`output` — library for formatted output

Table of contents

`output::ordinal` – ordinal numbers

`output::ordinal` converts an integer to the corresponding english ordinal number.

Call(s):

- `output::ordinal(integer)`

Parameters:

`integer` — integer

Return Value: a string with the english ordinal number

Related Functions: `info`, `userinfo`, `print`

Details:

- `output::ordinal` converts an integer to the corresponding english ordinal number. The return value is a string and can be used in messages.

Example 1. Convert some numbers to the corresponding english ordinal string:

```
>> map([0, 1, 2, 3, 4, 22, 134, 2001], output::ordinal)
 ["0th", "1st", "2nd", "3rd", "4th", "22nd", "134th", "2001st"]
```

Changes:

- `output::ordinal` is a new function.

`output::tableForm` – printing objects in table form

`output::tableForm(obj)` prints the object `obj` in table form.

Call(s):

- `output::tableForm(object <, seperator> <, options>)`

Parameters:

`object`	—	list or set of any MuPAD objects
`seperator`	—	string between columns
`options`	—	one single option or a set of options

Options:

`Unquoted`	—	strings will be printed without quotes
`Unique`	—	all columns are of the same width
`Left`, *`Center`*, *`Right`*	—	the entries will be aligned left, center or right
`Sort``= procedure`	—	the entries will be sorted
`Output``= file`	—	Output into a file.

Return Value: no return value

Related Functions: `output::tree`, `print`, `fopen`, `fprint`, `fclose`

Details:

- `output::tableForm` prints the elements of the given object in table form. The width of the table depends on the size of `TEXTWIDTH`. The width of a column depends on the widest entry in this column.
- If `seperator` is given then it is appended to each object. Appending spaces to the separator results additionally space between columns. By default the separator is one space.
- Without the option *`Sort`* the objects will be converted to strings an then sorted alphabetically.

Option <*`Unquoted`*>:

- The output function `fprint` will be calles with the option *`Unquoted`*. E. g. strings will be printed without quotes.

Option <*`Unique`*>:

- All columns will be printed with the same width, the widest column determines the width.

Option <*`Left`*, *`Center`*, *`Right`*>:

- The entries of each column will be aligned left, center or right.

Option <`Sort = procedure`>:

- The entries will be sorted with the given procedure. Without a procedure the entries will be printed unsorted.

Option <`Output = file`>:

- Output into a file. If `file` is a string, a file named `file` will be opened and overwritten and closed after writing. If `file` is a file descriptor the table will be appended to `file` without closing `file`.

Example 1.

```
>> output::tableForm(anames(DOM_DOMAIN))
```

```
 ClientObject DOM_ARRAY    DOM_BOOL   DOM_COMPLEX
 DOM_DOMAIN   DOM_EXEC     DOM_EXPR   DOM_FAIL
 DOM_FLOAT    DOM_FUNC_ENV DOM_IDENT  DOM_INT
 DOM_LIST     DOM_NIL      DOM_NULL   DOM_POINT
 DOM_POLY     DOM_POLYGON  DOM_PROC   DOM_PROC_ENV
 DOM_RAT      DOM_SET      DOM_STRING DOM_TABLE
 DOM_VAR      Plot         Pref       misc
 output       plot         polylib    specfunc
 stdlib
```

```
>> output::tableForm(map(anames(DOM_DOMAIN), expr2text), "  ", {Unique, Cen
```

```
 "ClientObject"   "DOM_ARRAY"      "DOM_BOOL"
 "DOM_COMPLEX"    "DOM_DOMAIN"     "DOM_EXEC"
   "DOM_EXPR"      "DOM_FAIL"     "DOM_FLOAT"
 "DOM_FUNC_ENV"   "DOM_IDENT"      "DOM_INT"
   "DOM_LIST"      "DOM_NIL"       "DOM_NULL"
  "DOM_POINT"      "DOM_POLY"    "DOM_POLYGON"
   "DOM_PROC"    "DOM_PROC_ENV"    "DOM_RAT"
   "DOM_SET"      "DOM_STRING"    "DOM_TABLE"
   "DOM_VAR"         "Plot"          "Pref"
     "misc"         "output"         "plot"
   "polylib"       "specfunc"       "stdlib"
```

```
>> output::tableForm(anames(DOM_DOMAIN), ", ", {Sort = FALSE, Right})
```

```
ClientObject,    DOM_ARRAY,   DOM_BOOL,  DOM_COMPLEX,
  DOM_DOMAIN,     DOM_EXEC,   DOM_EXPR,     DOM_FAIL,
   DOM_FLOAT, DOM_FUNC_ENV,  DOM_IDENT,      DOM_INT,
    DOM_LIST,      DOM_NIL,   DOM_NULL,    DOM_POINT,
    DOM_POLY,  DOM_POLYGON,   DOM_PROC, DOM_PROC_ENV,
     DOM_RAT,      DOM_SET, DOM_STRING,    DOM_TABLE,
     DOM_VAR,         Plot,       Pref,         misc,
      output,         plot,    polylib,     specfunc,
      stdlib
```

