****

**UNIVERSIDADE DO SUL DE SANTA CATARINA**

**FABIO DELA BRUNA**

**MÁRCIO OZÓRIO TEIXEIRA**

**LINGUAGENS DE PROGRAMAÇÃO E PARADIGMAS**

Tubarão

2009

**SUMÁRIO**

**1 INTRODUÇÃO**......................................................................................................................2

**2 LINGUAGENS DE PROGRAMAÇÃO**..............................................................................3

2.1 CONCEITO..........................................................................................................................3

2.2 BREVE HISTÓRICO...........................................................................................................3

**2.2.1 Linguagens de primeira geração**....................................................................................4

**2.2.2 Linguagens de segunda geração**......................................................................................4

**2.2.3 Linguagens de terceira geração**......................................................................................4

**2.2.4 Linguagens de quarta geração**........................................................................................5

2.3PROPRIEDADES DESEJÁVEIS.........................................................................................5

**3 PARADIGMAS DE PROGRAMAÇÃO**..............................................................................7

3.1 CONCEITO..........................................................................................................................7

3.2 PRINCIPAIS PARADIGMAS DE PROGRAMAÇÃO.......................................................7

**3.2.1 Imperativo**.........................................................................................................................7

**3.2.2 Orientado a Objetos**.........................................................................................................8

**3.2.3 Funcional**...........................................................................................................................9

**3.2.4 Lógico**..............................................................................................................................10

**4 CONCLUSÃO**......................................................................................................................11

**REFERÊNCIAS**.....................................................................................................................12

**1 INTRODUÇÃO**

Os computadores são máquinas que até então não possuem o poder de pensar sozinhos, e para isso eles dependem de nós, seres humanos, para que possam realizar tarefas específicas. Entretanto, eles conseguem entender a linguagem binária, porém, para nós é complicado realizar a programação para um computador somente com 0`s e 1`s, deste modo, foram criadas as linguagens de programação, ou seja, um conjunto de regras usadas para definir um programa de computador, utilizadas por nós numa linguagem adequada e traduzida para o computador para linguagem binária para que possa ser processada por ele. Contudo, as linguagens de programação que são compreensíveis por nós de maneira mais simples também necessitam de regras para que possam ser utilizadas de maneira adequada, desta forma este trabalho aborda o conceito e um breve histórico das linguagens de programação e os principais paradigmas de programação.

**2 LINGUAGENS DE PROGRAMAÇÃO**

2.1 CONCEITO

Segundo Linder (2009), uma linguagem de programação é um vocabulário e um conjunto de regras sintáticas e semânticas utilizadas por programadores e engenheiros de softwares para construir programas de computador. Sendo que, cada linguagem possui um conjunto único de palavras-chaves e uma sintaxe específica que permite ao programador especificar exatamente sobre quais informações um computador vai atuar, como essas informações serão armazenadas ou transmitidas e quais ações devem ser tomadas sobre determinadas situações. Além do mais, oferecem a eles uma linguagem mais próxima da linguagem natural, permitindo assim que tenham maior produtividade em relação à linguagem nativa do computador.

2.2 BREVE HISTÓRICO

De acordo com Rocha (2009), o primeiro projeto de linguagem de programação surgiu para um computador que não existia, sendo idealizada por Ada Lovelace, esposa de Charles Babbage, o idealizador do computador, que, após gastar fortunas e um longo tempo, não conseguiu concretizar o projeto. A linguagem de programação ADA foi batizada em homenagem a esta primeira programadora. Porém, a primeira linguagem de programação para computadores foi provavelmente Plankalkül, criada por Konrad Zuse na Alemanha Nazista, mas que teve pouco ou nenhum impacto no futuro das linguagens de programação.

Contudo, segundo Gonçalves (2009), existe um número muito grande de linguagens de programação que foram desenvolvidas desde o início da computação, desta forma, foram agrupadas em quatro gerações, de acordo com suas características e épocas em que foram desenvolvidas.

**2.2.1 Linguagens de primeira geração**

A primeira geração de linguagens surgiu no início da computação, nos anos 50, mais precisamente de 1950 a 1962, eram linguagens com codificação de máquina. O Assembly e a linguagem de máquina representam esta primeira geração.

**2.2.2 Linguagens de segunda geração**

A segunda geração das linguagens de programação, as quais foram à base para as modernas linguagens existentes, foram desenvolvidas de 1962 a 1974. Estas linguagens tinham como principais características a vasta utilização de bibliotecas de software, a programação multiusuário, sistemas de execução em tempo real e o inicio do desenvolvimento de sistemas gerenciadores de banco de dados.

As linguagens que representaram essa geração foram: Fortran, Cobol, Algol e algumas extensões como o Basic.

