

# Programare functionala. Fundamentele limbajului LISP

Ruxandra Stoean  
<http://inf.ucv.ro/~rstoean>  
[ruxandra.stoean@inf.ucv.ro](mailto:ruxandra.stoean@inf.ucv.ro)

# Bibliografie

- Stuart C. Shapiro, *Common Lisp: An Interactive Approach*, Computer Science Press, 1992.
- Internet.

# Introducere

- Vom opera cu mediul standard al limbajului LISP.
- Acest lucru presupune ca vom lucra in COMMON LISP.
- Common Lisp aduce o interfata simpla, de tip DOS.
- In particular, vom lucra cu implementarea CLISP 2.30.

# Introducere

- In momentul in care pornim Common Lisp, ne vom afla deja in fata prompterului Lisp.
- Prompterul va astepta sa introducem ceea ce, in cadrul programarii functionale, poarta numele de S-expresie (expresie simbolica).
- Dupa ce S-expresia este scrisa, apasam tasta ENTER.

# Ciclul citire-evaluare-scriere al Lisp

- Atunci cand dam Lisp-ului o S-expresie, acesta va produce urmatorii pasi:
  - Va **citi** S-expresia.
  - Va interpreta S-expresia drept **reprezentarea scrisa** a unui **forme** (**obiect** Lisp ce trebuie evaluat).
  - Va **evalua** forma drept alt (sau poate chiar acelasi) **obiect valoare**.

# Ciclul citire-evaluare-scriere al Lisp

- Va alege o reprezentare scrisa pentru obiectul valoare.
- Va **scrie** reprezentarea scrisa pe care a ales-o.
- Dupa ce intoarce valoarea, prompterul Lisp va reaparea si va astepta o noua expresie.

# Ciclul citire-evaluare-scriere al Lisp

- Acesta este modul de folosire al Lisp:
  - Utilizatorul introduce reprezentarea scrisa a unei forme.
  - Lisp o evalueaza.
  - Apoi, trimite inapoi o reprezentare scrisa a valorii formei.

# Un prim exemplu

- O S-expresie simpla pe care o vom introduce este numeralul in scriere araba, 3.
- Aceasta este una din reprezentarile scrise pe care le folosim pentru numarul 3.
- Oamenii folosesc si numeralul in scriere romana III.
- Aceasta este deci distinctia pe care o face si Lisp intre un obiect si diferitele sale posibilitati de reprezentare scrisa.



# Exemplu

- Lisp interpreteaza numeralul 3 ca reprezentand numarul 3.
- Evalueaza aceasta forma – adica obiectul numeric 3.
- In Lisp, numerele sunt evaluate in ele insele.
- Lisp va alege o reprezentare scrisa pentru 3 si va utiliza, de asemenea, numeralul arab 3.

# Interactiunea cu Lisp

```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
Microsoft Windows [Version 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.

D:\Kits\clisp-2.30>lisp.exe -M lispinit.mem
  i i i i i i i
  I I I I I I I
  I   \   +   /   I
  \   -   +   -   /
  -   -   -   -   -
  -----+-----
          00000  0  0000000  00000  00000
          8      8  8      8      8      8
          8      8  8      8      8      8
          8      8  8      00000  80000
          8      8  8      8      8
          8      0  8      8      8
          00000  8000000  0008000  00000  8

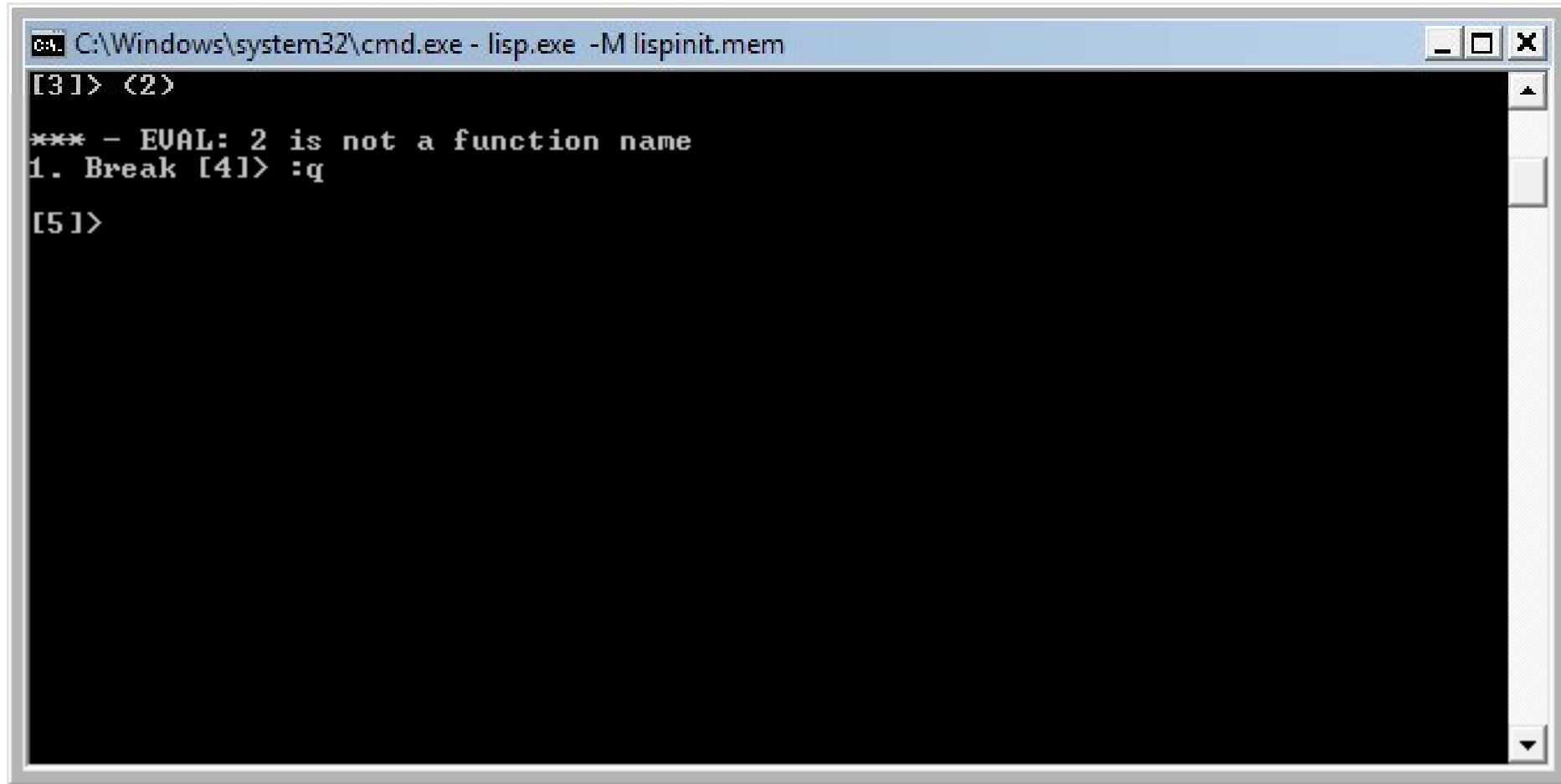
Copyright (c) Bruno Haible, Michael Stoll 1992, 1993
Copyright (c) Bruno Haible, Marcus Daniels 1994-1997
Copyright (c) Bruno Haible, Pierpaolo Bernardi, Sam Steingold 1998
Copyright (c) Bruno Haible, Sam Steingold 1999-2002

[1]> 3
3
[2]>
```

