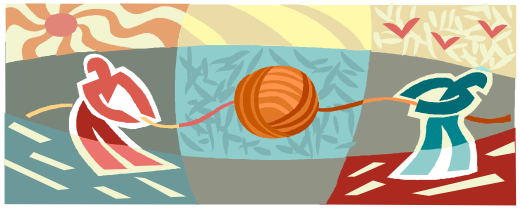
Caractere și stringuri



Ruxandra Stoean
http://inf.ucv.ro/~rstoean
ruxandra.stoean@inf.ucv.ro

Scrierea și citirea caracterelor

- Prologul are predicate predefinite folosite pentru scrierea şi citirea câte unui caracter.
- Predicatul *put/1* va scrie un singur caracter.
- Din păcate insă, argumentul trebuie să fie un întreg care reprezintă caracterul ASCII.

Scrierea caracterelor

- Deci, pentru scrierea mesajului *merge*, ar trebui să apelăm:

? - put(109), put(101), put(114), put(103), put(101).

- Evident însă, oricine ar prefera să folosească:

? – write('merge').

Scrierea caracterelor

- Dacă folosim însă ca argument la predicatul *write/1* un mesaj scris între ghilimele:
  - ne vor fi afişate chiar valorile întregi care reprezintă fiecare caracter în cod ASCII.

? – write(”merge”).

[109, 101, 114, 103, 101]

Scrierea caracterelor

- Având astfel lista de valori numerice, este simplu de realizat un predicat recursiv care sa scrie toate caracterele, rând pe rând.

```
scriestringul([]).
scriestringul([P|R]) :- put(P), tab(1),

        scriestringul(R).
```

Scrierea caracterelor

? - scriestringul(”merge”).

```
3 ?- scriestringul("merge").
m e r g e

Yes
```

Citirea caracterelor

- Pentru citirea unui caracter folosim predicatul *get/1:*
  - acesta citeşte doar primul caracter din cele introduse.
- Poate fi introdus orice caracter.

Citirea caracterelor

- La citire, nu mai trebuie să incheiem, după ce am introdus caracterul de citit, cu punct (.).
- Punctul poate fi chiar caracterul de citit.
- Dacă citirea se face dintr-un fişier, verificarea de sfârşit de fişier se face comparând valoarea dată ca argument la predicatul *get/1* cu –1.

Citirea caracterelor

- Predicatul *citesc/0* definit în continuare, citeşte un caracter şi afişează valoarea sa corespunzătoare în codul ASCII:

```
citesc :- write('Introduceti un caracter:'), get(C),
  nl, write(Valoarea caracterului '), put(C), write(’
  in cod ASCII este:’), write(C).
```

Citirea caracterelor

?-citesc.

```
5 ?- citesc.
Introduceti un caracter:c

Valoarea caracterului c in cod ASCII este:99

Yes
```

Predicate predefinite la stringuri

- Deja am folosit mai devreme stringuri:
  - ele sunt atomi simpli, scrişi însă între ghilimele

- Trecerea de la atomi la lista care conţine numere întregi în cod ASCII se face prin intermediul predicatului

- string_to_list(+String, -Ascii) sau
- string_to_list(-String, +Ascii)

String -> List

? – string_to_list(string, Valoare).

```
8 ?- string_to_list(string, Valoare).

Valoare = [115, 116, 114, 105, 110, 103]

Yes
```

? - string_to_list(String, [99,101,118,97]).

```
9 ?- string_to_list(String, [99,101,118,97]).

String = "ceva"

Yes
```

Concatenare stringuri

- string_concat(+String1, +String2, -String3)

- *string_concat* este un predicat similar lui *atom_concat/3*, cu singura deosebire că argumentul de ieşire aici este de tipul string.
  - De subliniat faptul că argumentele de intrare sunt date ca atomi.

