Introdução à Programação

Linguagens de Programação:
conceituação, classificação e paradigmas de linguagens de programação

Engenharia da Computação

Professor: Críston Pereira de Souza

Web: www.univasf.edu.br/~criston.souza/algoritmos.html

email: criston.souza@univasf.edu.br

Sumário

- Conceitos
- Classificação
- Paradigmas de LP

Conceitos

- Uma LP é uma ferramenta utilizada para escrever programas.
- As primeiras linguagens de programação eram muito simples.
  - LPs de baixo nível
    - Ex.: Linguagem de máquina
- Com o avança dos computadores e a necessidade de se desenvolver aplicações mais complexas as LPs passaram a ser mais robustas.
  - LPs de alto nível
    - Ex.: C, C++, Java, Pascal, ...

Conceitos

- **Exemplo 1:** Linguagem de alto nível

```
int A, B;
int main()
{
        A = 2;
        B = 1;
        A = A + B;
}
```

- **Exemplo 2:** Linguagem de baixo nível

```
MOV AX,0002
MOV BX,0001
ADD AX,BX
```

Conceitos

- As linguagens de programação surgiram da necessidade de tornar o processo de desenvolvimento de software mais produtivo.
- Processo de desenvolvimento de software:
  - Especificação de requisitos;
  - Projeto do software;
  - Implementação;
  - Validação;
  - Manutenção.

Conceitos

- Propriedades desejáveis de uma LP:
  - Legibilidade
    - Facilidade para ler e entender um programa.
    - Problemas com *goto* (programação *macarrônica*).
    - Ambiguidade:
      - Instrução *this* em Java (usado para referenciar um objeto e também para chamar um método construtor de um objeto), e
      - Operador * em C e C++ (indicação de conteúdo de memória apontado por um ponteiro e operação de multiplicação).

Conceitos

- Propriedades desejáveis de uma LP:
  - Redigibilidade
    - Esta propriedade melhor diferencia as linguagens de alto nível e de baixo nível.
    - Possibilita ao programador se concentrar unicamente nos algoritmos centrais do problema.
    - LPs que requerem muita programação de entrada e saída e que não dispõem de mecanismos para o tratamento de erros tendem a obscurecer os algoritmos centrais nos programas.

Conceitos

- Propriedades desejáveis de uma LP:
  - Confiabilidade
    - Importante que a LP permita a verificação automática durante o processo de compilação ou execução.
    - LPs que possuem mecanismos para detectar eventos indesejáveis e especificar respostas adequadas a tais eventos permitem a construção de programas mais confiáveis.

Conceitos

- Propriedades desejáveis de uma LP:
  - Eficiência
    - Dependendo da aplicação são necessários certos recursos da linguagem.
      - Ex.: Aplicações de tempo real, aplicações embarcadas (baixo consumo de memória e energia), ...
    - Algumas LP´s possuem bons compiladores que tendem a otimizar automaticamente o código gerado.
    - LP´s que requerem verificação de tipos durante a execução são menos eficientes do que aquelas que não fazem este tipo de verificação.

Conceitos

- Propriedades desejáveis de uma LP:
  - Facilidade de aprendizado
    - O programador deve ser capaz de aprender a LP com facilidade.
    - LPs com muitas características e múltiplas maneiras de realizar a mesma funcionalidade tendem a ser mais difíceis de aprender.
      - Ex.: incremento em C e C++
        - » c = c + 1, c += 1, c++, ++c

Conceitos

- Propriedades desejáveis de uma LP:
  - Ortogonalidade
    - Diz respeito à capacidade de a LP permitir ao programador combinar seus conceitos básicos sem que se produzam efeitos anômalos nessa combinação.
    - Em LPs ortogonais o programador pode prever, com segurança, o comportamento de uma determinada combinação de conceitos.

Conceitos

- Propriedades desejáveis de uma LP:
  - Reusabilidade
    - Possibilidade de reutilizar o mesmo código para diversas aplicações.
    - Quanto mais reusável for um código, maior será a produtividade de programação.
    - Uma forma de permitir a reusabilidade é através da parametrização de subprogramas (procedimentos e funções).
    - Uso de bibliotecas de subprogramas.

