**FUNDAÇÃO CENTRO DE ANÁLISE, PESQUISA E INOVAÇÃO TECNOLÓGICA.**
**INSTITUTO DE ENSINO SUPERIOR FUCAPI**
**COORDENAÇÃO DE GRADUAÇÃO EM**
**CIÊNCIA DA COMPUTAÇÃO, ENGENHARIA DA COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO**

# COMMON LISP

**GABRIEL GONÇALVES BATISTA GAMBARRA**
**IGOR FERNANDES GONÇALVES**
**ORLEY DO AMARAL FERREIRA**
**THIAGO TRIBUZY GAVANSKI**

**MANAUS**
**2017**

**GABRIEL GONÇALVES BATISTA GAMBARRA**
**IGOR FERNANDES GONÇALVES**
**ORLEY DO AMARAL FERREIRA**
**THIAGO TRIBUZY GAVANSKI**

# COMMON LISP

Trabalho apresentado ao Curso de Graduação em Ciência da Computação, Engenharia da Computação e Sistemas de Informação, do Instituto de Ensino Superior FUCAPI – CESF como requisito parcial para a disciplina de Paradigmas de Linguagem de Programação.

Orientador (a): Prof. William Malvezzi, M.Sc.

**MANAUS**
**2017**

**RESUMO**

Este trabalho aborda a linguagem LISP com seus principais conceitos com a finalidade de elaborar um processo de escolha fundamentada em inteligência artificial através da criação de uma árvore de decisão. Durante as fases de elaboração, são mostradas as principais funções e seus funcionamentos para em seguida abordar as funções que foram utilizadas. Uma árvore de decisão foi construída com o objetivo de explorar os recursos oferecidos pela linguagem. Buscando implementar tais objetivos, houveram pesquisas e reuniões que resultaram no desenvolvimento da aplicação que auxilia o usuário na busca de uma resposta de acordo com os dados fornecidos.

Palavras-chave: LISP, Inteligência artificial, Árvore de decisão.

**SUMÁRIO**

# 1. INTRODUÇÃO

O homem no decorrer dos anos sempre buscou maneiras mais customizadas de obter informação, durante esse período foram desenvolvidos métodos, assim como criados novas fontes de pesquisa. Com o passar do tempo a quantidade de informação se tornou exorbitante ao ponto de pequenas pesquisas serem direcionadas para variadas possibilidades de respostas e fontes, isso passou a causar imprecisão das respostas. Trabalhando nesse contexto, este projeto buscou explorar uma alternativa de ferramenta de busca por informações mais direcionadas. Neste caso, buscou-se implementar uma árvore de decisão, técnica de decisão baseada em inteligência artificial, para a tratativa deste problema. A técnica da árvore de decisão foi escolhida com a finalidade de desenvolver interação com o usuário. Nesse sentido, pela necessidade de usar uma linguagem confiável e rápida que fosse criada especialmente para implementação de técnicas de inteligência artificial e que fornecesse simplicidade na criação, foi adotada a linguagem LISt Processing(LISP). Vale ressaltar que LISP foi utilizado somente para implementar a técnica de inteligência. Nas escolha do modelo da interface buscou-se características de simplicidade e fácil interação, nesse caso construiu-se uma interface do tipo web em PHP e HTML. Por fim, esse este documento aborda os principais conceitos da linguagem LISP com o objetivo de trazer uma visão geral da linguagem.

## 1.1 PROBLEMA

A crescente quantidade de informação presente nos meios tecnológicos transformou a busca do conhecimento um processo exaustivo. Dentre esse nicho de procura, as pessoas passaram a pesquisar primeiramente em sites, redes sociais ou blogs por conteúdo que suprima suas dúvidas, porém tais fontes nem sempre são confiáveis e objetivas, além do alto acúmulo de informações disponíveis.

## 1.2 OBJETIVOS

Neste tópico será abordada a finalidade ao qual o trabalho proposto e em seguida os objetivos específicos necessários à sua construção.

### 1.2.1 Objetivo Geral

Criar uma aplicação que leva o usuário a determinar, com base nas perguntas e respostas, um possível diagnóstico.

### 1.2.2 Objetivos Específicos

- Criar um banco de dados das DSTs com seus respectivos sintomas
- Desenvolver a árvore de decisão baseada no banco de dados
- Programar a árvore utilizando a linguagem LISP.
- Desenvolver a interface em PHP
- Ligar a interface ao algoritmo de decisão do LISP.
- Testar a operabilidade da aplicação

## 1.3 JUSTIFICATIVA

A necessidade de ter informação precisa e a falta de meios simplificados impulsiona o ser humano a buscar métodos customizados que possibilitem o aprimoramento deste processo. O software proposto visa suprir a esta escassez e fornece através de sua interface simplificada um método mais simples e amigável ao usuário.