**2.2.3 Linguagens de terceira geração**

As linguagens de terceira geração também são conhecidas como linguagens modernas ou estruturadas foram desenvolvidas de 1974 a 1986. Estas linguagens tinham como características principais a capacidade procedural e estrutural dos dados, além do mais foram utilizadas para o desenvolvimento de sistemas distribuídos, incorporaram recursos mais inteligentes e exigiam um hardware menos robusto. A terceira geração foi dividida em duas categorias: linguagem de propósito geral e linguagens especializadas.

As linguagens que representaram essa geração foram: C, Pascal, Lisp, Smalltalk, Forth e dentre algumas outras.

**2.2.4 Linguagens de quarta geração**

A quarta geração das linguagens de programação vem sendo desenvolvidas desde 1986, tendo como fundamentais características a geração de sistemas especialistas, o desenvolvimento de inteligência artificial, e a execução paralela de programas. Esta geração trás um aumento no desenvolvimento voltado para linguagens de alto nível, apresentando um alto nível de abstração como principal característica.

As linguagens de quarta geração podem ser classificadas em três categorias: linguagens de consulta, geradoras de programas e outras linguagens (4GL).

Temos como exemplo de linguagens de quarta geração o SQL, XML, Delphi, Visual Basic, Java e dentre outras.

2.3PROPRIEDADES DESEJÁVEIS

Atualmente existem inúmeras linguagens de programação e cada uma possui suas peculiaridades, entretanto, existem propriedades que são almejáveis em qualquer linguagem de programação existente, desta forma Mattos (2009) define as seguintes propriedades:

* **Legibilidade**

A legibilidade numa linguagem de programação esta relacionada à facilidade na leitura dos códigos, da mesma maneira, quanto mais legível for mais fácil será entender o código e descobrir os erros na programação.

* **Redigibilidade**

A redigibilidade numa linguagem de programação está relacionada na facilidade em escrever programas, logo, está vinculada a legibilidade.

* **Confiabilidade**

A confiabilidade numa linguagem de programação está relacionada aos mecanismos que a linguagem oferece para se desenvolver programas confiáveis.

* **Eficiência**

A eficiência numa linguagem de programação está relacionada aos mecanismos que ela oferece para que se possam desenvolver aplicativos mais eficientes, fazendo com que o tempo de resposta de um aplicativo seja mais rápido.

* **Facilidade de Aprendizado**

A facilidade de aprendizado numa linguagem de programação está totalmente ligada ao programador, pois a linguagem deve oferecer mecanismos de fácil entendimento ao programador.

* **Ortogonalidade**

A ortogonalidade numa linguagem de programação é interessante porque desta forma o programador pode prever, com segurança, o comportamento de uma determinada combinação de conceitos. Ou seja, ela permite ao programador combinar conceitos básicos de forma que não produzem efeitos anormais nessa combinação.

* **Reusabilidade**

A reusabilidade numa linguagem de programação aumenta a produtividade de programação, pois ela possibilita a reutilização do mesmo código para diversas aplicações, desta forma, reduzindo a escrita de código desnecessário.

* **Modificabilidade**

A modificabilidade numa linguagem de programação é uma facilidade que ocorre quando se modifica determinado trecho de código, para implementar novas funcionalidades, e o mesmo não provoca mudanças em outras partes do programa. Desta forma, trazendo benefícios para o programador.

* **Portabilidade**

A portabilidade numa linguagem de programação é altamente desejável, pois desta forma, independente do hardware ou sistema operacional que está sendo utilizado durante o desenvolvimento do aplicativo nesta linguagem, ele funcionará corretamente no hardware ou sistema operacional do cliente final, evitando a modificação de código desnecessário na adaptação do aplicativo para um ambiente específico.

**3 PARADIGMAS DE PROGRAMAÇÃO**

3.1 CONCEITO

De acordo com Ferreira (2009) um paradigma de programação fornece e determina a visão que o programador possui sobre a estruturação e execução do programa, ou seja, os paradigmas de programação são técnicas utilizadas pelos programadores no desenvolvimento de aplicações na qual dizem respeito a como programar numa linguagem de programação. Pode-se comparar com a engenharia de software, pois, assim como diferentes grupos na engenharia de software propõem diferentes metodologias, diferentes linguagens de programação propõem diferentes paradigmas de programação, sendo que, cada uma delas apresenta técnicas específicas e cada programador escolhe a que melhor se encaixa para o desenvolvimento de uma solução para um problema.