Conceitos

- Propriedades desejáveis de uma LP:
  - Modificabilidade
    - Facilidade oferecida pela LP para possibilitar ao programador alterar o programa em função de novos requisitos, sem que tais modificações impliquem mudanças em outras partes do programa.
    - Uso de constantes simbólicas e separação entre interface e implementação.
      - Ex.:
        - *const float pi = 3.14;*
        - *#DEFINE COLUNAS 10*

Conceitos

- Propriedades desejáveis de uma LP:
  - Portabilidade
    - É desejável que programas escritos em uma determinada LP se comportem da mesma maneira independente da ferramenta utilizada para traduzí-los para a linguagem de máquina ou da arquitetura computacional sobre a que estão executados.
    - Um mesmo programa ou biblioteca deve ser utilizado em vários ambientes e diferentes situações sem que seja necessário despender tempo de programação para reescrevê-los ou adaptá-los ao novo ambiente de tradução ou execução.
      - Padronizações do tipo ISO ou IEEE
        - » Ex.: C padrão ANSI (*American National Standards Institute*)

Classificação

- Linguagem de máquina
  - Linguagem que a máquina é capaz de "entender" sem precisar ser traduzida.
  - Programas em linguagens de máquina são compostos por sequências de bits.
    - Ex.: 000110110
  - As sequências de bits referem-se a instruções e dados a serem executados.

Classificação

- Linguagem simbólica
  - Refere-se a uma abstração sobre as instruções e os dados.
  - As instrução são representadas por um “rótulo” simbólico que denota o verbo definido pela instrução.
    - Ex.: MOV AX, 0003
  - Um dado pode ser especificado diretamente na instrução ou pelo endereço de memória onde o dado é armazenado.
  - O endereço é denotado por um rótulo arbitrário definido pelo programador.
  - Programas em linguagem simbólica precisam ser traduzidos para outro equivalente em linguagem de máquina.
  - Os tradutores são chamados de montadores (*assemblers*).
  - Esse tipo de linguagem é utilizada em casos extremos onde é necessário extrair a máxima eficiência do hardware.

