

# Esercitazione 1

*Istruzioni trattate:* `number_properties`, `log2`, `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.

I dialoghi riportati nelle esercitazioni sono stati eseguiti utilizzando la versione 6.1.0 di *Scilab* su un calcolatore supportato da un sistema operativo *Linux*.

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  $\beta$ ,

```
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:

$$u = 2^{-53} = 10^{-15} \cdot 0.11102230246251565404236316680908203125$$

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

- **log2**

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

$$[f, e] = \text{log2}(\xi)$$

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

```

-->[f,e] = log2(1)
f =

    0.5
e =

    1.

```

```

-->[f,e] = log2(-3)
f =

    - 0.75
e =

    2.

```

```

-->[f,e] = log2(0)
f =

    0.
e =

    0.

```

- `nearfloat`

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

```
nearfloat('pred',  $\xi$ )
```

restituisce il predecessore di  $\xi$ ,

```
nearfloat('succ',  $\xi$ )
```

restituisce il successore di  $\xi$ . Ad esempio:

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

```
1.0000000
```

```
-->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 appartenente all'intervallo [2, 25] se *tipo* è 'v', all'intervallo [8, 25] se *tipo* è 'e'. Precisamente: per ogni intero positivo  $n$  si indichi con  $V_n$  l'insieme dei numeri reali  $x$  che ammettono almeno una tra le seguenti scritte in base dieci:<sup>1</sup>

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

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

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

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

- se *formato corrente* = ['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);
- 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).

<sup>1</sup>L'intero  $n$  indica il *numero massimo di simboli* che *Scilab* può usare per scrivere il risultato dell'arrotondamento.

<sup>2</sup>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*.

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

```
-->0.00390625
ans =
```

```
0.0039062
```

```
-->12345678
ans =
```

```
12345678.
```

```
-->-123456789
ans =
```

```
- 1.235D+08
```

Nel primo caso, la scrittura in formato variabile del numero immesso dall'utilizzatore ( $2^{-8} \in F(2, 53)$ ) è:  $+0.00390625$  e richiede undici simboli. I numeri reali più vicini al valore immesso e che ammettono una scrittura in formato variabile che utilizza *al più* dieci simboli sono:  $a = +0.0039062$  e  $b = +0.0039063$ . Entrambi distano  $5 \cdot 10^{-8}$  dal valore immesso. La scrittura in formato esponenziale del numero immesso dall'utilizzatore è:  $+3.90625D-03$  e richiede dodici simboli. Il numero reale più vicino al valore immesso e che ammette una scrittura in formato esponenziale che utilizza *al più* dieci simboli è:  $c = +3.906D-03$ , e dista  $25 \cdot 10^{-8}$  dal valore immesso. L'elemento mostrato da *Scilab* (in formato variabile) è  $a$ : l'arrotondato in  $V_{10}$  del valore immesso, con ultima cifra pari.

Nel secondo caso, il numero immesso è un elemento di  $F(2, 53)$  che ammette una scrittura in formato variabile che richiede dieci simboli. L'elemento mostrato da *Scilab* (in formato variabile) coincide con l'elemento immesso.

Nel terzo caso, la scrittura in formato variabile dell'elemento di  $F(2, 53)$  immesso dall'utilizzatore è:  $-123456789$ . e richiede undici simboli. Il numero reale più vicino al valore immesso e che ammette una scrittura in formato variabile che utilizza *al più* dieci simboli è:  $-99999999$ . che dista più di  $2 \cdot 10^7$  dal valore immesso. La scrittura in formato esponenziale del numero immesso dall'utilizzatore è:  $-1.23456789D+08$  e richiede quindici simboli. Il numero reale più vicino al valore immesso e che ammette una scrittura in formato esponenziale che utilizza *al più* dieci simboli è:  $-1.235D+08$  e dista meno di  $5 \cdot 10^4$  dal valore immesso. L'elemento mostrato da *Scilab* (in formato esponenziale perché non rappresentabile in formato variabile utilizzando al più dieci simboli) è dunque quest'ultimo. Si osservi che in tutti i casi *Scilab* mostra l'arrotondato in  $V_{10}$  del valore immesso.

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*<sup>3</sup> del valore della variabile in esame. Ad esempio:

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

---

<sup>3</sup>Più correttamente: *non meno accurata*. Vedere l'Esercizio 4.

```

1.    10.

-->x = 123456789
x =

1.235D+08

-->format('v',15)

-->x
x =

123456789.

```

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 =

1.00000000000000002220446

```

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 coincide di quello degli argomenti*. Inoltre, in *Scilab*, gli oggetti di tipo stringa sono racchiusi tra virgolette. Ad esempio:

```

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

3.1415927

1.4142136

-->x = 2^30;

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

"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 (anche) 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.<sup>4</sup>

Ad esempio:

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

```
1.
```

```
-->int(-%pi)
ans =
```

```
- 3.
```

```
-->int(-1)
ans =
```

```
- 1.
```

---

<sup>4</sup>Se  $\xi \in F_d(2, 53, -1021, 1024)$  è intero,  $\text{int}(\xi) = \xi$  altrimenti  $\text{int}(\xi)$  è l'intero adiacente a  $\xi$  più vicino a zero.



Discutere poi i valori ottenuti da:

```
[f,e] = log2(xi_norm_min)
```

e

```
[f,e] = log2(xi_min)
```

2. ★ Constatere 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.
```

disp: Il numero degli argomenti in uscita è sbagliato: ne erano attesi 0.

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  $1.074 \cdot 10^9$  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  $F(2, 53)$ ,  $\text{rd}$  è la funzione arrotondamento in  $F(2, 53)$  con RTTE e le pseudo funzioni aritmetiche sono definite nel modo usuale, si ha:

$$2 \otimes x = 2x \quad , \quad \text{int}(x) = \text{rd}(\lfloor x \rfloor) = \lfloor x \rfloor$$

e quindi:

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

(Suggerimento: dimostrare che  $2x$ ,  $\lfloor x \rfloor$  e  $2x - \lfloor 2x \rfloor$  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  $F_d(2, 53, -1021, 1024)$ .