Paradigmas de Programação

Gustavo Jungthon[1], Cristian Machado Goulart[1]

[1]Faculdade de Informática de Taquara (FIT)
Rua Oscar Martins Rangel, 4500 – 95600–000 – Taquara – RS – Brazil
guto@faccat.br, cmg3k@faccat.br

Resumo. *Este artigo tem como objetivo esboçar os paradigmas das diferentes linguagens de programação de computadores.*

1. Introdução

Um paradigma é o que determina o ponto de vista da realidade e como se atua sobre ela, os quais são classificados quanto ao seu conceito de base, podendo ser: Imperativo, funcional, lógico, orientado a objetos e estruturado. Cada qual determina uma forma particular de abordar os problemas e de formular respectivas soluções. Além disso, uma linguagem de programação pode combinar dois ou mais paradigmas para potencializar as análises e soluções. Deste modo, cabe ao programador escolher o paradigma mais adequado para analisar e resolver cada problema.

2. Paradigmas

2.1. Paradigma Imperativo

2.1.1. Conceito

O Paradigma Imperativo é baseado na arquitetura de Von Neumann. É o primeiro paradigma a existir e até hoje é o dominante.

Esse paradigma segue o conceito de um estado e de ações que manipulam esse estado, nele encontramos procedimentos que servem de mecanismos de estruturação. Podemos denominá-lo de procedural por incluir sub-rotinas ou procedimentos para estruturação.

2.1.2. Linguagens

Exemplos de linguagens de programação que baseiam-se no modelo imperativo:

- Ada;
- ALGOL;
- Assembler;
- Basic;
- C;
- Cobol;

- Fortran;
- Pascal;
- Python;
- Lua;

2.1.3. Aplicações

Exemplo:

```
function fatorial (n: integer):integer;
var fat: integer;
begin
   fat := 1;
   while (n>1) do
    begin
      fat := fat * n;
      n := n – 1;
    end;
   fatorial := fat;
end;
```

2.1.4. Vantagens

As vantagens desse paradigma são: eficiência (porque embute o modelo de Von Neumann); modelagem “natural” de aplicações do mundo real; paradigma dominante e bem estabelecido; e também muito flexível.

2.1.5. Desvantagens

As desvantagens são: difícil legibilidade; as instruções são centradas no como e não no o que.

2.2. Paradigma Estruturado

2.2.1. Conceito

Este paradigma preconiza que todos os programas possíveis podem ser reduzidos a apenas três estruturas: seqüência, decisão e iteração. Tendo, na prática, sido transformada na Programação modular, a Programação estruturada orienta os programadores para a criação de estruturas simples em seus programas, usando as subrotinas e as funções. Foi a forma dominante na criação de software entre a programação linear e a programação orientada por objetos.

Apesar de ter sido sucedida pela programação orientada por objetos, pode-se dizer que a programação estruturada ainda é marcantemente influente, uma vez que grande parte das pessoas ainda aprendem programação através dela.

2.2.2. Linguagens

- C;
- Basic;
- Pascal;
- Cobol;

2.2.3. Aplicações

```
open file;
while (reading not finished) {
  read some data;
  if (error) {
    stop the subprogram and inform rest of the program about the error;
  }
}
process read data;
finish the subprogram;
```

2.3.4. Vantagens

Os problemas podem ser quebrados em vários subproblemas, a boa legibilidade e a boa compreensão da estrutura deste paradigma motivam os programadores a iniciarem a programação pelo modelo estruturado.

2.3.5. Desvantagens

Os dados são separados das funções;

Mudanças na estrutura dos dados acarreta alteração em todas as funções relacionadas.

Gera sistemas difíceis de serem mantidos;

2.3. Paradigma Orientado a Objetos

2.3.1. Conceito

A programação Orientada a Objetos é baseada na composição e interação de diversas unidades de softwares denominados objetos. O funcionamento de um software orientado a objetos se dá através do relacionamento e troca de mensagens entre esses objetos. Esses objetos são classes, e nessas classes os comportamentos são chamados de métodos e os estados possíveis da classe são chamados de atributos. Nos métodos e nos atributos também são definidas as formas de relacionamento com outros objetos.

