# Programare funcţională – Laboratorul 12

Fişiere. Operaţii I/O

Isabela Drămnesc

May 21, 2014

## 1 Concepte

- read, read-line, read-string
- write, display
- open-input-file, open-output-file
- map, foldl, foldr
- operaţii cu fişiere

## 2 Read, write

(read-line [in mode])

- Se citesc caractere pana la intalnirea unui separator sau pana cand se citeste end-of-file.
- Daca nu se citeste nici un caracter inainte de end-of-file, atunci se returneaza eof.

## 3 Fisiere

### 3.1 Porturi

Fiecare procedura de input/output poate avea un argument extra pentru a specifica un fisier:

```
(show '(across the universe) file1)
(show-line '(penny lane) file2)
(read file3)
```

- inainte de a utiliza un fisier, trebuie sa-l deschideti
- Daca doriti sa cititi un fisier, sistemul trebuie sa verifice daca fisierul exista
- Daca doriti sa scrieti intr-un fisier, sistemul trebuie sa creeze un fisier gol

- Procedurile Racket care deschid fisierele returneaza un port, pe care Racket il utilizeaza pentru a memora fisierul pe care l-ati deschis
- Port-urile sunt utile doar ca si argumente pentru proceduri de input/output

### Exemplu:

```
> (let ((port (open-output-file "songs")))
    (show '(all my loving) port)
    (show '(ticket to ride) port)
    (show '(martha my dear) port)
    (close-output-port port))
```

close-output-port, ca si define, au cate o valoare returnata nespecificata (despre care nu discutam in exemplele noastre).

- Fişiere: functia **open-output-file** deschide un fisier pentru scriere
- **open-input-file** deschide un fisier pentru citire.

```
> (define out (open-output-file "output.txt"))
> (display "hello" out)
> (close-output-port out)
> (define in (open-input-file "data.txt"))
> (read-line in)
> (read in)
> (close-input-port in)
```

## 3.2 Scrierea fisierelor pentru citire

Daca un fisier exista deja, atunci open-output-file lanseaza o exceptie. Solutia este sa folosim una din urmatoarele:

```
> (define out (open-output-file "output.txt" #:exists 'truncate))
> (display "␣:)␣" out)
> (close-output-port out)
```

sau

```
> (define out (open-output-file "output.txt" #:exists 'update))
> (display "␣:)␣" out)
> (close-output-port out)
```

Care este diferenta dintre truncate si update?

**Exercitiu:**

Creati un fisier “songs.txt” si adaugati urmatoarele trei linii:
(ALL MY LOVING)
(TICKET TO RIDE)
(MARTHA MY DEAR)

```
> (define in (open-input-file "songs.txt"))
> (read in)
> (read in)
> (read in)
> (read in)
> (close-input-port in)
> (read in)
```

### 3.3 Utilizand un fisier ca o baza de date

Nu este foarte interesant sa se citeasca linie cu linie.
Hai sa utilizam o mica baza de date pentru a depista un anumit cantec folosind numere. Definim urmatoarele 2 functii:

```
(define (get-song n)
  (let ((port (open-input-file "songs")))
    (skip-songs (- n 1) port)
    (let ((answer (read-line port)))
      (close-input-port port)
      answer)))

(define (skip-songs n port)
  (if (= n 0)
      'done
      (begin (read-line port)
             (skip-songs (- n 1) port))))

> (get-song 2)
(TICKET TO RIDE)
```

**Dar daca intrebam de un cantec a carui numar este mai mare decat 3?**

In acest caz, read va returna un obiect end-of-file. Urmatorul program citeste tot dintr-un fisier pana la intalnirea eof si printeaza pe ecran rezultatul:

```
(define (print-file name)
  (let ((port (open-input-file name)))
    (print-file-helper port)
    (close-input-port port)
    'done))

(define (print-file-helper port)
  (let ((stuff (read-line port)))
    (if (eof-object? stuff)
        'done
        (begin (show-line stuff)
               (print-file-helper port)))))

> (print-file "songs")
```

ALL MY LOVING
TICKET TO RIDE
MARTHA MY DEAR
DONE

## 3.4 Alinierea textului

Cateva exemple:

- 1) Intr-un fisier avem un text care nu e aranjat si am dori sa aliniem textul in asa fel incat fiecare linie sa aiba aceeasi lungime cu toate celelalte, la fel ca intr-o carte
- 2) Intr-un fisier avem numele si note ale unor studenti si dorim sa facem suma totala a notelor, precum si media fiecarui student
- 3) Intr-un fisier avem numele si prenumele unor studenti si dorim sa rearanjam in forma prenume, nume.

