LISP: Exemple de L.P. Funcional

2721-Llenguatges de Programació

LISP

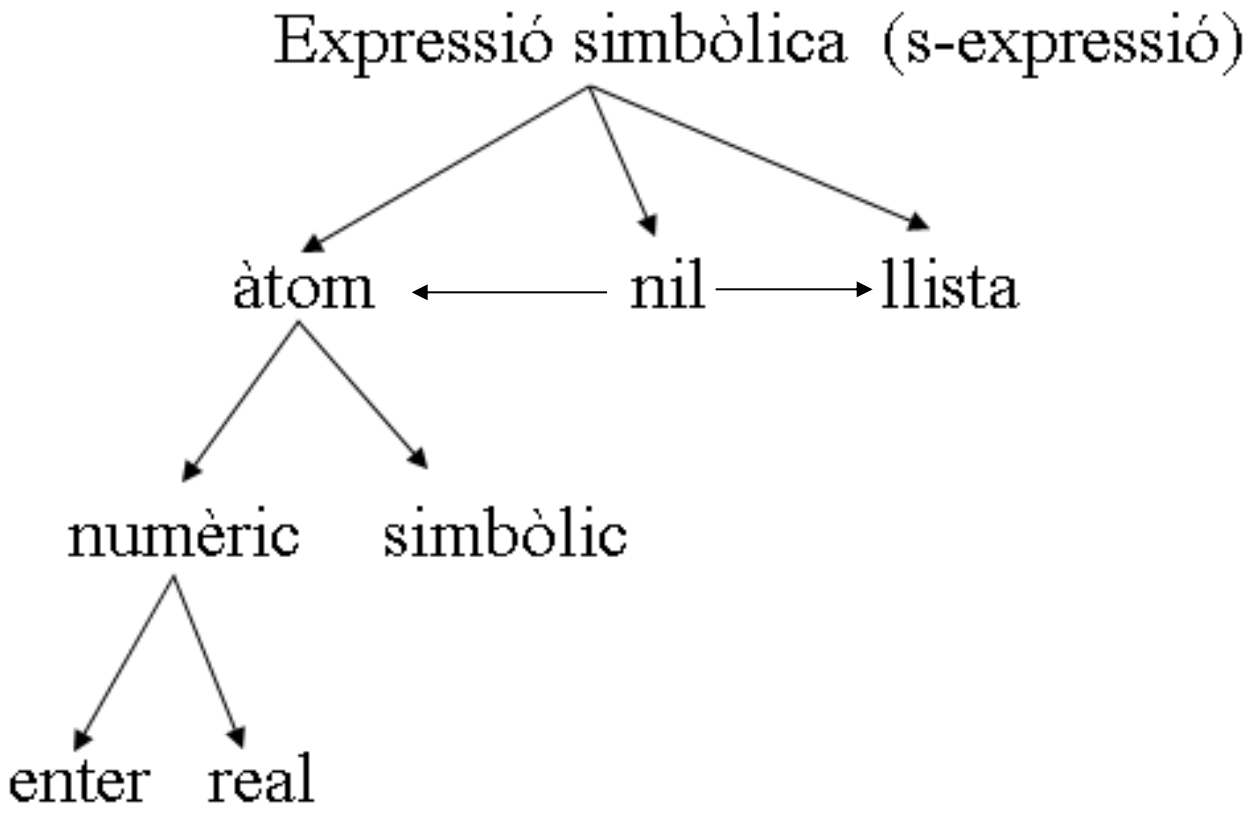
- Són les inicials de **LIS**t **P**rocessing language
- Creat per John McCarthy, al MIT (1958)
- Dissenyat per la Intel·ligència Artificial
- Estudiarem la part funcional de LISP

LISP

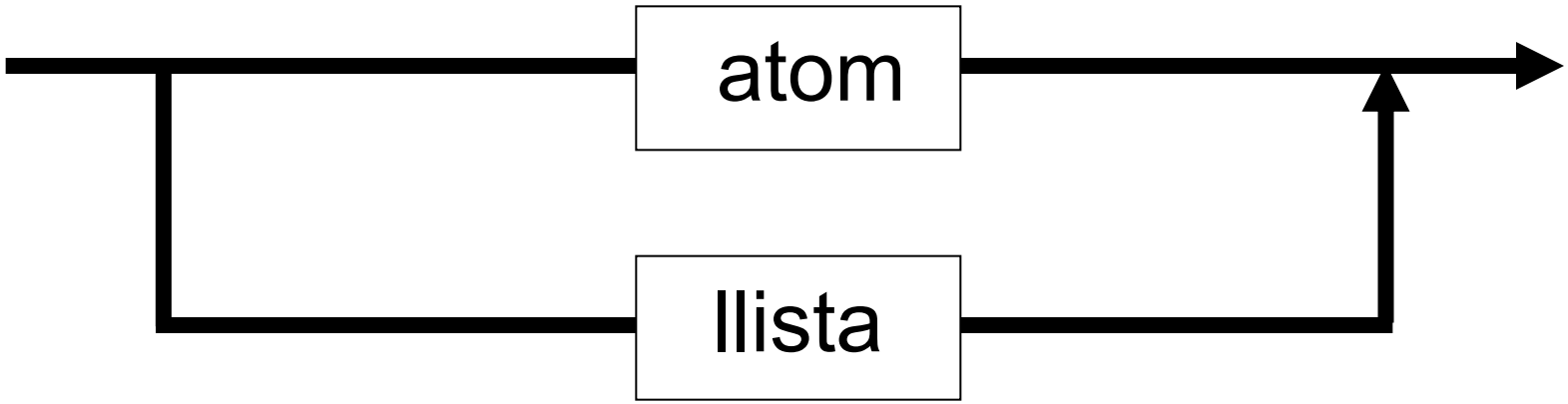
El LISP va introduir:

- L'equivalència entre estructures de dades i programes
- La **recursió** com a única estructura de control d'iteració
- Una estructura de dades bàsica, la **llista**
- La recollida d'escombraries (*garbage collection*)

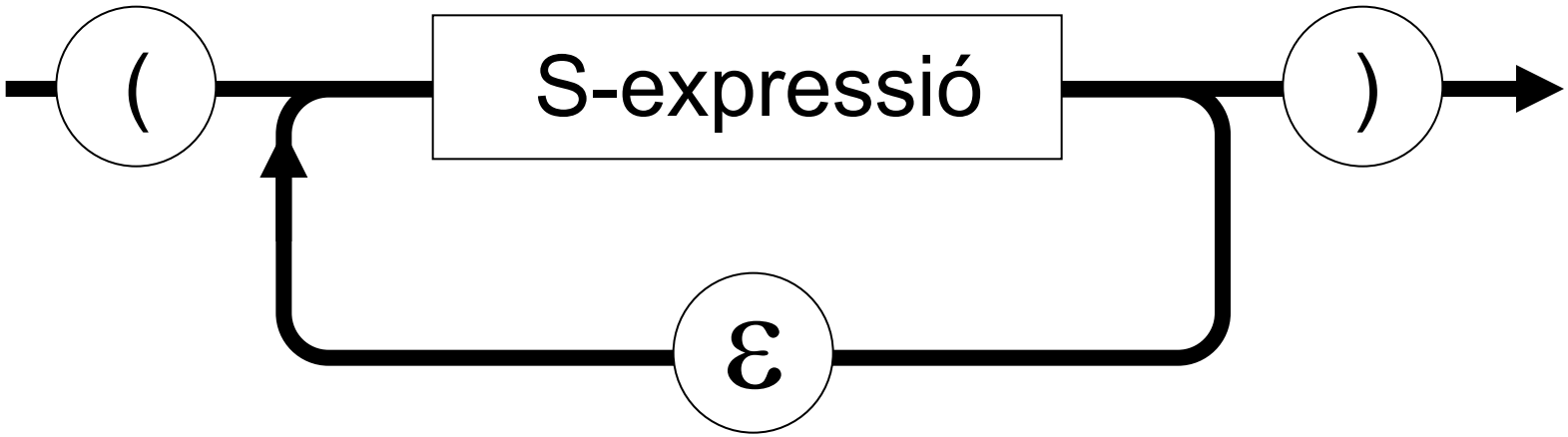
Tipus de dades



S-expressió

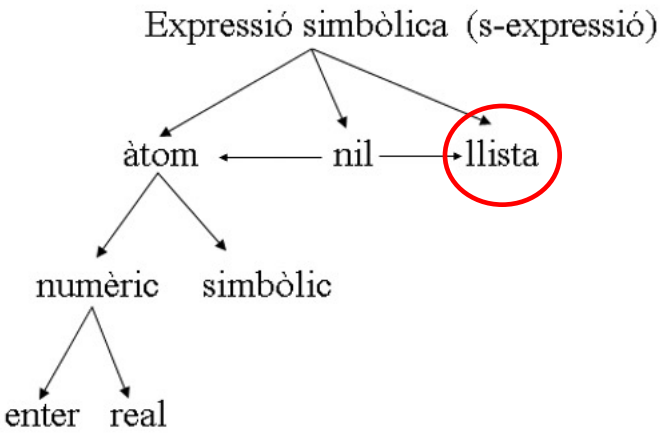


llista



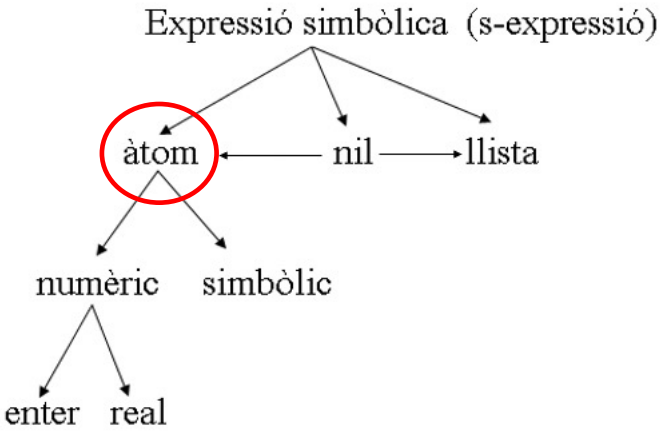
llistes

- (palma inca manacor)
- (a 35 b 44)
- ((antoni comes) (pep llopis))