## 1.4 METODOLOGIA

Para o desenvolvimento deste trabalho, foram elaboradas pesquisas em sites e livros sobre as doenças e seus sintomas. Além disso, foi elaborado análise de códigos e modelos posteriormente criados de árvores de decisão em LISP. Com o objetivo da criação do banco de dados foram utilizadas planilhas eletrônicas para a organização de informações previamente filtradas qualitativamente. Em relação ao desenvolvimento do projeto, foram estabelecidas três reuniões semanais, duas no âmbito acadêmico e uma em meio privado. As duas primeiras reuniões foram utilizadas para a discussão do andamento do projeto e a última para implementação do software. Para a criação da interface, foi-se utilizado servidor local (localhost) a fim de compilar a linguagem PHP: Hypertext Preprocessor (PHP). Na última fase, um protótipo foi elaborado para executar testes de aceitação, desempenho e precisão da árvore de decisão.

# 2. COMMON LISP

## 2.1 Valores e Tipos

Na linguagem LISP, variáveis não são tipadas, mas sim os dados relacionados aos objetos. Existem dois tipos de dados principais em LISP, são eles: escalares e as estruturas de dados. Os escalares são tipos de números, caracteres símbolos, etc. Por sua vez, as estruturas de dados podem ser listas, vetores, bit-vetores e strings. Qualquer variável pode receber um objeto seja qual for o seu tipo, a não ser que o mesmo já tenha sido declarado expressamente.

Os tipos de dados são organizados em uma hierarquia. Um tipo de dados é um conjunto de objetos LISP e diversos outros objetos podem pertencer a um tal conjunto. Um objeto pode ter qualquer tipo destes: array, fixnum, package, simple-string, atom, float, pathname, simple-vector, bignum, function, random-state, single-float, bit, hash-table, ratio, standard-char, bit-vector, integer, rational, stream, character, keyword, readtable, string, [common], list, sequence, [string-char], compiled-function, long-float, short-float, symbol, complex, nill, signed-byte, t, cons, null, simple-array, unsigned-byte, double-float, number, simple-bit-vector, vector. Além dos tipos predefinidos, a linguagem ainda permite a criação de estruturas, que podem ser definidas através da função defstruct, a partir deste momento, o nome da estrutura passa a ser um tipo aceito pelo LISP.

## 2.2 Declarações de constantes, variáveis, tipos, procedimentos

Para DEITAL (2011 p.24), “Uma variável é um local na memória do computador em que um valor pode ser armazenado para ser usado por um programa.” Em LISP, toda e qualquer variável é representada por um símbolo, o nome da variável passa a ser este símbolo e tem seu valor armazenado nesta célula.

Assim como em outras linguagem, as variáveis podem ser globais ou locais. Variáveis globais têm seus valores fixos durante a execução do programa ou até que tenham o mesmo especificamente alterado. Como não existe declaração de tipo em lisp, o valor deve ser especificado diretamente ao símbolo que o representa. As locais, por sua vez, são definidas dentro de um certo procedimento. Os argumentos chamados de parâmetros também são considerados variáveis locais e são acessíveis somente durante sua respectiva função. Tanto as globais quanto as locais podem ser criadas através da função setq, entretanto, as locais também podem ser criadas através dos comandos let ou prog.

Segundo Wagner, (Docs Microsoft. Constantes (Guia de programação em C#)), “As constantes são valores imutáveis que são conhecidos no tempo de compilação e não são alterados durante a vida útil do programa”. Em LISP, constantes nunca tem seu valor alterado e podem ser declaradas através da função defconstant.

## 2.3 Expressões e Atribuições

Para LUGER (2004, p.620), “Os elementos sintáticos da linguagem de programação LISP são expressões simbólicas, também conhecidas como expressões-s: uma expressão s pode ser um átomo ou uma lista. Átomos LISP são as unidades sintáticas básicas da linguagem e incluem tanto números como símbolos. Átomos simbólicos são compostos por letras, números e os seguintes caracteres alfanuméricos *-+/@$%^&_<>.”. Todas as operações básicas e avançadas podem ser realizadas em LISP, assim como as comparações. Por exemplo, a soma pode ser feita através do operador `+` da seguinte forma: (+ 2 5), o que resultará em 7. Uma comparação é feita do mesmo jeito.