O sa scriem o procedura file-map, care este similara cu map, doar ca lucreaza cu fisiere.

```
(define (file-map fn inname outname)
  (let ((inport (open-input-file inname))
        (outport (open-output-file outname)))
    (file-map-helper fn inport outport)
    (close-input-port inport)
    (close-output-port outport)
    'done))

(define (file-map-helper fn inport outport)
  (let ((line (read-line inport)))
    (if (eof-object? line)
        'done
        (begin (show-line (fn line) outport)
               (file-map-helper fn inport outport)))))
```

**Care e diferenta dintre acest exemplu si cel precedent cu print-file?**

**Exercitiu:**

Creati un fisier de intrare dddbmt care sa contina:
David Harmon
Trevor Davies
John Dymond
Michael Wilson
Ian Amey

iar ca si output trebuie sa obtineti un fisier care sa contina:
Harmon, David
Davies, Trevor
Dymond, John

Wilson, Michael
Amey, Ian

Trebuie scrisa o procedura care primeste ca si parametru un nume si care va returna acelasi nume, dar cu numele si prenumele inversate si adauga o virgula intre ele. Are forma de genul:

```
(define (lastfirst name)
  (se (word (last name) ",") (bl name)))
```

Utilizam butlast (bl) in loc de first in cazul in care mai exista si un al doile prenume. Pentru a utiliza aceasta procedura apelam file-map:

```
> (file-map lastfirst "dddbmt" "dddbmt-reversed")
DONE
```

Pentru a vedea rezultatul pe ecran:

```
> (print-file "dddbmt-reversed")
```

**Exercitiu: media notelor**

Creati un fisier de input care contine:

```
John 88 92 100 75 95
Paul 90 91 85 80 91
George 85 87 90 72 96
Ringo 95 84 88 87 87
```

Output-ul va fi:

```
John total: 450 average: 90
Paul total: 437 average: 87.4
George total: 430 average: 86
Ringo total: 441 average: 88.2
```

Functia pe care trebuie sa o definiti are scheletul de forma:

```
(define (process-grades line)
  (se (first line)
      "total:"
      (accumulate + (bf line))
      "average:"
      (/ (accumulate + (bf line))
         (count (bf line)))))

> (file-map process-grades "grades" "results")

> (print-file "results")
```

**Alinierea textului**

Creati un fisier care sa contina urmatorul text:

```
Programming languages should be designed not by
piling feature on top of feature, but by
removing the weaknesses and restrictions that
make additional features appear necessary.
Scheme demonstrates that a very small number of
rules for forming expressions, with no
restrictions on how they are composed, suffice
to form a practical and efficient programming
language that is flexible enough to support most
of the major programming paradigms in use today.
```

Rezultatul va trebui pus intr-un alt fisier care va contine:

```
Programming languages should be designed not by
piling feature on top of feature, but by
removing the weaknesses and restrictions that
make additional features appear necessary.
Scheme demonstrates that a very small number of
rules for forming expressions, with no
restrictions on how they are composed, suffice
to form a practical and efficient programming
language that is flexible enough to support most
of the major programming paradigms in use today.
```

## 3.5 Scrierea fisierelor pentru citire

- Read ignora spatiile si ne forteaza sa avem paranteze in fisiere.
- Read-line repara aceste probleme, dar pierde spatiile.

Un exemplu in care e mai benefic sa utilizam read. Avem un fisier care pe fiecare linie contine un cantec si albumul corespunzator

```
((love me do) (please please me))
((do you want to know a secret?) (please please me))
((think for yourself) (rubber soul))
((your mother should know) (magical mystery tour))
```

Daca utilizam read-line pentru citirea din fisier pierdem structura listelor; ne va returna pentru prima linie "((love". Read, va avea efectul dorit de noi.

**Care e diferenta dintre write si display?**

```
> (write '(a hard "day's" night))

> (display '(a hard "day's" night))
```

# 4 Concluzii

- Nu uitati dupa ce deschideti un fisier sa-l si inchideti

- Atentie la deschidere si inchidere fisiere cand definiti functii recursive
- Nu puteti citi aceeasi linie de doua ori

# 5 Temă:

- Scrieti o procedura care numara liniile dintr-un fisier. Va avea ca parametru numele fisierului si va returna numarul.
- Scrieti o procedura care numara cate cuvinte sunt intr-un fisier. Va avea ca parametru numele fisierului si va returna numarul.