Dpto. de Álgebra, Computación, Geometría y Topología
Universidad de Sevilla

Implementación de algoritmos y cálculo simbólico

(Curso 90–91)

José A. Alonso Jiménez

Sevilla, 1990

Contenido

Capítulo 1

Cálculo aritmético en Lisp

1.1 Operaciones aritméticas

1.1.1 Calcular en LISP:

1. $3 + 5$
2. $2 + 3 + 4$
3. $20/5$
4. $5/20$
5. $(24/3)/2$
6. $1/0.5$

1.1.2 Calcular en LISP:

1. $(1 - 2)(4 + 5)$
2. $(15 + 5) + (100 - 45)$

1.1.3 Calcular en LISP:

$$\frac{2(4-1)6}{18} + (8-6)7$$

1.1.4 Calcular el valor de las siguientes expresiones:

```
(+)
(+ 3)
(+ 3 7 5)
(+ 32000 32000)
(+ 32000.0 32000)

(- 3)
(- 123 7 5)

(ABS 3)
(ABS -3.6)

(*)
(* 3)
(* 3 7 5)
(* 32000 32000)
(* 32000.0 32000)

(/ 6 2)
(/ 5 2)

(/ 2)
(/ 0.5)

(MOD 7 2)

(MAX 1 2 3 4 5 2)
(MAX -2.3 7 0)

(MIN 3)
(MIN 1 2 3 4 5 2)
(MIN -2.3 7 0)
```

1.2 Asignación de valores a variables

1.2.1 Escribir el resultado de las siguientes formas:

```
(SETQ PRECIO 80)
(SETQ IVA 10)
PRECIO
```

```
(/ (* PRECIO IVA) 100)
```

1.2.2 Escribir el resultado de las siguientes formas:

```
(+ 1 2 X 3)
(SETQ X (* 2 5))
(+ 1 2 X 3 )
(SETQ X 1 Y 2 Z (+ X Y))
(+ X Y Z)
```

1.2.3 Calcular el valor de C después de la evaluación de la expresión

```
(SETQ A 3 C (+ (SETQ B 4) (+ A B)))
```

1.2.4 Calcular el valor de las siguientes expresiones:

```
(SETQ X 3)
(INCF X)
X
(DECF X 3)
X
```

1.3 Definición de nuevas funciones

1.3.1 Definir de la función

```
(CUBO X)
```

de forma que devuelva el valor del cubo de `X`. Por ejemplo,

```
(CUBO 2)        =>   8
(CUBO (+ 2 3))  =>  25
```

1.2.2 Definir de la función

```
(SUMA-BINARIA X Y)
```

que devuelva la suma del valor de `X` y el de `Y`. Por ejemplo,

```
(SUMA-BINARIA 2 3)                 =>  5
(SUMA-BINARIA 2 (* 2 3))           =>  8
(* (SUMA-BINARIA (+ 50 50) 1) 3)  =>  303
```

1.3.3 Calcular el valor de las siguientes expresiones:

```
(DEFUN NUMERO () (SETQ X (1+ X)))
(SETQ X 0)
(+ (NUMERO) (NUMERO) (NUMERO))
X
```

1.3.4 Calcular el valor de las siguientes expresiones:

```
(SETQ DOS 2)
(DEFUN DOS () 3)
(+ DOS (DOS))
```

1.4 Programación descendente

1.4.1 Definir la función

```
(MEDIA-CUADRADO X Y)
```

que devuelva la media de los cuadrados de `X` e `Y`. Por ejemplo,

```
(MEDIA-CUADRADO 2 4)  =>  10.0
```

1.4.2 Definir la función

```
(CUADRADO-MEDIA X Y)
```

que devuelva el cuadrado de la media de `X` e `Y`. Por ejemplo,

```
(CUADRADO-MEDIA 2 4)  =>  9.0
```

1.4.3 Escribir una función

```
(P2 A B C X)
```

que devuelva el valor del polinomio $ax^2 + bx + c$. Por ejemplo,

```
(P2 4 3 2 2)  =>  24
```

1.4.4 Definir, utilizando la función `P2`, la función

```
(P4 X)
```

que calcule el valor del polinomio $3x^4 + 7x^2 + 4$.

1.5 Las variables locales

1.5.1 Calcular el valor de las siguientes expresiones:

```
(SETQ X 1)
(LET ((X 2) (Y 3)) (+ X Y))
X
```

1.5.2 Definir la función

```
(F1 b c)
```

que calcule la media de los cuadrados de las raíces de la ecuación a $x^2 + bx + c = 0$. Por ejemplo,

```
(F1 1 -5 6)  =>  6.5
```

Capítulo 2

El cálculo simbólico

2.1 Los objetos básicos

2.1.1 Determinar si los siguientes objetos son números, símbolos, listas o no son expresiones Lisp:

```
123
ATOMO
(ESTO ES UN ATOMO)
((A B) (C D)) 3 (3)
(LISTA 3)
(/ (+ 3 1) (- 3 1))
)(
((()))
(() ())
((ABC
```

2.1.2 Determinar el número de elementos y la profundidad de las siguientes listas:

```
(1 DOS 3 CUATRO)
(SOCRATES (- 43 1))
((X 1) (Y 2) (Z 3))
(- (* 2 (+ A 3)))
(((1)))
(DEFUN SUMA (X Y) (+ X Y))
()
```

2.2 La función QUOTE

2.2.1 Determinar el valor de las siguientes expresiones:

```
A
(SETQ A 3)
(QUOTE A)
'A
```

2.2.2 Determinar el valor de las siguientes expresiones:

```
(A B C)
(QUOTE (A B C))
'(A B C)
(+ 2 3)
'(+ 2 3)
```

2.3 Las funciones SET y SETQ

2.3.1 Determinar el valor de las siguientes expresiones:

```
(SETQ HOMBRE 'SOCRATES)
HOMBRE
(SETQ SOCRATES 1)
HOMBRE
(SETQ MORTAL HOMBRE)
MORTAL
(SET 'HOMBRE 'PLATON)
HOMBRE
MORTAL
(SET HOMBRE 'SOCRATES)
HOMBRE
PLATON
```

2.4 Funciones de búsqueda en listas

2.4.1 Determinar el valor de las siguientes expresiones:

```
(CAR '(A B C))
```

```
(CAR '((D E) F (G H)))
(SETQ L '(10 20))
(CAR L)
(CAR (SETQ M '(+ 1 2)))
(CDR ' (A B C))
(CDR '((D ((E F))) G (H I)))
(CDR L)
(CDR M)
(CAR (CDR M))
(CAR (CDR (CDR M)))
(CDR (CAR (CDR (CAR '((D (E F)) G (H I))))))
```

2.4.2 Definir las funciones

```
(SEGUNDO L)
(TERCERO L)
```

que devuelven el segundo o tercer elemento de la lista `L`. Por ejemplo,

```
(SEGUNDO '(A B C D))  => B
(TERCERO '(A B C D))  =>  C
```

2.4.3 Determinar el valor de las siguientes expresiones:

```
(SETQ M '(A (B C) D E))
(CADR M)
(CADDR M)
```

2.4.4 Escribir los árboles binarios de la siguientes listas:

```
((A B) C (D E))
(DEFUN SUMA (X Y) (+ X Y))
```

y encontrar los `CADDR` y `CADADDR` de dichas listas.

