Hortolândia, 18 de fevereiro de 2014.

Aulas 7 e 8 – Paradigmas de Linguagens de Programação

Na área de Linguagens de Programação, existem diversos paradigmas que têm sido utilizados pela comunidade de computação. Cada um desses paradigmas tem induzido à construção de linguagens específicas, que exploram e evidenciam de forma mais direta e natural os conceitos correspondentes a cada paradigma (MELO et al, 2003).

Segundo Melo et al (2003), os principais paradigmas de programação são: **funcional**, **imperativo**, **orientado a objetos, baseado em lógica** e **baseado em satisfação de restrições**.

Algumas linguagens foram desenvolvidas para suportar um paradigma específico (**Smalltalk** e **Java** suportam o paradigma de orientação a objetos enquanto **Haskell** suportam o paradigma funcional), enquanto outras linguagens suportam múltiplos paradigmas (como o **LISP**, **Perl**, **Python**, **C++** e **Oz**).

Nesta aula, exploraremos as principais características de dois dos principais paradigmas dentre os acima citados, que são: o paradigma **imperativo** e o **orientado a objetos**.

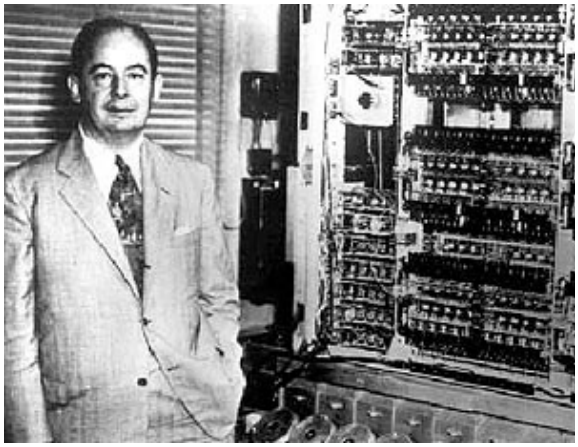
I. Paradigma Imperativo

O fundamento para a programação imperativa é o conceito de **Máquina de Turing**.

A Máquina de Turing é uma abstração matemática – proposta pelo pesquisador **Alan Turing** nos anos 30 – que corresponde ao conjunto de funções computáveis.

Alan Turing foi um matemático britânico, que viveu de 1912 a 1954.

A Figura 1, à direita, mostra o cientista **Alan Turing**.



Essa caracterização das funções computáveis foi aproximada por **John Von Neumann** a uma arquitetura de computadores que fundamenta os computadores construídos até hoje (MELO et al, 2003).

A Figura 2, à esquerda, representa o cientista **John Von Neumann**.

A essência da programação imperativa (e também da máquina de Turing) se resume a três conceitos:

1. A descrição de estados de uma máquina abstrata por valores de um conjunto de variáveis, sendo que uma variável é um identificador de um local – um endereço físico de memória, por exemplo – que atua como repositório para determinado conjunto de valores.

2. Reconhecedores desses estados, que são expressões compostas por relações entre valores e/ou resultados de operações utilizando valores. Alguns desses valores podem ser substituídos por variáveis e nesse caso o valor presente na variável será o valor utilizado na expressão.

3. Comandos, que podem ser de dois tipos:

a) **Comandos de atribuição**, que constroem valores efetuando operações a partir de valores preexistentes e atualizam os conteúdos de variáveis;

b) **Comandos de controle**, que determinam qual o próximo comando a ser executado.

A execução de um programa imperativo se assemelha, portanto, à simulação da operação de uma máquina física. Cada estado da máquina, uma vez reconhecido, leva a uma sequência de ações.

As ações alteram o estado da máquina, suscitando novas ações e assim por diante até que seja reconhecido um "estado final", que indica a conlusão de uma tarefa.

Exemplos de linguagens que seguem o paradigma imperativo: **Algol**, **Pascal**, **C**.

