

Programação 2

Jordana S. Salamon

jssalamon@inf.ufes.br

DEPARTAMENTO DE INFORMÁTICA
CENTRO TECNOLÓGICO
UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO




Ponteiros

Definição

- ▶ Ponteiros são tipos especiais de variáveis que armazenam não um dado diretamente (um inteiro, um real, um caractere) mas sim o endereço de memória onde um dado se encontra;

Memória							
(<u>int</u>)	100						
		(<u>* int</u>)			0x00		



Declarando um Ponteiro

- ▶ Ponteiros são declarados como se fossem variáveis normais do tipo desejado, porém usando um * para indicar que é um ponteiro:
 - ▶ `int *ponteiroParaInt;`
- ▶ Assim como variáveis normais, um ponteiro não inicializado (como o exemplo acima) possui um valor qualquer (lixo de memória) e, portanto, aponta para um local desconhecido.
- ▶ Usar um ponteiro assim pode causar graves erros no programa;

Inicializando um Ponteiro

- ▶ Para inicializar um ponteiro, precisamos atribuir um endereço de memória a ele.
- ▶ Como não sabemos o endereço das variáveis, utilizamos o operador & (que já vimos no scanf!):

```
int umInteiro = 100;  
int *ponteiroParaInt  
ponteiroParaInt = &umInteiro;
```

Acessando um Ponteiro

- Com o ponteiro apontando para um endereço válido, podemos usá-lo. Para acessar o valor apontado pelo ponteiro, utilizamos novamente o *, que é o operador de resolução de referências:

```
int umInteiro = 100;
int *ponteiroParaInt;
ponteiroParaInt = &umInteiro;
printf("%d\n", *ponteiroParaInt);           // Imprime 100.
*ponteiroParaInt = *ponteiroParaInt * 2;    // Note os dois usos de *
printf("%d\n", umInteiro);                  // Imprime 200!
printf("%p\n", ponteiroParaInt);            // Imprime o endereço
```

- Como visto no exemplo acima, modificar o valor apontado pelo ponteiro obviamente modifica também o valor da variável original, cujo endereço colocamos no ponteiro;

