Cflp

Laboratator

Facultatea de Automatica şi Calculatoare
Departamentul de Calculatoare

Lab 7

Contents

Association list

A list of association is a list which contains sublists of the form `(<key> <value>)`. The following expression creates a list of associations wich can describe the properties of an object:

```
(setq brick-a '((color red)
                (supported-by brick-b)
                (is-a brick)))
((color red) (supported-by brick-b) (is-a brick))
```

ASSOC

The primitive `ASSOC` returns the value associated with a key, from an association list. `ASSOC` takes two arguments: a key and an association list. `ASSOC` will search for the value associated with the given key, and will return the whole association in the form `(<key> <value>)`. If the key cannot be found, it returns `NIL`.

ASSOC

```
(assoc 'color brick-a)
(color red)
(assoc 'is-a brick-a)
(is-a brick)
(assoc 'size brick-a)
nil
```

An association list may contain multiple elements having the same key, but `ASSOC` returns only the first association it finds:

ASSOC

```
(setq brick-a '((supported-by brick-c)
                (color red)
                (supported-by brick-b)
                (is-a brick)))
((supported-by brick-c) (color red) (supported-by brick-b)
(is-a brick))
(assoc 'supported-by brick-a)
(supported-by brick-c)
```

Problem 1

Write the `FETCH` procedure, which takes as arguments a key and an association list, and if it finds the key it returns only its value. Otherwise it returns a question

mark.

```
(fetch 'temperature '((temperature 100)
                      (pressure (120 60))
                      (pulse 72)))
100
(fetch 'complaints '((temperature 100)
                      (pressure (120 60))
                      (pulse 72)))
?
```

Problem 2

Write the `LIST-KEYS` procedure, which takes as arguments an association list and returns all the keys contained in it.

Properties

A symbol can have properties. A property has a name and a value.
Note: in LISP the word *value* can have three meanings:

- we talk about the value returned at the evaluation of a form
- we talk about the value of a symbol's property
- we talk about the value of a symbol

The GET primitive

In order to return the value of a property we use `GET`, which takes as argument the name of a symbol and the name of a property. Suppose that the symbol `pyramid-d` has a property with the name `color`, which has the value `red`.

GET

```
(get 'pyramid-d 'color)
red
```

By using `SETF` we can set the value of a property.

SETF

```
(setf (get 'pyramid-d 'color) 'red)
red
(setf (get 'pyramid-d 'supported-by) 'brick-c)
brick-c
```

We are able to delete a property by using the combination:
`(setf (get <symbol> <property>) nil)`

Problem 3

Supposing there are symbols with the property `parent`, define the procedure `grandfather` which takes as parameter a symbol and returns the grandfather from the father's branch, or `nil` if it is now known.

Problem 4

Define the procedure `adam` which takes as argument a symbol and returns the furthest ancestor on the father's branch.

Problem 5

Define the procedure `ancestors` which returns a list containing the person,

together with all its known ancestors, searching two properties: father and mother.

Matrices

Matrices are built in LISP using the function `MAKE-ARRAY`. We need to specify the number of indexes and their domain.

MAKE-ARRAY

The following example builds a 2-dimensional matrix:

```
(SETQ M (MAKE-ARRAY '(64 64)))
```

Every index may take a value between `0` and `63`. We tied the matrix returned by `MAKE-ARRAY` as a value to the symbol `M`, in order to use it afterwad.
Once a matrix is constructed, we cand assign values to different locations. diferitelor locaţii.

AREF

The next example assigns the value `88` to the location in the matrix corresponding to indices `31` and `27`:

```
(SETF (AREF M 31 27) 88)
```

In this context, `AREF` returns a position in the matrix, whose content is being modified by `SETF`. We can use just `AREF` in other to obtain the value of a location.

```
(AREF M 31 27)
88
```

In LISP the indices of a matrix may contain not only numbers, but also expressions. Two different locations may contain expressions of different types. The function `ARRAY-DIMENSION` returns the domain of an index.

ARRAY-DIMENSION

```
(ARRAY-DIMENSION M 0)
64
(ARRAY-DIMENSION M 1)
64
Both indices of the matrix M have values smaller than 64.
```

The function `ARRAY-DIMENSIONS` returns the domains of all indices into a list.

ARRAY-DIMENSION

```
(ARRAY-DIMENSIONS M)
(64 64)
```

Problem 6

Write a procedure `PRINT-ARRAY` which print a 2D matrix taken as argument. Each row is going to be printed on a new line. Use `ARRAY-DIMENSION` to get the domain names.

Problem 7

Another way of representing a matrix is by using a list of lists, each sublist corresponding to a dimension. Write a procedure which transforms a matrix represented as a list of lists into a standard LISP matrix. Suppose that all sublists have the same dimension.

Problems