Changes:

- `output::tableForm` used to be `misc::tableForm`.
- output of elements of sets or lists, only one string as separator
- new options *`Unique`*, *`Left`*, *`Center`*, *`Right`*, *`Sort`* and *`Output`*
- enhanced alignment, column formatting, output to files

`output::tree` – display of trees

`output::tree` formats internally represented trees to display graphically.

Call(s):

- `output::tree(Tree <, indentdepth <, charlist> <, options>>)`

Parameters:

`Tree`	—	the tree, given as a special list
`indentdepth`	—	indent depth for each subtree
`charlist`	—	the chars that illustrate the tree structure
`options`	—	option *`Small`*

Options:

`Small` — suppresses the display of a space line between every tree entry

Return Value: an string object to display

Related Functions: `adt::Tree`, `prog::exprtree`

Details:

- `output::tree` displays trees given as specially MuPAD lists.
- The first object of the list is the root of the tree. All further objects are nodes or subtrees of the tree. A subtree is again a special list (as described), and any other MuPAD object will be interpreted as node of the tree.
- The elements of the tree will be printed by MuPAD, when the tree will be displayed, so it's recommended to use strings as objects or objects with a well defined display.
- The return value is a string that contains all chars to display the tree. With functions `print` and `fprint` *and* the option `Unquoted` the tree can be displayed.
- The parameter `charlist` is a list with five characters. The default value is `["|", "+", "-", "'", " "]`. The characters have the following meaning (described in the order of the list).
- The vertical lines of the tree, the connection between vertical and horizontal line (i.e., an arm, but not the last arm), an arm (vertical line), the last connection to an arm in a subtree, a char between an arm and the description of the arm.

Option <*Small*>:

- suppresses the display of a space line between every tree entry to reduce the height of the tree

---

Example 1. `output::tree` displayes special nested lists as trees:

```
>> TREE := ["a1", "a2", ["b1", "b2", ["c1", "c2"], "b3"],
                  ["d1", "d2", "d3"]]:
   print(Unquoted, output::tree(TREE))
```

```
a1
|
+-- a2
|
+-- b1
|   |
|   +-- b2
|   |
|   +-- c1
```

```
|   |   |
|   |   '-- c2
|   |
|   '-- b3
|
'-- d1
    |
    +-- d2
    |
    '-- d3
```

```
>> print(Unquoted, output::tree(TREE, 3, Small)):
```

```
a1
+- a2
+- b1
|  +- b2
|  +- c1
|  |  '- c2
|  '- b3
'- d1
   +- d2
   '- d3
```

The chars can be defined by the user:

```
>> print(Unquoted, output::tree(TREE, 6, ["|", "|", ".", "\\", " "])):
```

```
a1
|
|.... a2
|
|.... b1
|     |
|     |.... b2
|     |
|     |.... c1
|     |     |
|     |     \.... c2
|     |
|     \.... b3
|
\.... d1
      |
      |.... d2
      |
      \.... d3
```

Changes:

- `output::tree` is a new function.