Concatenare stringuri

? - string_concat(campu, lung, Tot).

```
10 ?- string_concat(campu, lung, Tot).

Tot = "campulung"

Yes
```

?- string_concat(campu, Ce, campulung).

```
11 ?- string_concat(campu, Ce, campulung).

Ce = "lung"

Yes
```

Lungime sir de caractere

- string_length(+String, -Lungime)

? - string_length('un string mare de tot', M).

```
14 ?- string_length('un string mare de tot', M).

M = 21

Yes
```

? - string_length(altceva, M).

```
13 ?- string_length(altceva, M).

M = 7

Yes
```

String <-> Atom

- string_to_atom(+String, -Atom) sau
- string_to_atom(-String, +Atom)

?- string_to_atom("string de transformat", Atom).

```
16 ?- string_to_atom("string de transformat", Atom).

Atom = 'string de transformat'

Yes
```

?- string_to_atom(String, exemplu).

```
1 ?- string_to_atom(String, exemplu).

String = "exemplu"
```

Determinarea subsirului unui sir

- sub_string(+String, +Start, +Lungime, -Rest, -Substring)
- Primul argument este sirul de caractere din care extragem un subşir.
- Start este un întreg pozitiv care dă poziţia de la care selectăm subşirul.
- Al treilea argument dă lungimea subşirului.
- *Rest* este tot un întreg pozitiv care spune câte poziţii mai sunt până la sfârşitul şirului iniţial.
- Iar ultimul argument reprezintă chiar subşirul căutat.

Determinarea subsirului unui sir

?- sub_string('Alin e tare nebun', 0, 11, Rest, Valoare).

```
19 ?- sub_string('Alin e tare nebun', 0, 11, Rest, Valoare).

Rest = 6
Valoare = "Alin e tare"

Yes
```

?- sub_string('Alin e tare nebun', _, 5, 0, Valoare).

```
20 ?- sub_string('Alin e tare nebun', _, 5, 0, Valoare).

Valoare = "nebun"

Yes
```

Determinarea subsirului unui sir

?- sub_string('Alin e tare nebun', 5, Cat, 6, Valoare).

```
21 ?- sub_string('Alin e tare nebun', 5, Cat, 6, Valoare).

Cat = 6
Valoare = "e tare"

Yes
```

?- sub_string('Alin e tare nebun', Inceput,Cat,Rest,nebun).

```
22 ?- sub_string('Alin e tare nebun', Inceput,Cat,Rest,nebun).

Inceput = 12
Cat = 5
Rest = 0

Yes
```

Determinarea subsirului unui sir

- De subliniat că şi în cazul acestui predicat, ca şi la *string_concat/3:*
  - argumentele de intrare sunt atomi
  - doar cel de ieşire reprezentând un string.

- După cum se vede din exemple:
  - numai primul argument trebuie să fie întotdeauna instanţiat
  - celelalte pot fi calculate.

Transformare data si ora in string

- convert_time(+Timp, -Valoare)

- Primul argument poate fi obţinut prin apelarea predicatului *get_time/1*: obţinem astfel data şi ora curente.

Transformare data si ora in string

?- get_time(Timp), convert_time(Timp, Data), nl, write('Data este '), write(Data).

```
23 ?- get_time(Timp), convert_time(Timp, Data), nl, write('Data este '), write(Data).

Data este Thu Mar 29 12:45:23 2007

Timp = 1.17516e+009
Data = "Thu Mar 29 12:45:23 2007"

Yes
```

Transformare data si ora in string

- Pentru manevrarea mai uşoară a datei şi orei, mai există şi predicatul

- *convert_time(+Timp, -Anul, -Luna, -Ziua, -Ora, -Minute, -Secunde, -Milisecunde)*.

Transformare data si ora in string

?- get_time(Timp), convert_time(Timp, An, Luna, Zi, Ora, Minute, Secunde, Milisec).

```
24 ?- get_time(Timp), convert_time(Timp, An, Luna, Zi, Ora, Minute, Secunde, Milisec).

Timp = 1.17516e+009
An = 2007
Luna = 3
Zi = 29
Ora = 12
Minute = 45
Secunde = 41
Milisec = 881

Yes
```

Palindrom

- Verificaţi cu ajutorul unui predicat dacă un cuvânt dat este palindrom.
- Un palindrom este un cuvânt care are aceeaşi formă, fie că este citit de la stânga, fie de la dreapta.
- Exemple: capac, lupul, cojoc, rar.