àtoms

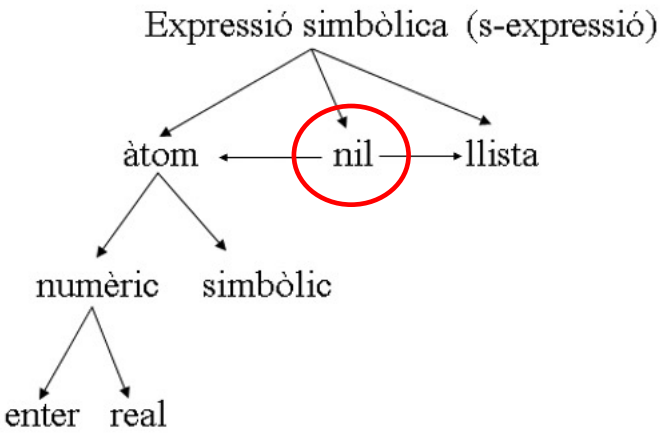
- Els **àtoms simbòlics** (literals) són identificadors (lletres, números i guions començant per una lletra)
- Els **àtoms numèrics** estan formats únicament per xifres (i pot ser el punt decimal)
- Exemples:
  - A Z80 PREU-COST 38 18.3



nil

Pot ser:

- El valor atòmic “fals”
- La llista buida (també representable amb “()”)



Operacions aritmètiques bàsiques

Originàriament:

PLUS, DIFFERENCE, TIMES i QUOTIENT

Ara:

+, -, *, /

Operacions relacionals bàsiques

Originàriament:

ZEROP, GREATERP, LESSP

Ara:

=, <, <=, >, >=

Es conserven:

EQ, EQUAL, NUMBERP, ATOM, LISTP

Operacions booleanes bàsiques

Són el AND, OR i NOT

Veritat es representa amb l'àtom: T

Fals es representa amb l'àtom: nil

Altres funcions: ODDP, EVENP

El procés d'avaluació

Els sistemes de LISP són intèrprets que permeten a l'usuari introduir una expressió, l'intèrpret la llegeix, l'avalua i ens mostra el resultat

Aquest procés anomenat "avaluació" es repeteix continuament

El procés d'avaluació

Com s'avaluen les expressions ?

Un **àtom numèric** avalua a ell mateix

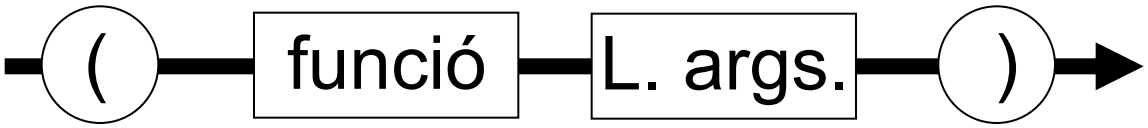
30 → 30

Un **àtom simbòlic** avalua al seu valor

PI → 3.1415926

L'avaluació **d'una llista** segueix el següent procés:

El procés d'avaluació



El primer element de la llista es considera el nom d'una funció definida, la resta d'elements són els arguments d'aquesta funció

Primer s'avaluen els arguments

Després es crida la funció:

(+ 4 3) → 7

El procés d'avaluació

Hi ha dos tipus de llistes: les dades i els programes, però les dues tenen el mateix format

(jo tu ell)

(+ 3 4)

Per evitar que les llistes de dades s'avaluïn hem d'utilitzar l'operador QUOTE ó ‘

El procés d'avaluació

Així doncs, '(+ 3 4) o (quote (+ 3 4)) s'avalua a (+ 3 4) i no a 7

LISP utilitza la **Notació Prefixa de Cambridge**, els operadors van al principi i els operants a continuació separats per espais en blanc

Com s'avalua (+ 5 (* 1 4)) ?

Com funciona l’intèrpret?

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

553 + 47

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

553 + 47 = 600

(+ 553 47)

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

386 – 70 + 38

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

386 – 70 + 38 = 354

(+ (- 386 70) 38)

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

$$4^3$$

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

$4^3 = 64$

(* 4 (* 4 4))

o

(* 4 4 4)

...

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

$$1024 + 2^3 - \left(\frac{27}{3} + 1\right)$$

Quin és el resultat?

Amb l'intèrpret de LISP calculau:

$$1024 + 2^3 - \left(\frac{27}{3} + 1\right) = 1022$$

(+ 1024 (* 2 (* 2 2)) (- (+ (/ 27 3) 1)))

o

(- (+ (* (* 2 2) 2) 1024) (+ (/ 27 3) 1))

...

Manipulació de llistes

Les operacions bàsiques són el CAR i el CDR

- El CAR retorna el primer element d'una llista
- El CDR retorna el resta de la llista (sense el primer element)

(car '(a b c d)) → a (pot ser un àtom o una llista)
(cdr '(a b c d)) → (b c d) (sempre és una llista)

Quin és el car i el cdr d'una llista buida ?

Exercici:

- Obteniu el segon element de la llista (a b c d)

Exercici:

- Obteniu el segon element de la llista (a b c d): b

(car (cdr ‘(a b c d)))

Exercici:

- Obteniu el tercer element de la llista (a b c d)

Exercici:

- Obteniu el tercer element de la llista (a b c d): c

(car (cdr (cdr ‘(a b c d))))

Manipulació de llistes

Podem trobar funcions amb el format CxxxR on x és una A o una D que permeten la composició de CAR i CDR per obtenir el segon, el tercer,...

cadr ≡ (car (cdr l))

caddr ≡ (car (cdr (cdr l)))

Manipulació de llistes

Altres funcions són:

- null : mira si una llista és buida o no
- append: s'utilitza per afegir dues llistes
- member: mira si un element pertany a una llista
- delete: borra un element d'una llista
- reverse: inverteix una llista

Exercici:

- Eliminau l'element *b* de la llista (a b c d)

Exercici:

- Eliminau l'element *b* de la llista (a b c d): (a c d)

(delete 'b '(a b c d))

Exercici:

- Afegiu les llistes (a b) i (a b c d)

Exercici:

- Afegiu les llistes (a b) i (a b c d): (a b a b c d)

(append ‘(a b) ‘(a b c d))

Exercici:

- Afegiu l'element *a* a la llista (b c d)

Exercici:

- Afegiu l'element *a* a la llista (b c d): (a b c d)

(append ‘(a) ‘(b c d))

Construcció de llistes

- Per construir llistes es poden utilitzar el CONS i el LIST
- CONS afegeix un nou element al principi d'una llista donada

  (cons ‘a ‘(b c d)) → (a b c d)

  (cons ‘(1 2) ‘(b c d)) → ((1 2) b c d)
- LIST crea una llista a partir del seus arguments

  (list ‘a ‘b ‘c ‘d) → (a b c d)

Exercici:

- Afegiu l'element *a* a la segona posició de la llista (b c d): (b a c d)

Exercici:

- Afegiu l'element *a* a la segona posició de la llista (b c d): (b a c d)

> (cons (car '(b c d)) (cons 'a (cdr '(b c d))))

Exercici:

- Afegiu l'element *a* a la segona posició de la llista (b c d): (b a c d)

> (cons (car '(b c d)) (cons 'a (cdr '(b c d))))
(b a c d)

> (append (list (car '(b c d)) 'a) (cdr '(b c d)))

Assignacions

- Encara que la seva utilització no és massa aconsellada, es poden donar valors als àtoms directament amb dues funcions: SET i SETQ

- SETQ vol dir “set quote”

```
(setq ciutats ‘(palma inca manacor))
(set ‘ciutats ‘(palma inca manacor))
```

- En aquesta assignatura només es permet la utilització del set i el setq per motius clars d’eficiència, és a dir, assignacions no destructives

Assignacions

Exemple:

(setq ciutats ‘(palma inca manacor))

(set ‘ciutats ‘(palma inca manacor))

Les dues són equivalents i l’avaluació és

ciutats →(palma inca manacor)

Diferència EQUAL i EQ

(set ‘l1 ‘(a b c)) → (a b c)

(set ‘l2 l1) → (a b c)

(set ‘l3 ‘(a b c)) → (a b c)

(eq l1 l2) →	(equal l1 l2) →
(eq l1 l3) →	(equal l1 l3) →
(eq l2 l3) →	(equal l2 l3) →

Diferència EQUAL i EQ

(set ‘l1 ‘(a b c)) → (a b c)

(set ‘l2 l1) → (a b c)

(set ‘l3 ‘(a b c)) → (a b c)

(eq l1 l2) → t	(equal l1 l2) → t
(eq l1 l3) → nil	(equal l1 l3) → t
(eq l2 l3) → nil	(equal l2 l3) → t

Definició de noves funcions

Per definir noves funcions s'utilitza l'identificador **defun**:

(defun nom-de-funció parametres expressió)

exemple:

(defun suma1 (x) (+ x 1))

(suma1 4) → 5

Exercici

Escriu la funció *quadrat*, que calculi el quadrat d'un nombre:

```
> (quadrat 3)
9
```

Exercici

Escriu la funció *quadrat*, que calculi el quadrat d'un nombre:

```
>(defun quadrat (n) (* n n))

> (quadrat 4)
16
```

Exercici

Escriu la funció *suma3*, que calculi la suma de tres nombres:

```
> (suma3 4 5 6)
15
```

Exercici

Escriu la funció *suma3*, que calculi la suma de tres nombres:

```
>(defun suma3 (x1 x2 x3)
     (+ x1 (+ x2 x3)))

> (suma3 4 5 5)
14
```

Exercici

Escriu la funció *tercer*, que retorni el tercer element d'una llista:

```
> (tercer '(a b c))
c
```

Exercici

Escriu la funció *tercer*, que retorni el tercer element d'una llista:

```
>(defun tercer (l)
      (car (cdr (cdr l))))

> (tercer '(a b c))
c
```

Exercici

Escriu la funció *darrer*, que retorni el darrer element d'una llista:

```
> (darrer '(a b c d))
d
```

Exercici

Escriu la funció *darrer*, que retorni el darrer element d'una llista:

```
> (defun darrer (l) (car (reverse l)))
darrer

> (darrer '(a b c d))
d
```

Exercici

Escriu la funció *darrer*, que retorni el darrer element d'una llista (**sense utilitzar el *reverse***):

> (darrer '(a b c d))