# Debugger-ul din Lisp

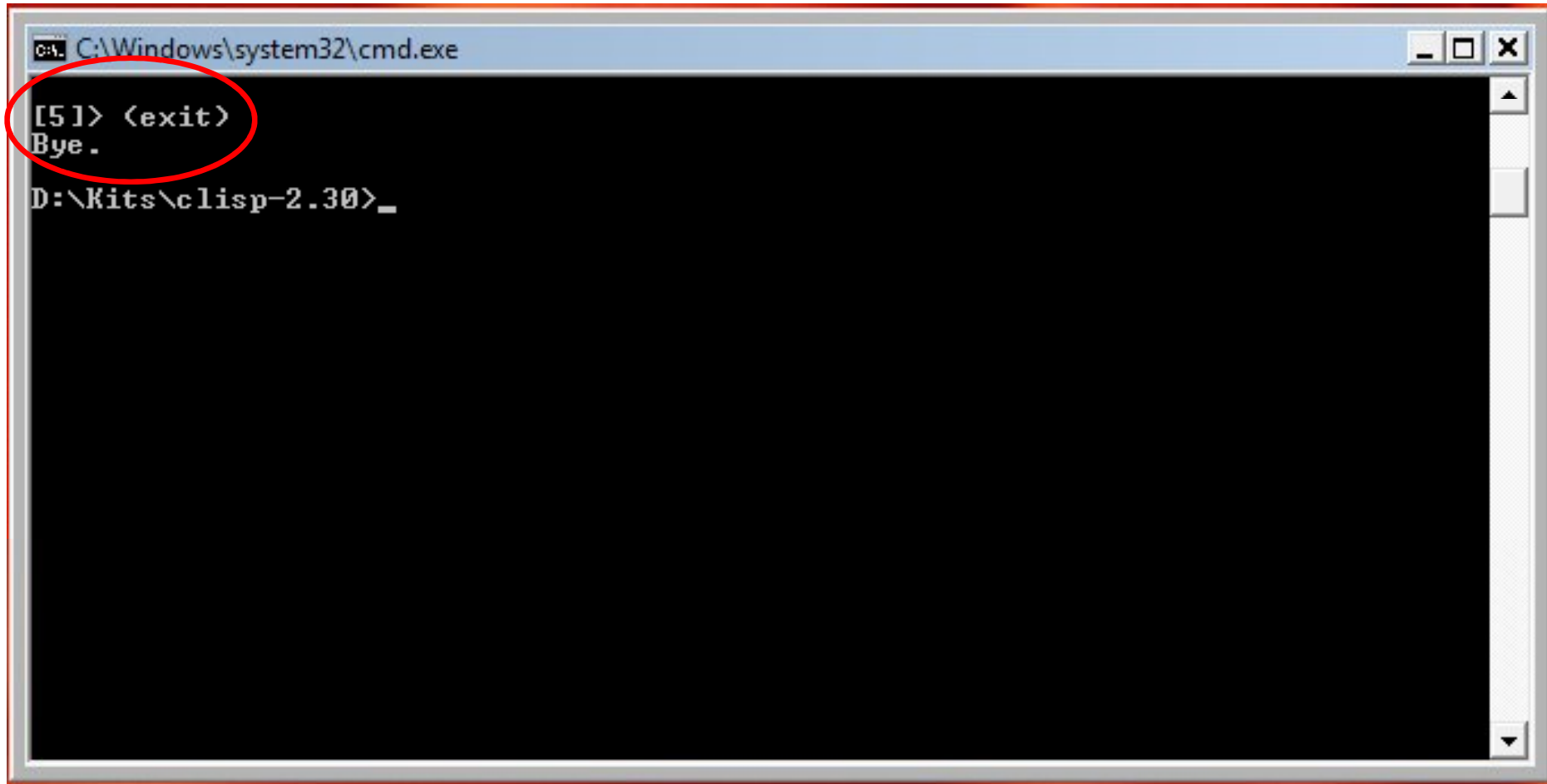
- Daca in introducerea unei S-expresii se face vreo greseala, va intra debugger-ul Lisp-ului.
- Acesta mai poarta numele si de ciclu (sau pachet) break.
- Acesta arata ca un prompter Lisp obisnuit, doar ca exista aici niste comenzi speciale pentru a obtine informatii despre ce presupune eroarea.
- Deocamdata, vom parasi aceste bucle break, tastand **:q.**