2.4.5 Determinar el valor de las siguientes expresiones:

```
(SETQ L '(A (B C) D E))
(NTH 0 L)
(NTH 2 L)
(NTH 8 L)
(NTHCDR 0 L)
```

```
(NTHCDR 2 L)
(NTHCDR 8 L)
(LAST L)
(LENGTH L)

(SETQ L '(DEFUN SUMA (X Y) (+ X Y)))
(NTH 2 L)
(NTHCDR 2 L)
(LAST L)
(LENGTH L)
```

2.4.6 Escribir sucesiones de `CAR` y `CDR` para obtener DINERO de las siguientes expresiones:

```
(AGUA PAPEL DINERO TIERRA)
((AGUA PAPEL) DINERO TIERRA)
((AGUA PAPEL) (DINERO TIERRA))
(AGUA (PAPEL DINERO) TIERRA)
((AGUA PAPEL) ((DINERO) TIERRA))
```

2.4.7 Definir la función

```
(ULTIMO L)
```

que devuelva el último elemento de la lista `L`. Por ejemplo,

```
(ULTIMO '(A B C))  =>  C
```

2.4.8 Definir la función

```
(PENULTIMO L)
```

que devuelva el penúltimo elemento de la lista `L`. Por ejemplo,

```
(PENULTIMO '(A B C))  =>  B
```

2.4.9 Definida la variable `AGENDA` por

```
(SETQ AGENDA '((PEPE SIERPES-12 "531420")
               (JUAN CUNA-5 "243568")
               (PABLO CERVANTES-45 "456345")))
```

Definir las funciones

```
(NOMBRE N)
(DIRECCION N)
(TELEFONO N)
```

de forma que devuelvan el nombre, la dirección o el teléfono n-ésimo de la lista. Por ejemplo,

```
(NOMBRE 1)
(DIRECCION 2)
(TELEFONO 3)
```

2.5 Funciones de construcción de listas

2.5.1 Determinar el valor de las siguientes expresiones:

```
(CONS 'A '(B C D))
(CONS '(A B) '(C D))
(CONS 'JB (CONS 0 (CONS 0 (CONS 7 NIL))))
(CONS NIL NIL)
(CONS (CAR '(A B)) (CDR '(B C)))
```

2.5.2 Determinar el valor de las siguientes expresiones:

```
(SETQ L (CONS (- 5 3) '(ES UN NUMERO)))
(CONS '7+ L)
L
```

2.5.3 Determinar el valor de las siguientes expresiones:

```
(LIST 'A 'B 'C)
(LIST 'A 1 'B 2)
(SETQ L (LIST 'EL 'NUMERO 3))
(LIST (CAR L) 'DIGITO (CADDR L))
(LIST '* (+ 2 3) '(/ X 2))
(LIST NIL)
```

2.5.4 Determinar el valor de las siguientes expresiones:

```
(APPEND '(A B) '(C D))
(APPEND '(1 2) ' (3 4) '(5 6 7))
(APPEND (LIST '* (+ 2 3)) '(4 5))
(APPEND NIL '(A B))
```

2.5.5 Determinar el valor de las siguientes expresiones:

```
(CONS '(A B) '(C D))
(LIST '(A B) '(C D))
(APPEND '(A B) '(C D))
```

2.5.6 Definir la función

```
(ROTAR-IZQ L)
```

de forma que devuelva la lista obtenida pasando el primer elemento de `L` al último lugar. Por ejemplo,

```
(ROTAR-IZQ '(A B C D))
```

2.5.7 Definir la función

```
(PASA-1-A-3 L)
```

de forma que coloque el primer elemento de la lista `L` en tercer lugar. Por ejemplo,

```
(PASA-1-A-3 '(A B C D))
```

2.6 Funciones de modificación física de listas

2.6.1 Determinar el valor de las siguientes expresiones:

```
(SETQ L '(Y Z))
(PUSH 'X L)
L
```

2.6.2 Determinar el valor de las siguientes expresiones:

```
(SETQ A '(X Y Z))
(POP A)
A
```

Capítulo 3

El control

3.1 Funciones de comparación de números

Las funciones =, /=, >=, >, <=, <

3.1.1 Calcular el valor de las siguientes expresiones:

```
(= 10 (+ 3 7))
(= 2 2.0 (+ 1 1))
(= 1 2 3)
(= 1 2 1)
(/= 10 (+ 3 7))
(/= 2 2.0 (+ 1 1))
(/= 1 2 3)
(/= 1 2 1)
(>= 4 3 3 2)
(>= 4 3 3 5)
(> 4 3 2 1)
(> 4 3 3 2)
(<= 2 3 3 4)
(<= 5 3 3 4)
(< 1 2 3 4)
(< 1 3 3 4)
```

Las funciones ZEROP, PLUSP, MINUSP, EVENP, ODDP

3.1.2 Calcular el valor de las siguientes expresiones:

```
(ZEROP (- 2 2))
(ZEROP (+ 2 2))
(PLUSP (+ 2 3))
(PLUSP (- 2 3))
(MINUSP (+ 2 3))
(MINUSP (- 2 3))
(EVENP (+ 2 3))
(EVENP (+ 2 4))
(ODDP (+ 2 3))
(ODDP (+ 2 4))
```

3.2 Funciones de comparación de símbolos y listas

La función EQ

3.2.1 Calcular el valor de las siguientes expresiones:

```
(EQ 'LISP 'LISP)
(EQ 'LISP 'LISA)
(EQ (CAR '(A B C)) 'A)
(EQ 3 3.0)
(EQ (CONS 'A '(B C)) (CONS 'A '(B C)))
```

La función EQUAL

3.2.2 Calcular el valor de las siguientes expresiones:

```
(EQUAL (CONS 'A '(B C)) '(A B C))
(EQUAL (+ 2 3) 5)
```

La función MEMBER

3.2.3 Calcular el valor de las siguientes expresiones:

```
(MEMBER 'X '(A X B X C))
(MEMBER 'X '(A (X) B))
(SETQ L' ((A B) (C D)))
```

```
(MEMBER '(C D) L)
(MEMBER '(C D) L :TEST #'EQUAL)
(MEMBER 2.0 '(1 2 3))
(MEMBER 2.0 '(1 2 3) :TEST #'=)
(MEMBER 2.0 '(1 2 3) :TEST #'<)
```

Los predicados ATOM, SYMBOLP, NUMBERP, CONSP, LISTP

3.2.4 Calcular el valor de las siguientes expresiones:

```
(SETQ N (1+ 3) S 'LISP L '(1 ES UN NUMERO))
(ATOM N)
(ATOM S)
(ATOM T)
(ATOM ())
(ATOM L)
(CONSP L)
(CONSP ())
(LISTP NIL)
(SYMBOLP N)
(SYMBOLP L)
(SYMBOLP S)
(SYMBOLP T)
(SYMBOLP NIL)
(NUMBERP N)
(NUMBERP (CAR L))
(NUMBERP S)
```