d

Recursivitat

L'estructura per fer iteracions en LISP és la recursivitat

>(darrer '(a b c d)) → d

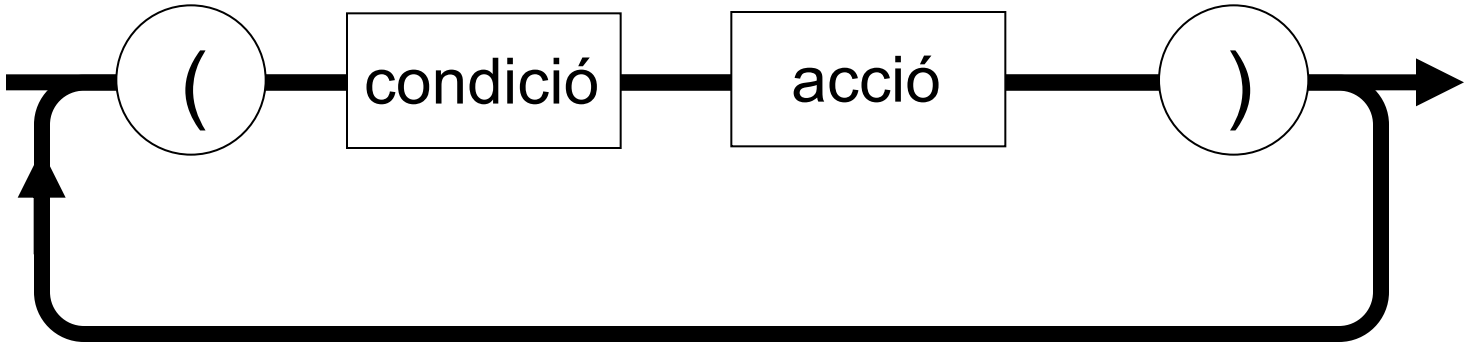
Recursivitat

Però per poder definir recursivament una funció necessitarem una instrucció condicional que ens permeti determinar quan s'acaba la recursivitat

Expressions condicionals

Cond

Condicions-accions



Expressions condicionals

Exemple:

```
(cond       (condició-1 acció-1)
            (condició-2 acció-2)
            ...
            (condició-i  acció-i)        )
```

s'avalua l'acció corresponent a la primera condició que es verifiqui

És un bon costum que la darrera condició sigui "t", per assegurar que al menys una de les condicions es compleix (equivalent a un default)

Exercici

Escriu la funció *darrer*, que retorni el darrer element d'una llista (sense utilitzar el *reverse*):

```
> (darrer '(a b c d))
d
```

Solució

Escriu la funció *darrer*, que retorni el darrer element d'una llista:

```
(defun darrer (L)
    (cond ((null (cdr L)) (car L))
          (t (darrer (cdr L)))))
```

Exercici

exemple: càlcul de la longitud d'una llista

(long '(a b c d)) → 4

Solució

exemple: càlcul de la longitud d'una llista

```
(defun long (l)
        (cond   ((null l) 0)
                (t (+ 1 (long (cdr l))))))
```

Exercici

- Definir la funció pertany que faci el mateix que member

  (pertany ‘a ‘(b a c)) →T

Solució

- Definir la funció pertany que faci el mateix que member

```
(defun pertany (x l)
        (cond ((null l) nil)
                ((equal x (car l)) t)
                (t (pertany x (cdr l)))))
```

Exercici

- Definir la funció exponenciació

  (exp 2 3) →8

Solució

- Definir la funció exponenciació

```
(defun exp (m n)
        (cond ((= n 0) 1)
                (t (* m (exp m (- n 1))))))
```

Exercici

- Escriure la funció fibonacci que retorna l'enèsim valor d'aquesta sèrie

  (fib 6) → 8

$$F(n) = \begin{cases} 0, & \text{si } n = 0; \\ 1, & \text{si } n = 1; \\ F(n-1) + F(n-2) & \text{altrament.} \end{cases}$$

Solució

- Escriure la funció fibonacci que retorna l'enèsim valor d'aquesta sèrie

```
(defun fib (n)
        (cond  ((= n 0) 0)
               ((= n 1) 1)
               (t (+ (fib (- n 1)) (fib (- n 2))))))
```

Exercici

- Escriure la funció dividir, que fa la divisió entera de M entre N (tots dos positius). Fixau-vos que aquesta funció dóna el número de vegades que N està contingut dins M

  (dividir 9 2) → 4

Solució

- Escriure la funció dividir, que fa la divisió entera de M entre N (tots dos positius). Fixau-vos que aquesta funció dóna el número de vegades que N està contingut dins M

```
(defun dividir (m n)
    (cond ((< m n) 0)
          (t (+ 1 (dividir (- m n) n)))))
```

Exercici

- Escriure també la funció parell a partir de dividir. Aquesta funció ens diu si un número és parell o no (igual que evenp)

  (parell 9) → nil

  (parell 6) → t

Solució

- Escriure la funció parell a partir de dividir. Aquesta funció ens diu si un número és parell o no (igual que evenp)

```
(defun parell (n)
        (cond  ((= (* 2 (dividir n 2)) n) t)
               (t nil)))
```

Exercici

- Escriure una funció recursiva "senars" que donada una llista de números retorna una llista amb tots els números parells eliminats

  (senars '(3 1 8 7 4 10)) → (3 1 7)

Solució

- Escriure una funció recursiva "senars" que donada una llista de números retorna una llista amb tots els números parells eliminats

```
(defun senars (l)
    (cond ((null l) nil)
          ((not (parell (car l))) (cons (car l) (senars (cdr l))))
          (t (senars (cdr l)))))
```

Exercici

- Definir la funció borra per eliminar la primera aparició d'un element dins una llista

  (borra 'a '(1 2 a 3 a b c)) → (1 2 3 a b c)

Solució

- Definir la funció borra per eliminar la primera aparició d'un element dins una llista

```
(defun borra (x l)
            (cond  ((null l) nil)
                   ((equal x (car l)) (cdr l))
                   (t (cons (car l) (borra x (cdr l))))))
```