# Debugger-ul din Lisp



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[3]> (2)
*** - EVAL: 2 is not a function name
1. Break [4]> :q
[5]>
```

# Terminarea sesiunii de Lisp



The image shows a Windows command prompt window with the title bar "C:\Windows\system32\cmd.exe". The window contains the following text:

```
[5]> (exit)  
Bye.  
D:\Kits\clisp-2.30>_
```

The text "[5]> (exit)" is circled in red. The prompt "D:\Kits\clisp-2.30>\_" indicates that the Lisp session has been terminated and the command prompt has returned to the directory where clisp was running.

# Numere in Lisp

- Numerele sunt unul dintre tipurile de baza ale Lisp-ului.
- Se pot folosi numere intregi sau reale.
- In cadrul intregilor, nu putem folosi insa virgule sau spatii:
  - 12 345 sau 12,345 sunt reprezentari incorecte de intregi.
  - Vom scrie direct 12345.

# Numere in Lisp

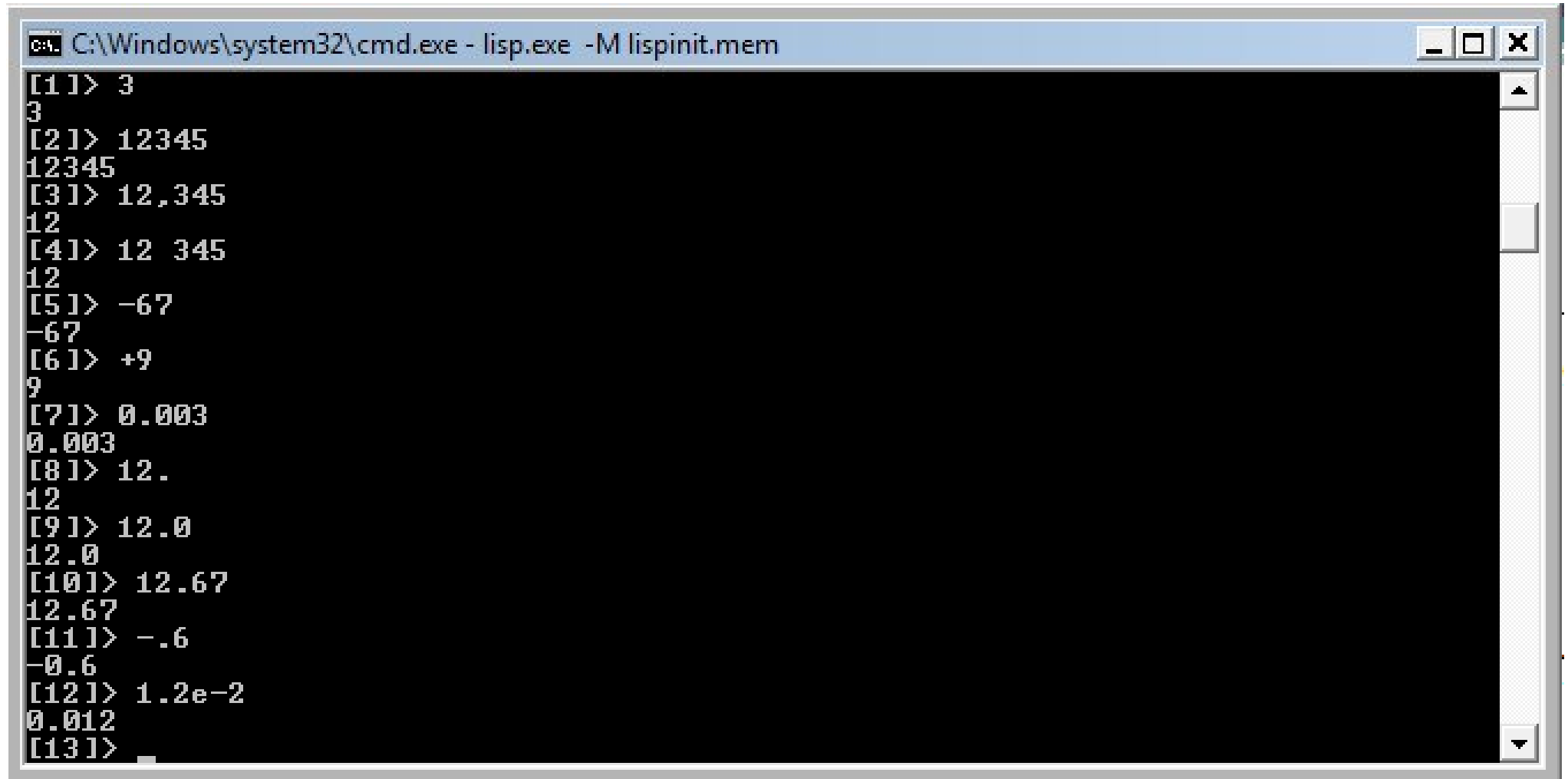
- Pentru a scrie un intreg negativ, vom insera semnul “-” in fata sa, iar pentru unul pozitiv putem de asemenea pune semnul “+”:
  - -34, +25 sunt expresii corecte de intregi.
- Un intreg se poate termina cu “.” – acesta va fi citit drept intreg:
  - 12. va fi egal cu a scrie 12.
  - 12.0 va fi inasa interpretat drept real.

# Numere in Lisp

- Numerele reale sunt construite cu ajutorul semnului “.” si cu cel putin o cifra dupa punct:
  - 12.9
  - 13.0
- Pot fi scrise si sub forma stiintifica, cu semnul de exponent:
  - 0.34e-2 – care inseamna  $0.34 \times 10^{-2}$ .



# Numere in Lisp



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[1] > 3
3
[2] > 12345
12345
[3] > 12,345
12
[4] > 12 345
12
[5] > -67
-67
[6] > +9
9
[7] > 0.003
0.003
[8] > 12.
12
[9] > 12.0
12.0
[10] > 12.67
12.67
[11] > -.6
-0.6
[12] > 1.2e-2
0.012
[13] >
```