3.3 Condicionales

La función IF

3.3.1 Calcular el valor de las siguientes expresiones:

```
(IF T 1 2)
(IF NIL 1 2)
(IF (= (SETQ A 3) 4) 1 0)
(IF (= A 3) 1 0)
(IF (= A 4) (+ A 2))
(IF (/ A 4) (- A 2))
```

3.3.2 Definir la función

```
(ABSOLUTO N)
```

que devuelva el valor absoluto del número `N`. Por ejemplo,

```
(ABSOLUTO -5)  =>  5
```

La función `COND`

3.3.3 Calcular el valor de las siguientes expresiones:

```
(SETQ X T Y NIL Z '(3 4) U NIL)
(COND (Y (SETQ U 'VAL1))
      ((CDR Z) (SETQ U 'VAL2))
      (X (SETQ Y (CAR Z))  (SETQ U 'VAL3))
      (T (SETQ U 'VAL 4)))
U
Y
```

3.3.4 Calcular el valor de las siguientes expresiones:

```
(COND ((1+ 2)) (T 5))
(COND (( = (+ 2 3) 6) 'UNO) ('DOS))
(COND ((ATOM  '(A)) 'UNO) ((LISTP  'A)  'DOS))
```

3.3.5 Redefinir la función `ABSOLUTO` usando la función `COND`.

3.3.6 Definir la función

```
(NOTA N)
```

que devuelva número la nota correspondiente al número `N`; es decir,

```
           | SUSPENSO     , si N < 5
           | APROBADO     , si 5 <= N < 7
(NOTA N) = | NOTABLE      , si 7 <= N < 9
           | SOBRESALIENTE, si 9 <= N <= 10
           | ERROR        , si N > 10
```

3.3.7 Definir la función

```
(TIPO-DE EXP)
```

que devuelva el tipo de la expresión `EXP` (i.e. lista-no-vacía, átomo-nil, número o símbolo). Por ejemplo:

```
(TIPO-DE  '(A B))
(TIPO-DE  NIL)
(TIPO-DE  (1+ 3))
(TIPO-DE  T)
(TIPO-DE  'A)
```

3.3.8 Supongamos que hemos definido una variable `METALES` que contiene algunos metales y sus densidades mediante la expresión:

```
(SETQ METALES  '(HIERRO 7.8 COBALTO 8.9 NIQUEL 8.9))
```

y otra variable `GASES-NOBLES` mediante la expresión

```
(SETQ GASES-NOBLES '(HELIO NEON ARGON)).
```

Definir una función

```
(AYUDA-QUIMICA SIMB)
```

de forma que devuelva:

- una lista de la forma `(METAL DE DENSIDAD N)`, si `SIMB` es un metal de la lista `METALES` y `N` es la densidad de `SIMB`.
- `GAS-NOBLE`, si `SIMB` está en la lista `GASES-NOBLES`.
- `DESCONOCIDO`, en otro caso.

Por ejemplo:

```
(AYUDA-QUIMICA  'COBALTO)   =>  (METAL DE DENSIDAD 8.9)
(AYUDA-QUIMICA  'HELIO)     =>  GAS-NOBLE
(AYUDA-QUIMICA  'ALUMINIO)  =>  DESCONOCIDO
```

La función CASE

3.3.9 Redefinir la función `AYUDA-QUIMICA` usando la función `CASE`.

3.3.10 Definir la función

```
(EXTREMOS L)
```

de forma que devuelva los extremos de la lista `L`. Por ejemplo,

```
(EXTREMOS ())         =>  NIL
(EXTREMOS '(A))       =>  (A A)
(EXTREMOS '(A B))     =>  (A B)
(EXTREMOS '(A B C))   =>  (A C)
```

Las funciones AND y OR

3.3.11 Calcular el valor de las siguientes expresiones:

```
(AND (SETQ L '(A B)) (SETQ M (CDDR L)) (SETQ L '(X)))
M
L
(OR NIL (SETQ A 2) (SETQ A 3))
A
```

3.3.12 Definir la función

```
(EMPIEZA-POR-NUMERO S)
```

de forma que devuelva `T`, si `S` es una lista que empieza por un número y `NIL`, en caso contrario. Por ejemplo:

```
(EMPIEZA-POR-NUMERO  '(1 A)) ---> T
(EMPIEZA-POR-NUMERO T)       ---> NIL
```

Las funciones WHEN y UNLESS

3.3.13 Calcular el valor de las siguientes expresiones:

```
(WHEN (= (+ 2 3) 5) (SETQ A 1) (SETQ B 5) (+ A B))
(WHEN (= (+ 2 3) 6) (SETQ A 1) (SETQ B 5) (+ A B))
(UNLESS ( = ( + 2 3) 5) (SETQ A 1) (SETQ B 5) (+ A B))
(UNLESS ( = ( + 2 3) 6) (SETQ A 1) (SETQ B 5) (+ A B))
```

Capítulo 4

La programación recursiva

4.1 Funciones recursivas

4.1.1 Definir la función

```
(FACT N)
```

de forma que devuelva el factorial de `N` y explicar el cálculo de `(FACT 3)`.

4.2 Las funciones de trazado

Las funciones `TRACE` y `UNTRACE`

4.2.1 Evaluar la función `FACT` usando la función `TRACE`.

4.3 Aritmética entera positiva

El objeto de este parágrafo es "redefinir" las funciones y predicados sobre los números enteros positivos usando solamente las funciones `1+`, `1-` *y el predicado* `=`.

4.3.1 Definir las funciones `SUC` y `PRED` tales que

```
(SUC N)  = N + 1
(PRED N) = N - 1
```

4.3.2 Definir los predicados `IGUAL-P` y `DESIGUAL-P` tales que

$$\texttt{(IGUAL-P N1 N2)} = \begin{cases} \texttt{T} & \text{si } \texttt{N1 = N2} \\ \texttt{NIL} & \text{en caso contrario} \end{cases}$$

$$\texttt{(DESIGUAL-P N1 N2)} = \begin{cases} \texttt{NIL} & \text{si } \texttt{N1 = N2} \\ \texttt{T} & \text{en caso contrario} \end{cases}$$

4.3.3 Definir el predicado `MENOR-P` tal que

$$\texttt{(MENOR-P N1 N2)} = \begin{cases} \texttt{T} & \text{si } \texttt{N1} < \texttt{N2} \\ \texttt{NIL} & \text{en otro caso} \end{cases}$$

Por ejemplo,

```
(MENOR-P 0 3) ---> T
(MENOR-P 3 3) ---> NIL
```

4.3.4 Definir la función `SUMA` tal que

```
(SUMA N1 N2) = N1 + N2
```

(Indicación: usar los esquemas

$$\texttt{(N1 + N2)} = \begin{cases} \texttt{N2} & \text{si } \texttt{N1 = 0} \\ \texttt{(N1 - 1) + (N2 + 1)} & \text{en caso contrario} \end{cases}$$

$$\texttt{(N1 + N2)} = \begin{cases} \texttt{N2} & \text{si } \texttt{N1 = 0} \\ \texttt{((N1 - 1) + N2) + 1} & \text{en caso contrario} \end{cases}$$

4.3.5 Determinar el número de llamadas a la función `SUMA` realizadas para calcular `(SUMA 0 8)`, `(SUMA 8 0)`, `(SUMA X Y)`.

