Subprogramas (função, método e procedimento)

Prof. Julio Arakaki

Ciência da Computação

PUC-SP

PUC-SP

Subprograma

- Cada subprograma possui um único ponto de entrada.

- O chamador (programa ou outro subprograma) é suspenso durante a execução do subprograma chamado.

- O controle sempre retorna ao chamador quando a execução do subprograma chamado termina.

PUC-SP

Subprograma

- A definição de um subprograma descreve a interface de utilização e as ações a serem executadas.

- Uma chamada de subprograma é uma solicitação explícita ("call") para que o subprograma seja executado.

- O cabeçalho de subprograma é a primeira parte da definição, inclui o nome, o tipo de retorno e os parâmetros. Exemplo em C*:*

```
void somar (int x, int y) {...}
```

- O perfil (também conhecido como assinatura) de um subprograma inclui o número, a ordem e os tipos de seus parâmetros.

PUC-SP

Definições básicas

- declarações de subprogramas em C e C++ são chamadas de protótipos (assinaturas).

Em java, utiliza-se interfaces:

```
interface IPilha {
    void push (Item item);
    Item pop();
    boolean isEmpty();
}
```

- na declaração de subprogramas tem apenas o protocolo de utilização e não o corpo.

- a declaração de subprograma permite a verificação do número, da ordem e dos tipos dos parâmetros. Além disso, fornece o seu tipo de retorno.

PUC-SP

Definições básicas

- Os subprogramas também descrevem computações.

- Eles podem acessar dados de:

  1. Variáveis locais

  2. Variáveis não locais (mas visíveis no subprograma).
     Uso de variáveis globais (não bom!!).

  3. Parâmetros – passagem de parâmetros (permite uma computação parametrizada)

PUC-SP

Parâmetros (argumentos) – valores padrão

- Em linguagens como Python, Ruby, C++ e PHP, os parâmetros nas funções, podem ter valores padrão.
Exemplos:

Em Python:

```
def compute_pay(income, exemptions = 1, tax_rate)
…
pay = compute_pay(20000.0, tax_rate = 0.15)
```

Em C++:

```
float compute_pay(float income, float tax_rate, int exemptions = 1)
...
pay = compute_pay(20000.0, tax_rate = 0.15)
```

PUC-SP

Parâmetros (argumentos) - variáveis

- Algumas linguagens possuem funções com parâmetros variáveis. Por exemplo printf **de C, C++ e Java.**

```
   .printf("%d, %f, %s", idade, nota, nome);
```

- em C#, o número de parâmetros podem ser variáveis desde que sejam do mesmo tipo:

Se DisplayList **foi definido numa classe** MyClass*:*

```
...
public void DisplayList(params int[] list) {
   foreach (int next in list) {
      Console.WriteLine("Next value {0}", next);
   }
}
...
```

PUC-SP

Parâmetros (argumentos) - variáveis

Sejam as declarações:

```
Myclass myObject = new Myclass;
int[] myList = new int[6] {2, 4, 6, 8, 10, 12};
```

DisplayList **poderia ser chamada:**

```
myObject.DisplayList(myList); // um parametro
myObject.DisplayList(2, 4, 3 * x – 1, 17); // tres parametros
```

PUC-SP

Procedimentos e Funções

- São consideradas estratégias para ampliar a linguagem.

- São conjunto de sentenças que permitem/definem computações parametrizadas.

- Funções retornam valores. Os Procedimentos não retornam valores.

- Variáveis definidas dentro de subprogramas são denominadas variáveis locais, pois seu escopo é o corpo do subprograma.

- Na maioria das linguagens, as variáveis locais são dinâmicas (por “default”). Em C, C++, Java tem o especificador `static`.

PUC-SP

Passagem de Parâmetros

- Três modos: in (de entrada), out (de saída), inOut (de entrada e saída).

- Por valor:
  Uma cópia do parâmetro é passada para o subprograma;

- Por referência:
  O endereço do parâmetro é passado para o subprograma;
  É eficiente em termos de tempo e gasto de memória.

