Fondamenti di Informatica, A.A. 2011-2012

07/09/2012

Esercizio 1

È dato il frammento di codice Matlab

```
n = 3;
v = zeros(n,1);
w = zeros(1,n);
for i=1:n
  v(i) = i;
  w(i) = i^2;
end
a = v*w;
disp(a);
```

Si chiede cosa viene visualizzato dall'interprete Matlab.

Soluzione

Il vettore `v` è un vettore colonna il quale assume come valori gli interi da 1 a 3; il vettore `w` è un vettore riga che assume come valori i quadrati degli interi da 1 a 3. Il prodotto di un vettore colonna per un vettore riga, di eguali dimensioni, darà come risultato una matrice quadrata, che viene quindi visualizzata come segue:

```
1     4     9
2     8    18
3    12    27
```

Esercizio 2

Descrivere le regole di visibilità delle variabili in Matlab, con particolare riguardo all'utilizzo delle funzioni.

Soluzione

Le variabili di Matlab sono normalmente visibili nel workspace in cui sono state create. Ciascuna invocazione di funzione (comprese quelle ricorsive) genera un nuovo workspace che inizialmente contiene solo le variabili presenti nella lista degli argomenti ed a cui è stato associato un valore dalla funzione chiamante; pertanto ogni accesso ad una variabile si riferisce puramente a quella locale, senza altra relazione con le variabili del workspace da cui la funzione è stata invocata.

La parola chiave `global` può essere usata per specificare esplicitamente una associazione diretta tra una variabile di una funzione ed una del workspace di partenza; questa associazione deve essere resa esplicita sia nello spazio di partenza che nella funzione.

Esercizio 3

È dato il frammento di codice Matlab

```
a=[1,2,3,4];
b=[-3,4,5,1];
c=[1,-1,0,-1];

d=a.*((a<b)&(b~=c));
disp(d);
```

Si chiede cosa visualizza l'interprete Matlab.

Soluzione

L'espressione `((a<b)&(b~=c))` è una espressione con operatori relazionali e logici applicati ad array; pertanto il suo valore sarà un array di valori logici rappresentati con valori 0 e 1; con i valori indicati per `a`, `b` e `c` si ottiene

```
(a<b) =     0     1     1     0
(b~=c) =    1     1     1     1
((a<b)&(b~=c)) =  0     1     1     0
```

Infine, questo vettore viene moltiplicato elemento per elemento con il vettore `a` risultando quindi

```
 disp(d)
    0     2     3     0
```

Esercizio 4

È data la seguente funzione Matlab

```
function [x]=mistery(x)
  n=length(x);
  for i=1:n
    for j=1:g(i)
       x(j) = x(j) + 4;
    end
  end
end
```

Stimare il numero di operazioni aritmetiche eseguite dal codice per le seguenti tre scelte di $g(i)$

1. $g(i) = 3$;
2. $g(i) = i$;
3. $g(i) = i/4$.

Può essere utile la seguente eguaglianza

$$\sum_{k=0}^{m} k = \frac{m(m+1)}{2}.$$

Soluzione

Ciascuna iterazione del ciclo più interno esegue una sola istruzione aritmetica; pertanto il numero totale di istruzioni sarà pari al numero di iterazioni interne. Distinguendo i tre casi:

1. Se $g(i) = 3$, il numero di iterazioni del ciclo interno è indipendente da i, per cui vale

$$OP = \sum_{i=1}^{n} \sum_{j=1}^{3} 1 = \sum_{i=1}^{n} 3 = 3n = O(n)$$

2. Se $g(i) = i$, il numero di iterazioni del ciclo interno è pari a i, per cui vale

$$OP = \sum_{i=1}^{n} \sum_{j=1}^{i} 1 = \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \approx \frac{n^2}{2} = O(n^2)$$

3. Se $g(i) = i/4$, il numero di iterazioni del ciclo interno è pari a $i/4$, per cui vale

$$OP \approx \sum_{i=1}^{n} \sum_{j=1}^{i/4} 1 = \sum_{i=1}^{n} \frac{i}{4} = \frac{1}{4} \frac{n(n+1)}{2} \approx \frac{n^2}{8} = O(n^2);$$

   si noti che in questo caso, avendo richiesto solo una stima, stiamo ignorando il fatto che in effetti il numero di iterazioni eseguito in un ciclo `for j=1:i/4` è pari a `floor(i/4)`, e quindi ad esempio per $i = 1, 2, 3$ il ciclo interno non esegue alcuna operazione

Esercizio 5

È dato il frammento di codice Matlab

```
s = 0;
idx = [];
for i=1:length(x)
  if (x(i) > 0)
     s = s+x(i);
     idx = [ idx i ];
     if (s>thresh)
       break;
     end
  end
end
```

Riscrivere il codice facendo uso del ciclo `while` e senza usare la istruzione `break`.

Soluzione

L'unica difficoltà dell'esercizio risiede nello stabilire la corretta condizione di uscita, che è determinata non solo dal valore di `length(x)` ma anche dalla accumulazione dei valori in `s`. La soluzione più semplice è la seguente

```
s = 0;
idx = [];
i = 1;
while ((i<=length(x)) && (s<=thresh))
  if (x(i) > 0)
     s = s+x(i);
     idx = [ idx i ];
  end
  i = i + 1;
end
```

L'unico punto delicato è che questa soluzione assume implicitamente che `thresh>0`, cosa che andrebbe controllata; ignoriamo per semplicità la soluzione che gestisce anche `thresh<0`.