# Liste in Lisp

- LisP = List Processing
- Care este reprezentarea scrisa a unei liste?
- Conform lui S. C. Shapiro, definitia unei S-expresii lista este:
  - O paranteza stanga urmata de zero sau mai multe S-expresii urmate de o paranteza dreapta este o S-expresie lista.
- S-expresiile se delimiteaza una de alta prin spatii.

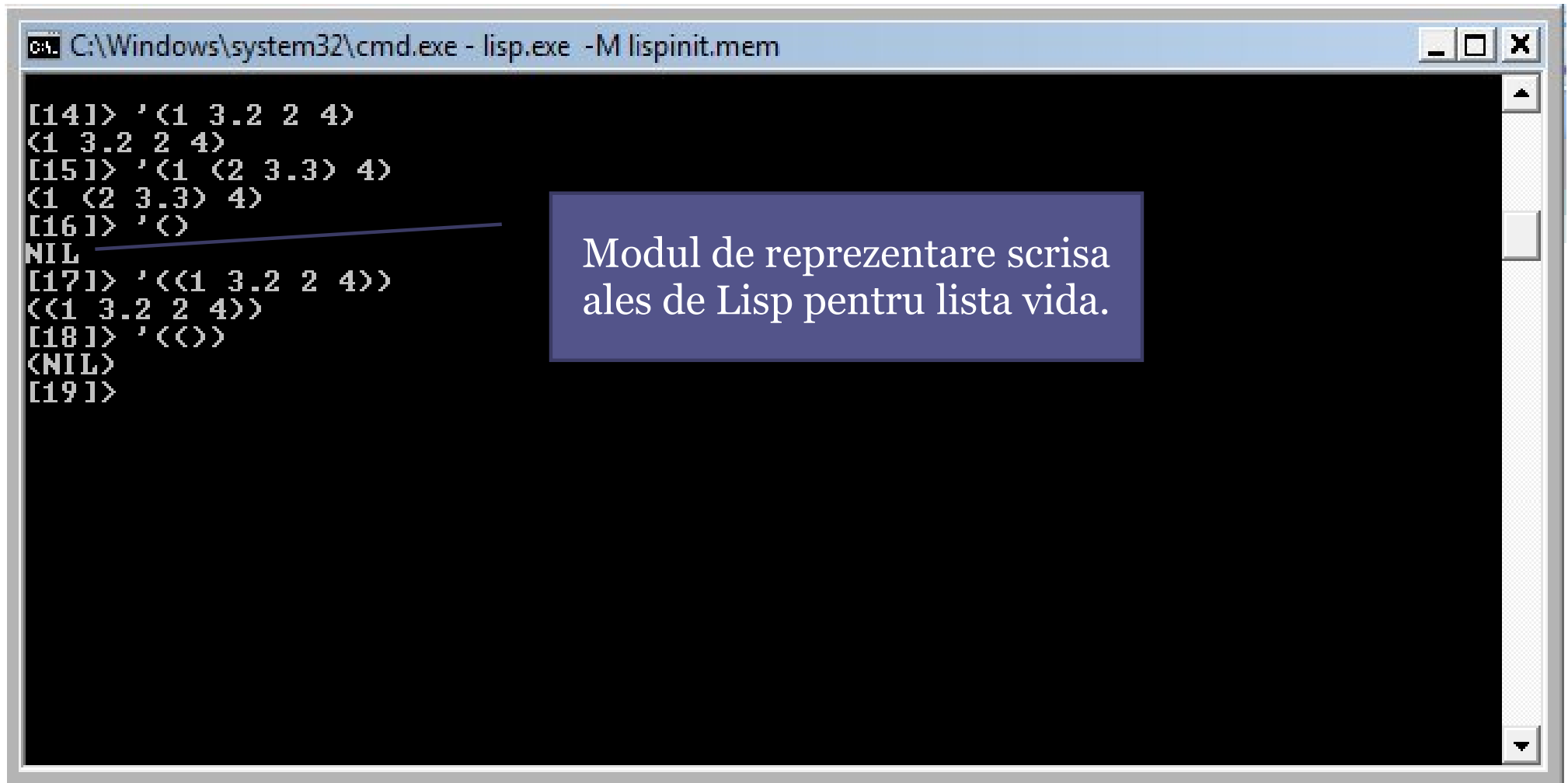
# Exemple

- $(1\ 3.2\ 2\ 4)$
- $(1\ (2\ 3.3)\ 4)$
- $()$
- $((1\ 3.2\ 2\ 4))$
- $((()))$

# Liste

- In acest moment, Lisp-ul citește expresia care este data de utilizator și încearcă să o evalueze.
- Pana la a evalua o lista, ii vom cere Lisp-ului doar sa ne afiseze lista introdusa.
- Putem impiedica evaluarea unei liste si, in loc, sa obtinem printarea ei folosind semnul de apostrof inaintea S-expresiei lista.