4.3.6 Redefinir la función `SUMA` de forma que el número de llamadas al calcular `(SUMA X Y)` sea el mínimo de `X` e `Y`.

4.3.7 Definir la función `RESTA` tal que

$$\texttt{(RESTA N1 N2)} = \begin{cases} \texttt{NIL} & \text{si } \texttt{N1} < \texttt{N2} \\ \texttt{N1 - N2} & \text{en otro caso} \end{cases}$$

usando las funciones definidas en este parágrafo.

4.3.8 Definir la función `PRODUCTO` tal que

```
(PRODUCTO N1 N2) = N1 * N2
```

usando las funciones definidas en este parágrafo.

4.3.9 Definir los predicados `PAR-P` e `IMPAR-P` tales que

$$\texttt{(PAR-P N)} = \begin{cases} \texttt{T} & \text{si N es par} \\ \texttt{NIL} & \text{en caso contrario} \end{cases}$$

$$\texttt{(IMPAR-P N)} = \begin{cases} \texttt{T} & \text{si N es impar} \\ \texttt{NIL} & \text{en caso contrario} \end{cases}$$

4.3.10 Definir la función `COCIENTE` tal que

```
(COCIENTE N1 N2) = Parte entera de N1/N2
```

Por ejemplo:

```
(COCIENTE 7 2) = 3
```

4.3.11 Definir la función `RESTO` tal que

```
(RESTO N1 N2) = Resto de dividir N1 entre N2.
```

Por ejemplo,

```
(RESTO 7 2) ---> 1
```

4.3.12 Definir la función

```
(DIVISION N1 N2)
```

de forma que devuelva una lista cuyo primer elemento sea la parte entera de `N1/N2` y el segundo, el resto de dividir `N1` entre `N2`. Por ejemplo,

```
(DIVISION 7 2) ---> (3 1)
```

4.4 La aritmética ordinaria del lisp

4.4.1 Definir la función

$$\texttt{(POTENCIA N1 N2)} = \begin{cases} \texttt{INDET} & \text{si N1 = N2 = 0} \\ \texttt{0} & \text{si N1 = 0 y N2} \neq \texttt{0} \\ \texttt{0} & \text{si N1} \neq \texttt{0 y N2 = 0} \\ N1^{N2} & \text{en otro caso} \end{cases}$$

Por ejemplo,

```
(POTENCIA 2 3) ---> 8
```

4.4.2 Redefinir la función `SQRT`.

4.5 Simulación de primitivas sobre listas

4.5.1 Redefinir la función `LENGTH`.

4.5.2 Redefinir la función `NTH`.

4.5.3 Redefinir la función `APPEND`.

4.5.4 Redefinir la función `MEMBER` de forma que

```
(N-MEMBER 'B '(A B C))          ---> (B C)
(N-MEMBER '(B) '((A) (B) (C))) ---> ((B) (C))
```

4.5.5 La función

```
(REVERSE L)
```

devuelve la lista obtenida escribiendo los elementos de `L` en orden inverso. Por ejemplo,

```
(REVERSE '(A B C)) ---> (C B A).
```

Redefinir la función `REVERSE`.

4.5.6 La función

```
(SUBST E1 E2 L)
```

devuelve la lista obtenida a partir de `L` en la que todas las ocurrencias de la expresión `E2` se han sustituido por la expresión `E1`. Por ejemplo:

```
(SUBST 5 'A '(A B (A C))) ---> (5 B (5 C))
```

Redefinir la función `SUBST`.

4.5.7 La función

```
(REMOVE S L)
```

devuelve la lista obtenida a partir de `L` borrando las estancias, de primer nivel, de la expresión S. Por ejemplo,

```
(REMOVE 'A '(A B (A C) A)) ---> (B (A C))
```

Redefinir la función `REMOVE`.

4.6 Definición de funciones sobre listas

4.6.1 Definir la función SUBSTOP que trabaje como `SUBST` pero sólo en el primer nivel. Por ejemplo,

```
(SUBSTOP 5 'A '(A (A B) A)) ---> (5 (A B) 5)
```

4.6.2 Definir la función `REV` que actúe como `REVERSE` pero a todos los niveles. Por ejemplo,

```
(REV '(A (B C) (D (E)))) ---> (((E) D) (C B) A)
```

4.6.3 Definir la función

```
(LINEALIZA L)
```

que devuelva la lista obtenida suprimiendo todos los paréntesis internos de `L` salvo los correspondientes a `()`. Por ejemplo:

```
(LINEALIZA '(A (B (C)) () E)) ---> (A B C NIL E)
```

4.6.4 Definir la función

```
(CUENTA-ATOMOS L)
```

de forma que devuelva el número de átomos que contiene la lista `L`. Por ejemplo,

```
(CUENTA-ATOMOS '(A (B (C)) () E)) ---> 5
```

4.6.5 Definir la función

```
(MISMA-FORMA E1 E2)
```

que devuelva `T` o `NIL`, según que las expresiones `E1` y `E2` tengan o no la misma forma. Por ejemplo,

```
(MISMA-FORMA '(A (B C)) '(1 (2 3))) ---> T
```

4.6.6 Definir la función `MEM` que devuelva `T` o `NIL` y actúe como `MEMBER` pero a todos los niveles. Por ejemplo,

```
(MEM 'A '((A B) C)) ---> T
```

4.6.7 Definir la función `SEPARA` que tome una lista (de los niveles que sea) y separe las letras de los números. Por ejemplo,

```
(SEPARA '(A (1 2) ((B)) (C (3)))) ---> ((A B C) (1 2 3))
```

4.6.8 Escribir la función tt DOBLA-EL que duplique todos los elementos de la lista que toma por argumento. Por ejemplo,

```
(DOBLA-EL '(A B))   ---> (A A B B)
(DOBLA-EL '((A) B)) ---> ((A) (A) B B)
```

4.6.9 Escribir la función `DOBLA-ATOMOS` que duplique todos los átomos de la lista que toma como argumento en el nivel que se encuentre. Por ejemplo,

```
(DOBLA-ATOMOS '((A) B)) ---> ((A A) B B)
```

4.6.10 Escribir una función `AGRUPA` que agrupe los elementos sucesivos de dos listas. Por ejemplo,

```
(AGRUPA '(A B C) '(1 2 3)) ---> ((A 1) (B 2) (C 3))
(AGRUPA '(A B) '(1 2 3))   ---> ((A 1) (B 2) 3)
(AGRUPA '(A B C) '(1 2))   ---> ((A 1) (B 2) C)
```

4.7 Funciones sobre árboles

Los árboles pueden representarse por listas. Por ejemplo, el árbol

.

A . . E

B C D .

A

se representa por la lista (A (B C) (D (A)) E). En esta sección estudiaremos funciones sobre árboles.