Passagem de Parâmetros

. Em C para fazer passagem por referência utiliza-se os ponteiros.

. Em C++ utiliza-se o “ponteiro especial” & (referência)

. Em Java, todos os parâmetros são passados por valor. Objetos são passados por referência.

. Em Fortran 95+, os parâmetros podem ser declarados com in, out, inout.

. Em C#, o padrão é passagem por valor. Para passar por referência, utiliza-se o especificador `ref`.

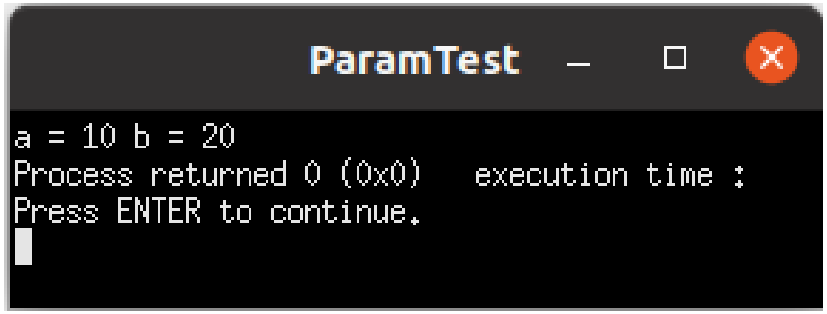
. PHP é similar a C#.

PUC-SP

Passagem de Parâmetros - Exemplos

Em C:

```
#include <stdio.h>
#include <stdlib.h>

void troca(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}

int main() {
    int a = 10, b = 20;
    troca (a, b);
    printf("a = %d b = %d", a, b); // o que será impresso??
}
```

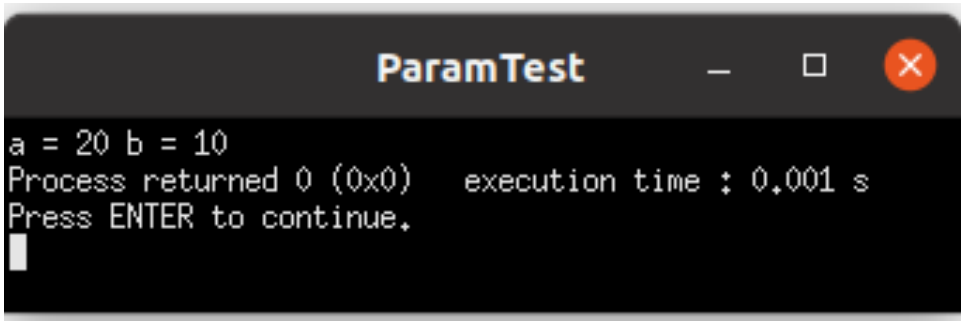


PUC-SP

Passagem de Parâmetros - Exemplos

Solução em C:

```
#include <stdio.h>
#include <stdlib.h>

void troca(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int a = 10, b = 20;
    troca (&a, &b);
    printf("a = %d b = %d", a, b); // o que será impresso??
}
```

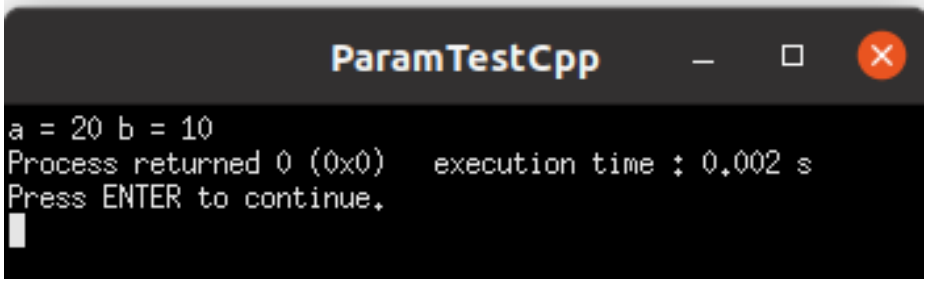


PUC-SP

Passagem de Parâmetros - Exemplos

Solução em C++:

```
#include <iostream>

using namespace std;

void troca(int &a, int &b) {
        int temp = a;
        a = b;
        b = temp;
}

int main() {
        int a = 10, b = 20;
        troca (a, b);
        printf("a = %d b = %d", a, b); // a=? b=?
}
```

