Calcolo numerico e **programmazione**
Introduzione a Scilab 2

Tullio Facchinetti
<tullio.facchinetti@unipv.it>

14 maggio 2012

10:12

http://robot.unipv.it/toolleeo

Cos'è una stringa

- una stringa consiste in una sequenza ordinata di caratteri
- una costante stringa on Scilab è delimitata dai doppi apici
- le variabili striga sono create come tutte le altre variabili
- esistono varie funzioni per manipolare le stringhe

```
-->nome = "Albert"
 nome  =
 Albert
-->cognome = "Einstein"
 cognome  =
 Einstein
```

Concatenazione di stringhe

- l'operatore "+" viene usato per concatenare due o più stringhe

```
-->nome + " " + cognome
 ans  =
 Albert Einstein
```

Lunghezza di una stringa

`length(st)` ritorna la lunghezza (numero di caratteri) della stringa `st`

```
-->length("abcde")
 ans  =
    5.
-->length("abcde" + "abcde")
 ans  =
    10.
```

Identificazione di caratteri

`strchr(st, ch)` trova la prima occorrenza del carattere `ch` all'interno della stringa `st`

```
-->strchr("Albert Einstein", "a")
 ans  =

-->strchr("Albert Einstein", "E")
 ans  =
 Einstein
-->strchr("Albert Einstein", "n")
 ans  =
 nstein
```

Comparazione di stringhe

si può fare semplicemente usando l'operatore "=="

```
-->"serio" == "faceto"
 ans  =
  F
-->"serio" == "SERIO"
 ans  =
  F
-->"serio" == "serio"
 ans  =
  T
```

Comparazione di stringhe

la funzione `strcmp(st1, st2)` permette di determinare se una stringa è maggiore/uguale/minore dell'altra (usa il metodo di comparazione "del dizionario")

```
-->strcmp("alfa", "beta")
 ans  =
  - 1.
-->strcmp("beta", "alfa")
 ans  =
    1.
-->strcmp("beta", "BETA")
 ans  =
    1.
-->strcmp("beta", "beta")
 ans  =
    0.
```

Costrutti disponibili in Scilab

- in Scilab tutti i valori numerici sono normalmente memorizzati in forma di numero in virgola mobile a doppia precisione (64 bit)
- è possibile forzare la rappresentazione di un numero a valore intero

Creazione di variabili intere

è necessario usare delle funzioni dedicate per creare una variabile intera

```
y = int8(x)   -> 8-bit signed
y = uint8(x)  -> 8-bit unsigned
y = int16(x)  -> 16-bit signed
y = uint16(x) -> 16-bit unsigned
y = int32(x)  -> 32-bit signed
y = uint32(x) -> 32-bit unsigned
```

Creazione di variabili intere

```
-->i = uint32(0)
 i  =
  0
-->j = i - 1
 j  =
  4294967295
-->k = j + 1
 k  =
  0
```

si noti che il valore stampato **non presenta** il punto che indica che si tratta di un numero in virgola mobile

Overflow e circolarità degli interi

```
-->uint8(0 + (-4:4))
 ans  =
  252  253  254  255  0  1  2  3  4
-->uint8(2^8 + (-4:4))
 ans  =
  252  253  254  255  0  1  2  3  4
--> int8(2^7 + (-4:4))
 ans  =
  124  125  126  127 -128 -127 -126 -125 -124
```

quando si verifica un overflow, il numero assume valore partendo dall'estremo opposto dell'intervallo di valori

Overflow, circolarità e problemi di portabilità

- altri programm simili a Scilab, come Matlab o Octave, non si comportano allo stesso modo in caso di overflow
- sia Matlab che Octave saturano il valore di in overflow

```
octave:1> uint8(0 + (-4:4))
ans =
  0   0   0   0   0   1   2   3   4
octave:2> uint8(2^8 + (-4:4))
ans =
  252   253   254   255   255   255   255   255   255
octave:3> int8(2^7 + (-4:4))
ans =
  124   125   126   127   127   127   127   127   127
```

Costrutti disponibili in Scilab

- costrutto `if`
  permette di valurare una condizione
- variante `elseif`
  permette di "concatenare" più istruzioni `if` alternative
- costrutto `select`
  permette di selezionare tra più opzioni

L'istruzione `if`

la condizione è sempre verificata:

```
if (%t) then
  disp("Condizione vera!")
end
```

la condizione verifica che `a` sia o meno inclusa nell'intervallo $[0, 100]$ (attenzione: la variabile `deve esistere`):

```
if (a > 0 & a < 100)
  disp("Condizione vera!")
else
  disp("Condizione falsa!")
end
```

La variante elseif

la variante `elseif` permette di scegliere tra più condizioni:

```
if (ora >= 6 & ora < 12) then
  disp ("Mattina!")
elseif (ora >= 12 & ora < 18) then
  disp ("Pomeriggio!")
elseif (ora >= 18 & ora < 22) then
  disp ("Sera!")
else
  disp ("Notte!")
end
```

Determinare se una variabile esiste

se una variabile non è stata definita si ottiene un errore in esecuzione del seguente tipo:

```
-->exec('/home/toolleeo/if-elseif.sce', -1)
if (ora >= 6 & ora < 12) then
      !--error 4
Undefined variable: ora

at line       1 of exec file called by :
exec('/home/toolleeo/if-elseif.sce', -1)
```

Determinare se una variabile esiste

con `isdef` si può controllare l'esistenza di una variabile:

```
if (~isdef('ora'))
  disp("Calcolo non possibile!")
elseif (ora >= 6 & ora < 12) then
...
```

Il costrutto select

```
select ora
case 8
  disp ("Colazione!")
case 12
  disp ("Pranzo!")
case 17
  disp ("The!")
case 19
  disp ("Cena!")
else
  disp ("Non e‘ ora di mangiare!")
end
```

L'istruzione for

calcolo del numero di Fibonacci:

```
n = 10;
fibo = zeros(1,n);
fibo(1) = 1;
fibo(2) = 1;
for i = 3:n
  fibo(i) = fibo(i-1) + fibo(i-2);
end
disp(fibo)
```

L'istruzione `while`

```
i = 11;   // per esempio...
while (i ~= 1)
    if (modulo(i, 2) ~= 0) then
        i = 3 * i + 1;
    else
        i = int(i / 2);
    end
    disp(i)
end
```

la funzione `modulo(x, y)`
calcola il resto della divisione intera x/y

Le istruzioni break e continue

```
while (%t)
    a = input('inserisci il tuo valore [0, 10]: ');
    if (a < 0 | a > 10) then
        continue;
    end
    b = int(10 * rand(1, 1))
    if (modulo(int(a + b), 2) ~= 0) then
        disp('somma dispari: hai vinto!');
        continue;
    end
    disp('somma pari: hai perso!');
    break;
end
```

Cicli annidati

sono costituiti da un ciclo all'interno di un altro

```
row = 10;
col = 10;
A = zeros(row, col);
for i = 1:row
    for j = 1:col
        A(i, j) = i * j;
    end
end
```