Passagem por cópia x Passagem por referência

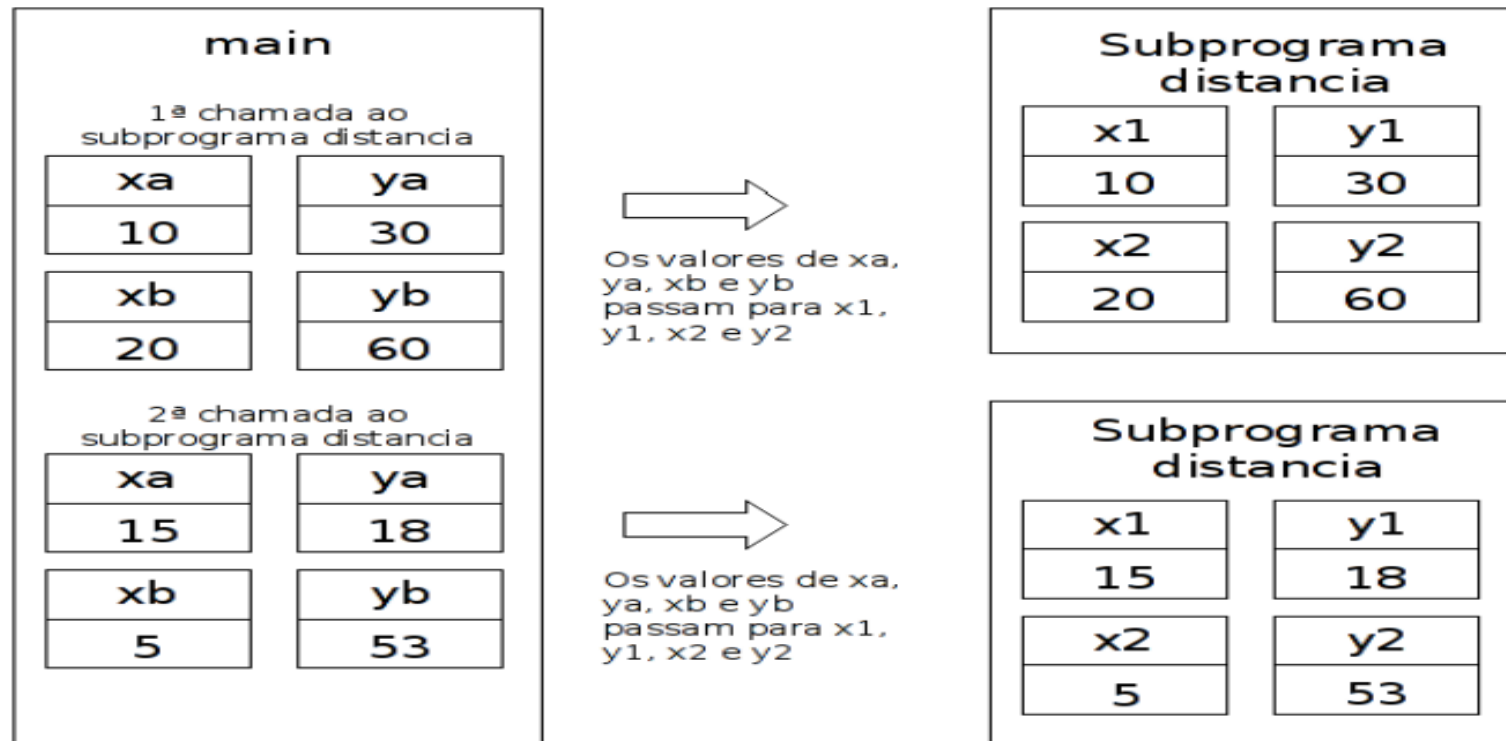
Passagem por cópia em funções

- ▶ Anteriormente nós aprendemos que quando uma variável é passada como parâmetro para um função, é realizada uma cópia do valor da variável que está sendo passada para a variável do parâmetro. Ex:

```
int soma (int a, int b) {  
    return a + b;  
}  
  
int main() {  
    int x, y, r;  
    printf("Digite 2 numeros: ");  
    scanf("%d %d", &x, &y);  
    r = soma (x,y);  
    printf("Resultado = %d\n", r);  
    r = soma (10,5);  
    printf("Resultado = %d\n", r);  
}
```


Passagem por cópia em funções

- ▶ Para cada chamada da função com seus respectivos parâmetros de entrada é feita uma instanciação da mesma.
- ▶ Não é possível modificar o valor das variáveis da função que efetuou a chamada.



Passagem por referência em funções

- Utilizando ponteiro torna-se possível passar a localização na memória da variável para a função, permitindo que a mesma seja modificada dentro da função. Ex:

```
void soma (int a, int b, int *r) {  
    *r = a + b;  
}  
  
int main() {  
    int x, y, r;  
    printf("Digite 2 numeros: ");  
    scanf("%d %d", &x, &y);  
    soma (x, y, &r);  
    printf("Resultado = %d\n", r);  
    soma (10, 5, &r);  
    printf("Resultado = %d\n", r);  
}
```

Passagem por referência em funções

- Fazer uma função que troque os valores de duas variáveis:

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

```
void troca(float x, float y) {  
    float aux;  
    aux = x;  
    x = y;  
    y = aux;  
}
```

```
void main() {  
    float a, b;  
  
    a = 3.56;  
    b = 2.4;  
  
    printf("a = %.2f e b = %.2f\n", a, b);  
    troca(a,b);  
    printf("a = %.2f e b = %.2f\n", a, b);  
}
```

Uma função só pode devolver um valor para aquela que faz a chamada. No entanto, a função troca precisa devolver dois valores. Como podemos resolver isso?