# Exemple de interactiune cu Lisp



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem

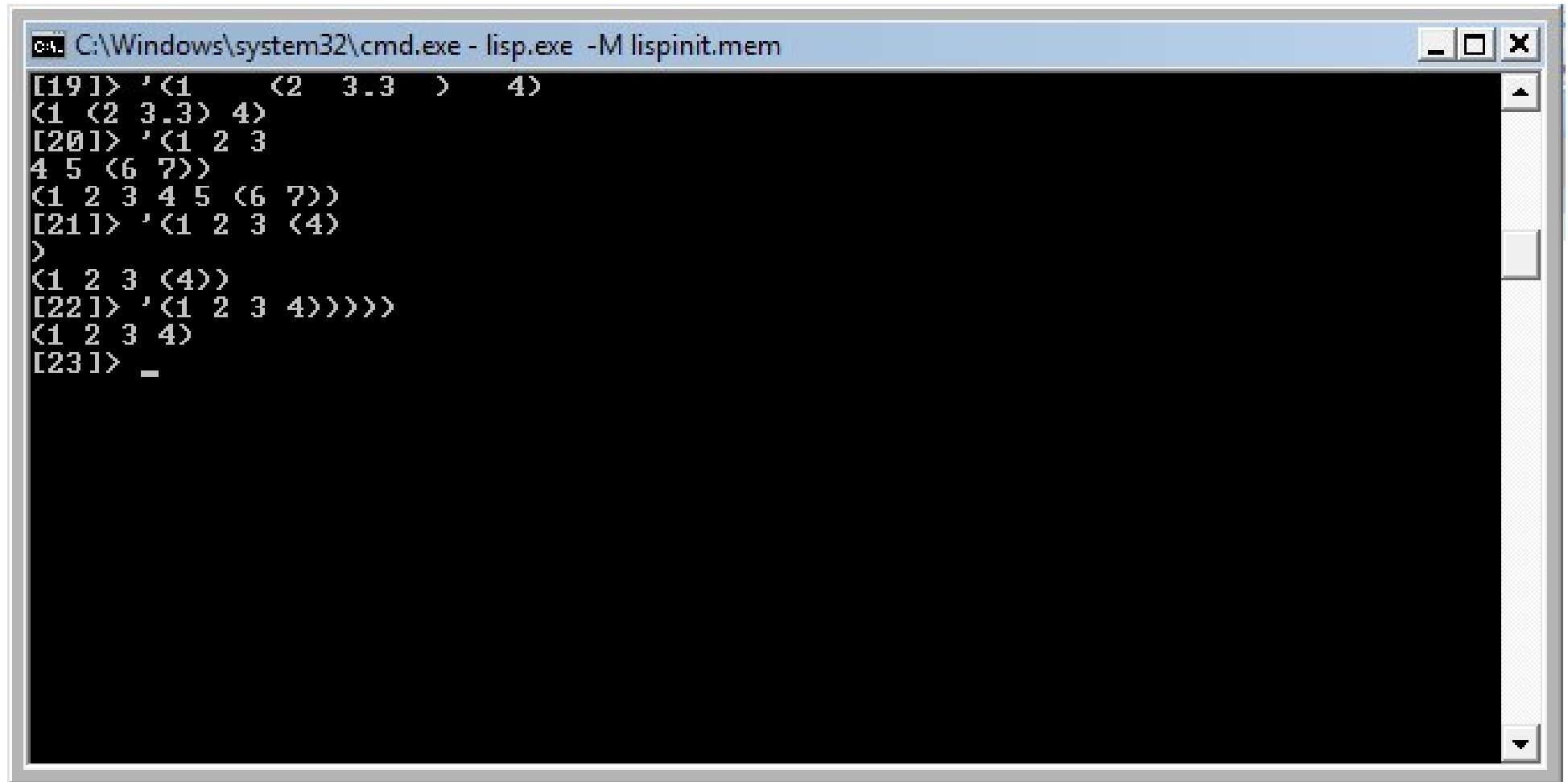
[14]> '(1 3.2 2 4)
<1 3.2 2 4>
[15]> '(1 (2 3.3) 4)
<1 (2 3.3) 4>
[16]> '(<>)
NIL
[17]> '<<1 3.2 2 4>>
<<1 3.2 2 4>>
[18]> '(<>)
<NIL>
[19]>
```

Modul de reprezentare scrisa  
ales de Lisp pentru lista vida.

# Liste

- Lisp-ul va ignora de asemenea spatiile in plus sau ENTER-urile.
- Daca toate parantezele deschise nu sunt inchise de utilizator, Lisp-ul va astepta in continuare paranteze dreapta.
- Se pot pune mai multe paranteze dreapta decat stanga; Lisp-ul le va ignora pe cele in plus.

# Exemple



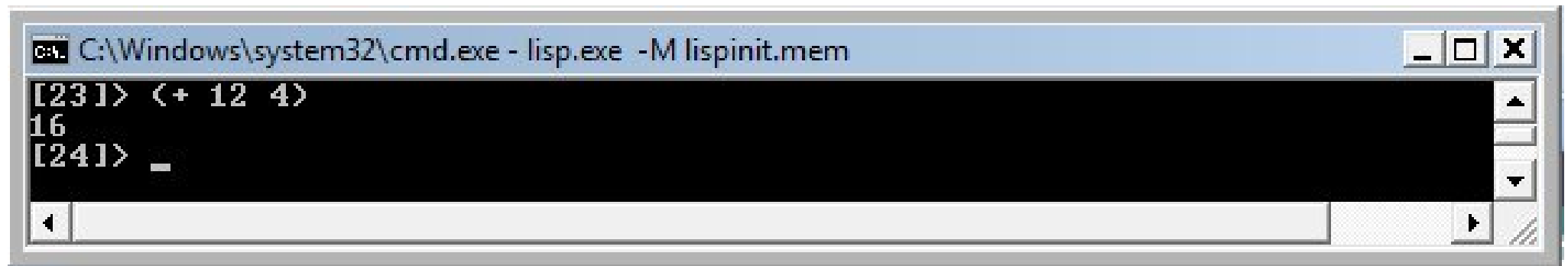
```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[19]> '(1 (2 3.3) 4)
(1 (2 3.3) 4)
[20]> '(1 2 3
4 5 (6 7))
(1 2 3 4 5 (6 7))
[21]> '(1 2 3 (4)
)
(1 2 3 (4))
[22]> '(1 2 3 4)))))
(1 2 3 4)
[23]> _
```



# Expresii aritmetice in Lisp

- Evaluarea obiectelor lista este operatia de baza in Lisp.
- Conform lui S. C. Shapiro:
  - Valoarea unei liste este cea obtinuta prin aplicarea functiei denumita de primul argument (membru) al listei asupra valorilor celorlalti membri ai listei.
- Vom incepe evaluarea listelor cu ajutorul operatorilor matematici de baza: +, -, \*, /.

# Exemplu



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem". The command prompt shows a Lisp interpreter session with the following text:

```
[231]> <+ 12 4>  
16  
[241]> _
```

The window includes standard Windows window controls (minimize, maximize, close) and a scrollbar at the bottom.

# Notatia prefixa Cambridge

- Formatul sub care expresiile aritmetice sunt scrise sub forma de lista poarta numele de notatie prefixa Cambridge.
- Numele provine de la cel care a dezvoltat aceasta notatie – John McCarthy de la MIT, Cambridge, MA – si de la faptul ca operatorul “prefixeaza” (este inaintea) operanzilor sai.

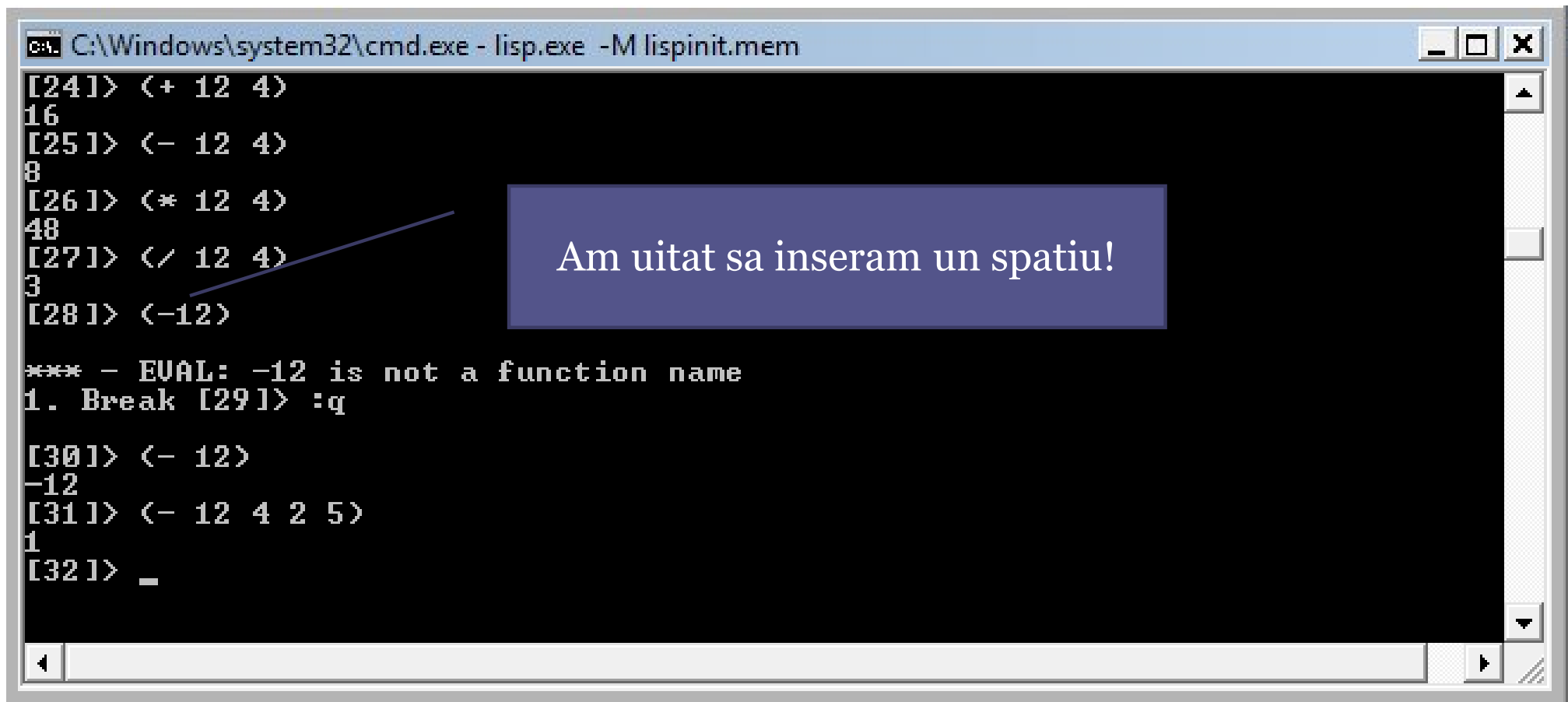
# Notatia prefixa Cambridge

- Termenul a fost preluat de la notatia poloneza prefixa, unde functia e scrisa inaintea argumentelor sale.
- Acest format difera de cel matematic clasic de tip infix:
  - In care operatorul este scris intre operanzii sai (de ex.  $12 + 4$ )
  - sau in care functia este scrisa inainte de argumente dar nu in paranteza cu ele (de ex.  $f(x, y)$ ).
  - Aceasta din urma se va scrie in Lisp sub forma:  $(f\ x\ y)$

# Notatia prefixa Cambridge

- Avantajul major al acestei notatii este ca scrierea ramane foarte simplu de utilizat indiferent de numarul de argumente:
  - 1,
  - 2
  - sau chiar mai multe ducand la operatii succesive

# Exemple de interactiune



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[241]> (+ 12 4)
16
[251]> (- 12 4)
8
[261]> (* 12 4)
48
[271]> (/ 12 4)
3
[281]> (-12)

*** - EVAL: -12 is not a function name
1. Break [291]> :q

[301]> (- 12)
-12
[311]> (- 12 4 2 5)
1
[321]> _
```