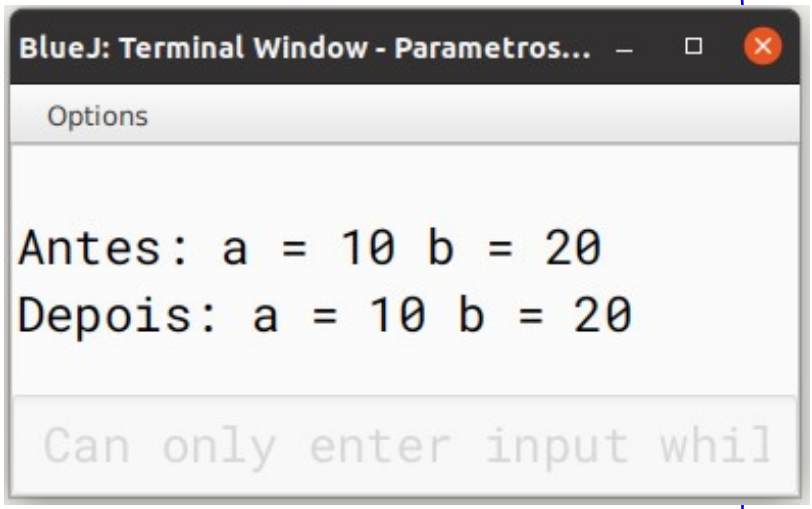


PUC-SP

Passagem de Parâmetros - Exemplos

Em Java:

```
public class Parametros {
    public void troca(int a, int b) {
        int temp = a;
        a = b;
        b = temp;
    }
}

public class Aplicacao {
    public static void main(String args[]) {
        Parametros param = new Parametros();
        int a = 10;
        int b = 20;

        System.out.printf("\nAntes: a = %d b = %d", a, b);
        param.troca(a, b);
        System.out.printf("\nDepois: a = %d b = %d", a, b);
    }
}
```