4.7.1 Escribir la función

```
(MISMO-ARBOL L1 L2)
```

que devuelva `T` si `L1` y `L2` representan el mismo árbol y `NIL`, si no lo representan. Por ejemplo

```
(MISMO-ARBOL '(1 (2 (3))) '(A (B (C)))) ---> T
```

4.7.2 Definir la función

```
(N-HOJAS L)
```

que devuelva el número de hojas del árbol asociado a la lista `L`. Por ejemplo,

```
(N-HOJAS '(A (B C) (C (A)) E)) ---> 6
```

4.7.3 Definir la función

```
(N-ARCOS L)
```

que devuelva el número de arcos del árbol asociado a L. Por ejemplo,

```
(N-ARCOS '(A (B C) (C (A)) E)) ---> 9
```

4.7.4 Definir la función

```
(PROFUNDIDAD L)
```

que devuelva el número de niveles que existen en `L`; es decir, el número de arcos de la raíz a la hoja más lejana. Por ejemplo,

```
(PROFUNDIDAD '(A (B C) (D (A)) E)) ---> 3
```

4.8 Funciones sobre conjuntos

Representamos los conjuntos como listas sin elementos repetidos. Por ejemplo, `(A B C)` *es un conjunto; pero,* `(A B A C)` *no lo es. Además, el orden de los elementos de un conjunto no es esencial. Así,* `(A B C)` *y* `(A C B)` *representan el mismo conjunto.*

4.8.1 Definir la función `REDUCE-CONJUNTO` que tome una lista y elimine los elementos repetidos; esto es, la convierta en "conjunto". Por ejemplo,

```
(REDUCE-CONJUNTO ' (A B A C)) ---> (B A C)
```

4.8.2 Definir

```
(UNION C1 C2)
```

que devuelva la unión de los conjuntos `C1` y `C2` (i.e. una lista sin elementos repetidos que contenga los elementos que están en `C1` o en `C2`). Por ejemplo,

```
(UNION '(0 3 6 9 12) '(0 4 8 12)) ---> (3 6 9 0 4 8 12)
```

4.8.3 Definir la función

```
(INTERSECCION C1 C2)
```

que devuelva la intersección de los conjuntos `C1` y `C2` (i. e. la lista de sus elementos comunes). Por ejemplo,

```
(INTERSECCION '(0 3 6 9 12) '(0 4 8 12)) ---> (0 12)
```

4.8.4 Definir la función

```
(DIF C1 C2)
```

que devuelva la diferencia de los conjuntos `C1` y `C2` (i.e. la lista de los elementos de `C1` que no son elementos de `C2`). Por ejemplo,

```
(DIF '(0 3 6 9 12) '(0 4 8 12)) ---> (3 6 9)
```

4.8.5 Definir la función

```
(DIF-SIM C1 C2)
```

que devuelva la diferencia simétrica de `C1` y `C2` (i.e. la lista de los elementos que sólo están en uno de los conjuntos `C1` ó `C2`).Por ejemplo,

```
(DIF-SIM '(0 3 6 9 12) '(0 4 8 12)) ---> (3 6 9 4 8)
```

4.8.6 Definir la función

```
(IGUAL-CONJ C1 C2)
```

que devuelva `T` si los conjuntos `C1` y `C2` son iguales y `NIL`, si no lo son. Por ejemplo,

```
(IGUAL-CONJ '(A B C) '(C A B)) ---> T
```

Capítulo 5

La iteración en Lisp

5.1 El grupo PROG, GO, RETURN

5.1.1 Redefinir iterativamente, usando PROG, las siguientes funciones:

1. FACTORIAL.
2. POTENCIA.
3. LENGTH.
4. NTHCDR.
5. MEMBER.
6. REVERSE.
7. APPEND.

5.2 Las funciones DO y DO*

5.2.1 Redefinir iterativamente, usando DO, las siguientes funciones:

1. LENGTH
2. FACTORIAL

Las funciones LET*, PROG* y DO*

5.2.2 Redefinir iterativamente, usando `DO*`, las siguientes funciones:

1. `FACTORIAL`
2. `REVERSE`
3. `POTENCIA`

5.2.3 Definir la función

```
(SIMBOLOS-DE L)
```

que devuelva una lista de los símbolos de la lista `L`. Por ejemplo,

```
(SIMBOLOS-DE '(1 + 1 = 2))  =>  (= +)
```

5.2.4 Definir la función

```
(PONE-PARENTESIS L)
```

que devuelva una lista en la que los elementos de L están entre paréntesis. Por ejemplo,

```
(PONE-PARENTESIS '(A B C))  =>  ((A)(B)(C))
```

5.3 Las funciones DOTIMES y DOLIST

5.3.1 Redefinir iterativamente, usando `DOTIMES`, las siguientes funciones:

1. `FACTORAL`
2. `POTENCIA`
3. `REVERSE`

5.3.2 Redefinir iterativamente, usando `DOLIST`, la función `REVERSE`.

5.3.3 Definir iterativamente, usando `DOLIST`, la función

```
(CUENTA-APROBADOS L)
```

de forma que si `L` es una lista de notas, devuelva el número de aprobados. Por ejemplo,

```
(CUENTA-APROBADOS '(1 6 5 3 7))   =>  3
```

5.4 La función MAPCAR

5.4.1 Redefinir, usando `MAPCAR`, la función `SUBST`.

5.5 La función APPLY

5.5.1 Redefinir, usando APPLY, la función `CUENTA-ATOMOS`.

5.5.2 Definir la función

```
(SUMA-VECTORES V1 V2)
```

que devuelva la suma de los vectores `V1` y `V2`. Por ejemplo,

```
(SUMA-VECTORES '(3 5) '(4 3))  =>  (7 8)
```

5.5.3 Definir la función

```
(NORMA V)
```

que devuelva la norma del vector `V`; es decir, la raíz cuadrada de la suma de los cuadrados de sus argumentos. Por ejemplo,

```
(NORMA '(3 4))  =>  5.0
```

5.5.4 Definir la función

```
(PRODUCTO-ESCALAR V1 V2)
```

que devuelva el producto escalar de `V1` y `V2`; es decir,

```
(PRODUCTO-ESCALAR '(A1 ... AN) '(B1 ...BN)) => A1.B1 +...+ AN.BN
```

Por ejemplo,

```
(PRODUCTO-ESCALAR '(2 3) '(4 5))  =>  23
```

5.5.5 Ejercicio: Redefinir la función

```
(REV L)
```

que invierta los elementos de una lista a todos los niveles. Por ejemplo,

```
(REV '(A (B C) (D (E))))  =>  (((E) D) (C B) A)
```

Capítulo 6

Funciones anónimas

6.1 La función LAMBDA

6.1.1 Calcular el valor de las siguientes expresiones:

```
((LAMBDA (M N) (+ M N)) 2 3)
((LAMBDA (X Y) (+ (* X X) (* Y Y))) 3 4)
(SETQ N1 10)
((LAMBDA (N) (* N 2)) N1)
((LAMBDA (X) (LIST X X)) 2)
((LAMBDA (X Y) (CONS Y X)) '(B) 'A)
((LAMBDA (L) (CAR L)) '(X Y))

(MAPCAR #'(LAMBDA (N) (* N N)) '(1 2 3))
((LAMBDA (N) (* N N)) '(3))
(APPLY #'(LAMBDA (N) (* N N)) '(3))
```

6.1.2 Definir la función

```
(PARTES-DE L)
```

que devuelva los subconjuntos de L. Por ejemplo,

```
(PARTES-DE '(A B C)) => (NIL (C)(B)(B C)(A)(A C)(A B)(A BC))
```

