r*r*3.14 ;
}
double rectangle_area(int w, int
    h) throw (char, double) {
    if (w<0 && h<0) throw ‘a’ ;
   if (w<0 || h < 0) throw 3.14 ;
    return w*h ;
}
```

```
void main() {
    double r, w, h, a = 0 ;
    try {
          cin >> r >> w >> h ;
          a = circle_area(r) +
               rectangle_area(w,h);
    } catch (int x) {
          // circle_area()中的錯誤
    } catch (char c) {
           // rectangle_area()中的錯誤
    } catch (double d) {
          // rectangle_area()中的錯誤
    }
    cout << “area=“ << endl ;
}
```

練習: 考慮overflow, underflow

```
int data[10], top = -1 ;
void push(int x) {
    data[++top]=x ;
}
int pop() {
    return data[top--] ;
}
```

```
void main() {
    int n, i ;
    cin >> n ;
    for (i = 0 ; i<n; i++)
        push(n) ;
    cin >> n ;
    for (i=0; i<n; i++)
        cout << pop() ;
}
```

練習: 改成class stack

```
class stack{
int data[10], top;
public:
stack(){top=-1;}
void push(int x) throw (int)
{
   if(x>9)
   {
      throw -1;
   }
      data[++top]=x ;
}
int pop() throw (int)
{
   if(top<0)
   {
      throw -2;
   }
      return data[top--] ;
}
void show()
{
   for(int i=0;i<top+1;i++)
         cout<<data[i]<<endl;
}
};
```

```
void main() {
    stack s;
      int n, i ;
      cin >> n ;
  try {

      for (i = 0; i<n; i++)
      {      s.push(i) ;

      }
      s.show();

  cin >> n ;
      for (i=0; i<n; i++)
      {
                cout << s.pop() <<endl;

      }
  }
  catch(int x)
  {
    if(x==-1)
    { cout<<"overflow"<<endl;}
    if(x==-2)
    {   cout<<"underflow"<<endl;}
  }
      //show();
```

EX: 故意不接例外

```
void main() {
    stack s;
    for (int i = 0 ; i <20 ; i++)
        s.push(i) ; //可能產生例外
}
```

EX: 沒接到

```
double circle_area(double r) throw (int)  {
    if (r<0) { throw –1; }
    return r*r*3.14 ;
}
void main() {
    try {
          ......
          circle_area(r) ;
    } catch (char c) {
          ......
    }
}
```

討論

- 每個函數都使用throw (int)
  - 延續錯誤碼概念: 大家共用容易弄錯
- throw (int, double, float...)
  - 時而有窮.....

→ 丟回自訂類別

丟出自訂類別: 服務端

```
class Exception {
public:
    string description ;
    Exception(string s) { description = s ; }
} ;
double circle_area(double r) throw (Exception){
    if (r < 0) throw Exception("circle_area(): r<0") ;
    return r*r*3.14 ;
}
```

```
Exception e("circle_area(): r<0");
throw e ;
```

接收自訂類別: 客戶端

```
void main() {
    try {
        ...
        circle_area(r) ;
        ...
    } catch (Exception e) {
        cout << e.description <<endl ;
        // 錯誤處理, maybe exit(1) ;
    }
    ...
}
```

每個函數均自訂例外型態

```
class CircleException: public Exception {
    public: CircleException(string s): Exception(s) { }
} ;
class RectException: public Exception {
    public: RectException(string s): Exception(s) { }
} ;
double circle_area(double r) throw (CircleException) {
    if (r < 0) throw CircleException("circle, r<0") ;
    return 3.14 * r * r ;
}
double rectangle_area(double w, double h) throw (RectException) {
    if (w<0||h<0) throw RectException("Rect, w<0 || h<0") ;
    return w*h ;
}
```

練習: 客戶端程式

```
void main() {
    double r, w, h ;
    cin >> r >> w >> h ;
    cout << circle_area(r) <<endl ;
    cout << rectangle_area(r) <<endl ;
}
```

客戶端程式:使用繼承概念

```
void main() {
    int r, w, h ;
    cin >> r >> w >> h ;
    try {
        cout << circle_area(r) <<endl ;
        cout << rectangle_area(w, h) <<endl ;
    } catch (Exception e) {
        cout << e.description <<endl ;
    }
}
```

catch(...)

```
void fun1() throw (int) {
    throw 1 ;
}
void fun2() throw (char) {
    throw ‘a’ ;
}
void fun3() throw (double) {
    throw 3.14 ;
}
```

```
void main() {
    try {
        fun1(); fun2(); fun3();
    } catch (int) {
        //例外處理
    } catch (char) {
        // 例外處理
    } catch(double) {
        // 例外處理
    }
}
```

catch(...)使用範例

```
void fun1() throw (int) {
    throw 1 ;
}
void fun2() throw (char) {
    throw ‘a’ ;
}
void fun3() throw (double) {
    throw 3.14 ;
}
```

```
void main() {
    try {
        fun1(); fun2(); fun3();
    } catch (...) {
        //例外處理
    }
}
```

catch(...)使用範例

```
void fun1() throw (int) {
    throw 1 ;
}
void fun2() throw (char) {
    throw ‘a’ ;
}
void fun3() throw (double) {
    throw 3.14 ;
}
```

```
void main() {
    try {
        fun1(); fun2(); fun3();
    } catch (int x) {
        //例外處理
    } catch (...) { //其他例外
        //例外處理
    }
}
```

巢狀例外處理

- 把不能處理的例外再丟出去