2.3.2. Linguagens

Exemplos de linguagens de programação que baseiam-se no modelo orientado a objetos:

- Smalltalk;
- Python;
- Ruby;
- C++;
- Object Pascal;
- Java;
- C#;
- Oberon;
- Ada;
- Eiffel;
- Simula;
- .NET

2.3.3. Aplicações

Exemplo:

```
package exemplo;


public class Conta {
   String nrDaCOnta;
   String descricao;
   double saldo;
```

```
    double limite;
    Cliente cliente = new Cliente();

    public boolean saque (double valor){
        if (valor<=(saldo+limite)){
            saldo-=valor;
            return true;
        } else{
            return false;
        }

    }

    public boolean deposita (double valor){
        if (valor<=(saldo+limite)){
            saldo+=valor;
            return true;
        } else{
            return false;
        }
    }

    public boolean transfere (Conta c, double valor){
        if (valor<=(saldo+limite)){
            c.saldo+=valor;
            saldo-=valor;
            return true;
        } else{
            return false;
        }
    }
}
```

2.3.4. Vantagens

Esse paradigma possui todas as vantagens do paradigma imperativo entre outras: a alteração de um módulo não incorre na modificação de outros módulos; quanto mais um módulo for independente, maior a chance dele poder ser reutilizado em outra aplicação.

2.3.5. Desvantagens

Por exigir formas de pensar relativamente complexas, a programação orientada a objetos até hoje ainda não é bem compreendida ou usada pela maioria.

2.4. Paradigma Funcional

2.4.1. Conceito

Este paradigma trata a computação como uma avaliação de funções matemáticas. Este método enfatiza a aplicação de funções, as quais são tratadas como valores de primeira importância, ou seja, funções podem ser parâmetros ou valores de entrada para outras funções e podem ser os valores de retorno ou saída de uma função.

2.4.2. Linguagens

Exemplos de linguagens de programação que baseiam-se no modelo funcional:

- Lambda (não implementado para computadores);
- LISP;
- Scheme (tentativa de simplificar e melhorar o LISP);
- ML (Criada em universidade);
- Miranda (também criada em universidade);
- Haskell;

2.4.4. Desvantagens

Na programação funcional parecem faltar diversas construções freqüentemente (embora incorretamente) consideradas essenciais em linguagens imperativas, como C. Por exemplo, não há alocação explícita de memória nem de variáveis.

2.4.5. Vantagens

Devido ao processo automático de alocação de memória, então efeitos colaterais no cálculo da função são eliminados. Sem estes efeitos, a linguagem assegura que o resultado da função será o mesmo para um dado conjunto de parâmetros não importando onde, ou quando, seja avaliada e é empregado em computações independentes para execução paralela. A recursividade em programação funcional pode assumir várias

formas e é em geral uma técnica mais poderosa que o uso de laços do paradigma imperativo.

2.5. Paradigma Lógico

2.5.1. Conceito

Nesse paradigma programas são relações entre Entrada/Saída. Possui estilo declarativo, como o paradigma funcional. Inclui características imperativas, por questões de eficiência. Aplicações em prototipação em geral, sistemas especialistas, bancos de dados, etc.

2.5.2. Linguagens

Exemplos de linguagens de programação que baseiam-se no paradigma lógico:

- Popler;
- Conniver;
- QLISP;
- Planner;
- Prolog;
- Mercury;
- Oz;
- Frill.

2.5.3. Aplicações

Exemplo em Prolog:

```
star(sun).
star(sirius).
star(betelgeuse).
orbits(mercury, sun).
orbits(venus, sun).
orbits(earth, sun).
orbits(mars, sun).
orbits(moon, earth).
orbits(phobos, mars).
orbits(deimos, mars).
planet(X) :- orbits(X,sun).
```

satellite(B) :- orbits(B,P), planet(P).

?-

satellite(phobos).

yes

?- satellite(S).

S=moon

S=phobos

S=deimos

2.5.4. Vantagens

Possui a princípio todas as vantagens do paradigma funcional. E permite concepção da aplicação em um alto nível de abstração (através de associações entre E/S).

2.5.5. Desvantagens

Variáveis de programa não possuem tipos, nem são de alta ordem.

Referências

Sampaio, A. (2008) “Paradigmas de Linguagens de Programação”, http://www.cin.ufpe.br/~in1007/transparencias/aulaIntroducaoPLP.ppt, Agosto.

Paula, A. (2008) “Paradigmas de Linguagens de Programação Motion Capture White Paper”, http://http://www.inf.unisinos.br/~anapaula/disciplinas/60023/, Agosto.

Fernandes, E., Carvalho, K., Villar, L., Getirana, N. e Gaudêncio, V. (2008) “Paradigmas de linguagem de programação Motion Capture White Paper”, http://http://www.inf.unisinos.br/~anapaula/disciplinas/60023/, Agosto.

Programação funcional

http://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_funcional

Programação estruturada

http://labes.inf.ufes.br/vsouza/sites/default/files/CursoOOSlides03.pdf