Esercitazione 1

Istruzioni trattate: `number_properties`, `frexp`, `nearfloat`, `format`, `disp`, ciclo `for`, `string`, `int`, `length`.

Lo scopo delle esercitazioni è di *introdurre all'uso dell'ambiente di programmazione per calcolo numerico Scilab* e di *verificare l'efficacia del modello proposto per prevedere e spiegare il comportamento di Scilab* in alcuni casi elementari.

A chi legge si raccomanda di riprodurre al calcolatore i "dialoghi" con *Scilab* proposti e di prendere spunto da essi per crearne di nuovi. Gli esercizi contrassegnati dal simbolo ★ sono leggermente più astratti rispetto agli altri.

Nella prima parte di questa esercitazione presenteremo alcune *funzioni predefinite*: di ciascuna daremo una breve descrizione e semplici esempi. Tutte le *funzioni predefinite* trattate hanno una *pagina di help* in *Scilab*. A tali pagine si deve far riferimento per maggiori dettagli. Nella seconda parte utilizzeremo le *funzioni predefinite* introdotte per insegnare a *Scilab* come costruire la stringa che rappresenta la scrittura posizionale in base due della frazione di un elemento di $F(2, 53)$.

Prima parte

- `number_properties`

  Questa *funzione predefinita* consente di conoscere alcune proprietà dell'insieme dei numeri in virgola mobile e precisione finita utilizzato da *Scilab*. In particolare:

  `number_properties('radix')`

  restituisce *la base* β,

  `number_properties('digits')`

  restituisce *la precisione* m,

  `number_properties('maxexp')`

  restituisce *il valore massimo dell'esponente* $b_{\max}$,

  `number_properties('minexp')`

  restituisce *il valore minimo dell'esponente* $b_{\min}$,

  `number_properties('denorm')`

  dichiara *se sono presenti elementi denormalizzati*,

  `number_properties('eps')`

  restituisce *la precisione di macchina*. Si ottiene:

```
-->base = number_properties('radix')
 base  =

    2.

-->precisione = number_properties('digits')
 precisione  =

    53.

-->bmax = number_properties('maxexp')
 bmax  =
```

```
    1024.

-->bmin = number_properties('minexp')
 bmin  =

  - 1021.

-->denormalizzati = number_properties('denorm')
 denormalizzati  =

  T

 -->u = number_properties('eps')
 u  =

    1.110D-16
```

La precisione di macchina in $F(2, 53)$ vale:

$$\mathtt{u} = 2^{-53} = 10^{-15}\ 0.111022302462515654042363166809082031 25$$

Come vedremo, *il valore mostrato da Scilab è un'approssimazione di* `u`.

- `frexp`

  Questa *funzione predefinita* restituisce *la frazione* (con segno) e *l'esponente* di un assegnato numero di macchina ξ. Precisamente:

  $$\texttt{[f,e] = frexp}(\xi)$$

  assegna ad `f` ed `e` valori tali che: per $\xi \neq 0$, $\frac{1}{2} \leqslant |\mathtt{f}| < 1$ e $\xi = 2^{\mathtt{e}} \cdot \mathtt{f}$; per $\xi = 0$, $\mathtt{f} = \mathtt{e} = 0$. Ad esempio:

```
-->[f,e] = frexp(1)
 e  =

    1.
 f  =

    0.5

-->[f,e] = frexp(-3)
 e  =

    2.
 f  =

  - 0.75

-->[f,e] = frexp(0)
 e  =

    0.
 f  =

    0.

-->[f,e] = frexp(u)
 e  =
```

```
    -52.
 f  =

    0.5
```

- `nearfloat`

  Questa *funzione predefinita* restituisce *il predecessore* o *il successore* di un assegnato numero di macchina ξ. Precisamente:

  `nearfloat('pred',`ξ`)`

  restituisce il predecessore di ξ,

  `nearfloat('succ',`ξ`)`

  restituisce il successore di ξ. Ad esempio:

```
-->s1 = nearfloat('succ',1)
 s1  =

      1.

-->s1 == 1
 ans  =

  F

-->s1 > 1
 ans  =

  T

-->1 == nearfloat('pred',s1)
 ans  =

  T
```