Am uitat sa inseram un spatiu!

# Evaluarea listelor

- Dacă argumentele funcțiilor aritmetice sunt întregi, rezultatul va fi întreg.
- Dacă unul dintre argumente este real, atunci rezultatul va fi real.

# Evaluarea listelor

- Exceptie se face daca incercam sa impartim un intreg la un alt intreg si valoarea rezultata nu este exacta:
  - Rezultatul va fi ceea ce poarta numele de fractie: 2 numere separate de semnul “/”, pozitive sau negative.
  - Fractia va fi reprezentata de catre Lisp sub forma simplificata.
- Si utilizatorul poate introduce fractii, chiar si sub forma nesimplificata.



# Example

```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[32]> </ 12 4>
3
[33]> </ 12.0 8>
1.5
[34]> </ 12 8>
3/2
[35]> 12/8
3/2
[36]> _
```

# Evaluarea listelor

- Putem avea expresii aritmetice incluse in alte expresii aritmetice – cum este natural in matematica, de exemplu,  $5 \times (3 + 4)$ .
- In Lisp, aceasta expresie se va scrie sub forma:  
 $(\times 5 (+ 3 4))$
- In schimb,  $5 \times 3 + 4$  se scrie:  
 $(+ (\times 5 3) 4)$
- In general  $f(x, g(y))$  se va scrie sub forma:  
 $(f x (g y))$

