PARADIGMAS E LINGUAGENS DE PROGRAMAÇÃO

ENGENHARIA DA COMPUTAÇÃO – UFC/SOBRAL

Prof. Danilo Alves

danilo.alves@alu.ufc.br

INTRODUÇÃO AO CONCEITO DE SUBPROGRAMAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Modularização
- Passagem de parâmetros
- Variáveis locais e globais
- Recursividade

INTRODUÇÃO

UNIVERSIDADE FEDERAL DO CEARÁ

- Dois recursos fundamentais de abstração
  - Abstração de processos
    - Desde o início da história das linguagens de programação
  - Abstração de dados
    - Desde o início dos anos 1980

FUNDAMENTOS DE SUBPROGRAMA

UNIVERSIDADE FEDERAL DO CEARÁ

- Cada subprograma tem um único ponto de entrada
- A unidade de programa chamadora é suspensa durante a execução do subprograma chamado
- O controle sempre retorna para o chamador quando a execução do subprograma termina

DEFINIÇÕES BÁSICAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Uma definição de subprograma descreve a interface e as ações da abstração de um subprograma
- Uma chamada a subprograma é a requisição explícita que diz que o subprograma deve ser executado
- Um cabeçalho de subprograma é a primeira parte da definição, incluindo o nome, o tipo de subprograma e os parâmetros formais
- O perfil de parâmetros (ou assinatura) de um subprograma contém o número, a ordem e os tipos de seus parâmetros formais
- O protocolo é o perfil de parâmetros mais, se for uma função, seu tipo de retorno

DEFINIÇÕES BÁSICAS

UNIVERSIDADE FEDERAL DO CEARÁ

- Declarações de funções em C e C++ são chamadas de protótipos
- Uma declaração de subprograma fornece o protocolo, mas não inclui seu corpo
- Um parâmetro formal é uma variável listada no cabeçalho do subprograma e usado nele
- Um parâmetro real representa um valor ou endereço usado na sentença de chamada do subprograma

CORRESPONDÊNCIA ENTRE OS PARÂMETROS REAIS E FORMAIS

- Posicional

  – A vinculação dos parâmetros reais a parâmetros formais é por posição: o primeiro real é vinculado ao primeiro formal e assim por diante

– Seguro e efetivo

- Palavra-chave

  – O nome do parâmetro formal a que um parâmetro real deve ser vinculado é especificado com o parâmetro real

  – Vantagem: Parâmetros podem aparecer em qualquer ordem, evitando erros de correspondência

  – Desvantagem: O usuário deve saber os nomes dos parâmetros formais

VALORES PADRÃO DE PARÂMETROS FORMAIS

UNIVERSIDADE FEDERAL DO CEARÁ

- Em certas linguagens (como C++, Python, Ruby, Ada e PHP), parâmetros formais podem ter valores padrão (se nenhum parâmetro real é passado)

  – Em C++, parâmetros padrão devem aparecer por último, já que os parâmetros são posicionalmente associados

- Número variável de parâmetros
  - – C# permite que os métodos aceitem um número variável de parâmetros, desde que sejam do mesmo tipo — o método especifica seu parâmetro com o modificador **params**
  - – Em Python, o real é uma lista de valores e o parâmetro formal correspondente é um nome com um asterisco
  - – Em Lua, um número variável de parâmetros é representado por um parâmetro formal com reticências

BLOCOS EM RUBY

UNIVERSIDADE FEDERAL DO CEARÁ

- Ruby inclui métodos de iteração para suas estruturas de dados
- Iteradores são implementados com blocos, que podem ser definidos por aplicações
- Os blocos podem ter parâmetros formais, que são especificados entre barras verticais; o bloco que é passado para o subprograma chamado é chamado com uma sentença **yield**

```
def fibonacci(last)
  first, second = 1, 1
  while first <= last
    yield first
    first, second = second, first + second
  end
end

puts "Fibonacci numbers less than 100 are:"
fibonacci(100) {|num| print num, " "}
puts
```

PROCEDIMENTO E FUNÇÕES

UNIVERSIDADE FEDERAL DO CEARÁ

- Existem duas categorias de subprogramas
  - **Procedimento** são coleções de sentenças que definem computações parametrizadas
  - **Funções** se parecem estruturalmente com os procedimentos, mas são semanticamente modeladas como funções matemáticas