L'incongruenza nelle prime tre risposte di *Scilab* dell'ultimo esempio è solo apparente. Quando a *Scilab* viene chiesto di visualizzare il valore di una variabile, esso mostra una stringa che rappresenta (in base dieci per comodità dell'utilizzatore) *l'arrotondato* di tale valore in un opportuno sottoinsieme di numeri reali. *In quale sottoinsieme avviene l'arrotondamento* e *il formato da utilizzare per scrivere il risultato dell'arrotondamento* è stabilito dal valore del parametro *formato corrente* costituito da un vettore [*tipo*, *lunghezza*] in cui *tipo* è una stringa che può assumere valori `'v'` ('formato di tipo variabile') oppure `'e'` ('formato di tipo esponenziale') e *lunghezza* è un numero intero in $[2, 25]$. Precisamente: per ogni intero positivo n si indichi con V_n l'insieme dei numeri reali x che ammettono almeno una tra le seguenti scritture in base dieci:[1]

(a) $x = (-1)^s\, c_1 \cdots c_k.c_{k+1} \cdots c_{n-2}$ con $s \in \{\,0, 1\,\}$ e $1 \leqslant k \leqslant n-2$;

(b) $x = (-1)^s\, c_1.c_2 \cdots c_{n-6}\, 10^b$ con $s \in \{\,0, 1\,\}$, $c_1 \neq 0$ e b numero intero in $[-99, 99]$;

(c) $x = (-1)^s\, c_1.c_2 \cdots c_{n-7}\, 10^b$ con $s \in \{\,0, 1\,\}$, $c_1 \neq 0$ e b numero intero in $[-324, 308]$;[2]

e con E_n l'insieme dei numeri reali x che ammettono almeno una tra le precedenti scritture (b) e (c). Allora:

- se *formato corrente* $= [\texttt{'v'}, n]$ l'arrotondamento avviene in V_n e il risultato dell'arrotondamento viene espresso, in ordine di preferenza, in una forma corrispondente ad (a) oppure a (b) oppure a (c);

[1] L'intero n indica il *numero massimo di simboli* che *Scilab* può usare per scrivere il risultato dell'arrotondamento.

[2] La limitazione dell'esponente nel caso (b) deriva dal fatto che *Scilab* usa *due cifre* per scrivere $|b|$; la limitazione nel caso (c) è conseguenza della limitatezza dell'esponente dell'insieme dei numeri in virgola mobile utilizzati da *Scilab*.

- se *formato corrente* = [`'e'`, n] l'arrotondamento avviene in E_n e il risultato dell'arrotondamento viene espresso, in ordine di preferenza, in una forma corrispondente a (b) oppure a (c).

Il valore predefinito del parametro *formato corrente* è [`'v'`, 10]. Si ha, ad esempio:[3]

```
-->0.12345678
 ans  =

    0.1234568

-->12345678
 ans  =

    12345678.

-->-123456789
 ans  =

  - 1.235D+08
```

L'utilizzatore può *conoscere* e *modificare* il valore del parametro *formato corrente* utilizzando la *funzione predefinita* `format`.

- `format`

  Questa *funzione predefinita* consente di conoscere e modificare il valore del parametro *formato corrente*. Precisamente:

  `format()`

  restituisce il valore del parametro *formato corrente* con il *tipo* così codificato: 0 significa `'e'` e 1 significa `'v'`;

  `format(`s, n`)`

  assegna al parametro *formato corrente* il valore `[`s, n`]`.

