```
?- write('Hello World!'), nl.
Hello World!
true.
```

# PROLOG

Achilles

Professor: Marco Rodrigo Costa

# INTRODUÇÃO

- O TERMO PROLOG É DERIVADO DA EXPRESSÃO "PROGRAMMING IN LOGIC", UMA VEZ QUE É BASEADO EM LÓGICA DE PREDICADOS.
- PROGRAMAR EM PROLOG É BEM DIFERENTE DE PROGRAMAR EM UMA LINGUAGEM PROCEDIMENTAL.
- EM PROLOG SE FORNECEM FATOS E REGRAS PARA UMA BASE DE DADOS.
- ENTÃO SE EXECUTAM CONSULTAS OU (QUERIES) A ESSA BASE DE DADOS.

```
gato(tom).
rato(jerry).
cachorro(spike).
gato(butch).
```

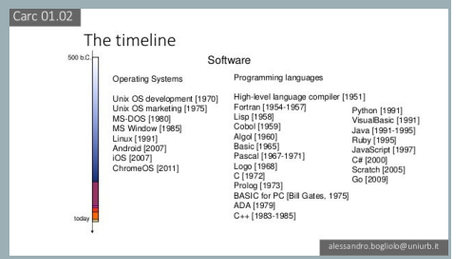
```
?- rato(tom).
 No.
```

# HISTÓRIA

Foi introduzido entre 1971 e 1973 por **Alain Colmerauer** e **Phillipe Roussel**, e recebeu a colaboração de **Robert Kowalski**. Com o propósito inicial de traduzir linguagens naturais.

Em 1975 a equipe recebeu uma tarefa: cada um dos indivíduos deveria traduzir duas páginas de Fortran para a linguagem de máquina e fazer compilar na máquina T1600. A equipe conseguiu realizar a tarefa e assim terminou a história do nascimento de Prolog.

Em 1977, David Warren, especialista em Inteligência Artificial da Universidade de Edimburgh, implementou uma versão eficiente de Prolog, batizada de Prolog-10




# PARADIGMA

## Paradigma Declarativo ^ Paradigma Lógico

Muito utilizado em aplicações de inteligência artificial. Esse paradigma chega no resultado esperado a partir de avaliações lógico-matemáticas. Pessoas que possuem facilidade em lógica de predicados, se sentem confortáveis em entender como uma linguagem nesse paradigma opera.

Descrevem os resultados desejados sem listar explicitamente os comandos ou etapas que devem ser executados.

Consiste em instruções lógicas e o programa é executado procurando provas das instruções

Principais elementos desse paradigma:

- **Proposições**: base de fatos concretos e conhecidos.
- **Regras de inferência**: definem como deduzir proposições.
- **Busca**: estratégias para controle das inferências.

# CARACTERÍSTICAS

## Programar em Prolog consiste em:

- Declarar fatos.
- Definir regras.
- Fazer perguntas.

Todos os dados são tratados como sendo de um único tipo, conhecido como termo, que pode ser:

**Átomos**
**Números**
**Variáveis**
**Termos compostos**
**Listas**
**Strings**
**Fatos**
**Regras**
**Regras Recursivas**
**Avaliação**

## Átomos

Um átomo é uma sequência constituída de letras, números e underscore, mas iniciando com uma letra minúscula.

## Fatos

A estrutura de um fato é formada por um predicado, seus argumentos (objetos) e a instrução é finalizada com um ponto(.)

O predicado é a relação sobre os quais os objetos irão interagir.

Os nomes dos predicados e dos objetos devem sempre começar por letra . A ordem dos objetos poderá interferir no resultado de uma aplicação.

## Variáveis

A variável é uma incógnita, cujo valor é desconhecido, mas quando instanciamos um objeto a ela, a mesma não poderá mudar o objeto.

A busca é realizada na ordem em que os fatos foram passados.

## Questões

A sintaxe das questões varia de acordo com os compiladores existentes, mas basicamente é um fato antecedido de um ponto de interrogação ou o comando apropriado para o tipo de compilador.

Quando uma questão é feita, o Prolog realiza uma busca na sua base de conhecimento, procurando um fato que seja igual ao da questão. Se o fato for encontrado é retornado "YES", caso contrário o programa retornará a saída "NO".

## Regras

Utiliza-se as regras para fazer a construção de questões complexas. Deve especificar situações que podem ser verdadeiras se algumas condições forem satisfeitas.

Para utilizar uma regra, é necessário utilizar o símbolo ":-" que indica uma condição (se).

## OPERADORES DE CONTROLE

# Backtracking

É um procedimento dentro da linguagem Prolog. Uma busca inicial em um programa nesta linguagem segue o padrão Busca em profundidade, ou seja, a árvore é percorrida sistematicamente de cima para baixo e da esquerda para direita. Quando essa pesquisa falha, ou é encontrado um nó terminal da árvore, entra em funcionamento o mecanismo de backtracking. Esse procedimento faz com que o sistema retorne pelo mesmo caminho percorrido com a finalidade de encontrar soluções alternativas.

```
?- pai(roberto,X), mae(vera,X)
```

# CUT (!)

Permite indicar ao Prolog quais sub-objetivos já satisfeitos não necessitam ser reconsiderados ,ou seja ele aborta o processo de backtracking.