Palindrom

```
palindrom :- write('Introduceti cuvantul de
   verificat:'), read(Cuvant), string_to_list(Cuvant,
   Lista), palindrom(Lista).

palindrom(Lista) :- invers(Lista, Lista), write('Da,
   cuvantul este un palindrom!').
palindrom(_) :- write('Cuvantul nu este un
   palindrom!').
```

Inversarea unei liste - recapitulare

```
invers(L1, L2) :- inv(L1, [], L2).

inv([], L, L).
inv([X|R], Lt, L) :- inv(R, [X|Lt], L).
```

Palindrom

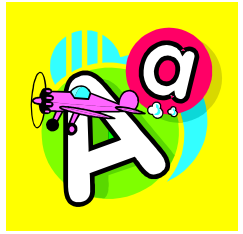
```
28 ?- palindrom.
Introduceti cuvantul de verificat:cojoc.
Da, cuvantul este un palindrom!

Yes
29 ?- palindrom.
Introduceti cuvantul de verificat:conac.
Cuvantul nu este un palindrom!

Yes
```

Vocale

- Având un text introdus în fişierul *intrare.txt*, să se numere câte vocale se găsesc în acesta.
- Pregatesc fisierul intrare.txt.



- Il deschid in citire.
- Colectez caracter de caracter.
  - Daca acela care este curent este membru in multimea data de vocale, il numar;
  - Altfel, parcurgem fisierul mai departe.

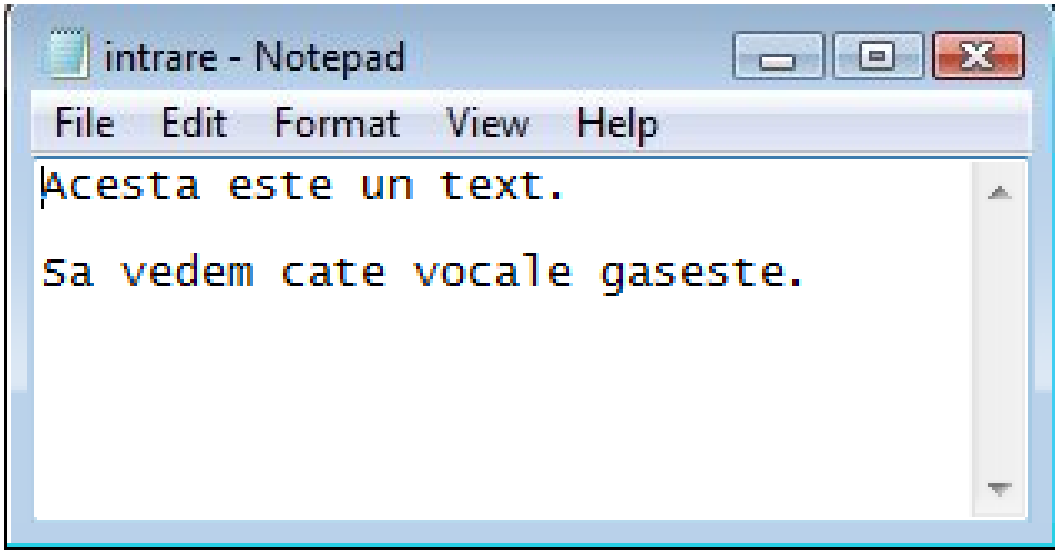
Vocale

```
vocale :- see('intrare.txt'), numar(0), seen.

numar(N) :- get(Caracter), Caracter \= -1,
   verific(Caracter, N, N1), numar(N1).
numar(N) :- write('Am gasit '), write(N), write('
   vocale.').

verific(Caracter, N, N1) :-
   string_to_list(aeiouAEIOU,Lista),
   member(Caracter, Lista), N1 is N + 1.
verific(_, N, N).
```

Vocale



38 ?- vocale.
Am gasit 18 vocale.

Yes

Permutari de caractere

- Având un sir de caractere introdus de la consolă, să se genereze şi să se afişeze toate permutările de caractere posibile.

Permutari de caractere

```
permut :- write('Introduceti cuvantul pentru care
   generam permutarea:'), read(Cuvant),
   string_to_list(Cuvant, CLista), permut(CLista).

permut(L) :- permut(L, Rez), scr(Rez), nl, fail.
permut(_).

scr([]).
scr([P|R]) :- put(P), scr(R).
```

Permutari de caractere

```
permut([], []).
permut(L, P) :- selecteaza(X, L, Rest),
   permut(Rest, RestPerm), P = [X|RestPerm].

selecteaza(X, [X|R], R).
selecteaza(X, [Y|R], [Y|R1]) :- selecteaza(X, R,
   R1).
```

Permutari de caractere

```
39 ?- permut.
Introduceti cuvantul pentru care generam permutarea:ion.
ion
ino
oin
oni
nio
noi

Yes
```

Pe saptamana viitoare!