Linguagens de Programação

PROF. ME. ANDERSON VANIN

Prof. Me. Anderson Vanin

Formação Técnica: Técnico em Eletrônica

Graduação: Bacharel em Ciência da Computação

Pós Graduação: Banco de Dados

Mestrado: Gestão do Conhecimento e Informática (Aplicado a Visão Computacional)

Atuação como professor de Ensino Técnico e Médio no CPS.

Atuação como professor de Ensino Superior no CPS.

Aulas em disciplinas ligadas a Programação e WebDesign.

Cursos extracurriculares em Inteligência Artificial e IoT.

Ementa

Apresentação dos Conceitos de Linguagens de Programação Modernas; Linguagens e seus Diferentes Paradigmas de Programação; Estudo Comparativo de Linguagens: Estrutura de Dados, Estruturas de Controle, Ambiente de Execução. Projeto de Linguagens: Características de uma Boa Linguagem de Programação, Sintaxe e Semântica; Seleção de Linguagens para Aplicações Específicas. Atividades Práticas: (1) Formulação recursiva de Algoritmos: Técnicas recursivas de programação; Linguagens puramente funcionais; Máquinas aderentes à programação funcional; **Linguagem LISP**; exercícios. Formulação de problemas usando lógica: Fatos, regras e especificações; Estruturas de dados; Cálculos de predicados e sua relação com a programação em lógica; **A linguagem PROLOG**. Aplicações. (2) Paralelismo e Concorrência; Programação Concorrente e Paralela: Modelos de Programação e Conceitos; Linguagens Paralelas e Concorrentes: Técnicas de Programação e Ferramentas.

AULA 01

1. Introdução às Linguagens de Programação

No passado escrevia-se programas utilizando apenas linguagens de baixo nível. A escrita é engessada, complexa e muito específica, sendo pouco acessível para os desenvolvedores no geral. Esse tipo de linguagem exige muito conhecimento de quem a programa (inclusive relacionado à forma com que o processador opera uma instrução-máquina).

1. Introdução às Linguagens de Programação

Recentemente foi liberado o código-fonte utilizado no computador que guiou a missão *Apollo* que teve como principal objetivo levar o homem à lua (na tão famigerada corrida espacial entre a União Soviética e os EUA), o *Apollo Guidance Computer*.

https://github.com/chrislgarry/Apollo-11

1. Introdução às Linguagens de Programação

Se você navegar no repositório acima encontrará diversos códigos-fonte com instruções como essas:

```
PROGLARM    CS  DSPTAB +11D
        MASK    OCT40400
        ADS DSPTAB +11D


MULTEXIT    XCH ITEMP1      # OBTAIN RETURN ADDRESS IN A
        RELINT
        INDEX   A
        TC  1

MULTFAIL    CA  L
        AD  BIT15
        TS  FAILREG +2

        TCF MULTEXIT
```

1. Introdução às Linguagens de Programação

São instruções da linguagem ***AGC Assembly Language***, uma variante da ***Assembly***, que por sinal, é de baixo nível.

Um programa escrito em uma dessas linguagens, chamadas de baixo nível, é composto por uma série de instruções de máquina que determinam quais operações o processador deve executar. Essas instruções são convertidas para a linguagem que o processador entende, que é a linguagem binária (sequência de bits 0 e 1), que é categorizada como First-generation programming language (1GL), em livre tradução: linguagem de programação de primeira geração.

1. Introdução às Linguagens de Programação

As linguagens de programação são sistemas de comunicação estruturados que permitem aos humanos instruir computadores. Elas evoluíram desde códigos binários (linguagem de máquina) até linguagens de alto nível, como Python, Java e LISP, simplificando a escrita e manutenção de software. Essa evolução foi impulsionada pela necessidade de abstração, eficiência e adaptação a diferentes paradigmas.

2. Evolução das Linguagens de Programação

A história das linguagens de programação reflete a busca contínua por formas mais eficientes e intuitivas de comunicar instruções aos computadores. Inicialmente, a programação era realizada diretamente em código de máquina, utilizando sequências binárias que o hardware podia interpretar. Com o tempo, surgiram linguagens de montagem (*assembly*), que introduziram mnemônicos para representar instruções, facilitando o trabalho dos programadores.

2. Evolução das Linguagens de Programação

A definição do dicionário Aurélio para “paradigma”:

- **Algo que serve de exemplo geral ou de modelo.**
- Conjunto das formas que servem de modelo de derivação ou de flexão.
- Conjunto dos termos ou elementos que podem ocorrer na mesma posição ou contexto de uma estrutura.

2. Evolução das Linguagens de Programação

Na década de 1950, emergiram as primeiras linguagens de alto nível. O FORTRAN (FORmula TRANslation) foi desenvolvido para cálculos científicos e engenharia, permitindo que os programadores escrevessem fórmulas matemáticas de maneira mais próxima da notação humana. Pouco depois, o COBOL (COmmon Business-Oriented Language) foi criado para aplicações comerciais, focando em processamento de dados empresariais.

2. Evolução das Linguagens de Programação

Nos anos 1960 e 1970, novos paradigmas começaram a se formar. O LISP (LISt Processing) introduziu conceitos de programação funcional e manipulação de listas, tornando-se fundamental em pesquisas de inteligência artificial. O Prolog (PROgramming in LOGic) trouxe a programação lógica, sendo utilizado em sistemas especialistas e processamento de linguagem natural.