Si osservi che se $n < N$ allora $V_n \subset V_N$ e $E_n \subset E_N$: aumentando il valore del parametro *lunghezza* nel *formato corrente Scilab* mostra un'approssimazione *più accurata*[4] del valore della variabile in esame. Ad esempio:

```
-->formato_corrente = format()
 formato_corrente  =

    1.    10.

-->x = 123456789
 x  =

    1.235D+08

-->format('v',15)

-->x
 x  =

    123456789.

-->123456789000000
 ans  =
```

[3] Esercizio: constatare che in ciascuno degli esempi *Scilab* mostra il risultato dell'arrotondamento utilizzando al più tanti simboli – incluso un segno `+` *implicito* ed un *punto decimale* `.` – quanto specificato dal valore della *lunghezza*.

[4] Più correttamente: *non meno accurata*. Vedere l'Esercizio 4.

```
    1.23456789D+14

-->format('e',10)

-->1
 ans  =

    1.000D+00

-->0
 ans  =

    0.000D+00
```

Per ottenere l'approssimazione più accurata del valore di `s1` che *Scilab* possa mostrare imponiamo al parametro *formato corrente* un valore con *lunghezza* pari a 25:

```
-->format('v',25)

-->s1
 s1  =

    1.000000000000000222046
```

Questo rende manifesta la coerenza delle risposte dell'esempio relativo alla *funzione predefinita* `nearfloat`: l'arrotondato di `s1` in V_{10} è 1, ma `s1` > 1. Si ricordi che il valore del parametro *formato corrente* non modifica il *valore* di una variabile, ma solo la stringa che *Scilab* produce quando deve *visualizzare* tale valore.

- `disp`
  Questa *funzione predefinita*, visualizza, con le modalità stabilite dal valore del parametro *formato corrente*, il valore degli argomenti. Si osservi che l'*ordine di visualizzazione è l'inverso di quello degli argomenti*. Ad esempio:

  ```
  -->disp(%pi,sqrt(2))

       1.4142136

       3.1415927

  -->x = 2^30;

  -->disp(x,'x vale circa')

   x vale circa

      1.074D+09
  ```

  È utile osservare che la *funzione predefinita* `disp` visualizza *ma non restituisce* il valore degli argomenti (vedere l'Esercizio 3).

- ciclo `for`
  Questo costrutto predefinito consente di definire *iterazioni*. La sequenza:

  `for` ⟨contatore⟩ `=` ⟨valore iniziale⟩`:`⟨valore finale⟩`,` ⟨istruzioni⟩ `end;`

  equivale a:

  ⟨contatore⟩ `=` ⟨valore iniziale⟩`;` ⟨istruzioni⟩
  ⟨contatore⟩ `=` ⟨valore iniziale⟩ $+ 1$`;` ⟨istruzioni⟩
  ⋮
  ⟨contatore⟩ `=` ⟨valore finale⟩`;` ⟨istruzioni⟩

Ad esempio:

```
-->for i=1:4, x = i; disp(x); end;

    1.

    2.

    3.

    4.
```

- `string`

  Questa *funzione predefinita* restituisce la *stringa* che rappresenta, con le modalità stabilite dal valore del parametro *formato corrente*, il valore dell'argomento. Ad esempio (`%e` è, in *Scilab*, l'arrotondato in $F(2, 53)$ del numero di Nepero e):

```
-->nep = string(%e)
 nep  =

 2.7182818

-->frase = 'e vale circa ' + nep + ' :-)'
 frase  =

 e vale circa 2.7182818 :-)
```

  La finestra che mostra le variabili evidenzia che `nep` e `frase` sono variabili di tipo *stringa*. Si osservi che la *funzione predefinita* `+` opera su stringhe: il risultato è la stringa ottenuta *concatenando* gli argomenti.