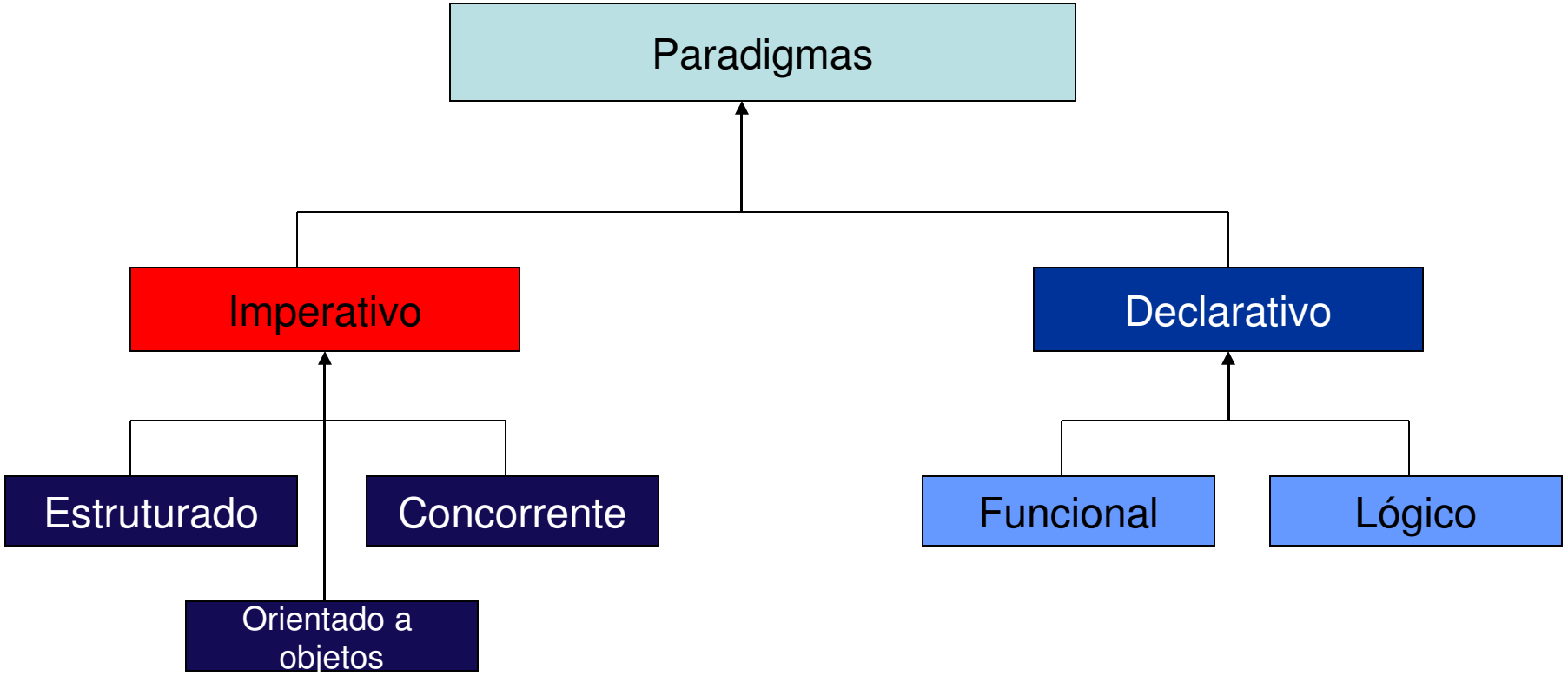
Classificação

- Linguagem de alto nível
  - Em uma LP de alto nível o grau de abstração é bem alto, próximo ao modo de o ser humano pensar.
  - Nas LPs de alto nível os programadores trabalham com o conceito de comando e não de instruções de máquina, como acontece em LPs de baixo nível.
  - Um único comando em uma LP de alto nível pode se referir a vários comandos em uma LP de baixo nível.
  - O acesso a memória é realizado por meio de variáveis e constantes.
  - Os programas escritos em LPs de alto nível precisam ser *traduzidos* ou *interpretados* para serem executados.

Paradigmas de LP

- Paradigma é um conjunto de características que servem para categorizar um grupo de linguagens.
- Os paradigmas mais comum para LPs são:
  - Estruturado
  - Orientado a objetos
  - Funcional
  - Lógico.

Paradigmas de LP



Paradigmas de LP

- Paradigma imperativo
  - Fundamentado na idéia de computação como um processo que realiza mudanças de estado.
  - Um estado representa uma configuração qualquer da memória do computador.
  - Os programas de LPs incluídas nesse paradigma especificam como uma computação é realizada por uma sequência de alterações no estado da memória do computador.
  - O foco deste paradigma é especificar como um processamento deve ser feito no computador.

Paradigmas de LP

- Paradigma imperativo
  - Estruturado
    - Oriundo da necessidade em se eliminar os desvios incondicionais (*goto*) dos programas.
    - Baseia-se na idéia de desenvolvimento de programas por refinamentos sucessivos (*top-down*).
    - A programação estruturada consegue organizar o fluxo de controle de execução dos programas.
    - Desestimula o uso de comandos de desvio incondicionais e incentiva a divisão dos programas em subprogramas e em blocos aninhados de comandos.
      - Ex.: as linguagens Pascal e C são exemplos deste paradigma.

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Estruturado: Programa em Linguagem Pascal

```
PROGRAM RaizQuadrada(INPUT,OUTPUT);
VAR n,raiz,erro: REAL;
BEGIN
   WRITE('Qual o número de que quer calcular a raiz quadrada ?');
   READLN(n);
   raiz:=SQRT(n);
   raiz:=(n/raiz+raiz)/2;
   erro:=(n/SQR(raiz))-1;
   IF erro < 1E-06 THEN
            WRITELN('A raiz de ',n,' , ',raiz,' ',erro)
   ELSE
            WRITELN('Existe um erro > 10E-06')
END.
```

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Estruturado: Programa em Linguagem C

```
#include <stdio.h>
void main()
{
        int t;
        for(t = 0; t < 100; t++) {
                printf("%d",t);
                if(t == 10) break;
        }
}
```

Paradigmas de LP

- Paradigma imperativo
  - Orientado a Objetos
    - Oferece conceitos que objetivam tornar mais rápido e confiável o desenvolvimento de sistemas.
    - Esse paradigma enfoca a abstração de dados como elemento básico programação.
    - Classes são abstrações que definem uma estrutura de dados e um conjunto de operações que podem ser realizadas sobre elas, chamadas *métodos*.
    - Os objetos são instâncias de classes.
      - Ex.: as linguagens C++, Smalltalk e Java são exemplos deste paradigma.

