OS PARADIGMAS DAS LINGUAGENS DE PROGRAMAÇÃO DA QUINTA GERAÇÃO (5 G)

Jonas K. Kahumba

Mestre em Ciencias de Processos Docente Educativo em Instituições de Ensino Superior Militar .

Mestrado em Tecnologia da Informação

Angola , Lobito, AMEx

E-mail: eng.kahumba@gmail.com

Palavras-chave: *Linguagens de Programação,Geraçóes,Paradigmas.*

RESUMO: *O presente artigo pretende contribuir para a formação dos Estudantes e profissionais de analise de sistemas e programação em torno das Novas Tecnologias e conhecimentos das Gerações de linguagens de programação de acordo com diferentes tipos e niveis as mesma , neste caso especificamente a utilização de diferentes linguagens de programação nas distintas áreas do conhecimento humano como profissionais e estudantes dos cursos de engenharia INFORMÁTICA ou Ciências da computação.*

Palabras clave*: lenguajes de programación, las generaciones, Paradigmas*

RESUMEN: *En este artículo se pretende contribuir a la formación de estudiantes y profesionales de sistemas de análisis y programación en torno a las nuevas tecnologías y las generaciones de conocimiento de lenguajes de programación de acuerdo a los diferentes tipos y niveles de la misma, en este caso en concreto utilizando diferentes a lenguajes de programación en diferentes áreas del conocimiento humano como profesionales y estudiantes en cursos de ingeniería informática o la informática.*

Keywords*: Programming languages, generations, Paradigms.*

ABSTRACT: *This article aims to contribute to the training of students and systems analysis professionals and programming around new technologies and knowledge Generations of programming languages according to different types and levels the same, in this case specifically using different to programming languages in different areas of human knowledge as professionals and students in engineering courses computer or computer Science.*

1. Introdução

Uma das principais metas das linguagens de programação é que programadores tenham uma maior produtividade, permitindo expressar suas intenções mais facilmente do que quando comparado com a linguagem que um computador entende nativamente (código de máquina).
Assim, linguagens de programação são projetadas para adotar uma sintaxe de nível mais alto, que pode ser mais facilmente entendida por programadores humanos. Linguagens de programação são ferramentas importantes para que programadores e engenheiros de software possam escrever programas mais organizados e com maior rapidez. A medida em que foi surgindo as gerações dos computadores ,também na mesma senda as linguagens de programação foram surgindo com as suas respeitivas classificações e categorias. Hoje nótorio a evolução das linguagens de programação, quando as suas gerações actualmente fala-se em 5 G. [3]

2. Paradigma das linguagens de Programação da 5G

5ª Geração: linguagens do conhecimento São usadas principalmente na área de Inteligência Artificial.Facilitam a representação do conhecimento que é essencial para a simulação de comportamentos inteligentes.

O termo 5ª geração refere-se, especialmente, a sistemas que usam mecanismos da área de inteligência artificial (IA), ou seja, sistemas especialistas, processadores de língua natural e sistemas com bases de conhecimento. Um sistema de 5ª geração armazena conhecimento complexo de modo que a máquina pode obter inferências a partir da informação codificada.

Um LP (Linguagem de Programação) é uma linguagem destinada a ser usada por uma pessoa para expressar um processo através do qual um computador pode resolver um problema.

Paradigma é um modelo interpretativo (ou conceitualização) de uma realidade. Permite organizar as ideias com vista: Ao entendimento dessa realidade. À determinação de qual a melhor forma de atuar sobre essa realidade. Pode dizer-se que um paradigma é um ponto de vista: um ponto de vista que determina como uma realidade é entendida e como se atua sobre ela.

As linguagens de programação podem ser classificadas em relação a três critérios:

Em relação ao nível:

- Baixo nível;
- Médio nível;
- Alto nível;

Em relação à geração:

- 1ª Geração;
- 2ª Geração;
- 3ª Geração;
- 4ª Geração;
- 5ª Geração;

Em relação ao paradigma:

- Imperativo;
- Funcional;
- Lógico;
- Orientado a Objetos;

3. **linguagens baseado no conhecimento:**

Linguagem de programação para resolução de problemas a partir de restrições dadas para o programa em vez de desenvolver algoritmos. São usadas principalmente na área de Inteligência Artificial. Facilitam a representação do conhecimento, o que é essencial para a simulação de comportamentos inteligentes.

Linguagens de programação lógica e linguagens baseadas em restrições geralmente pertencem a 5ª geração.

• Exemplo: **Prolog**

Na década de 80, as linguagens de 5ª geração eram consideradas o futuro da computação. Sendo surgido que elas substituiriam as linguagens de gerações anteriores. **Problema:** desenvolver algoritmos genéricos e eficientes é um problema complexo.