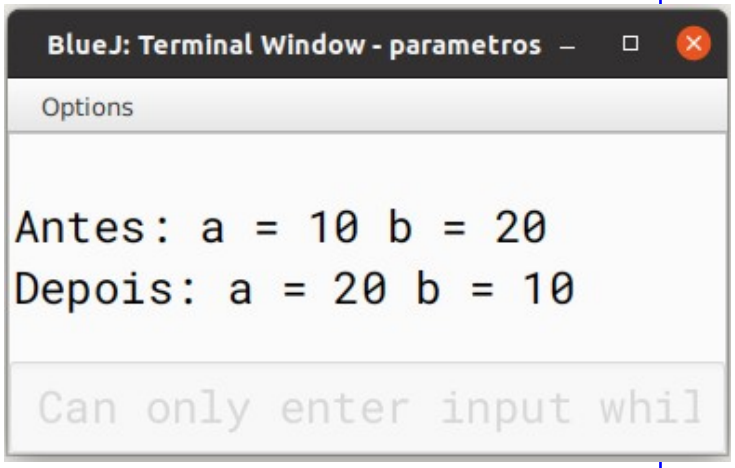


PUC-SP

Passagem de Parâmetros - Exemplos

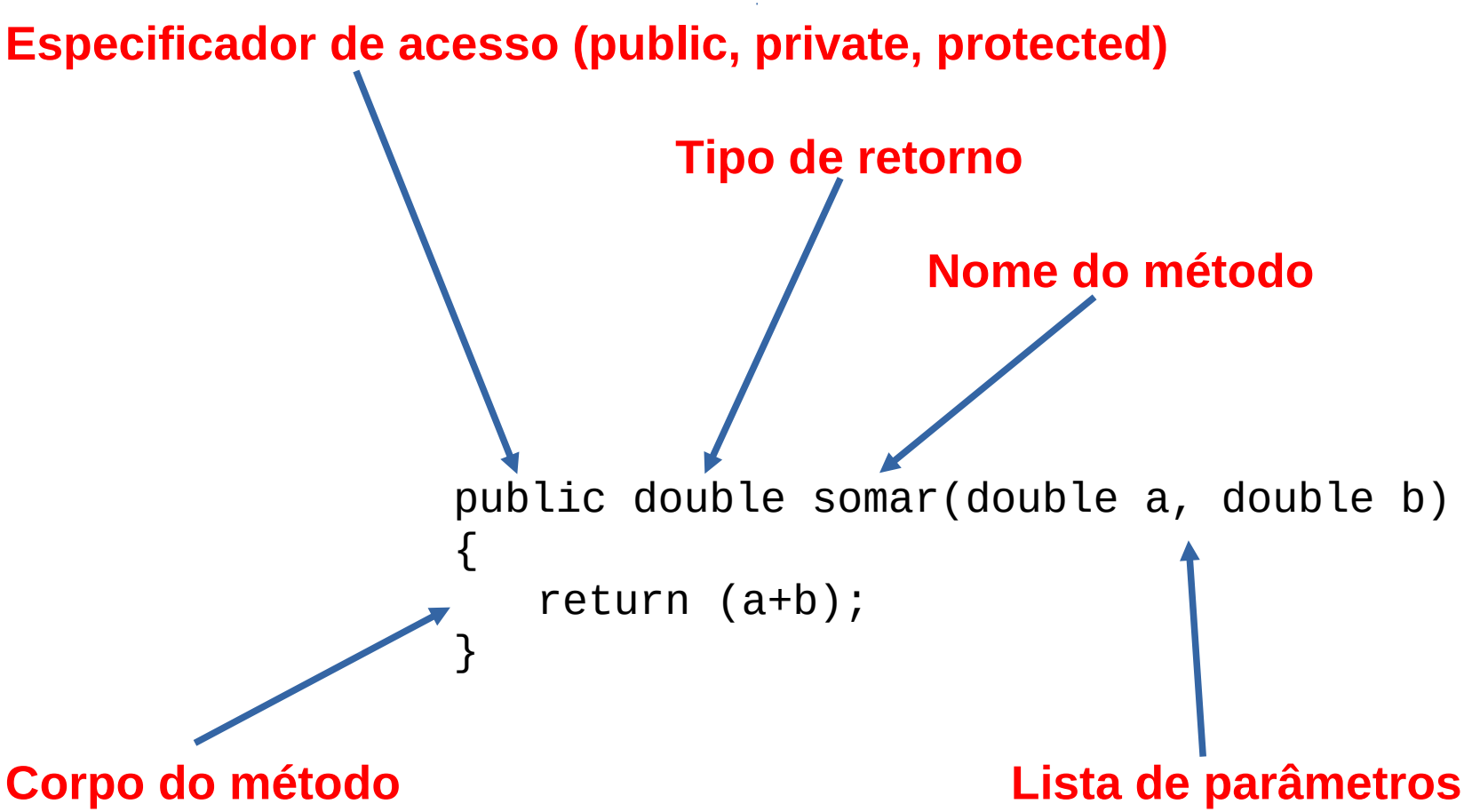
Solução em Java:

```
public class Parametros {
    public void troca(int a[], int b[]) {
        int temp = a[0];
        a[0] = b[0];
        b[0] = temp;
    }
}

public class Aplicacao {
    public static void main(String args[]) {
        Parametros param = new Parametros();
        int a[] = {10};
        int b[] = {20};

        System.out.printf("\nAntes: a = %d b = %d", a[0], b[0]);
        param.troca(a, b);
        System.out.printf("\nDepois: a = %d b = %d", a[0], b[0]);
    }
}
```



PUC-SP

Em Java: método (definição)



PUC-SP

Referências

1. Sebesta, Robert. “Conceitos de Linguagem de Programação”, 11a. Ed. Porto Alegre, Bookman, 2018.

2. Horstmann, Cay, “Conceitos de Computação com Java”. 5a. Ed. – Dados eletrônicos – Porto Alegre, Bookman, 2009.