Pregunta

- Com es modificaria la funció borra si en volem eliminar totes les aparicions ?

```
(defun borra (x l)
            (cond  ((null l) nil)
                   ((equal x (car l)) (cdr l))
                   (t (cons (car l) (borra x (cdr l))))))
```

Pregunta

- Com es modificaria la funció borra si en volem eliminar totes les aparicions ?

```
(defun borra (x l)
            (cond  ((null l) nil)
                   ((equal x (car l)) (borra x (cdr l)))
                   (t (cons (car l) (borra x (cdr l))))))
```

Exercici

- Definir la funció rdc que torna tots els elements d'una llista menys el darrer

  (rdc '(a b c)) → (a b)

Solució

- Definir la funció rdc que torna tots els elements d'una llista menys el darrer

```
(defun rdc (l)
        (cond  ((null (cdr l)) nil)
               (t (cons (car l) (rdc (cdr l)))))))
```

Exercici

- Definir la funció snoc que afegeix un element al final d'una llista

  (snoc 'a '(b c)) → (b c a)

Construcció/manipulació de llistes

- CONS afegeix un nou element al principi d'una **llista** donada

  (cons ‘a ‘(b c d)) → (a b c d)

  (cons ‘(1 2) ‘(b c d)) → ((1 2) b c d)
- LIST crea una llista a partir del seus arguments

  (list ‘a ‘b ‘c ‘d) → (a b c d)
- APPEND afegeix dues llistes

  (append ‘(a b) ‘(c d)) → (a b c d)

Solució

- Definir la funció snoc que afegeix un element al final d'una llista

```
(defun snoc (x l)
        (cond  ((null l) (cons x nil))
               (t (cons (car l) (snoc x (cdr l))))))
```

Solució

- Definir la funció snoc que afegeix un element al final d'una llista:

```
(defun snoc (x l)
        (cond  ((null l) (list x))
               (t (cons (car l) (snoc x (cdr l))))
        )
)
```

Exercici

- Definir la funció escala per multiplicar tots els elements d'una llista per un número

  (escala 2 '(3 5 6)) → (6 10 12)

Solució

- Definir la funció escala per multiplicar tots els elements d'una llista per un número

```
(defun escala (x l)
        (cond ((null l) nil)
                (t  (cons (* x (car l))
                          (escala x (cdr l))))))
```

Exercici

- Definir les funcions màxim i mínim d'una llista de números

  (maxim '(4 1 3 8 5)) → 8

  (minim '(4 1 3 8 5)) → 1

Solució

- Definir les funcions màxim i mínim d'una llista de números

```
(defun maxim (l)
        (cond   ((null (cdr l)) (car l))
                ((>= (car l) (maxim (cdr l))) (car l))
                (t (maxim (cdr l)))))
```

Solució

- Definir les funcions màxim i mínim d'una llista de números

```
(defun minim (l)
        (cond   ((null (cdr l)) (car l))
                ((< (car l) (minim (cdr l))) (car l))
                (t (minim (cdr l)))))
```

Exercici

- Escriure una funció per ordenar una llista de números amb el mètode de selecció directa (trobar el mínim a cada passa i posar-ho al principi)

  (ordena ‘(4 1 3 8 5)) → (1 3 4 5 8)

Solució

- Escriure una funció per ordenar una llista de números amb el mètode de selecció directa (trobar el mínim a cada passa i posar-ho al principi)

```
(defun ordena (l)
        (cond   ((null l) nil)
                (t (cons (minim l)
                         (ordena (borra (minim l) l))))))
```

Exercici

- Escriure la funció invertir que donada una llista la gira al revés (sense utilitzar reverse)

  (invertir ‘(a e i o u)) → (u o i e a)

Solució

- Escriure la funció invertir que donada una llista la gira al revés (sense utilitzar reverse)

```
(defun invertir(l)
      (cond  ((null l) nil)
                  ((null (cdr l)) (list (car l)))
                  (t (append (invertir (cdr l)) (list (car l)) ))
  )
)
```

Solució

- Escriure la funció invertir que donada una llista la gira al revés (sense utilitzar reverse)

```
(defun invertir (l)
        (cond ((null l) nil)
                (t (snoc (car l) (invertir (cdr l))))))
```

Exercici

- Escriure una funció per borrar l'enèsim element d'una llista

  (borrar 3 '(a e i o u)) → (a e o u)

Solució

- Escriure una funció per borrar l'enèsim element d'una llista

```
(defun borrar (n l)
        (cond   ((= n 1) (cdr l))
                (t (cons (car l)
                (borrar (- n 1) (cdr l))))))
```

Exercici

- Escriure una funció que compti el número de vegades que una expressió apareix a una llista

  (vegades 'a '(1 2 a 3 (a b) a)) → 2

  (vegades 'x '(s t (x) 3)) → 0

Solució

- Escriure una funció que compti el número de vegades que una expressió apareix a una llista

```
(defun vegades (x l)
    (cond ((null l) 0)
          ((equal x (car l)) (+ 1 (vegades x (cdr l))))
          (t (vegades x (cdr l)))))
```

Exercici

- Escriure una funció que compti el número total d'àtoms que hi ha dins una llista

  (atoms '(a (b (c d) e) f g (h i))) → 9

Solució

- Escriure una funció que compti el número total d'àtoms que hi ha dins una llista

```
(defun atoms (l)
    (cond ((null l) 0)
          ((listp (car l)) (+ (atoms (car l))
                              (atoms (cdr l))))
          (t (+ 1 (atoms (cdr l))))))
```