6.2 Las funciones EVERY y SOME

6.2.1 Calcular el valor de las siguientes expresiones:

```
(EVERY #'ATOM '(1 A))
(EVERY #'ATOM '((1) A))
(EVERY #'= '(1 2) '(1 2))
(SOME #'ATOM '((1) A))
(SOME #'ATOM '((1) (A)))
```

6.2.2 Definir la función

```
(SUBCONJUNTO A B)
```

que devuelva `T` si A es un subconjunto de B y `NIL` en caso contrario. Por ejemplo,

```
(SUBCONJUNTO '(A C) '(A B C))  =>  T
(SUBCONJUNTO '(A D) '(A B C))  =>  NIL
```

6.2.3 Definir la función

```
(DISJUNTOP A B)
```

que devuelva `T` si A y B son disjuntos y `NIL`, en caso contrario. Por ejemplo,

```
(DISJUNTOP '(A C) '(A B))  =>  NIL
(DISJUNTOP '(A C) '(D B))  =>  T
```

Capítulo 7

Las A–listas (listas de asociación)

7.1 Pares punteados

7.1.1 Calcular el valor de las siguientes expresiones:

```
(CONS 'A 'B)
(CAR '(A . B))
(CDR '(A . B))
'(A . NIL)
'(A . (B))
'(A . (B C))
'((A) . (B))
```

7.2 A–listas

7.2.1 Calcular el valor de las siguientes expresiones:

```
(SETQ L '((B . 2) (C . 3)))
(ACONS 'A 1 L)
```

7.2.2 Redefinir la función `ACONS`.

7.2.3 Calcular el valor de las siguientes expresiones:

```
(PAIRLIS '(A B C) '(1 2 3) ())
```

```
(PAIRLIS '(A B) '(1 2) '((C . 3)))
(PAIRLIS '(A B) '((1) (2)) ())
```

7.2.4 Redefinir la función `PAIRLIS`.

7.2.5 Calcular el valor de las siguientes expresiones:

```
(ASSOC 'B '((A . 1) (B . 2) (C . 3))))
(ASSOC '(B) '((A . 1) ((B) 1) (C D)))
```

7.2.6 Redefinir la función `ASSOC`.

7.2.7 Calcular el valor de las siguientes expresiones:

```
(RASSOC 1 '((A) (B . 1) (C D E))))
(RASSOC '(D E) '((A) ((B) 1) (C D E)))
(RASSOC '(D E) '((A) ((B) 1) (C D E)) :TEST #'EQUAL)
```

7.2.8 Redefinir la función `RASSOC`.

7.2.9 Calcular el valor de las siguientes expresiones:

```
(SETQ DICCIONARIO '((2 . DOS)(4 . CUATRO)(+ . MAS)(= . IGUAL-A)))
(SUBLIS DICCIONARIO '(2 + 2 = 4))
```

7.2.10 Se consideran las siguientes listas:

```
NUMEROS = (1 2 3)
CASTELLANO = (UNO DOS TRES)
INGLES  = (ONE TWO THREE)
FRANCES = (UN DEUX TROIS)
```

Definir las A–listas

```
NUMEROS-CASTELLANO, CASTELLANO-INGLES, CASTELLANO-FRANCES
```

que asocien los primeros elementos con los segundos.

7.2.11 Definir las funciones

```
(EN-CASTELLANO N),  (EN-INGLES N),  (EN-FRANCES N)
```

de forma que si N es un número, devuelva su nombre en el idioma apropiado. Por ejemplo,

```
(EN-CASTELLANO 2)    =>  DOS
(EN-INGLES 2)        =>  TWO
(EN-FRANCES 2)       =>  DEUX
```

7.2.12 Definir la función

```
(NOMBRE N I)
```

de forma que si N es un número e I es un idioma, devuelva el nombre de N en el idioma I. Por ejemplo,

```
(NOMBRE 2 'CASTELLANO)     =>  DOS
(NOMBRE 2 'INGLES)         =>  TWO
(NOMBRE 2 'FRANCES)        =>  DEUX
```

7.2.13 Definir las funciones:

```
(VALOR-CASTELLANO S), (VALOR-INGLES S), (VALOR-FRANCES S)
```

de forma que si S es el nombre de un número en el idioma indicado, devuelva su valor. Por ejemplo,

```
(VALOR-CASTELLANO 'DOS)    =>  2
(VALOR-INGLES 'ONE)        =>  1
(VALOR-FRANCES 'TROIS)     =>  3
```

7.2.14 Definir la función

```
(VALOR N I)
```

de forma que si N es el nombre de un número en el idioma I es un idioma, devuelva su valor. Por ejemplo,

```
(VALOR 'DOS 'CASTELLANO)     =>  2
(VALOR 'ONE 'INGLES)         =>  1
(VALOR 'TROIS 'FRANCES)      =>  3
```

7.2.15 Definir la función

```
(FRANCES-A-CASTELLANO S)
```

que traduzca números en francés a números en español. Por ejemplo,

```
(FRANCES-A-CASTELLANO 'TROIS)  =>  TRES
```

Capítulo 8

Las P–listas (listas de propiedades)

8.1 P–listas

8.1.1 Calcular el valor de las siguientes expresiones:

```
(SETF (SYMBOL-PLIST 'PEPE) '(EDAD 40 HIJOS (PEPITO PEPITA)))
(SYMBOL-PLIST 'PEPE)
(SYMBOL-PLIST 'JUAN)
(GET 'PEPE 'EDAD)
(GET 'PEPE 'HIJOS)
(GET 'PEPE 'PADRES)
(SETF (GET 'PEPE 'EDAD) 41)
(SYMBOL-PLIST 'PEPE)
(SETF (GET 'PEPE 'MADRE) 'ANA)
(SYMBOL-PLIST 'PEPE)
(REMPROP 'PEPE 'EDAD)
(SYMBOL-PLIST 'PEPE)
```

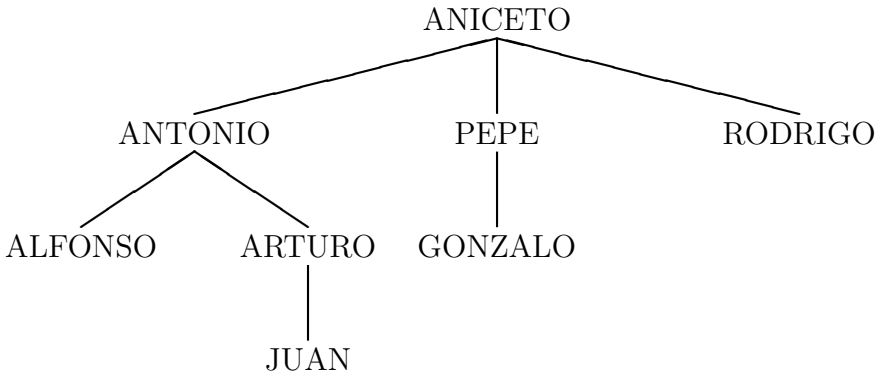
8.1.2 Escribir tres P–listas `JUAN`, `SOFIA` y `TOMAS` que recojan los siguientes datos:

JUAN	SOFIA	TOMAS
EDAD 35	EDAD 30	EDAD 5
MUJER SOFIA	MARIDO JUAN	PADRE JUAN
HIJO TOMAS	HIJO TOMAS	MADRE SOFIA

8.1.3 Consultar la base de datos anterior para obtener:

1. La edad de Tomás.
2. El hijo de Sofía.
3. La mujer del padre de Tomás.
4. La edad del marido de la madre de Tomás.
5. El hijo de la mujer del padre de Tomás.
6. El padre del hijo de Sofía.