## 2.4 Comandos

As funções podem ser definidas através do comando defun, a mesma, precisa de três argumentos, um nome para a função, os seus parâmetros e um corpo. Sua sintaxe é (defun nome (lista de parâmetros) " corpo). Uma função não necessariamente precisa de parâmetros, quando for o caso ela deve ser declarada como uma lista vazia, por exemplo: lista(). O corpo de uma função pode contém um número infinito de expressões.

Entretanto, em LISP, existe outro método de criar um função, através do comando lambda. *“A linguagem Lisp fornece as lambdas. Uma lambda é uma função com todas as características das restantes mas que não está associada a nenhum símbolo. Pode ser vista como uma função sem nome.” (Leitão, 1995)*

Ela é usa usada da mesma forma que qualquer outra função, onde toda e qualquer função corresponde a uma lambda.

## 2.5 Abstrações, parâmetros

*“A abstração de dados é uma forma de aumentar a modularidade.” (Leitão, 1995).*

A abstração de dados está presente em LISP, fazendo com que a construção do código consiga ficar mais clara, procurando se aproximar mais da realidade, permitindo isolar como os dados vão ser utilizados através das barreiras de abstração.

Existem três tipos de parâmetros em LISP, os opcionais, de resto e de chave. Os opcionais podem ser omitidos de uma função, e são definidos através do &optional na lista de parâmetros formais. Isto quer dizer que todos os parâmetros que forem declarados após, serão opcionais e se não forem declarados, serão atribuídos do valor nulo. Os de resto, por sua vez, qualifica somente o último parâmetro da função e indica que o mesmo vai ficar ligado a uma lista com todos os parâmetros restantes. Existem ainda os de chave, o qualificador &key informa o avaliador que os parâmetros qualificados são ligados através de uma indicação explícita de quem chama a função. Essa indicação é feita designando o nome de cada parâmetro precedido por dois pontos e indicando qual o valor a que ele deve estar ligado. O restante dos parâmetros se comportam como opcionais.

## 2.6 Construções de encapsulamento

O encapsulamento fica evidente quando se trabalha com Common Lisp Object System (CLOS), uma aplicação de LISP voltada a orientação de objetos que permite criar classe e objetos. É através da função defclass que a linguagem consegue criar tais classes, e possui a seguinte sintaxe:

```
(defclass class-name (superclass-name*)
        (slot-description*)
        class-option*))
```

Slots são variáveis que armazenam dados ou campos.

## 3. IMPRESSÕES A RESPEITO DA LINGUAGEM LISP

LISP é a segunda mais antiga linguagem de programação de alto nível. Com o decorrer dos anos, a linguagem foi achando o seu caminho na área de Inteligência Artificial, visto que a informação é processada através de símbolos de forma mais efetiva. LISP é uma linguagem multi-paradigma que fornece suporte tanto para programação funcional quanto procedural.

Embora tenha uma estrutura diferente do que conhecemos, LISP oferece uma diversidade de funções que ajudam a processar informações de maneira eficaz, mantendo também, o código bem enxuto e de clara compreensão.

A linguagem traz consigo uma ampla biblioteca de funções que foram sendo incorporadas para a linguagem ao longo dos anos.

O conhecimento em LISP se torna um grande diferencial por toda sua história de evolução e aplicação específica em Inteligência Artificial.

## 4. REFERÊNCIAS

LUGER, George F. Inteligência Artificial: estruturas e estratégias para a solução de problemas complexos. 4. ed. Porto Alegre: Bookman, 2004.

SEBESTA, Robert. Conceitos de Linguagens de Programação 9. ed. Porto Alegre: Bookman, 2003.

DEITAL, Paul. C Como Programar, 6. ed. São Paulo: Pearson.

SEIBEL, Peter. Practical Common Lisp. Disponível em: <http://www.gigamonkeys.com/book/>. Acesso em 29 out 2017

LEITÃO, Antônio. Introdução à linguagem Lisp. Disponível em: <http://www.dca.fee.unicamp.br/courses/EA072/lisp9596/Lisp9596.html>. Acesso em: 29 out 2017

WAGNER, Bill. Microsoft Docs. Disponível em: <https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/classes-and-structs/constants>. Acesso em 03 nov 2017.

w3ii. Disponível em: <http://www.w3ii.com/pt/lisp/default.html>. Acesso em: 29 out 2017.

tutorialspoint. Disponível em: <https://www.tutorialspoint.com/lisp/>. Acesso em: 29 out 2017