3.2 PRINCIPAIS PARADIGMAS DE PROGRAMAÇÃO

Quando se desenvolve uma linguagem de programação existem conceitos que são omitidos e outros que são incluídos a ela, deste modo, gerando linguagens diferentes, algumas mais simples que outras, porém cada uma possui características específicas para um estilo de programação peculiar, ou seja, cada linguagem possui um paradigma de programação. Desta forma, de acordo com Pinto (2009) têm-se como os principais paradigmas de programação atual detalhados nos itens a seguir.

**3.2.1 Imperativo**

Para Ferreira (2009) o paradigma imperativo baseia-se no conceito de comandos e atualização de variáveis. Como os programas são escritos para modelar processos e objetos do mundo real e tais objetos freqüentemente possuem estados que variam com o tempo, variáveis naturalmente modelam tais objetos. Desse modo, pode-se considerar que programas imperativos modelam tais processos e objetos com bastante eficiência. Além disso, todas as arquiteturas de computadores existentes hoje se baseiam na arquitetura de Von Newmann, caracterizada pelo acesso direto e possibilidade de alteração de valores armazenados em posições de memória. Isso torna os ambientes de execução das linguagens imperativas bastante eficientes. Programas imperativos são excelentes, por exemplo, para programação de baixo nível.

Alguns exemplos de linguagens de programação que baseiam-se no modelo imperativo são: Ada, Algol, Basic, C, Cobol, Fortran, Pascal, Python e dentre outras.

**3.2.2 Orientado a Objetos**

Segundo Manssour (2009) o paradigma de orientação a objetos tornou-se popular apenas na década de 80. Uma linguagem orientada a objetos deve fornecer suporte para três características “chave”: tipos abstratos de dados, herança e um tipo particular de amarração dinâmica. Um modelo orientado a objetos tem como entidade fundamental o “objeto”, que recebe e envia mensagens, executa processamentos e possui um estado local que ele pode modificar. Problemas são resolvidos através de objetos que enviam mensagens uns para os outros. Assim, pode-se dizer que um modelo orientado a objetos é formado por quatro componentes básicos: objetos, mensagens, métodos e classes.

A seguir tem-se a descrição desses principais componentes do paradigma de orientação a objetos:

* **Objetos –** é um conjunto de operações encapsuladas (métodos) e um “estado” (determinado pelo valor dos atributos) que grava e recupera os efeitos dessas transações.
* **Mensagens** – são requisições que um objeto envia a outro, de forma que o objeto “receptor” forneça algum resultado desejado através da execução de uma operação.
* **Métodos** – são as descrições das operações que um objeto executa quando recebe mensagens.
* **Atributos –** um atributo éum dado, para o qual cada objeto de uma classe tem seu próprio valor.
* **Classes -** define as características de uma coleção de objetos.
* **Encapsulamento** – cada objeto é visto como o encapsulamento do seu estado interno, suas mensagens e seus métodos.
* **Polimorfismo –** é a propriedade que permite que uma mensagem seja enviada a diferentes objetos e que cada objeto execute a operação que é apropriada a sua classe.
* **Herança –** é a propriedade mais importante do conceito de orientação a objetos. Ela é permitida através da definição de uma hierarquia de classes, sendo que uma subclasse herda as propriedades de uma superclasse.

Entre os principais objetivos do paradigma de orientação a objetos, são destacados a reutilização de código e a restrição de acesso a detalhes dos componentes do software. Contudo, a programação orientada a objeto consiste em resolver um problema, através da análise das entidades e seus relacionamentos, dentro do conjunto onde se situa o problema. A partir dessa análise é estabelecido um Modelo de Resolução que represente aquilo que acontece no mundo real. A análise e projeto orientados a objetos têm como meta identificar o melhor conjunto de objetos para descrever um sistema de software. O funcionamento deste sistema se dá através do relacionamento e troca de mensagens entre estes objetos.

Além do mais temos as seguintes linguagens que utilizam o conceito de paradigma orientado a objetos: C++, C#, Java, Object Pascal, Objective-C, Python, Ruby e Smalltalk.

**3.2.3 Funcional**

Silva (2009) conceitua como sendo o paradigma funcional um modelo que consiste em desenvolver funções que resolvem um determinado problema. Essas funções correspondem aos princípios matemáticos, apesar de que nem sempre podem ser consideradas funções totais. O paradigma funcional trabalha de forma avaliativa, assim como as calculadoras, lê uma expressão, calcula o seu valor e apresenta o resultado.

A característica predominante da programação funcional é que o significado de uma expressão é o seu valor, e o papel do computador é obtê-lo.

As linguagens funcionais são naturalmente recursivas e implementam de forma mais rápida o conceito de recursão. Este fato dependendo do contexto pode torná-las mais eficientes que as linguagens imperativas para alguns problemas. Exemplos de linguagens funcionais são: Lisp, ML, Miranda e Haskell.