4. **Os principais paradigmas das linguagens programação 4 G e 5 G**

Paradigma é um modelo interpretativo (ou conceitualização) de uma realidade. • Permite organizar as ideias com vista: Ao entendimento dessa realidade;

À determinação de qual a melhor forma de atuar sobre essa realidade. Pode dizer-se que um paradigma é um ponto de vista: um ponto de vista que determina como uma realidade é entendida e como se atua sobre ela.

Algumas linguagens criadas durante a história introduziram novas formas de se pensar sobre programação, resultando em formas distintas de modelagem de soluções de software.

- FORTRAN (imperativa);
- LISP (funcional); –
- Simula (orientadas a objetos);
- Prolog (lógica).

Outras linguagens são o resultado da evolução de linguagens mais antigas, muitas vezes mesclando características de diferentes linguagens existentes. – Por exemplo, C++ é uma evolução do C com características de orientação a objetos, importadas de Simula.

- **Paradigma imperativo** (sequência, atribuição, estado): Basic, Pascal, C, Ada, Fortran, Cobol, Assembly.
- **Paradigma funcional** (função, aplicação, avaliação): Lisp, Haskell, Erlang, Scheme.
- **Paradigma lógico** (relação, dedução): Prolog.
- **Paradigma orientado a objetos** (objeto, estado, mensagem): C++, Java, C#, Eiffel, Smalltalk, Python.
- **Paradigma concorrente** (processo, comunicação (síncrona ou assíncrona)): C++, C#, Java.

Paradigma Imperativo: As linguagens imperativas são orientadas a ações, onde a computação é vista como uma sequência de instruções que manipulam valores de variáveis (leitura e atribuição). Os programas são centrados no conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado. – Paradigma também denominado de procedural, por incluir subrotinas ou procedimentos como mecanismo de estruturação.

Baseia-se na arquitetura de computadores Von Neumann:

- Programas e dados são armazenados na mesma memória;
- Instruções e dados são transmitidos da CPU para a memória, e vice-versa;
- Resultados das operações executadas na CPU são retornadas para a memória.

Subdivide-se em estruturado e não-estruturado. Linguagens não-estruturadas geralmente fazem uso de comandos goto ou jump. Exemplos – Assembly e Basic:

As linguagens estruturadas surgiram objetivando facilitar a leitura e execução de algoritmos – não fazem o uso do **goto**.

Paradigma Funcional:

Características da programação funcional:

- Programas são funções que descrevem uma relação explícita e precisa entre E/S;
- Estilo declarativo: não há o conceito de estado nem comandos como atribuição;

Exemplos de Linguagens Funcionais:

• Haskell, Scheme, Common LISP,CLOS (Common LISP Object System), Miranda;

Vantagens:

- Simplifica a resolução de alguns tipos problemas:
- Prova de propriedades;

- Resolução de programas de otimização;

Desvantagens:

- Problema: o mundo não é funcional!
- Implementações ineficientes;
- Mecanismos primitivos de E/S;

Paradigma Lógico: Paradigma de programação baseado em lógica formal;

– Um programa lógico é equivalente à descrição do problema expressa de maneira formal, similar à maneira que o ser humano raciocinaria sobre ele;

– A programação lógica consiste em declarar fatos, que podem ser relações (associações) ou regras que produzem fatos deduzidos a partir de outros.

Programação em linguagens lógicas requer um estilo mais descritivo;

– O programador deve conhecer os relacionamentos entre as entidades e conceitos envolvidos para descrever os fatos relacionados ao problema;

– Programas descrevem um conjunto de regras que disparam ações quando suas premissas são satisfeitas;

Principal linguagem lógica: Prolog

Vantagens:

- Permite a concepção da aplicação em alto nível de abstração;
- Linguagem mais próxima do raciocínio humano; – Desvantagens:
- Dificuldade em expressar algoritmos complexos;
- Complexidade exponencial;

Paradigma Orientado a Objetos:

- Tratam os elementos e conceitos associados ao problema como objetos;
- Objetos são entidades abstratas que embutem dentro de suas fronteiras, as características e operações relacionadas com a entidade real;

Sugere a diminuição da distância entre a modelagem computacional e o mundo real:

- O ser humano se relaciona com o mundo através de conceitos de objetos;
- Estamos sempre identificando qualquer objeto ao nosso redor;
- Para isso lhe damos nomes, e de acordo com suas características lhes classificamos em grupos;

Sistemas são vistos como coleções de objetos que se comunicam, enviando mensagens, colaborando para dar o comportamento global dos sistemas.

