Classe 5 A Abacus – Funzioni e Subroutine in ASP

Riepilogo della lezione del 5-12-12

Function, Sub e VbScript

Come Visual Basic, da cui, almeno sintatticamente, deriva direttamente, VbScript ha la possibilità di scrivere codice in modo modulare in routine chiamate Function e Sub (corrispondenti in modo piuttosto vicino a Function e Procedure del Pascal). In realtà Visual Basic tratta questi elementi sintattici in modo del tutto tipizzato, nel senso, che sia i parametri in uscita, sia gli eventuali risultati hanno un tipo ben preciso. Differentemente VbScript, pur mantenendo una sintassi del tutto analoga, abolisce del tutto le dichiarazioni di tipo sia nei parametri che per i risultati, confermando di essere un linguaggio debolmente tipizzato.

Fare un parallelismo tra queste due tipi di routine e le funzioni del C è piuttosto facile. Una Function VbScript ha corrispondenza con una funzione che ritorni un valore (int, char o altro), mentre alla Sub corrispondono a funzioni C che non ritornano valore (tipo void).

La sintassi generale è sempre quella Visual Basic, ossia:

```
Function <nome>(<parametri>)
    ...
End Function
Sub <nome>(<parametri>)
...
End Sub
```

Nel primo caso, ossia quello della Function la routine fornisce un valore di ritorno. Tale valore viene comunicato verso l'esterno, assegnandolo ad una variabile, sempre presente implicitamente, che ha lo stesso nome della funzione stessa. In pratica, per esempio (con sintassi VbScript):

```
Function Somma(X,Y)
    Somma = X + Y
End Function
```

Si noti già in questo esempio che il tipo dei parametri non è specificato, proprio per il motivo detto in precedenza, ossia che VbScript è un linguaggio debolmente tipizzato. Lo stesso tipo di ritorno della funzione, presente in Visual Basic, qui non è indicato. Le segnature di tipo quindi sono del tutto assenti.

Per richiamare una Function , ovviamente la chiamata prevede di norma una assegnazione del valore di ritorno, che viene recuperato con una assegnazione o utilizzato direttamente in una espressione. La chiamata ad una Function è quindi del tipo `<variabile> = Somma(<parametri>)`.

```
<%
    S = FSomma(10, 24)
```

```
        Response.Write S
        Function FSomma(N1, N2)
           FSomma = N1 + N2
        End Function
%>
```

Viceversa una Sub è una routine che non è previsto intrinsecamente renda un valore, ma ciò può o non può accadere. Vi è anche da specificare che la Sub nel caso renda valori li rende tramite il meccanismo dei parametri passati per riferimento (indicati con ByRef in VbScript). L'indicazione ByRef deve in questo caso essere anteposta al parametro, e permette al parametro stesso di funzionare sia come valore di ingresso che come valore in uscita, ossia in modo bidirezionale.

Le chiamate alla Sub può essere fatta in 2 modi (ad esempio per la Sub StampaSomma):

```
...
Sub StampaSomma(N1,N2)
      Response.Write (N1 + N2)
End Sub
...
Call StampaSomma(3,5)
...
StampaSomma 3, 5
```

I parametri in questo caso, essendo senza indicazioni si presuppongono passati con il metodo ByValue. Nel caso uno dei parametri fosse da ritornare, come si è detto, esso deve essere definito con tipo di passaggio ByRef. Una apposita variabile si deve preoccupare di recuperare il valore ritornato per utilizzarlo. Non ha inoltre senso che ad un parametro con passaggio di tipo ByRef, corrisponda, come valore in ingresso una costante, seppur VbScript non evidenzia questa incongruenza con una indicazione di errore.

Vi è inoltre da notare che la posizione delle Sub o Function non è importante, mentre alcuni linguaggi pretendono tassativamente che ciò che viene richiamato sia già stato dichiarato e definito (tipicamente il Pascal), VbScript non pone vincoli sulla posizione delle routine nel listato.

Variabili locali e globali

Data la presenza delle Sub e Function si pone il problema quando e come le variabili presenti sulla pagina siano visibili all'interno delle stesse o le variabili interne alle Sub e Function siano visibili all'esterno. Anche in VbScript abbiamo il concetto di variabile locale e globale. Le variabili dichiarate a livello di listato principale sono usualmente visibili anche nelle Sub E Function e quindi sono automaticamente globali. Ad esempio il listato:

```
V = 10
S = FSomma(26)
Response.Write S
```

```
Function FSomma(N2)
   FSomma = V + N2
End Function
```

stampa il valore 36 (= 10 + 26), il che evidenzia che V si "vede" all'interno della funzione Fsomma, ossia il valore che ha nel corpo della pagina permane anche all'interno di Fsomma. Quindi si può concludere che V è globale alle funzioni e sub prensenti nella pagina.

Tuttavia variabili che operano solo all'interno di funzioni e procedure, di default, non sono visibili all'esterno:

```
'N = 0
Somma 10, 26
Response.Write N
Sub Somma(A, B)
    N = A + B
End Sub
```

Nel listato qui sopra N non risulta visibile nel programma principale, pur essendo inizializzato nella Sub Somma. Quindi si conferma che variabili assegnate solo nelle Sub e Function sono da considerarsi locali (non esistono fuori dalle relative routine). Differente situazione se la variabile con lo stesso nome è utilizzata anche nel listato principale (ad esempio nel listato sopra se N = 0 viene de-commentata). Allora essa diviene globale e quindi le modifiche all''interno delle Sub e Function si rispecchiano all'esterno. Un possibile modo, anche in questa situazione, per forzare l'esistenza di una variabile interna e di una esterna separate, seppur con lo stesso nome, è dichiarare la variabile interna N localmente, così:

```
N = 0
Somma 10, 26
Response.Write N
Sub Somma(A, B)
Dim N
    N = A + B
End Sub
```

La visibilità delle Function e Sub si limita al listato, ossia al codice della pagina in cui si stà operando. Non è ovviamente possibile in ASP richiamare Sub e Function presenti su altre pagine ASP. In alcuni casi, tramite inclusione ripetuta di file contenenti le funzioni sulle diverse pagine che ne hanno necessità, si ottiene l'effetto di avere a disposizione quanto necessario, senza dover continuamente riscrivere (o fare il cut & paste) del codice relativo alle routine necessarie.