Permite que o programa:

- Rode mais rápido
- Ocupe menos memória.
- Em alguns casos, o cut evita que o programa entre em laço infinito.

```
primogenito(X,Y) :- pai(Y,X), masculino(X), !
```

Principais aplicações do cut:

- Unificação de padrões, de forma que quando um padrão é encontrado os outros padrões possí-veis são descartados
- Na implementação da negação como regra de falha
- Para eliminar da árvore de pesquisa soluções alternativas quando uma só é suficiente
- Para encerrar a pesquisa quando a continuação iria conduzir a uma pesquisa infinita, etc.

# Fail

Comando Inversamente ao comando cut, o predicado pré-definido fail sempre falha. O operador de corte pode ser combinado com o predicado fail para produzir uma falha forçada.

É usada para informar: Se a execução chegou até esse ponto, então pode abandonar a tentativa de satisfazer a regra. A conjunção falha devido ao fail, e o objetivo-pai falha devido ao cut.

```
 cabeça :- objetivo<sub>1</sub>, ..., objetivo<sub>n</sub>, !, fail.
```

```
mamifero(X) :- gato(X).
mamifero(X) :- cachorro(X).
mamifero(X) :- rato(X).

gato(tom).
rato(jerry).
cachorro(spike).

gosta(ana,X) :- gato(X),!,fail.
gosta(ana,X) :- mamifero(X).
```

# LINGUAGENS RELACIONADAS

# *Linguagens semelhantes*

## Datalog

Inicialmente introduzida em 1977 por Hervé Gallaire e Jack Minker, Datalog é sintaticamente um subconjunto de Prolog. É principalmente utilizada em querys para consultar bases de dados dedutíveis. Entre principais divergências estão limitações no uso da negação e da recursão, e o fato de Datalog não ser uma linguagem Turing-completa (não pode ser usada para resolver todo e qualquer problema computável).

## Visual Prolog

Também conhecido como *Turbo Prolog,* é uma linguagem baseada em Prolog que, em contraste, é fortemente tipada e apresenta múltiplos paradigmas. Entre os principais objetivos dessa linguagem está a simplificação da programação em paradigma lógico, através da aproximação da linguagem de paradigmas mais populares, eliminando algumas dificuldades da programação nesse paradigma.

# EXEMPLOS

# BIBLIOGRAFIA


WATT, David A. **Programming Language Design Concepts.** Chichester: John Wiley & Sons Ltd., 2004

PEREIRA, Fernando C.N.; SHIEBER, Stuart M. **Prolog and Natural Language Analysis.** Massachussets: Microtone Publishing, 2005. Disponível em: <http://www.mtome.com/Publications/PNLA/prolog-digital.pdf>. Acesso em: 02 abr. 2019

ENDRISS, Ulle. **An Introduction to Prolog Programming**. 2018. Disponível em: <https://staff.science.uva.nl/u.endriss/teaching/prolog/prolog.pdf > Acesso em: 02 abr. 2019

# REFERÊNCIAS


ENDRISS, Ulle. **An Introduction to Prolog Programming**. 2018. Disponível em: <https://staff.science.uva.nl/u.endriss/teaching/prolog/prolog.pdf > Acesso em: 02 abr. 2019

BARANAUSKAS, José Augusto. **Sintaxe e Semântica de Programas Prolog.** Disponível em: <http://dcm.ffclrp.usp.br/~augusto/teaching/ia/IA-Prolog-Sintaxe-Semantica.pdf> Acesso em: 02 abr. 2019

Prolog. In: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Prolog> . Acesso em: 02 abr. 2019

Datalog. In: Wikipédia: a enciclopédia livre. Disponível em: <https://pt.wikipedia.org/wiki/Prolog> . Acesso em: 02 abr. 2019

COLMERAUER, A.; ROUSSEL, P. **The birth of Prolog.** Disponivel em: <http://alain.colmerauer.free.fr/alcol/ArchivesPublications/PrologHistory/19november92.pdf>. Acesso em: 02 Abril.2019.

KOWALSKI, Robert A. **The Early Years of Logic Programming**. Disponivel em: <http://www.doc.ic.ac.uk/~rak/papers/the%20early%20years.pdf> . Acesso em: 02 Abril.2019.

ALBUQUERQUE, Eduardo S.de. **Prolog.** Disponível em: <http://www.inf.ufg.br/~eduardo/lp/alunos/prolog/prolog.html>. Acesso em: 02 abr.2019

A HISTÓRIA DO PROLOG PROGRAMMING LANGUAGE. Disponível em: <http://ptcomputador.com/P/computer-programming-languages/88505.html>. Acesso em: 02 abr. 2019

ROSEBRUGH, Robert. **A Brief History of Prolog.** Disponível em: <https://www.mta.ca/~rrosebru/oldcourse/371199/prolog/history.html>. Acesso em: 02 abr. 2019

PROLOG. 2019. Disponível em: <https://www.cleverism.com/skills-and-tools/prolog/> Acesso em: 02 abr. 2019

ROUCHY, Philippe. **Aspects of Prolog history: Logic Programming and Professional Dynamics.** Disponível em: < https://www.researchgate.net/publication/277325585_Aspects_of_Prolog_history_Logic_Programming_and_Professional_Dynamics>. Acesso em: 02 abr. 2019

INDIKA. **Difference Between Prolog and Lisp.** 20 maio 2011. Disponível em: <https://www.differencebetween.com/difference-between-prolog-and-lisp/> Acesso em: 02 abr. 2019

CArcMOOC 01.02 - Brief history of computing. [S.1.]: Computer Architecture,2016. Disponível em: <https://www.slideshare.net/alessandrobogliolo/carcmooc-0102-brief-history-of-computing> Acesso em: 01 abr. 2019