Uma aplicação é estruturada em módulos (classes) que agrupam um estado (atributos) e operações (métodos) sobre este;

A classe é o modelo ou molde de construção de objetos. Ela define as características e comportamentos que os objetos irão possuir.[2]

Vantagens:

- Organização do código;
- Aumenta a reutilização de código;
- Reduz tempo de manutenção de código;
- Ampla utilização comercial; – Desvantagens:
- Menos eficientes;

5. Linguagens Funcionais

Linguagens funcionais são linguagens que evidenciam um estilo de programação diferente das linguagens procedurais, chamado de programação funcional. A programação funcional enfatiza a avaliação de expressões, ao invés da execução de comandos. As expressões nessas linguagens são formadas utilizando-se funções para combinar valores básicos. As linguagens funcionais mais conhecidas são o LISP e o Prolog. A linguagem Scheme também é frequentemente citada, por ser uma variante simplificada do LISP. Diversas outras linguagens funcionais são encontradas na literatura, por exemplo, ASpecT, Caml, Clean, Erlang, FP, Gofer, Haskell, Hope, Hugs, Id, IFP, J, Miranda, ML, NESL, OPAL,e Sisal.

O Prolog é uma linguagem de programação prática e eficiente, introduzida em 1973 por Alain Colmerauer e seus associados na Universidade de Marseille, com o propósito inicial de traduzir linguagens naturais. Em 1977, David Warren da Universidade de Edimburgo, implementou uma eficiente versão do Prolog, batizada de Prolog-10. A partir daí, tornou-se uma escolha natural para a resolução de problemas que envolvem a representação simbólica de objetos e relações entre objetos. O fundamento básico por trás do Prolog é a noção de programação em lógica, onde o processo de computação pode ser visto como uma sequência lógica de inferências. Esta noção desenvolve-se então de modo a resultar em um mecanismo automático de prova de teoremas. Atualmente, o Prolog é utilizado em diversas aplicações na área de computação simbólica, incluindo-se aí: bases de dados relacionais, sistemas especialistas, lógica matemática, prova automática de teoremas, resolução de problemas abstratos e geração de planos, processamento de linguagem natural, projeto de arquiteturas, logística, resolução de equações simbólicas, construção de compiladores, análise bioquímica e projeto de fármacos.

O Prolog é uma linguagem diferente, mas de uma simplicidade marcante. Essa diferença decorre do fato de o Prolog ser uma linguagem funcional, e não uma linguagem procedural. Sua estrutura conceitual está intimamente ligada com a lógica matemática, o que a torna uma ferramenta indispensável para o estudo de lógica. Versões diferentes do

Prolog podem ser encontradas nas mais diversas plataformas, tanto na forma de compiladores como de interpretadores.

O LISP (LISt Processor) é uma linguagem de programação que foi desenvolvida utilizando-se idéias oriundas do estudo da inteligência artificial. Sua constituição é tal que programas escritos em LISP tendem a ser um modelo que emula as habilidades cognitivas do ser humano. O elemento básico utilizado pelo LISP são os símbolos, eminentemente símbolos alfanuméricos. A partir de 1984, o LISP começou a desenvolver-se em diversos dialetos, sem que um deles dominasse marcadamente os outros. Em um esforço conjunto de um seleto grupo de pesquisadores em linguagens de programação, representando diversas instituições, desenvolveu-se o Common Lisp, integrando com sucesso [1].

6. Novas direções das linguagens de programação

As novas direções das linguagens de programação, especialmente das linguagens de quarta geração é a sua aplicação com metodologias orientadas a objetos. Essas linguagens são baseadas nos conceitos de objetos, que agrupam comandos de programação com dados em objetos que podem ser usados o tempo todo durante a execução do programa, o que é muito útil em ambientes de execução paralela.Outra tendência dos ambientes de desenvolvimento de programas é fornecer além das linguagens de programação, um ambiente de geração automática de código, onde o programador especifica através de ferramentas visuais as características do programa e a ferramenta se encarrega de gerar a codificação na linguagem específica. Estas ferramentas são muito difundidas na programação para Windows, e são também chamados de RAD (Desenvolvimento Rápido de Aplicativos). A nova geração das linguagens de programação, que já é chamada por muitas pessoas de quinta geração, é baseada em métodos de consulta e utilizam comandos escritos em linguagens naturais, permitindo uma fácil comunicação com o computador.

7. Conclusão

A forma como as Tecnologias estão a avançar no campo da informática as linguagens de programação estão também a evoluir em gerações distintas para melhor compreensão dos seus paradigmas na programação Hoje é notável o uso das linguagens de programação virada em inteligência artificial e especializações ,para resolução dos problemas específicos.

8. Referencias

[1]**http://edirlei.3dgb.com.br/aulas/clp/CLP_Aula_02_Classificacao_Linguagens_Programacao_2015.pdf**

[2]http://www2.ufersa.edu.br/portal/view/uploads/setores/160/disciplinas/20092/informatica_aplicada/unidade01/02.3_Informatica_Aplicada%20-%20Linguagens.pdf

[3]https://pt.wikipedia.org/wiki/Linguagem_de_programa%C3%A7%C3%A3o_de_quarta_gera%C3%A7%C3%A3o