# Interactiune



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[36]> < * 5 < + 3 4 > >
35
[37]> < + < * 5 3 > 4 >
19
```

# Exercitiu

- Sa calculam radacinile ecuatiei:

$$2x^2 + 7x + 5 = 0$$

- Acestea sunt:

$$\frac{-7 \pm \sqrt{7^2 - 4 \times 2 \times 5}}{2 \times 2}$$

# Exercitiu

- Le vom scrie Lisp-ului sub forma:

- `(/ (+ -7.0 (sqrt (- (expt 7 2) (* 4 2 5)))) (* 2 2))`

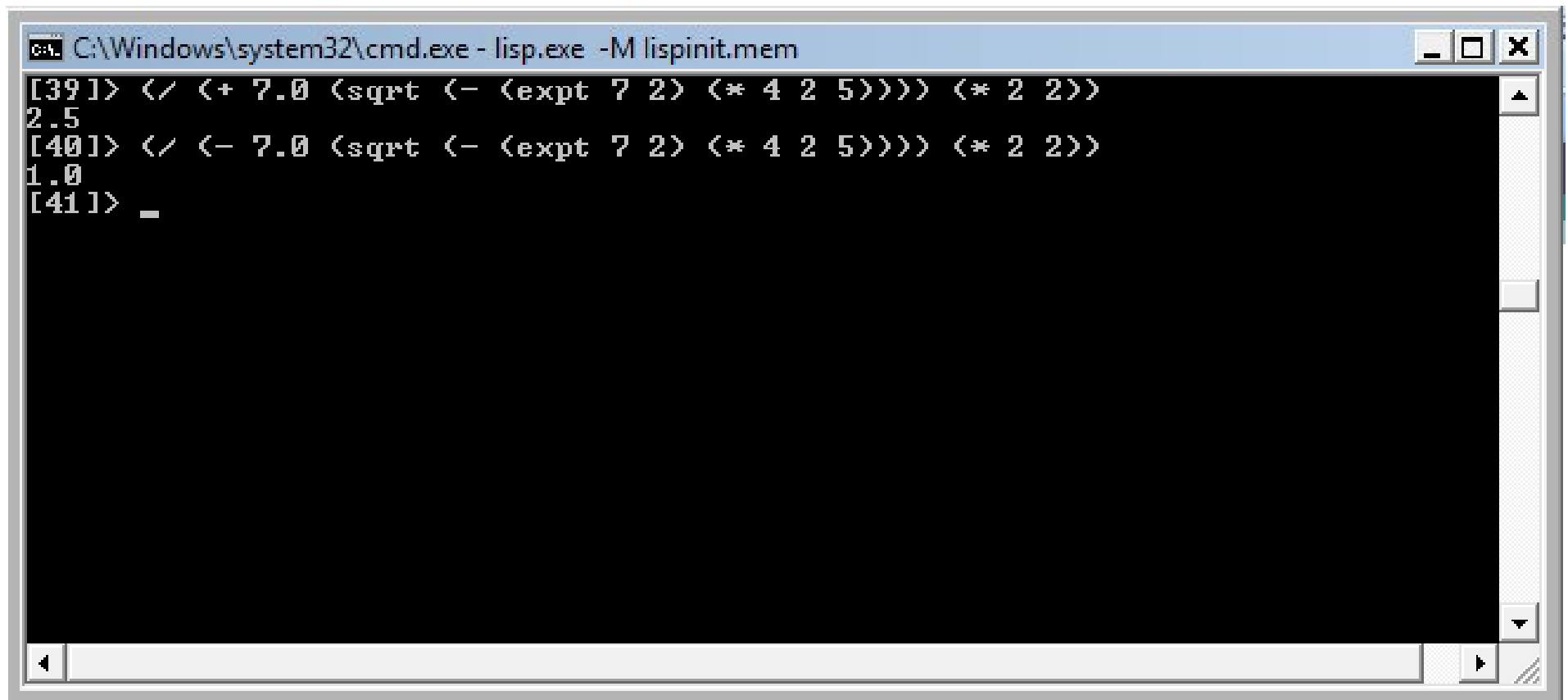
Functia radical –  
un singur  
argument

Functia ridicare  
la putere – 2  
argumente

- **si**

- `(/ (- -7.0 (sqrt (- (expt 7 2) (* 4 2 5)))) (* 2 2))`

# Interactiune

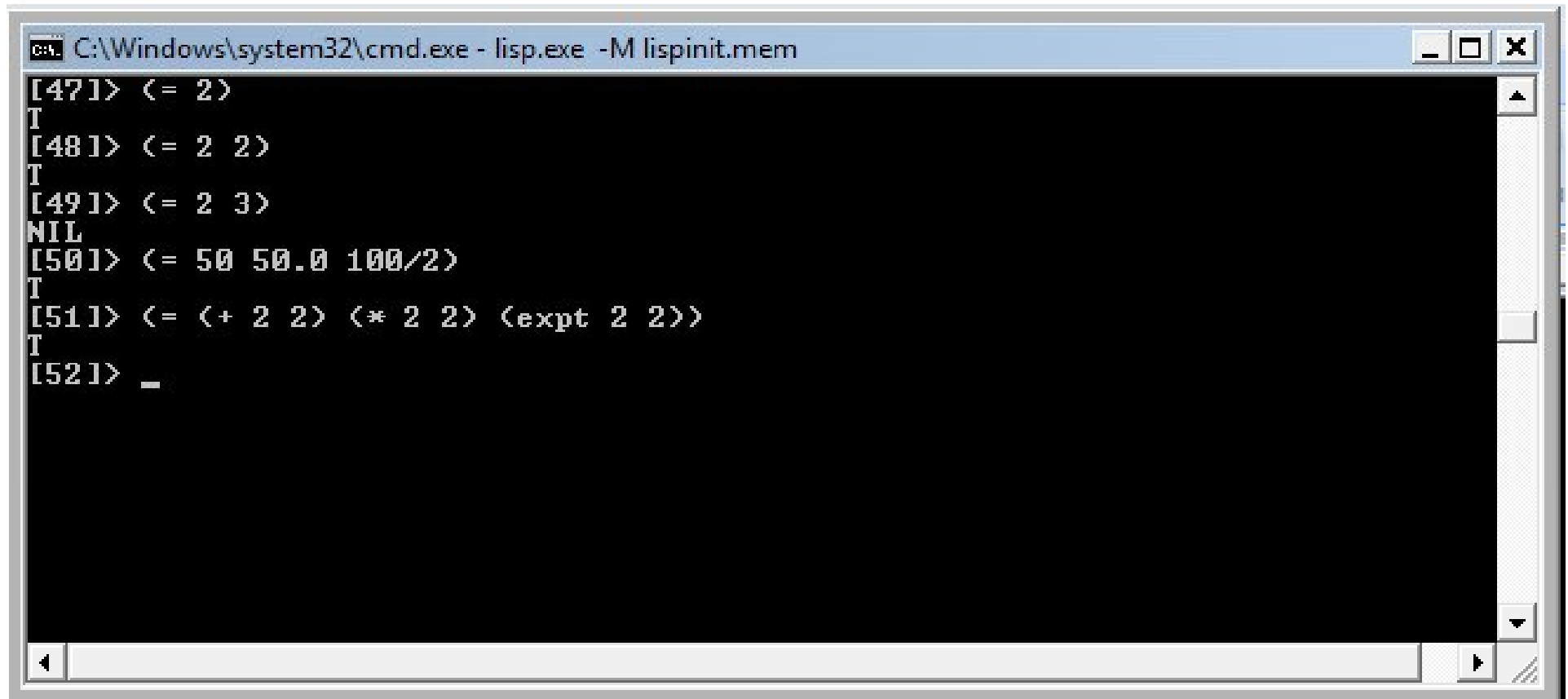


```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[39]> </ (<+ 7.0 (sqrt (- (expt 7 2) (* 4 2 5)))) (* 2 2))
2.5
[40]> </ (- 7.0 (sqrt (- (expt 7 2) (* 4 2 5)))) (* 2 2))
1.0
[41]> _
```

# Testarea egalitatii

- Verificarea egalitatii se face cu operatorul “=”.
- I se pot da 1, 2 sau mai multe argumente.
- Argumentele pot fi de tipuri numerice diferite – “=” testeaza numai egalitatea numerica.
- Intoarce TRUE (T) daca numerele sunt egale si FALSE (NIL) altfel.

# Interactiune



```
C:\Windows\system32\cmd.exe - lisp.exe -M lispinit.mem
[47]> (= 2)
T
[48]> (= 2 2)
T
[49]> (= 2 3)
NIL
[50]> (= 50 50.0 100/2)
T
[51]> (= (+ 2 2) (* 2 2) (expt 2 2))
T
[52]> _
```



# Exercitiu

- Utilizand Lisp-ul, gasiti valorile pentru:
  - $(25 + 30) \times 15/2$
  - $6 \times 3.1416$
  - Media numerelor 5, 6.7, -23.2, 75 si 100.3

Pe saptamana viitoare...