AMBIENTES DE REFERENCIAMENTO LOCAL

UNIVERSIDADE FEDERAL DO CEARÁ

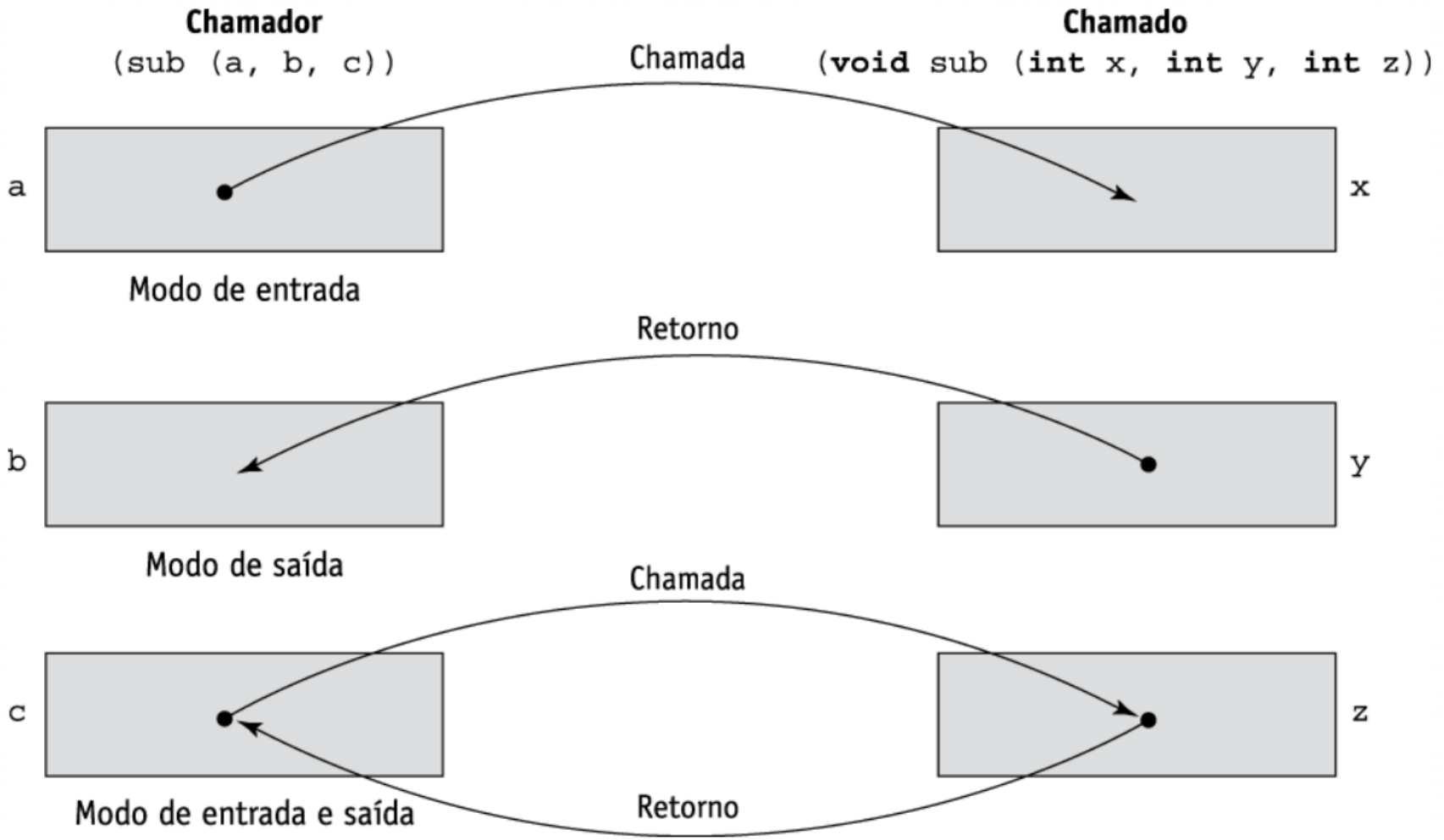
- Variáveis locais podem ser dinâmicas da pilha
  - Vantagens
    - Suporte para recursão
    - Armazenamento para variáveis locais é compartilhado entre alguns subprogramas
  - Desvantagens
    - Custo para alocação, liberação, tempo de inicialização
    - Endereçamento indireto
    - Variáveis locais podem ser estáticas

MODELOS SEMÂNTICOS DE PASSAGEM DE PARÂMETROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Modo de entrada
- Modo de saída
- Modo de entrada e saída