8.1.4 Escribir una P–lista HIJOS-DE que recoja el siguiente árbol genealógico:



8.1.5 Escribir una lista PADRES que contenga los padres de árbol anterior.

8.1.6 Definir la función

```
(PADRE X)
```

que devuelva el padre de `X`. Por ejemplo,

```
(PADRE 'GONZALO)  =>  PEPE
```

8.1.7 Definir la función

```
(ABUELO X)
```

que devuelva el abuelo de `X`. Por ejemplo,

```
(ABUELO 'JUAN)  =>  ANTONIO
(ABUELO 'PEPE)  =>  NIL
```

8.1.8 Definir la función

```
(DECANO X)
```

que devuelva el primer antepasado de `X`. Por ejemplo,

```
(DECANO 'JUAN)  =>  ANICETO
(DECANO 'LUIS)  =>  LUIS
```

8.1.9 Definir la función

```
(ANCESTROS X)
```

que devuelva la lista de los antepasados de `X`. Por ejemplo,

```
(ANCESTROS 'JUAN)  =>  (JUAN ARTURO ANTONIO ANICETO)
```

8.1.10 Utilizar la función `ANCESTROS` para redefinir la función `DECANO`.

8.1.11 Definir la función

```
(HERMANOS X)
```

que devuelva la lista de los hermanos de `X`. Por ejemplo,

```
(HERMANOS 'PEPE)  =>  (ANTONIO RODRIGO)
(HERMANOS 'JUAN)  =>  NIL
```

8.1.12 Definir la función

```
(TIOS X)
```

que devuelva la lista de los tíos de `X`. Por ejemplo,

```
(TIOS 'GONZALO)  =>  (ANTONIO RODRIGO)
```

8.2 Bases de datos

8.2.1 Deseamos construir una base de datos bibliográfica. Para ello, utilizamos las P–listas `LIBRO-N` que contiene el autor, título y editorial de libro de referencia `N` y la variable global `LIBROS` que contiene las P–listas almacenadas. Por ejemplo, si nuestra base comienza por el libro

```
(FARRENY:  ``Programmer en LISP'', ed.  MASSON)
```

escribimos

```
(SETF (SYMBOL-PLIST 'LIBRO-1)
      '(AUTOR "FARRENY"
      TITULO "Programmer en LISP"
      EDITORIAL "Masson"))

(SETQ LIBROS '(LIBRO-1))
```

Definir la función

```
(ADD-LIBRO REFERENCIA AUTOR TITULO EDITORIAL)
```

que añada el libro especificado a la base de datos y devuelva `REFERENCIA`. Por ejemplo,

```
(ADD-LIBRO 'LIBRO-2
           "WINSTON Y HORN"
           "LISP"
           "ADDISON WESLEY")  =>  LIBRO-2
LIBROS                        =>  (LIBRO-2 LIBRO-1)
(GET 'LIBRO-2 'TITULO)        =>  "LISP"
```

8.2.2 Definir la función

```
(BUSCAR-POR PROPIEDAD VALOR)
```

que busque en la base de datos `LIBROS` y dé la lista de las referencias de los libros que verifican las condiciones. Por ejemplo,

```
(BUSCAR-POR 'AUTOR "FARRENY")  =>  (LIBRO-1)
(BUSCAR-POR 'TITULO "LISP")    =>  (LIBRO-2)
```

8.3 Programación dirigida por los datos

8.3.1 Calcular el valor de las siguientes expresiones:

```
(EVAL '(+ 2 3))
(CONS '+ '(2 3))
(EVAL (CONS '+ '(2 3)))
```

8.3.2 Definir la función `CALCULAR` que permita sumar o restar números reales o complejos; es decir,

```
(CALCULAR 'A 'MAS B)           = A + B
(CALCULAR '(A B) 'MAS '(C D)   = (A + C  B + D)
(CALCULAR 'A 'MENOS 'B)        = A - B
(CALCULAR '(A B) 'MENOS '(C D) = (A - C  B - D)
```

Por ejemplo,

```
(CALCULAR 2 'MAS 3)              =>  5
(CALCULAR '(5 4) 'MENOS '(2 0))  =>  (3 4)
```

8.3.3 Ampliar la anterior definición par que acepte la multiplicación real y compleja; es decir,

```
(CALCULAR 'A 'POR 'B)          = AB
(CALCULAR '(A B) 'POR '(C D)) = (AC - BD  AD + BC)
```

Por ejemplo,

```
(CALCULAR 2 'POR 3)             =>  6
(CALCULAR '(1 2) 'POR '(3 4))   =>  (-5 10)
```

8.4 Funciones con memoria

8.4.1 Redefinir `FACTORIAL` como función con memoria.

8.4.2 Mediante `TRACE`, comprobar el número de llamadas a la función `FACTORIAL` cuando se calcula `(FACTORIAL 7)` y `(FACTORIAL 10)`.

8.4.3 Comparar el tiempo empleado en calcular los factoriales de 100, 150 y 200 con las definiciones anteriores.

8.4.4 La función de `FIBONACCI` está definida por

$$\texttt{FIB (N)} = \begin{cases} \texttt{1} & \text{si N = 0 ó N = 1} \\ \texttt{FIB(N-1) + FIB(N-2)} & \text{en otro caso} \end{cases}$$

Los primeros términos de la sucesion son 1, 1, 2, 3, 5, 8, 13, 21,... Escribir la función

```
(FIB N)
```

que devuelva el término N-ésimo de la función de FIBONACCI. Por ejemplo,

```
(FIB 6)  =>  13
```

8.4.5 Redefinir la función de `FIBONACCI` como función con memoria.

8.4.6 Comparar mediante `TRACE` y `TIME` las dos definiciones anteriores de la función `FIBONACCI`.

8.4.7 La función de 91 de McCarthy está definida por:

$$\texttt{F91(N)} = \begin{cases} \texttt{N - 10} & \text{si } \texttt{N} > 100 \\ \texttt{F91(F91(N + 11))} & \text{si } 0 \leq \texttt{N} \leq 100 \end{cases}$$

Escribir la definición de dicha función de forma recursiva, recursiva con memoria y no recursiva.