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Orientado a Objetos: Programa em Linguagem C++

```
class moeda
{
   private:
          static float US;
   public:
          static void usvalor() {
                 cout << "\nDigite o valor do dólar: ";
                 cin >> US;
          }
};
```

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Orientado a Objetos: Programa em Linguagem Smalltalk

```
anObject aMessage.
anObject firstMessage;
         secondMessage;
         lastMessage.
variableName := anObject aMessage.
```

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Orientado a Objetos: Programa em Linguagem Java

```
public class Hello
{
   public static void main(String[] args)  {
            System.out.println("Hello, world!");
   }
}
```

Paradigmas de LP

- Paradigma imperativo
  - Concorrente
    - A programação concorrente ocorre quando vários ***processos*** executam simultaneamente e concorrem por recursos.
    - Permitem utilizar uma única unidade de processamento ou várias unidades em paralelo.
    - As linguagens Pascal Concorrente e Java (*entre outras*) são exemplos de LPs que permitem desenvolver sistemas concorrentes.

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Concorrente: Programa em Pascal Concorrente

```
PROGRAM Exemplo;
PROCESS TYPE hello;
VAR i : integer;
BEGIN
FOR i:=1 TO 5 DO
    writeln('Hello World!');
END;
VAR first, second : hello;
BEGIN
COBEGIN
    first;
    second
COEND;
END.
```

```
VAR
msg : ARRAY [1..N] OF hello;
BEGIN
COBEGIN
            FOR i:=1 TO N DO
                        msg[i](params, ...);
COEND;
END.
```

Paradigmas de LP

- Paradigma imperativo
  - Exemplo: Paradigma Concorrente: Programa em Linguagem Java

```
public class SimpleThread extends Thread {
   public SimpleThread(String str) {
                    super(str);
   }
   public void run()  {
                    for (int i = 0; i < 10; i++) {
                                     System.out.println(i + " " + getName());
                                     try {
                                                      sleep((long)(Math.random() * 1000));
                                     } catch (InterruptedException e) {}
                    }
                    System.out.println("DONE! " + getName());
   }
}
public class TwoThreadsDemo {
   public static void main (String[] args) {
                    new SimpleThread("Jamaica").start();
                    new SimpleThread("Fiji").start();
   }
}
```

Paradigmas de LP

- Paradigma declarativo
  - Este paradigma os programas são especificações sobre o que é a tarefa.
  - O programador não precisa se preocupar sobre como o computador é implementado, nem sobre a maneira pela qual ele é melhor utilizado para realizar uma tarefa.
  - O programador deve descrever de forma abstrata a tarefa a ser resolvida.
  - Tipicamente, programas em LPs declarativas são especificações de relações ou funções.

Paradigmas de LP

- Paradigma declarativo
  - Funcional
    - LPs funcionais operam somente sobre funções, as quais recebem listas de valores e retornam um valor.
    - O objetivo é definir uma função que retorne um valor como a resposta do problema.
    - As principais operações nesse tipo de programação são a composição de funções e a chamada recursiva de funções.
    - Funções podem ser passadas como parâmetros a outras funções.
      - Ex.: Lisp, Haskell e ML

Paradigmas de LP

- Paradigma declarativo
  - Exemplo: Paradigma Funcional: Programa em Lisp

```
(defun bar (x) (
    setq x (* x 3))
    (setq x (/ x 2))
    (+ x 4)
)
```

Paradigmas de LP

- Paradigma declarativo
  - Exemplo: Paradigma Funcional: Programa em Haskell

```
factorial :: Integer -> Integer
factorial 0 = 1
factorial n | n > 0 = n * factorial (n-1)
```

Paradigmas de LP

- Paradigma declarativo
  - Exemplo: Paradigma Funcional: Programa em ML

```
fun fib 0 = 1
    | fib 1 = 1
    | fib n = fib(n-1) + fib(n-2);

fun square x = x * x;
fun inc x = x + 1;
```

Paradigmas de LP

- Paradigma declarativo
  - Lógico
    - LPs lógicas são normalmente baseadas em um subconjunto do cálculo de predicados.
    - Um predicado define uma relação entre constantes ou variáveis.
    - Um programa lógica é composto por cláusulas que definem predicados e relações factuais.
    - A execução de um programa lógico corresponde a um processo de dedução automática.
      - Ex.: Prolog

Paradigmas de LP

- Paradigma declarativo
  - Exemplo: Paradigma Lógico: Programa em Prolog

```
empresta(X,maria,livro(P)) :- dono(X,livro(P)).
dono(joao,livro(linguagens_de_programcao)).

?- empresta(joao,maria,livro(X)).
```

Paradigmas de LP