2. Evolução das Linguagens de Programação

Com o avanço da computação, surgiram linguagens que incorporavam múltiplos paradigmas e abstrações mais poderosas, como o C++, que adicionou conceitos de orientação a objetos ao C, e o Java, que popularizou a programação orientada a objetos com portabilidade entre plataformas.

3. Paradigmas da Programação

Paradigmas de programação são modelos ou estilos que definem a forma como os programas são construídos e estruturados. Eles influenciam diretamente a maneira como os desenvolvedores pensam e resolvem problemas. Os principais paradigmas incluem:

- **Imperativo**
- **Funcional**
- **Lógico**
- **Orientado a Objetos**

3. Paradigmas da Programação

Imperativo: Baseia-se na execução de instruções sequenciais que alteram o estado do programa. Os programas descrevem como realizar tarefas através de comandos que modificam variáveis e controlam o fluxo de execução. Exemplos de linguagens imperativas incluem C, Java e Python.

Linguagens clássicas como C, C++, PHP, Perl, C#, Ruby etc, "suportam" esse paradigma. Ele é focado na mudança de estados de variáveis.

```
if(option == 'A') {
    print("Opção 'A' selecionada.");
}
```

A impressão só é realizada se o valor da variável *option* for 'A'

3. Paradigmas da Programação

Funcional: Foca na aplicação de funções matemáticas, evitando estados mutáveis e efeitos colaterais. Os programas descrevem o que deve ser computado, enfatizando expressões e declarações em vez de comandos. Linguagens funcionais puras incluem Haskell e Erlang, enquanto linguagens como JavaScript e Python suportam paradigmas funcionais.

É possível desenvolver de forma "funcional" mesmo em linguagens não estritamente funcionais. Por exemplo, no PHP, que é uma linguagem multi paradigma, teríamos:

```
<?php
    $sum = function($value) {
            return $value + 2;
    };

    echo $sum(2); // 4
```

3. Paradigmas da Programação

Lógico: Baseia-se na definição de fatos e regras sobre um domínio de conhecimento, permitindo que o programa derive conclusões através de inferência lógica. O Prolog é a linguagem mais representativa desse paradigma.

Principais elementos desse paradigma:

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

3. Paradigmas da Programação

Lógico:

Exemplo:

- Proposição: Chico é um gato.
- Regra de inferência: Todo gato é um felino.
- Busca: Chico é um felino?

A resposta para a Busca acima precisa ser verdadeira. A conclusão lógica é:

"**Se Chico é um gato e todo gato é felino, então Chico é um felino.**"

3. Paradigmas da Programação

Orientado a Objetos: Organiza o código em "objetos" que encapsulam dados e comportamentos relacionados. Promove a reutilização e modularidade através de conceitos como herança, polimorfismo e encapsulamento. Linguagens como Java, C++ e Python suportam este paradigma.

O paradigma orientado a objetos tem uma grande preocupação em esconder o que não é importante e em realçar o que é importante. Nele, implementa-se um conjunto de classes que definem objetos. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento entre eles.

4. Visão Geral das Linguagens Prolog e LISP

LISP:

Desenvolvida por John McCarthy em 1958, o LISP é uma das linguagens de programação mais antigas ainda em uso. Seu nome deriva de "***LISt Processing***", refletindo sua forte capacidade de manipulação de listas. Características principais incluem:

4. Visão Geral das Linguagens Prolog e LISP

LISP:

- **Programação Funcional**: Enfatiza funções, permitindo passá-las como argumentos e retorná-las de outras funções.
- **Sintaxe Simples**: Utiliza uma notação prefixa uniforme, onde tanto código quanto dados são representados como listas, facilitando a metaprogramação.
- **Aplicações em IA**: Devido à sua flexibilidade e poder de abstração, o LISP tem sido amplamente utilizado em pesquisa de inteligência artificial, incluindo sistemas especialistas e processamento de linguagem natural.

4. Visão Geral das Linguagens Prolog e LISP

Prolog:

Criado na década de 1970 por Alain Colmerauer e Robert Kowalski, o Prolog é uma linguagem de programação lógica que se destaca por:

4. Visão Geral das Linguagens Prolog e LISP

Prolog:

- **Base em Lógica de Predicados**: Os programas são compostos por fatos e regras que descrevem relações entre entidades, permitindo a derivação de conclusões através de mecanismos de inferência.
- **Resolução de Consultas**: Os usuários podem fazer perguntas ao sistema, que tenta prová-las verdadeiras ou falsas com base nos fatos e regras fornecidos.
- **Aplicações em IA**: O Prolog é utilizado em áreas como sistemas especialistas, onde é necessário raciocínio lógico, e no processamento de linguagem natural, devido à sua capacidade de lidar com estruturas gramaticais complexas.

5. Comparação Prolog e LISP

Característica	LISP	PROLOG
Paradigma	Funcional	Lógico
Estrutura de Dados	Baseada em listas; suporte a estruturas complexas	Baseada em termos; fatos e regras definem relações
Execução	Avaliação de expressões; controle explícito de fluxo	Resolução de consultas; controle implícito de fluxo
Aplicações	Inteligência artificial, processamento de linguagem natural	Sistemas especialistas, processamento de linguagem natural
Sintaxe	Notação prefixa; código e dados representados como listas	Declarações de fatos e regras; consultas realizadas ao sistema

6. Exercício Proposto

Elaborar uma tabela comparativa entre os paradigmas de programação mencionados. Inclua na sua tabela dados de: Paradigma, Descrição, Vantagens, Desvantagens, Exemplos de Linguagens.