8.4.8 La función de `ACKERMANN` está definida por

$$\texttt{ACK(M, N)} = \begin{cases} \texttt{N + 1} & \text{si } \texttt{M} = 0 \\ \texttt{ACK(M - 1, 1)} & \text{si } \texttt{M} > 0 \text{ y } \texttt{N} = 0 \\ \texttt{ACK(M - 1, ACK(M, N-1))} & \text{en otro caso} \end{cases}$$

Escribir la definición de

```
(ACK M N)
```

y calcular

```
(ACK 3 4)
(TIME '(ACK 3 4))
```

Capítulo 9

Lectura y escritura

9.1 Funciones de lectura y escritura

La función PRINT

9.1.1 Escribir una función

```
(IMPRIME-TODO L)
```

tal que escriba los elementos de `L` uno en cada línea. Por ejemplo,

```
* (IMPRIME-TODO '(A (B C) D))
A
(B C)
D
NIL
```

Las funciones PRIN1, PRINC y TERPRI

9.1.2 Definir la función

```
(TAB-CUAD N)
```

de forma que imprima la lista de los cuadrados de 1 a N. Por ejemplo,

```
* (TAB-CUAD 3)
EL CUADRADO DE 1 ES 1
```

```
EL CUADRADO DE 2 ES 4
EL CUADRADO DE 3 ES 9
NIL
```

La función READ

9.1.3 Ejercicio: Definir la función

```
(CALCULA-CUADRADOS)
```

tal que

1. lea una expresión S;
2. si el valor de S es un número N, devuelva el cuadrado de N y vuelve a 1;
3. si el valor de S no es un número, escriba FIN y pare.

Por ejemplo,

```
* (CALCULA-CUADRADOS)
EL CUADRADO DE 3
ES 9

EL CUADRADO DE 5
ES 25

EL CUADRADO DE A
FIN

NIL
```

La función FORMAT

9.1.4 Calcular el valor de las siguientes expresiones:

```
* (FORMAT T "~%LINEA 1 ~%LINEA 2)
* (FORMAT T "~%EL CUADRADO DE ~D ES ~D" 3 (* 3 3))
* (SETQ L '(A B C))
* (FORMAT T
```

```
          "~%LA LONGITUD DE LA LISTA ~A ES ~D"
          L (LENGTH L))
* (FORMAT T
          "~%10 EN BINARIO ES ~B, EN OCTAL ES ~O Y EN HEXADECIMAL ES ~X"
          10 10 10)
```

9.1.5 Definir la función

```
(CUADRADOS M N)
```

que escriba la lista de los cuadrados desde M hasta N. Por ejemplo,

```
* (CUADRADOS 2 5)
EL CUADRADO DE 2 ES 4
EL CUADRADO DE 3 ES 9
EL CUADRADO DE 4 ES 16
EL CUADRADO DE 5 ES 25
FIN
```

9.2 Funciones de lectura y escritura sobre ficheros

9.2.1 Calcular el valor de las siguientes expresiones:

```
(SETQ FICH1 (OPEN "FICHERO.DAT" :DIRECTION :OUTPUT))
(FORMAT FICH1 "LINEA 1~%LINEA 2")
(FORMAT FICH1 "~%LINEA 3")
(CLOSE FICH1)
(SETQ A (OPEN "FICHERO.DAT" :DIRECTION :INPUT))
(READ-LINE A)
(READ-LINE A)
(READ-LINE A)
```

9.2.2 Calcular el valor de las siguientes expresiones:

```
* (WITH-OPEN-FILE (FICH1 "FICHERO.DAT" :DIRECTION :OUTPUT)
     (FORMAT FICH1 "UNO ~%DOS ~%TRES") )
* (WITH-OPEN-FILE (A "FICHERO.DAT" :DIRECTION :INPUT)
     (PRINT (READ A))
     (PRINT (READ A))
     (PRINT (READ A))
     'FIN )
```

9.2.3 Supongamos que el contenido del fichero NOTAS.DAT es:

```
((AGUADO CASAS) (JUAN JOSE) 5)
((AGUILA PUENTE) (RAFAEL) 2)
((ALBA ADAME) (CARLOS) 9)
((ALCALDE PEREZ) (MARIA LUISA) 3)
```

Definir la función que

```
(LEE)
```

que lea el contenido del fichero. Por ejemplo:

```
* (LEE)
((AGUADO CASAS) (JUAN JOSE) 5)
((AGUILA PUENTE) (RAFAEL) 2)
((ALBA ADAME) (CARLOS) 9)
((ALCALDE PEREZ) (MARIA LUISA) 3)

NIL
```

9.2.4 Definir la función

```
(APROBADOS)
```

que lea el fichero NOTAS.DAT y escriba en el fichero APROBADOS.DAT la lista de aprobados.

9.2.5 Supongamos que tenemos una lista

```
(SETQ AGENDA '((PEPE SIERPES-12   "4531420")
               (JUAN CUNA-5       "4243568") ))
```

Escribir las siguientes funciones:

`(BUSCAR NOMBRE)` Devuelve (NOMBRE DIRECCION TELEFONO), si NOMBRE está en AGENDA y NIL en caso contrario.

`(CAMBIAR-DIR NOMBRE NUEVA-DIR)` cambia la dirección de NOMBRE.

`(CAMBIAR-TEL NOMBRE NUEVO-TEL)` cambia el teléfono de NOMBRE.

`(PONER NOMBRE DIRECCION TELEFONO)` añade a la AGENDA (NOMBRE DIRECCION TELEFONO).

`(QUITAR NOMBRE)` quita (NOMBRE DIRECCION TELEFONO) a AGENDA.

`(CERRAR)` abre el fichero AGENDA.DAT, guarda el nuevo valor de AGENDA y lo cierra.

`(ABRIR)` abre el fichero AGENDA.DAT, lee el valor de AGENDA y lo cierra.

`(AGENDA)` presenta un menú para controlar el programa.

Por ejemplo,

```
* (BUSCAR 'PEPE)
(PEPE SIERPES-12 "4531420")
* (CAMBIAR-DIR 'PEPE 'TARFIA-SN)
(PEPE TARFIA-SN "4531420")
* (BUSCAR 'PEPE)
(PEPE TARFIA-SN "4531420")
* (CAMBIAR-TEL 'PEPE "4615600")
(PEPE TARFIA-SN "4615600")
* (BUSCAR 'PEPE)
(PEPE TARFIA-SN "4615600")
* (QUITAR 'PEPE)
((JUAN CUNA-5 "4243568"))
* (PONER 'PEPE 'TARFIA-SN "4615600")
((PEPE TARFIA-SN "4615600") (JUAN CUNA-5 "4243568"))
* (CERRAR)
NIL
* AGENDA

ERROR:
Unbound variable: AGENDA
1> <Control C>
* (ABRIR)
* AGENDA
((PEPE SIERPES-12 "4531420")(JUAN CUNA-5 "4243568")))

* ;;; EJEMPLO DE SESION:
* (AGENDA)
   1: LEER BASE DE DATOS        5: QUITAR NOMBRE
   2: BUSCAR UN NOMBRE          6: PONER NOMBRE
   3: CAMBIAR DIRECCION         7: GRABAR BASE DE DATOS
   4: CAMBIAR TELEFONO          8: SALIR
```

```
ESCRIBE OPCION: 1

ESCRIBE OPCION: 6
ESCRIBE NOMBRE: JUAN
ESCRIBE DIRECCION: TARFIA-SN
ESCRIBE TELEFONO: 4616500

ESCRIBE OPCION: 6
ESCRIBE NOMBRE: PEPE
ESCRIBE DIRECCION: CUNA-12
ESCRIBE TELEFONO: 4227700

ESCRIBE OPCION: 2
ESCRIBE EL NOMBRE: JUAN
NOMBRE:    JUAN
DIRECCION: TARFIA-SN
TELEFONO:  4616500

ESCRIBE OPCION: 7

ESCRIBE OPCION: 8
"FIN"
```