- `int`

  Questa *funzione predefinita* restituisce *l'arrotondato verso zero in* $\mathbb{Z}$ del valore dell'argomento.[5] Ad esempio:

```
-->int(sqrt(2))
 ans  =

    1.

-->int(-%pi)
 ans  =

  - 3.

-->int(-1)
 ans  =

  - 1.
```

- `length`

  Questa *funzione predefinita* restituisce, quando applicata ad una stringa, la sua *lunghezza*, ovvero il *numero di caratteri* che la costituisce. Ad esempio:

```
-->length('123456789')
 ans  =
```

[5] Se $\xi \in F_d(2, 53, -1021, 1024)$ è intero, `int(`ξ`)` $= \xi$ altrimenti `int(`ξ`)` è l'intero adiacente a ξ più vicino a zero.

```
    9.

-->length('1 + 1 = 2')
 ans  =

    9.

-->length('')
 ans  =

    0.
```

Seconda parte

La sequenza di istruzioni che segue determina la stringa s che rappresenta la scrittura posizionale in base due della frazione f di un elemento di $F(2, 53)$:[6]

```
s = '0.';
for i = 1:53, s = s + string(int(2*f)); f = 2*f - int(2*f); end;
```

Si ha, ad esempio:

```
-->s1 = nearfloat('succ',1);

-->[f,e] = frexp(s1);

-->s = '0.';

-->for i = 1:53, s = s + string(int(2*f)); f = 2*f - int(2*f); end;

-->s
 s  =

 0.10000000000000000000000000000000000000000000000000001

-->length(s)
 ans  =

   55.
```

Esercizi

---

1. Dedurre dalla pagina di *help* della funzione predefinita `number_properties` il significato degli assegnamenti seguenti:

   ```
   xi_norm_min = number_properties('tiny')
   ```

   e

   ```
   xi_min = number_properties('tiniest')
   ```

   Discutere poi i valori ottenuti da:

   ```
   [e,f] = frexp(xi_norm_min)
   ```

   e

   ```
   [e,f] = frexp(xi_min)
   ```

[6] Vedere l'Esercizio 5.

2. Constatare che in ciascuno degli esempi il risultato mostrato da *Scilab* è l'arrotondato del valore richiesto nell'insieme specificato dal valore del parametro *formato corrente*.

3. Spiegare il seguente comportamento di *Scilab*:

```
-->x = 123
 x  =

    123.

-->sx = disp(x)

    123.

-->sx
  !--error 4
C'è una variabile non definita: sx
```

4. ★ Se *formato corrente* = ['v', 10], in *Scilab* si ha:

```
-->x = 2^30
 x  =

    1.074D+09
```

Dimostrare che *tutti gli interi in* $[-2^{53}, 2^{53}]$ *sono in* $F_d(2, 53, -1021, 1024)$ e dedurne che anche 2^{30} e $1.074\ 10^9$ lo sono. Discutere poi il seguente "dialogo" con *Scilab*:

```
-->x == 1.074d9
 ans  =

  F

-->ERR_10 = abs(x - 1.074d9)/x
 ERR_10  =

    0.0002404

-->format('v',11)

-->x
 x  =

    1.0737D+09

-->x == 1.0737d9
 ans  =

  F

-->ERR_11 = abs(x - 1.0737d9)/x
 ERR_11  =

    0.00003895
```

5. ★ Mostrare che se `x` è un elemento di $F(2, 53)$, rd è la funzione arrotondamento in $F(2, 53)$ con RTTE e le pseudo funzioni aritmetiche sono definite nel modo usuale si ha:

$$2 \otimes \mathtt{x} = 2\mathtt{x} \qquad , \qquad \mathtt{int(x)} = \mathrm{rd}(\lfloor \mathtt{x} \rfloor) = \lfloor \mathtt{x} \rfloor$$

e quindi:

$$(\,2 \otimes \texttt{x}\,) \ominus \texttt{int(}2 \otimes \texttt{x)} = \text{la parte frazionaria di } 2\texttt{x}$$

(Suggerimento: dimostrare che $2\texttt{x}, \lfloor\,\texttt{x}\,\rfloor$ e $2\texttt{x} - \lfloor\,2\texttt{x}\,\rfloor$ sono in $F(2, 53)$.)

6. Verificare, costruendo la stringa che rappresenta la frazione di `x`, che l'istruzione:

   ```
   -->x = 0.1;
   ```

   assegna ad `x` il valore: arrotondato – con RTTE – di un decimo in $F_d(2, 53, -1021, 1024)$.