

Semana 3 – Tópicos em Programação Imperativa

Prof. Cassiano Moralles

O Paradigma Imperativo

- Solução do problema especificada na ótica do Computador
 - Programas centrados no conceito de um estado (modelado por variáveis) e ações (comandos) que manipulam o estado;
 - Paradigma também denominado de procedural, por incluir sub-rotinas ou procedimentos como mecanismo de estruturação;
 - Primeiro paradigma a surgir e ainda é o dominante;

Vinculações de Armazenamento e Tempo de Vida

- Em linguagens de programação imperativas, a estrutura das vinculações de armazenamento para variáveis é fundamental. Este processo de vinculação é crucial e envolve dois processos: **alocação** e **liberação**. A alocação é a obtenção de uma célula de memória de um conjunto disponível, enquanto a liberação é o retorno dessa célula ao conjunto de disponíveis.
- **Tempo de vida** de uma variável refere-se ao período em que ela está vinculada a uma célula de memória específica, começando na vinculação inicial e terminando quando é desvinculada.



Vinculações de Armazenamento e Tempo de Vida

- Com base nisso, as variáveis podem ser classificadas em quatro categorias:

1. Variáveis Estáticas: Vinculadas a células de memória antes da execução do programa e permanecem assim até o final da execução. São eficientes devido ao endereçamento direto, mas reduzem a flexibilidade e não permitem o uso de subprogramas recursivos.

2. Variáveis Dinâmicas da Pilha: Alocadas quando suas declarações são elaboradas durante a execução e liberadas quando a execução termina. São usadas para variáveis locais em subprogramas, permitindo a recursão e compartilhamento de espaço de memória, embora impliquem maior sobrecarga de execução.



Vinculações de Armazenamento e Tempo de Vida

Variáveis Dinâmicas do Monte (Heap):


3. Explícitas: Alocadas e liberadas por comandos específicos do programador, como malloc e free em C.

4. Implícitas: Gerenciadas automaticamente pela linguagem, como em Java, onde o coletor de lixo cuida da liberação. As variáveis dinâmicas do monte oferecem grande flexibilidade, mas podem ser menos eficientes devido ao gerenciamento de memória.



Vinculações de Armazenamento e Tempo de Vida

java

 Copiar código

```
// Exemplo de variável estática
public class Exemplo {
    static int valorEstatico = 10; // variável estática

    public static void main(String[] args) {
        int valorLocal = 5; // variável dinâmica da pilha
        System.out.println("Valor Estático: " + valorEstatico);
        System.out.println("Valor Local: " + valorLocal);
    }
}
```

Tópicos em Programação Imperativa



A essência das linguagens de programação imperativas é o papel dominante das sentenças de atribuição.



O propósito de uma sentença de atribuição é modificar o valor de uma variável. Então, uma parte integral de todas as linguagens imperativas é o conceito de variáveis cujos valores mudam durante a execução de um programa.



Linguagens não imperativas algumas vezes incluem variáveis de um tipo diferente, como os parâmetros de funções em linguagens funcionais.

Tópicos em Programação Imperativa



Um operador pode ser unário por possuir um único operando, binário, dois operandos, ou ternário, três operandos. Na maioria das linguagens de programação imperativas, os operadores binários usam a notação convencional (infix), ou seja, eles aparecem entre seus operandos.



Uma exceção é Perl, possuindo alguns operadores pré-fixados* (prefix), precedendo seus operandos.

Tópicos em Programação Imperativa

Operadores aritméticos são usados para mais de um propósito. Por exemplo, nas linguagens de programação imperativas, o sinal + é usado para especificar a adição tanto de inteiros quanto de valores de ponto flutuante.

Algumas linguagens – Java, por exemplo – também o usam para concatenação de cadeias. Esse uso múltiplo de um operador é chamado de sobrecarga de operadores, uma prática considerada aceitável, desde que nem legibilidade nem confiabilidade sejam comprometidas.

Tópicos em Programação Imperativa

```
public class OperadoresAritmeticos {  
  
    public static void main(String[] args) {  
        // Adição de inteiros  
        int a = 10;  
        int b = 20;  
        int somaInteiros = a + b;  
        System.out.println("Soma de inteiros: " + somaInteiros);  
  
        // Adição de valores de ponto flutuante  
        double x = 5.5;  
        double y = 3.2;  
        double somaPontoFlutuante = x + y;  
        System.out.println("Soma de valores de ponto flutuante: " + somaPontoFlutuante);  
  
        // Concatenação de cadeias  
        String str1 = "Olá, ";  
        String str2 = "mundo!";  
        String concatenacao = str1 + str2;  
        System.out.println("Concatenação de cadeias: " + concatenacao);  
    }  
}
```

Tópicos em Programação Imperativa



A maioria das linguagens de programação imperativas restringe os tipos que podem ser retornados por suas funções.



Variáveis locais em funções em linguagens de programação imperativas mantêm o estado da função. Na matemática, não existe o conceito do estado de uma função.



O controle de fluxo em definições de funções matemáticas é consideravelmente diferente do de programas em linguagens de programação imperativas.



Enquanto funções em linguagens imperativas são definidas como coleções de sentenças que podem incluir diversos tipos de sentenças de controle de fluxo, funções matemáticas não têm sentenças múltiplas e usam apenas recursão e expressões condicionais para a avaliação do controle de fluxo.

Vantagens e Desvantagens

Vantagens:

- Eficiência (embute modelo de Von Neumann);
- Modelagem “natural” de aplicações do mundo real;
- Paradigma dominante e bem estabelecido;

Desvantagens:

- erros introduzidos durante **manutenção**
- descrições demasiadamente operacionais focalizam **o como** e não **o que**

Considerações

Para aqueles que estão confortáveis com programação imperativa, a troca para a funcional pode ser difícil é desestimulante. Por outro lado, quem começa com uma linguagem funcional não estranha os programas funcionais.

Obviamente programadores imperativos, que acreditam que, por estarem de bem com a programação imperativa, ela é de certa forma uma maneira intrinsecamente mais natural de programar. Alguns programadores funcionais se sentem da mesma forma a respeito dos programas funcionais.