**3.2.4 Lógico**

De acordo Ribeiro (2009) a idéia fundamental da programação em lógica é que um algoritmo é constituído de dois elementos distintos: a lógica, que define a solução do problema, e o controle, que define como a solução do problema é alcançada. Sendo que, o programador deve somente listar os passos do elemento lógico, para a solução do problema, enquanto o controle é feito automaticamente pelo sistema de programação em lógica.

Desta maneira, um programa especificado no paradigma lógico é basicamente um conjunto finito de sentenças lógicas, nomeadas cláusulas que expressam o problema-meta, a ser solucionado.

Freqüentemente, a prova de teoremas consiste de um repertório de três ferramentas:

* Um conjunto de axiomas: fatos que são aceitos como verdade para a prova do teorema;
* Um conjunto de regras de inferência: regras pelas quais se podem obter novas informações a partir das informações conhecidas;
* Uma questão (query): uma consulta submetida aos fatos conhecidos.

Logo, uma prova então será composta de um conjunto de produções que tem base nos fatos assumidos (axiomas), então se deriva novos fatos (inferência) e finalmente finaliza apresentando que uma questão (query) que pode ser derivada a partir das informações iniciais. A programação em lógica, que define em toda sua extensão o paradigma lógico também compõe o paradigma declarativo, o qual engloba também o paradigma funcional. Entretanto, o paradigma funcional é bem mais disseminado nas comunidades de estudo de linguagens de programação. O paradigma lógico ganhou impulso por volta dos anos 80, quando foi selecionado como base do projeto japonês de computadores de quinta geração. A linguagem de programação que utiliza a abordagem proposta pelo paradigma lógico é a Prolog.

**4 CONCLUSÃO**

Pôde-se concluir através da realização deste trabalho que a criação das linguagens de programação foi um grande passo para a computação, pois, possibilitou aos programadores que pudessem desenvolver aplicações de uma maneira eficaz, sem a necessidade de conhecer a linguagem binária e apenas conhecendo a linguagem natural é possível desenvolver soluções para determinados problemas através da utilização de uma linguagem de programação específica.

Além do mais, as linguagens possuem paradigmas de programação, ou seja, é a forma que o programador deve pensar para desenvolver o código para determinada linguagem, porém, todo paradigma de programação, assim como tudo existente, tem seus prós e contras e cabe a cada programador determinar o melhor paradigma para o desenvolvimento de soluções.

**REFERÊNCIAS**

LINDER, Marcelo. **Linguagens de Programação**.Disponível em: <http://www.univasf.edu.br/~marcelo.linder/arquivos\_pc/aulas/aula5.pdf>. Acesso em: 15 ago. 2009.

GONÇALVES, Luiz Marcos G. **Algoritmo e Lógica de Programação**: Conceitos de Linguagens de Programação. Disponível em: <http://www.dca.ufrn.br/~lmarcos/courses/DCA800/pdf/linguagens.pdf >. Acesso em: 15 ago. 2009.

ROCHA, Jaison. **Poligrafe**: Programação: A expressão padrão. Disponível em: <http://fotografeumaideia.com.br/site/index.php?option=com\_content&task=view&id=629&Itemid=140>. Acesso em: 15 ago. 2009.

MATTOS, Renato. **Linguagem de Programação.** Disponível em: <http://www.linhadecodigo.com.br/Artigo.aspx?id=489&pag=1>. Acesso em: 15 ago. 2009.

FERREIRA, Tiago Juliano. **Paradigmas de Programação.** Disponível em: <http://www.etaj.com.br/~tiago/paradigmas.doc>. Acesso em: 16 ago. 2009.

PINTO, Bruno Queiroz. **Conceitos Iniciais e Características de uma Linguagem de Programação.** Disponível em: <http://www.inf.fesurv.br/~bruno/disciplinaPP/arquivos/aulas/aula02.ppt>. Acesso em: 16 ago. 2009.

MANSSOUR, Isabel Harb. **Paradigmas de Linguagens I.** Disponível em: <http://www.inf.pucrs.br/~gustavo/disciplinas/pli/material/paradigmas-aula12.pdf>. Acesso em: 16 ago. 2009.

SILVA, José Mauro da. **Paradigma Funcional.** Disponível em: < http://informacaocomdiversao.blogspot.com/2009/02/paradigma-funcional.html >. Acesso em: 16 ago. 2009.

RIBEIRO, Monael Pinheiro. **Prolog.** Disponivel em: < http://www.de9.ime.eb.br/~mpribeiro/aspectos\_formais/Prolog/Trabalho/Prolog\_afc.doc >. Acesso em: 16 ago. 2009.