## Function: array - create an array

### Calling sequence:

array( indexfcn, bounds, list)

### Parameters:

`indexfcn` - (optional) an indexing function
`bounds` - (optional) sequence of ranges
`list` - (optional) list of initial values

## Description:

- An array is a specialization of a table, with zero or more specified dimensions, where each dimension is an integer range. The result of executing the array function is to create an array. For example, V := array(1..10) creates a one dimensional array (a Maple vector) of length 10 but with no explicit entries. The command A := array(1..m,1..n) creates a two dimensional array (a Maple matrix) with m rows and n columns.
- All parameters to the array function are optional and may appear in any order. The bounds parameter is a sequence of integer ranges which must appear consecutively. If the bounds are not specified then they are deduced from the list of initial values.
- The indexfcn can be a procedure or a name specifying how indexing is to be performed - see indexfcn for more information. The built-in indexing functions are symmetric, antisymmetric, sparse, diagonal, and identity. If indexfcn is not specified, then ``ordinary'' indexing is used.
- The list of initial values may be a list of equations (cf. tables), or a list of values (one-dimensional), or a nested list of lists (row-by-row).
- The map function can be used to apply a function to each entry of an array. For example, map (simplify, A) simplifies each entry of the array A.
- Arrays have special evaluation rules (like procedures) so that if the name A has been assigned an array then A evaluates to the name A and eval(A) yields the actual array structure.
- The op function can be used to pick apart an array structure. op(1,eval(A)) yields indexfcn, if one is specified. If no indexfcn has been specified, it returns nothing. op(2,eval(A)) yields bounds; and op(3,eval(A)) yields entries, where entries is a list of equations corresponding to the explicit entries in the array (cf. entries).

## Examples:

```
> v := array(1..4):
  for i to 3 do v[i] := i^2 od:
  print(v);
```

$$[1, 4, 9, v_4]$$

```
> v[2];
```

$$4$$

```
> v[0];
```

```
Error, 1st index, 0, smaller than lower array bound 1
```

```
> A := array(1..2,1..2):
  A[1,2] := x:
  A[1,1];
```

$$A_{1,1}$$

```
> A[1,2];
```

$$x$$

```
> print(A);
```

$$\begin{bmatrix} A_{1,1} & x \\ A_{2,1} & A_{2,2} \end{bmatrix}$$

```
> A := array( symmetric, 1..2,1..2, [ [1,x], [x,x^2] ] ):
  op(1,eval(A));
```

$$symmetric$$

```
> op(2,eval(A));
```

$$1 .. 2, 1 .. 2$$

```
> op(3,eval(A));
```

$$[(1, 2) = x, (1, 1) = 1, (2, 2) = x^2]$$

```
> map(diff,A,x);
```

$$\begin{bmatrix} 0 & 1 \\ 1 & 2\,x \end{bmatrix}$$