ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES

Disciplina: 113476

Profa. Carla Denise Castanho

Universidade de Brasília – UnB
Instituto de Ciências Exatas – IE
Departamento de Ciência da Computação – CIC

5. SUBALGORITMOS

PASSAGEM DE PARÂMETROS

Passagem de Parâmetros

- Quando chamamos uma função, podemos passar **valores** que ela utilizará em seu processamento. Esses valores são chamados formalmente de **parâmetros**.
- Por padrão, parâmetros são passados **por valor**, ou seja, quando a função é chamada, a expressão colocada na **posição** daquele parâmetro é calculada e o **valor** resultante é **copiado** para as respectivas variáveis locais da função.

Passagem de Parâmetros

- Quando chamamos uma função, podemos passar **valores** que ela utilizará em seu processamento. Esses valores são chamados formalmente de **parâmetros**.
- Por padrão, parâmetros são passados **por valor**, ou seja, quando a função é chamada, a expressão colocada na **posição** daquele parâmetro é calculada e o **valor** resultante é **copiado** para as respectivas variáveis locais da função.

Exemplo - Passagem por Valor

```
Algoritmo PassagemPorValor

Função Quadrado(n : inteiro) : inteiro
Início
    retorne n * n
Fim

Variáveis
    num : inteiro
Início
    Leia (num)
    Escreva (Quadrado(num))
Fim
```

Passagem de Parâmetros

- Quando chamamos uma função, podemos passar **valores** que ela utilizará em seu processamento. Esses valores são chamados formalmente de **parâmetros**.
- Por padrão, parâmetros são passados **por valor**, ou seja, quando a função é chamada, a expressão colocada na **posição** daquele parâmetro é calculada e o **valor** resultante é **copiado** para as respectivas variáveis locais da função.

Exemplo - Passagem por Valor

```
Algoritmo PassagemPorValor

Função Quadrado(n : inteiro) : inteiro
Início
    retorne n * n
Fim

Variáveis
    num : inteiro
Início
    Leia (num)
    Escreva (Quadrado(num))
Fim
```

Quando Quadrado é chamada, o valor de num é copiado para a variável local n.
Lembra que parâmetros de funções também são variáveis? É por esse motivo!

Passagem de Parâmetros

- Portanto, quando chamamos uma função, podemos passar como parâmetro qualquer **expressão** (inclusive variáveis, constantes e o retorno de outras funções). Apenas o valor final importa!
- Também é fácil concluir que, a cada chamada de uma função, seu bloco inicia com valores diferentes para os parâmetros. Cada chamada da função é **independente** das outras!
- E, por fim, que, quando passamos uma variável, passamos apenas seu valor, a função chamada **não altera o valor da variável original**!
- Mas e se quiséssemos justamente isso? Por exemplo, como a função `scanf` de C faz para alterar o valor da variável que passamos?

Ponteiros

- Vamos ver se você lembra: o que é uma variável mesmo?

Ponteiros

- Vamos ver se você lembra: o que é uma variável mesmo?
- Não se esqueça jamais! Uma variável é uma região de **memória**, que recebeu um nome e um tipo.

Ponteiros

- Vamos ver se você lembra: o que é uma variável mesmo?
- Não se esqueça jamais! Uma variável é uma região de **memória**, que recebeu um nome e um tipo.
- Mas se uma variável é simplesmente memória, ela certamente tem um **endereço**!

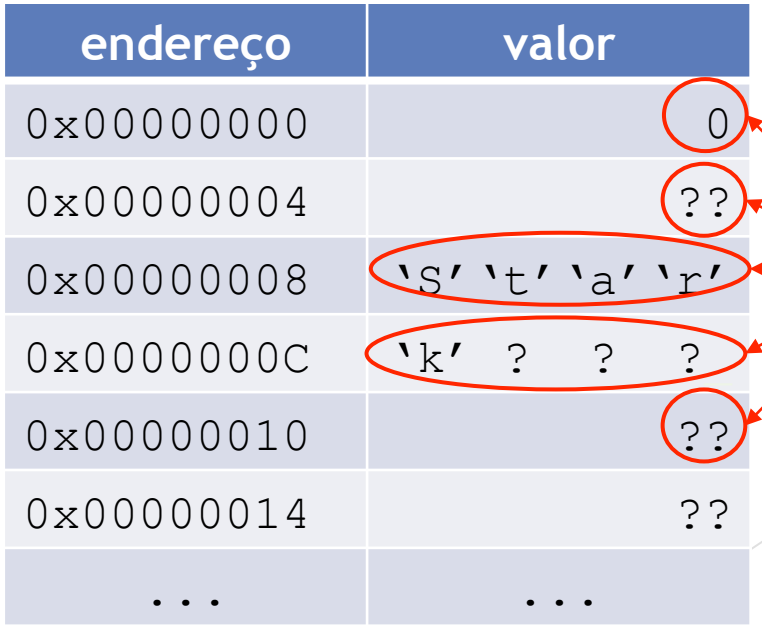
Ponteiros

- E como podemos saber o endereço de uma variável? Utilizamos **ponteiros**!
- Um **ponteiro** é um tipo especial de variável, ele serve para guardar o **endereço** de outras variáveis.
- Por esse motivo, ponteiros também são chamados de **referências**, porque eles referenciam, isto é, eles “apontam” para outra variável.
- Vamos entender melhor...

Ponteiros

- Lembra desse exemplo?