MODELOS DE PASSAGEM DE PARÂMETROS



MODELOS CONCEITUAIS DE TRANSFERÊNCIA E DADOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Um valor real é copiado (movimento físico)

- Um caminho de acesso é transmitido

PASSAGEM POR VALOR

UNIVERSIDADE FEDERAL DO CEARÁ

- O valor do parâmetro real é usado para inicializar o parâmetro formal correspondente
  - Normalmente implementada por cópia
  - Poderia ser implementada transmitindo um caminho de acesso para o valor do parâmetro real no chamador, mas isso requereria que o valor estivesse em uma célula com proteção contra escrita (uma que pudesse ser apenas lida)
  - **Desvantagens** (se cópias são usadas): é necessário armazenamento adicional para o parâmetro formal e o movimento pode ser custoso

PASSAGEM POR RESULTADO

UNIVERSIDADE FEDERAL DO CEARÁ

- Quando um parâmetro é passado por resultado, nenhum valor é transmitido para o subprograma
- O parâmetro formal correspondente age como uma variável local, mas logo antes de o controle ser transmitido de volta para o chamador

def primeiroCliente() *result Cliente.first* *end* *x = primeiroCliente();*	*Neste exemplo numa linguagem imaginária, a função "primeiroCliente" usa a palavra-chave **result** para retornar o resultado da sua execução ao procedimento chamador. O objeto retornado será atribuido a **x**.*
procedure teste(int a, int b) *...* *end* *teste(x, y);*	*Neste novo exemplo, na chamada ao procedimento **teste** os valores dos argumentos **x** e **y** não serão passados para **a** e **b**. Ao contrário, ao final do procedimento **teste** os valores atualizados de **a** e **b** é que serão retornados para os argumentos **x** e **y**.*

PASSAGEM POR VALOR-RESULTADO

UNIVERSIDADE FEDERAL DO CEARÁ

- Modelo de implementação para parâmetros no modo de entrada e saída no qual os valores reais são copiados
- Chamada de passagem por cópia
- Parâmetros formais têm armazenamento local

procedure troca3 (a : in out Integer, b : in out Integer) is *temp : Integer;* *begin* *temp := a;* *a := b;* *b := temp;* *end troca3;* *troca3(c, d);*	*Neste exemplo, em sintaxe similar a ADA, em se tratando de passagem por valor-resultado, ocorre o seguinte:* *1. os endereços dos argumentos **c** e **d** são guardados;* *2. os valores dos argumentos **c** e **d** são repassados para os parâmetros **a** e **b**.* *3. as trocas entre os valores de **a** e **b** são realizadas dentro do procedimento.* *4. os resultados (novos valores de **a** e **b**) são retornados e copiados nos endereços originais de **c** e **d**, que foram guardados no passo 1.* *Ou seja, **os valores de c e d foram realmente trocados**. Esse comportamento equivale à passagem por referência.*

PASSAGEM POR REFERÊNCIA

UNIVERSIDADE FEDERAL DO CEARÁ

- Transmite um caminho de acesso
- **Vantagem:** processo de passagem é eficiente (não são necessárias cópias nem espaço duplicado)
- **Desvantagens**
  - Acessos mais lentos (do que na passagem por valor) a parâmetros formais
  - Potenciais efeitos colaterais (colisões)
  - Apelidos podem ser criados

PASSAGEM POR NOME

UNIVERSIDADE FEDERAL DO CEARÁ

- Por substituição textual
- Quando os parâmetros são passados por nome, o parâmetro real é, na prática, textualmente substituído pelo parâmetro formal correspondente em todas as suas ocorrências no subprograma

```
#define mult(a,b) a*b
mult(3+4,6)
```

IMPLEMENTANDO MÉTODOS DE PASSAGEM DE PARÂMETROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Na maioria das linguagens contemporâneas, a comunicação via parâmetros ocorre por meio da pilha de tempo de execução
- Passagem por referência é a mais simples de implementar; apenas seu endereço deve ser colocado na pilha
- Um erro sutil, mas fatal, pode ocorrer com parâmetros com passagem por referência e passagem por valor-resultado

MÉTODOS DE PASSAGEM DE PARÂMETROS DAS PRINCIPAIS LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- C
  - Passagem por valor
  - A passagem por referência é atingida por meio do uso de ponteiros como parâmetros
- C++
  - Inclui o tipo referência para passagem por referência
- Java
  - Todos os parâmetros têm passagem por valor
  - Parâmetros objetos têm passagem por referência