Um exemplo de programa escrito em **Linguagem C** está representado na Listagem 1 a seguir.

```
// Exemplo de programa em C
#include <stdio.h>      //  Arquivo de cabeçalho (header)
void main()
{
   int contador;        // declarações simples
   float PrecoDoQuilo;
   double TaxaDeCambio;
   char LetraDigitada;
   int IdadeManoel, IdadeJoao, IdadeMaria;     // Pode-se
colocar mais de uma
                                    // variável na mesma linha
  double  TaxaDoDolar,
          TaxaDoMarco,
          TaxaDoPeso,
          TaxaDoFranco;

.......
}
```

Listagem 1: Exemplo de programa escrito em Linguagem C (trecho de código).

II. Paradigma Orientado a Objetos

O paradigma da orientação a objetos tem por princípio a solução de problemas pela cooperação de vários elementos, da mesma forma que usamos a prestação de serviço de outras pessoas para resolver vários de nossos problemas.

Com o aumento da complexidade dos problemas computacionais a serem resolvidos veio a necessidade de dividir as soluções computacionais em unidades menores. As abstrações de processos e módulos conseguidas nas linguagens puramente imperativas não distribuem a responsabilidade dos estados dos programas para essas unidades – elas agrupam abstrações, mas é responsabilidade do programa usá-las (MELO et al, 2003).

Como consequência disso, os módulos não são unidades computacionalmente independentes.

Daí surgiu a ideia, com **David Parnas**, no início dos anos 70, de ocultar informação como uma disciplina de programação para que as unidades passassem a controlar seus estados e os problemas pudessem ser resolvidos por colaboração.

Os principais fundamentos das linguagens de programação com o objetivo de prover objetos estão centrados nos conceitos de abstrações. Cada **objeto** é um elemento abstrato responsável pelo seu estado e faz as transformações sobre tal estado mediante um conjunto fixo de regras de comportamento.

O conceito de **orientação a objetos** teve seu início com a linguagem **Simula 67**, mas teve um melhor alcance com **Smalltalk** na década de 80.

Sob o ponto de vista computacional devemos ter **objetos** que podem prover serviços sob a sua própria responsabilidade. Isso requer que os objetos sejam proprietários dos seus estados e possam dar soluções próprias aos serviços requisitados.

As linguagens de programação orientadas a objetos permitem a criação de **objetos**, os quais são entidades computacionalmente ativas que guardam um conjunto de **dados** (os **atributos**), e os **serviços** (os **métodos**) que ele pode prover.

Exemplos de linguagens que seguem o paradigma orientado a objetos: **Smalltalk**, **Java**, **C++**, **C#**.

Um exemplo de programa escrito em Java está representado na Listagem 2 a seguir.

```
package exemplos;

/*    O primeiro programa em Java: Hello World
      Autor: Fulano de Tal  */

public class Hello {
  public static void main(String[] args) {
      // Todo programa tem um ponto de entrada: o "método" main
de alguma "classe"
      System.out.println("Hello, world!");
  }
}
```

Listagem 2: Exemplo de programa escrito em Linguagem Java (classe Hello).

Classes e Objetos

O processo de **abstração**, quando aplicado na programação orientada a objetos, leva o desenvolvedor primeiramente a representar um determinado problema computacional a ser resolvido na forma de um modelo de **classes de objetos**.

Este modelo de classes obtido é então transformado, por meio da atividade de programação, em estruturas de classes, escritas de acordo com as regras de uma determinada linguagem de programação, escolhida pelo desenvolvedor.

Tais **classes** são então compiladas (ou seja, transformadas em código executável) e então executadas pelo computador.

As **classes**, uma vez instanciadas pelo computador (ou seja, espaço de memória foi alocado para as mesmas), são chamadas de **objetos**.

A representação dos objetos é semelhante à das classes, sendo que a única diferença é que os objetos possuem valores para seus **atributos** e podem ter seus **métodos** executados por outros objetos.

Na próxima aula será visto como se representam as classes, em um modelo **orientado a objetos**, bem como será iniciado o estudo da linguagem **Java**, que implementa o **paradigma de orientação a objetos**.

III. Referências Bibliográficas

MELO, Ana Cristina Vieira de, SILVA, Flávio Soares Corrêa. **Princípios de Linguagens de Programação**. São Paulo: Edgard Blücher, 2003.

WIKIPEDIA. **Paradigma de Programação**. URL: http://pt.wikipedia.org/wiki/Paradigma_de_programa%C3%A7%C3%A3o Última consulta: 10/02/2014.