Exercici

- Escriure una funció que donada una llista retorni la mateixa llista sense els n primers elements

  (treuprimers 3 ‘(a e i o u)) → (o u)

Solució

- Escriure una funció que donada una llista retorni la mateixa llista sense els n primers elements

```
(defun treuprimers (n l)
        (cond ((= n 0) l)
                (t (treuprimers (- n 1) (cdr l)))))
```

Exercici

- Escriure una funció que donada una llista torni els primers n elements

  (tornaprimers 3 ‘(a e i o u)) → (a e i)

Solució

- Escriure una funció que donada una llista torni els primers n elements

```
(defun tornaprimers (n l)
    (cond ((= n 0) nil)
          (t (cons (car l)
                   (tornaprimers (- n 1) (cdr l))))))
```

Exercici

- Escriure una funció per insertar un element a la posició enèsima d'una llista

  (inserta 'aqui 3 '(i jo que faig)) → (i jo aqui que faig)

Exercici

- Escriure una funció per canviar l'enèsim element d'una llista per l'element donat

  (canviar 3 '(a e i o u) 'mig) → (a e mig o u)

Exercici

- Escriure les funcions necessàries per insertar un element dins una llista:
  - a l'esquerra d'un element donat

    (inserta-esquerra 'e 'a '(o r d e n a d o r)) → (o r d a e n a d o r)

  - a la dreta d'un element donat

    (inserta-dreta 'e 'a '(o r d e n a d o r)) → (o r d e a n a d o r)

  - en substitució d'un element donat

    (substituir 'e 'a '(o r d e n a d o r)) → (o r d a n a d o r)

Exercici de conjunts

- Donats dos conjunts (representats per llistes d'elements), escriure les funcions:

(conjunt-correcte '(a b c d)) → t
(conjunt-correcte '(a b c c d a)) → nil
(unio '(a b c) '(a d e c)) → (a b c d e)
(interseccio '(a b c) '(a d e c)) → (a c)
(diferencia '(a b c) '(a d e c)) → (b)
(diferencia-simetrica '(a b c) '(a d e c)) →'(b d e)
(producte-cartesia '(a b) '(d e)) →((a d) (a e) (b d) (b e))

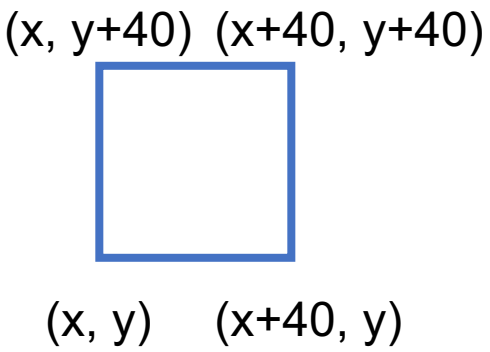
Enunciat de la pràctica

Exemple funcions de dibuix

- Com pintaríem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.

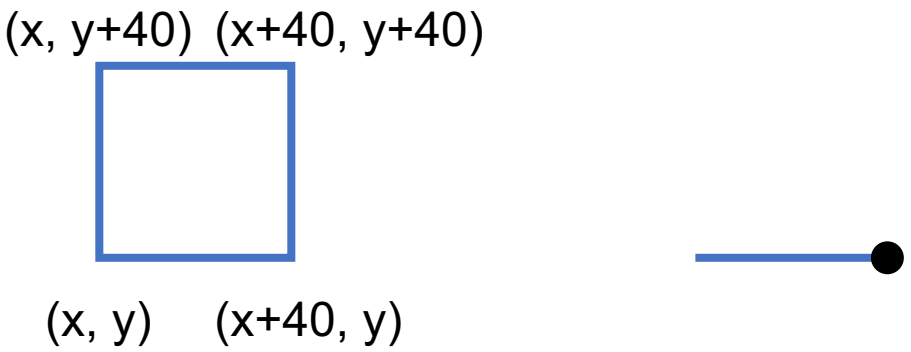
Exemple funcions de dibuix

- Com pintaríem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.



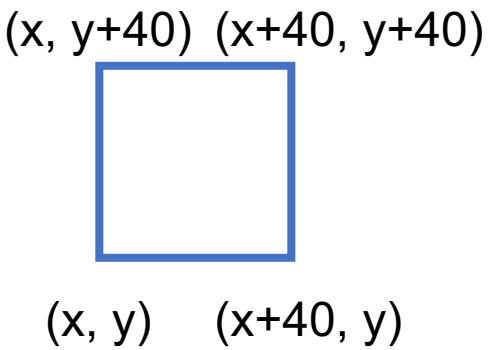
Exemple funcions de dibuix

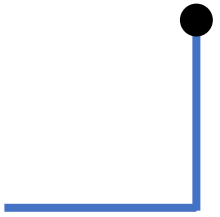
- Com pintaríem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.



Exemple funcions de dibuix

- Com pintariem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.

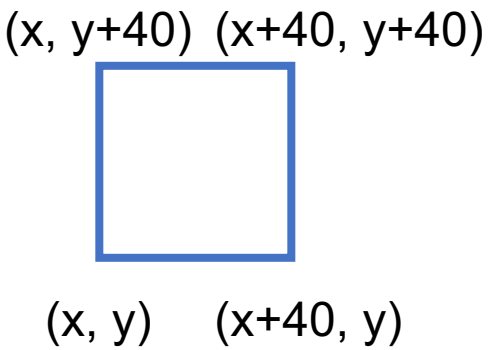


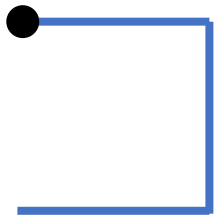


(drawrel 40 0)
(drawrel 0 40)

Exemple funcions de dibuix

- Com pintariem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.

