ESTUDO PRÁTICO DIRIGIDO – ROTEIRO LISP 01

MATA56 – Paradigmas de Linguagens de Programação (2023.1)

Profº Claudio Junior

- https://onecompiler.com/commonlisp

1) Avalie as expressões abaixo utilizando o interpretador LISP:

(+ 2 3)	(-5 -1)	(* 3 12)	(/ 12 4)
(/ 18 5)	(> 7 1)	(< 1 4)	(<= 7 7)
(= 4 4)	(>= 9 1)	(/= 6 1)	(or (< 2 3) (> 2 3))
(= 'a 'b)	(eq 'a 'b)	(eq 'a 'a)	(eq '(a b) '(a b))
(equal '(a b) '(a b))	(cons 1 '(2 3 4)	(car '(1 2 3))	(cdr '(1 2 3))
(car (cdr '(1 2 3)))	(cdr (cdr '(1 2 3)))	(length '(1 2 3))	(nth 2 '(1 2 3))
(cond ((= 1 2) 'a) ((> 2 3) 'b) ((<3 4) ‘c))	(if (> 5 3) 'maior 'menor)	(cond ((> 5 3) 'maior)	(let ((x 5) (y 3)) (+ x y))

Esta atividade tem por objetivo apresentar ao aluno o funcionamento das avaliações de expressões básicas em LISP. O aluno deve aproveitar a oportunidade e realizar testes diversos, criando e alterando as expressões apresentadas.

2. Digite e avalie o código abaixo, fazendo alterações para testar o comportamento da função:

```
(defun calcular-area-retangulo (base altura)
  (* base altura))

; Solicitar entrada do usuário para base e altura
(format t "Digite a base do retangulo: ")
(let ((base (read)))
  ;imprimir uma linha em branco
  (terpri)
  (format t "Digite a altura do retangulo: ")
  (let ((altura (read)))
    ;imprimir uma linha em branco
    (terpri)
    ; Chamar a função calcular-area-retangulo com os valores digitados pelo usuário
    (format t "A area do retangulo e: ~d m2"
            (calcular-area-retangulo base altura))))
```

Esta atividade tem por objetivo aproximar o aluno à escrita e desenvolvimento do programa em LISP ao utilizar comandos e estruturas básicas de leitura, atribuição e apresentação de dados. O aluno é convidado a alterar a estrutura base, realizando testes para aprimorar seu conhecimento.

ESTUDO PRÁTICO DIRIGIDO – ROTEIRO LISP 01

MATA56 – Paradigmas de Linguagens de Programação (2023.1)

Profº Claudio Junior

3. **Altere o programa LISP abaixo de modo que o valor do número seja digitado, ou seja, a execução da rotina deverá constar a alteração da chamada da função para verifica-numero num**

```
(defun verifica-numero (valor)
  (when (= valor 0)
    (format t "O valor é igual a zero.~%"))
  (when (> valor 0)
    (format t "O valor é maior que zero.~%"))
  (when (< valor 0)
    (format t "O valor é menor que zero.~%")))

(verifica-numero 0)   ; Exibe "O valor é igual a zero."
(verifica-numero 5)   ; Exibe "O valor é maior que zero."
(verifica-numero -3)  ; Exibe "O valor é menor que zero."
(verifica-numero 10)  ; Não exibe nenhuma mensagem, nenhuma condição é verdadeira.
```

Se no exercício anterior o aluno foi convidado a repetir os códigos apresentados, agora existe a oportunidade do aluno alterar um programa e implementar conhecimentos desenvolvidos anteriormente.

4. **Altere o programa abaixo para que só sejam impressos números ímpares da lista:**

```
(defun percorrer-lista (lista)
  (dolist (elemento lista)
    (if (listp elemento)
        (percorrer-lista elemento) ; Chamada recursiva se o elemento for uma lista
        (format t "~a~%" elemento)))) ; Realiza alguma ação com o elemento

(percorrer-lista '(1 2 (3 4 5) 6 7))
```

O objetivo desse exercício é fazer com que o aluno tenha conhecimento da programação LISP usando lista, assunto que será abordado na próxima aula.

5. **Você precisa criar um programa em LISP que verifica a idade de uma pessoa e exibe uma mensagem de acordo com a faixa etária. O programa deve pedir ao usuário que insira sua idade e, em seguida, exibir uma mensagem apropriada com base na idade fornecida. Utilize as estruturas de controle if e when para implementar a lógica do programa.**

Requisitos:
O programa deve solicitar ao usuário que insira sua idade.
O programa deve utilizar a função read para ler a entrada do usuário.
O programa deve utilizar a expressão let para armazenar o valor lido em uma variável.
O programa deve utilizar as estruturas de controle if e when para verificar a idade e exibir a mensagem apropriada de acordo com as seguintes condições:

ESTUDO PRÁTICO DIRIGIDO – ROTEIRO LISP 01

MATA56 – Paradigmas de Linguagens de Programação (2023.1)

Profº Claudio Junior

Se a idade for menor que 18, exiba a mensagem "Você é menor de idade."
Se a idade estiver entre 18 e 64 (inclusive), exiba a mensagem "Você é um adulto."
Se a idade for maior que 64, exiba a mensagem "Você é um idoso."
O programa deve imprimir a mensagem correspondente na saída.

O objetivo desse exercício é apresentar uma situação problema para que o aluno, como seu próprio esforço, desenvolva um programa em LISP. Os recursos necessários para desenvolver o programa foram abordados nos exercícios anteriores.

6. **Problema: Um médico está precisando criar um programa para calcular imediatamente o Indice de Massa Corporal (IMC) de uma pessoa e que apresente a classificação de acordo com o IMC calculado. Sua tarefa é desenvolver esse programa para o médico usando a Paradigma Funcional e a Linguagem LISP.**

Requisitos:
O programa deverá solicitar que o usuário informe o peso (kg) da pessoa
O programa deverá solicitar que o usuário informe a altura (m) da pessoa
O programa deverá usar a expressão let para armazenar os valores lidos do peso e da altura em suas respectivas variáveis
O programador deverá avaliar qual a melhor opção como estrutura de fluxo de controle (cond, if ou when) para determinar a classificação da pessoa de acordo com o IMC:
Abaixo do peso quando o IMC for menor do que 18.5
Peso ideal quando o IMC for menor do que 24.9 e maior ou igual a 18.5
Sobrepeso quando o IMC for menor do que 29.9 e maior ou igual a 24.9
Obesidade Grau 1 caso o IMC seja menor do que 34.9 e maior ou igual a 29.9
E se for maior ou igual a 34.9 a classificação será Obesidade Grau 2
O programa deve emitir duas mensagens:
1 - Seu IMC é <apresentar o imc calculado com duas casas decimais>
2 - Sua classificação é <apresentar a mensagem de acordo com a classificação>

Dicas:
- Divida as responsabilidades, cada funçao com um objetivo (lembram do PROLOG?);
- O let pode ser usado, mas atentem para o escopo;
- ~2,2f~% pode ser usado para formatar o imc.
- ~a~%% pode ser usado para apresentar mensagens
- Apos testar o seu programa e comprovar que está funcionando, experimente criar outras versões dele usando outras estrutura de controle. Por exemplo: se usou if, tente usar cond e/ou when. Faça testes usando let e let* caso seja possível.

O desafio aqui é a criação do programa LISP, com utilização de mais funções e exigirá a capacidade de organização das ideias