Passagem por referência em funções

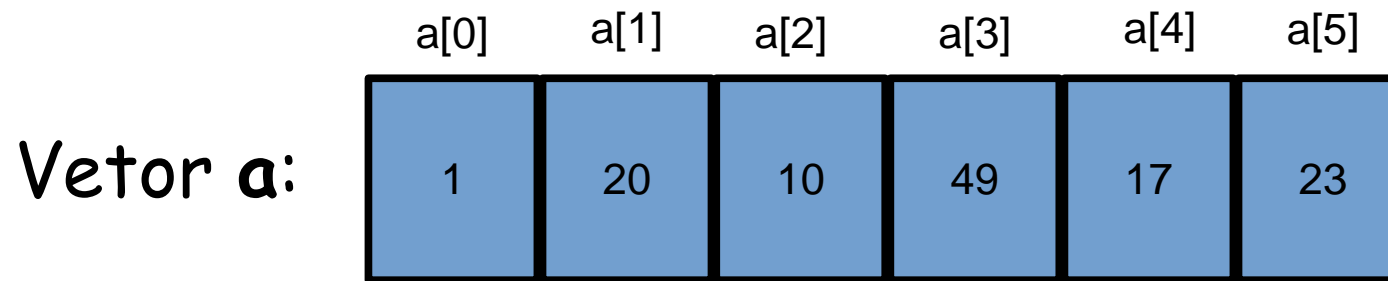
- Fazer uma função que troque os valores de duas variáveis.

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

```
void troca(float *x, float *y) {  
    float aux;  
    aux = *x;  
    *x = *y;  
    *y = aux;  
}
```

```
void main() {  
    float a, b;  
  
    a = 3.56;  
    b = 2.4;  
  
    printf("a = %.2f e b = %.2f\n", a, b);  
    troca(&a,&b);  
    printf("a = %.2f e b = %.2f\n", a, b);  
}
```

Ponteiros e vetores



pa

```
int *pa, x, y, z, a[6];
```

```
pa = &a[0];
```

```
x = *pa;
```

```
y = (pa+1);
```

&a[1]

```
z = *(pa+1);
```

a[1]

Ponteiros e vetores

- ▶ Por definição, o valor de uma variável do tipo vetor é o endereço do elemento 0 do vetor:

▶ `pa = &a[0];`  `pa = a`

- ▶ Diferença entre o nome de um vetor e um apontador:
 - ▶ Apontador: é uma variável (`pa = a` e `pa++` são operações válidas)
 - ▶ O nome de um vetor não é uma variável (`a = pa` e `a++` são operações inválidas)

Ponteiros e vetores

- ▶ Quando o nome de um vetor é passado como parâmetro de uma função, a variável que representa o parâmetro é um apontador e é instanciada pelo nome do vetor.

```
void inicializaVetor(int *vet,  
int n) {  
    int i;  
    for (i=0;i<n;i++)  
        vet[i] = i;  
}
```

```
#include <stdio.h>  
#define tam 100  
  
int main() {  
    int i, n, v[tam]= {0};  
    printf("Forneça um valor de n (<=  
        100): ");  
    scanf("%d", &n);  
    inicializaVetor(v, n);  
    printf("Vetor v: ");  
    for (i=0;i<n;i++)  
        printf("%d ", v[i]);  
    return 0;  
}
```

Qual o comportamento dos códigos abaixo?

```
int *pa, a[10];
```

```
pa = a;
```

```
a = pa;
```

```
pa++;
```

```
a++;
```

```
int *pa, a[10];
```

```
pa = a;
```

```
*pa += 2;
```

```
*pa++;
```



nemo

Qual o comportamento dos códigos abaixo? - Resposta

```
int *pa, a[10];
```

```
pa = a;
```

```
a = pa; (erro!)
```

```
pa++;
```

```
a++; (erro!)
```

```
int *pa, a[10];
```

```
*pa += 2;
```

```
*pa++;
```



Exercícios

- ▶ 1) Um ponteiro pode ser usado para dizer a uma função onde ela deve depositar o resultado de seus cálculos. Escreva uma função `hm` que converta minutos em horas-e-minutos. A função recebe um inteiro “`mnts`” e os endereços de duas variáveis inteiras, digamos `h` e `m`, e atribui valores a essas variáveis de modo que `m` seja menor que 60 e que $60 \cdot h + m$ seja igual a “`mnts`”. Escreva também uma função `main` que use a função `hm`.
- ▶ 2) Escreva uma função `mm` que receba um vetor inteiro `v[0..n-1]` e os endereços de duas variáveis inteiras, digamos `min` e `max`, e deposite nessas variáveis o valor de um elemento mínimo e o valor de um elemento máximo do vetor. Escreva também uma função `main` que use a função `mm`.

That's all Folks!



nemo