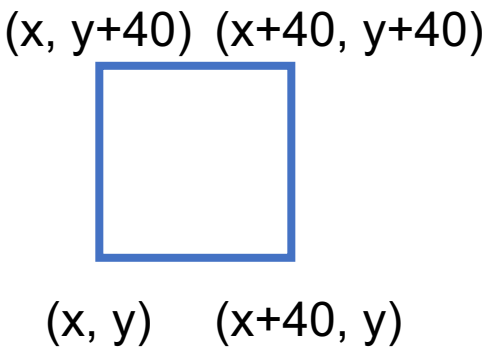


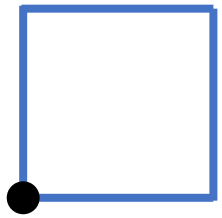


(drawrel 40 0)
(drawrel 0 40)
(drawrel -40 0)

Exemple funcions de dibuix

- Com pintariem un quadrat de 40x40?
  - **(move x y)** : mou el punt on es troba el llapis a la posició (x,y).
  - **(moverel x y)** : mou el punt on es troba el llapis x píxels horitzontalment i y píxels verticalment sense pintar.
  - **(draw x y)**: desplaça el llapis d'on està a la posició (x, y) pintant tot el recorregut.
  - **(drawrel x y)** : desplaça el llapis x píxels horitzontalment i y verticalment, pintant tot el recorregut.





(drawrel 40 0)
(drawrel 0 40)
(drawrel -40 0)
(drawrel 0 -40)

Exemple funcions dibuix

```
(defun quadrat (l)
        (drawrel l 0)
        (drawrel 0 l)
        (drawrel (- 0 l) 0)
        (drawrel 0 (- 0 l))
)
```

Exemple funcions dibuix

- Què que fa aquesta funció?

```
(defun quadrats (n d1 d2)
        (cond       ((= n 0) nil)
                    (t      (moverel d1 d2)
                            (quadrat 40)
                            (quadrats (- n 1) d1 d2))
        )
)
```

Exemple funcions dibuix

- Què que fa aquesta funció?

```
(defun quadrats (n d1 d2)
        (cond       ((= n 0) nil) ; n vegades
                    (t      (moverel d1 d2) ; es desplaça d1 d2
                            (quadrat 40) ; pinta quadrat de 40x40
                            (quadrats (- n 1) d1 d2))
        )
)
```

Exemple funcions dibuix

- Què que fa aquesta funció?

```
(defun m ()
        (cls)
        (move 150 100)
        (quadrats 20 4 4)
        (quadrats 20 4 -4)
        (quadrats 20 4 4)
        (quadrats 20 4 -4)
)
```

Exemple funcions dibuix

- Què que fa aquesta funció?

```
(defun m ()
        (cls) ; esborra la pantalla
        (move 150 100) ; es mou a la posició (150, 100)
        (quadrats 20 4 4) ; 20 quadrats diagonal per amunt
        (quadrats 20 4 -4) ; 20 quadrats diagonal per avall
        (quadrats 20 4 4) ; 20 quadrats diagonal per amunt
        (quadrats 20 4 -4) ; 20 quadrats diagonal per avall
)
```

Exemple llista de propietats

- Per a què les podem utilitzar?

```
>(setprop ‘cercle ‘(255 0 0) ‘color)        ; posa al valor de la
                                            ; propietat color el codi
                                            ;del vermell

>(color (get ‘cercle ‘color)) ?    és correcte?
```

Exemple llista de propietats

- Per a què les podem utilitzar?

```
>(setprop ‘cercle ‘(255 0 0) ‘color)        ; posa al valor de la
                                            ; propietat color el codi
                                            ;del vermell

>(color (get ‘cercle ‘color)) ?    ; és correcte?  NO
                          ;  (color ‘(255 0 0)) !!!
                          ;  volem: (color 255 0 0)
                          ;  com es pot fer ?
```

Exemple llista de propietats

- Per a què les podem utilitzar?

```
>(putprop ‘cercle ‘(255 0 0) ‘color)        ; posa al valor de la
                                            ; propietat color el codi
                                            ;del vermell

>(color (get ‘cercle ‘color)) ?    ; és correcte?  NO
                          ;  (color ‘(255 0 0)) !!!
                          ;  volem: (color 255 0 0)
                          ;  com es pot fer ?
>(eval (cons ‘color (get ‘cercle ’color)))  ; així sí!
```

Funcions útils del sistema

>(load ‘nom) ; nom és un arxiu amb extensió .lsp (nom.lsp)

>(savefun nom) ; guarda la funció dins un arxiu amb el mateix nom

>(room) ; mostra la memòria disponible

>(expand n) ; incrementa el número de segments de memòria

>(alloc n m) ; incrementa el número de nodes i apuntadors

>(gc) ; activa el garbage collector

>(dribble ‘nom) ; guarda tot el que es va fent dins un arxiu (nom)

>(dribble) ; acaba el dribble

LISP: Exemple de L.P. Funcional

2721-Llenguatges de Programació