

Paradigmas de Programação

Prof. Maicon R. Zatelli

LISP - Programação Funcional
Listas e Tuplas

Universidade Federal de Santa Catarina
Florianópolis - Brasil

2018/1

Construtores de listas

- '
- cons
- list

LISP

Exemplos

- Lista vazia
()
- Constrói uma lista de inteiros
'(1 2 3 4)
Resulta em (1 2 3 4)
- Constrói uma lista de caracteres
'(#\a #\b #\c #\d)
Resulta em (#\a #\b #\c #\d)
- Constrói uma dupla
(cons 2 3)
Resulta em (2 . 3)

Exemplos

- Adiciona o primeiro parâmetro à lista do segundo parâmetro

```
(cons 3 '(4 5 6))
```

Resulta em: (3 4 5 6)

```
(cons '(1 2 3) '(4 5 6))
```

Resulta em: ((1 2 3) 4 5 6)

- Constrói uma lista a partir de n argumentos

```
(list 1 2 3 4)
```

Resulta em: (1 2 3 4)

```
(list '(1 2 3) '(4 5 6))
```

Resulta em: ((1 2 3) (4 5 6))

LISP

```
(defun comprimento (lista)
  (if (null lista) ;testa se lista é vazia
      0
      (+ 1 (comprimento (cdr lista))) ;cdr retorna cauda
  )
)

(defun main()
  (write-line (write-to-string (comprimento '(1 2 3 6 5))))
)

(main)
```

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

LISP

```
(defun igual (lista1 lista2)
  (if (and (null lista1) (null lista2))
      T
      (if (or (null lista1) (null lista2))
          NIL
          (if (= (car lista1) (car lista2))
              (igual (cdr lista1) (cdr lista2))
              NIL)
      )
  )
)
```

- Se as duas listas forem vazias, retorna T
- Se uma ou outra for vazia, retorna NIL
- Se a cabeça for igual, retorna se a cauda é igual, caso contrário retorna NIL

Continua ...

LISP

```
(defun main()
  (write-line (write-to-string (igual '(1 2 1 4 5) '(1 2 3 4 5))))
  (write-line (write-to-string (igual '(1 2 3 4 5) '(1 2 3 4 5))))
)

(main)
```

- Se as duas listas forem vazias, retorna T
- Se uma ou outra for vazia, retorna NIL
- Se a cabeça for igual, retorna se a cauda é igual, caso contrário retorna NIL

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

LISP

```
(defun igual2 (lista1 lista2)
  (cond
    ((and (null lista1) (null lista2)) T)
    ((or (null lista1) (null lista2)) NIL)
    ((= (car lista1) (car lista2)) (igual2 (cdr lista1) (cdr lista2)))
    (t NIL)
  )
)

(defun main()
  (write-line (write-to-string (igual2 '(1 2 1 4 5) '(1 2 3 4 5))))
  (write-line (write-to-string (igual2 '(1 2 3 4 5) '(1 2 3 4 5))))
)

(main)
```

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

LISP

```
(defun dobro (lista)
  (if (null lista)
      ()
      (cons (* (car lista) 2) (dobro (cdr lista)))
  )
)

(defun main()
  (write-line (write-to-string (dobro '(1 2 3 4 5))))
)

(main)
```

- Constrói uma nova lista formada pelo dobro de cada elemento da lista recebida

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

LISP

```
(defun ordenacao (lista)
  (if (null lista)
      ()
      (add (car lista) (ordenacao (cdr lista)))
  )
)

(defun add (i lista)
  ;...
)
```

- Caso a lista for vazia, retorna ela mesma
- Caso contrário, adiciona a cabeça da lista na cauda já ordenada (recursivamente)

Continua ...

LISP

```
(defun add (i lista)
  (if (null lista)
      (cons i ())
      (if (<= i (car lista))
          (cons i lista)
          (cons (car lista) (add i (cdr lista)))
      )
  )
)
```

- A função add adiciona um elemento numa lista já ordenada.
- Se adicionando em uma lista vazia, retorna uma lista formada pelo próprio elemento.
- Se o elemento é menor que a cabeça da lista, adiciona ele no começo da lista.
- Se o elemento é maior que a cabeça da lista, adiciona ele entre a cabeça e alguma posição na cauda da lista (recursivamente).

Continua ...

LISP

```
(defun main()
  (write-line (write-to-string (ordenacao '(2 43 5 2 1 5 1 2)))))
)

(main)
```

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

LISP

```
(defun filtrar (f lista)
  (if (null lista)
      ()
      (if (funcall f (car lista))
          (cons (car lista) (filtrar f (cdr lista)))
          (filtrar f (cdr lista)))
      )
  )
)
```

- A função filtrar acima aplica a função *f* em cada elemento da lista e se retornar True, adiciona o elemento na lista e prossegue fazendo isso recursivamente na cauda da lista, até que não haja mais nenhum elemento.

LISP

```
(defun ehpar (n)
  (= (mod n 2) 0)
)

(defun main ()
  (write-line (write-to-string
    (filtrar (function ehpar) '(2 43 5 54 2 1 5 6 1 2)))
  )
)

(main)
```

Tente executar este código em:

http://www.compileonline.com/execute_lisp_online.php

Outras operações com listas:

- Concatenação: `concatenate`
(`concatenate 'list '(1 2 3 4) '(4 6) "abc"`)
Resulta em (1 2 3 4 4 6 a b c)
- Retorna a cabeça da lista: `car` ou `first`
(`car '(1 2 3 4)`)
Resulta em 1
- Retorna a cauda da lista: `cdr` ou `rest`
(`cdr '(1 2 3 4)`)
Resulta em (2,3,4)
- Retorna uma lista com o último elemento: `last`
(`last '(1 2 3 4)`)
Resulta em (4)

LISP - Alguns Links Úteis

- <https://www.tutorialspoint.com/lisp/index.htm>
- https://www.tutorialspoint.com/lisp/lisp_lists.htm

Ver atividade no Moodle