MÉTODOS DE PASSAGEM DE PARÂMETROS DAS PRINCIPAIS LINGUAGENS

UNIVERSIDADE FEDERAL DO CEARÁ

- Fortran 95

  – Parâmetros podem ser declarados para serem dos modos de entrada, de saída ou ambos

- C#

  – Método padrão: passagem por valor

  – Passagem por referência é especificada precedendo um parâmetro formal e seu real correspondente com **ref**

- PHP: similar a C#

- Perl

  – parâmetros reais são implicitamente colocados em uma matriz pré-definida chamada de @_

VERIFICAÇÃO DE TIPOS DOS PARÂMETROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Considerado importante para confiabilidade
- FORTRAN 77 e versão original do C: não tinham
- Pascal, FORTRAN 90, Java e Ada: sempre requerem
- ANSI C e C++: escolha é feita pelo usuário
- Perl, JavaScript e PHP não requerem verificação de tipos
- Em Python e Ruby, variáveis não têm tipos (objetos sim), então verificação de tipos dos parâmetros não é possível

MATRIZES MULTIDIMENSIONAIS COMO PARÂMETROS

UNIVERSIDADE FEDERAL DO CEARÁ

- Se uma matriz multidimensional é passada para um subprograma e o subprograma é compilado separadamente, o compilador precisa saber o tamanho declarado dessa matriz para construir a função de mapeamento de armazenamento

- Matriz pode ser passada como um ponteiro, e as dimensões reais da matriz podem ser incluídas como parâmetros

MATRIZES MULTIDIMENSIONAIS COMO PARÂMETROS: JAVA/C#

UNIVERSIDADE FEDERAL DO CEARÁ

- Matrizes são objetos;
- Cada matriz herda uma constante nomeada (**length** em Java, **Length** em C#) que é configurada para o tamanho da matriz quando o objeto matriz é criado

PARÂMETROS QUE SÃO SUBPROGRAMAS: VERIFICAÇÃO DE TIPOS DOS PARÂMETROS

UNIVERSIDADE FEDERAL DO CEARÁ

- C e C++: as funções não podem ser passadas como parâmetros, mas ponteiros para funções podem

- FORTRAN 95 possui um mecanismo para fornecer tipos de parâmetros para subprogramas que são passados como parâmetros

- Ada não permite que subprogramas sejam passados como parâmetros; a funcionalidade é fornecida pelos recursos de tipos genéricos

SUBPROGRAMAS SOBRECARREGADOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Um **subprograma sobrecarregado** é um subprograma que possui o mesmo nome de outro subprograma no mesmo ambiente de referenciamento
- C++, Java, C# e Ada incluem subprogramas sobrecarregados pré-definidos
- Em Ada, o tipo de retorno e uma função sobrecarregada podem ser usados para desambiguar chamadas (logo, duas funções sobrecarregadas podem ter o mesmo perfil de parâmetros e diferirem apenas em seus tipos de retorno)
- Ada, Java, C++ e C# permitem aos usuários escrever múltiplas versões de subprogramas com o mesmo nome

SUBPROGRAMAS GENÉRICOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Um subprograma **genérico** ou **polimórfico** recebe parâmetros de diferentes tipos em diferentes ativações
- Subprogramas sobrecarregados fornecem um tipo particular de polimorfismo chamado de **poliformismo ad hoc**
- O **polimorfismo paramétrico** é fornecido por um subprograma que recebe parâmetros genéricos que são usados em expressões de tipo que descrevem os tipos dos parâmetros do subprograma

SUBPROGRAMAS GENÉRICOS

UNIVERSIDADE FEDERAL DO CEARÁ

- Java 5.0

1. Parâmetros genéricos em Java 5.0 podem ser classes
2. Métodos genéricos em Java 5.0 são instanciados apenas uma vez, como métodos realmente genéricos
3. As restrições podem ser especificadas na faixa de classes que podem ser passados para o método genérico como parâmetros genéricos

EXEMPLO DE POLIMORFISMO DE PARÂMETROS: C++

UNIVERSIDADE FEDERAL DO CEARÁ

```
template <class Type>
Type max(Type first, Type second) {
    return first > second ? first : second;
}
```

- O modelo acima pode ser instanciada para qualquer tipo em que o operador > é definido

```
int max (int first, int second) {
    return first > second? first : second;
}
```