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : booleano
```



Ponteiros

- Lembra desse exemplo?
- E se eu criar uma variável para guardar o endereço de a?

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : boolean
```

endereço	valor
0x00000000	0
0x00000004	??
0x00000008	'S' 't' 'a' 'r'
0x0000000C	'k' ? ? ?
0x00000010	??
0x00000014	??
...	...

conteúdo

MEMÓRIA

Ponteiros

- Lembra desse exemplo?
- E se eu criar uma variável para guardar o endereço de a?

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : boolean
ptr_a : *real = &a
```

endereço	valor
0x00000000	0
0x00000004	??
0x00000008	'S' 't' 'a' 'r'
0x0000000C	'k' ? ? ?
0x00000010	??
0x00000014	0x00000004
...	...

conteúdo

MEMÓRIA

Ponteiros

- Lembra desse exemplo?
- E se eu criar uma variável para guardar o endereço de a?

A variável ptr_a é do tipo ponteiro para real.

```
n : inteiro = 0
a : real
sobrenome : string = "Stark"
achou : boolean
ptr_a : *real = &a
```

ptr_a é inicializada com o endereço de a

endereço	valor
0x00000000	0
0x00000004	??
0x00000008	'S' 't' 'a' 'r'
0x0000000C	'k' ? ? ?
0x00000010	??
0x00000014	0x00000004
...	...

MEMÓRIA

Ponteiros

- Dado que eu tenho o endereço de uma variável, como faço para alterar seu valor?
- Nesse caso, vamos **derreferenciar** o ponteiro, isto é vamos mexer na variável apontada por ele:

Ponteiros

- Dado que eu tenho o endereço de uma variável, como faço para alterar seu valor?
- Nesse caso, vamos **derreferenciar** o ponteiro, isto é vamos mexer na variável apontada por ele:

Exemplo - Acessando variáveis via ponteiros

```
*ptr_a ← 5.0
```

Ponteiros

- Dado que eu tenho o endereço de uma variável, como faço para alterar seu valor?
- Nesse caso, vamos **derreferenciar** o ponteiro, isto é vamos mexer na variável apontada por ele:

Exemplo - Acessando variáveis via ponteiros

```
*ptr_a ← 5.0
```

Essa expressão significa "faça o endereço apontado por ptr_a receber o valor 5.0"!
No nosso exemplo, estamos alterando o valor da variável original a!

Passagem por Referência

- Portanto, se precisarmos que uma função **altere** o valor de uma variável, podemos utilizar ponteiros!
- Nesse caso, estamos passando parâmetros **por referência**.

Passagem por Referência

- Portanto, se precisarmos que uma função **altere** o valor de uma variável, podemos utilizar ponteiros!
- Nesse caso, estamos passando parâmetros **por referência**.

Exemplo - Passagem por Referência

```
Algoritmo TrocaValores
Função Troca (a : *inteiro, b : *inteiro)
Variáveis
    aux : inteiro
Início
    aux ← *a
    *a ← *b
    *b ← aux
Fim
Variáveis
    num1, num2 : inteiro
Início
    Leia (num1, num2)
    Troca(&num1, &num2)
    Escreva (num1, num2)
Fim
```

Passagem por Referência

- Portanto, se precisarmos que uma função **altere** o valor de uma variável, podemos utilizar ponteiros!
- Nesse caso, estamos passando parâmetros **por referência**.

Exemplo - Passagem por Referência

```
Algoritmo TrocaValores
Função Troca (a : *inteiro, b : *inteiro)
Variáveis
    aux : inteiro
Início
    aux ← *a
    *a ← *b
    *b ← aux
Fim
Variáveis
    num1, num2 : inteiro
Início
    Leia (num1, num2)
    Troca(&num1, &num2)
    Escreva (num1, num2)
Fim
```

Os parâmetros a e b são os endereços de variáveis inteiras, passadas por referência.

Faça aux receber o valor da variável apontada por a.

Faça a variável apontada por a receber o valor da variável apontada por b.

Faça a variável apontada por b receber o valor de aux.

Troca é chamada recebendo os endereços de num1 e num2.

Passagem por Referência em C

- Escolhemos utilizar em pseudocódigo exatamente a mesma sintaxe da Linguagem C. Vamos ver o mesmo exemplo agora em C.

Exemplo - Passagem por Referência em C

```c
#include <stdio.h>

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

int main () {
    int num1, num2;
    scanf("%d%d", &num1, &num2);
    troca(&num1, &num2);
    printf("%d, %d\n", num1, num2);
    return 0;
}
```

Passagem de Parâmetros – Exercício

1. Escreva um algoritmo que leia três números A, B e C (parâmetros de uma função do 2º grau) e chame uma função que RETORNE o número de raízes reais da equação em função de A, B e C, bem como o valor das raízes.

Para resolver esse problema, calcule o

$$\text{DELTA} = B^2 - 4AC$$

Se **DELTA > 0**, existem duas raízes reais:

$$x_1 = \frac{-B + \sqrt{DELTA}}{2A} \qquad x_2 = \frac{-B - \sqrt{DELTA}}{2A}$$

Se **DELTA = 0**, existem duas raízes reais iguais:

$$x_1 = x_2 = \frac{-B}{2A}$$

Se **DELTA < 0**, não existem raízes reais.

SUGESTÃO: como parâmetros, passe A, B e C por VALOR e x1 e x2 por REFERÊNCIA. Dentro da função altere x1 e x2 e eles estarão automaticamente alterados na main. Você pode ter um último parâmetro (chamado de raizes, por ex.) que retorna 0, 1, ou 2. (0= não existem raízes, 1= existem duas raízes iguais e 2= existem duas raízes diferentes). Depois você testa o quê o parâmetro raizes retornou na main e mostra a msg adequada.