Universidade Federal do Espírito Santo

Programação II

Prof.ª Claudia Boeres (boeres@inf.ufes.br)

Departamento de Informática
Centro Tecnológico
Universidade Federal do Espírito Santo

Apontadores e Vetores

Relembrando o conceito de variáveis

- Contém valores inteiros, reais, char, endereços de memória...

- Operadores que manipulam endereços da memória:
  - &: fornece o endereço de uma variável
  - *: operador de indireção. Quando aplicado a um endereço, acessa o conteúdo que está armazenado neste endereço.

```
#include <stdio.h>
#define tam 10
main()
{
        int x = 1, y = 2; //declaração de variáveis inteiras
        int z[10]; // declaração de um vetor de inteiros
        int *p;  //declaração de uma variável que recebe um endereço de
                 //memória que armazena um inteiro

        printf("x = %i\n", x);
        printf("y = %i\n", y);
        p= &x;
        y = *p;
        *p = 0;
        p = &z[0];
}
```

Relembrando Passagem de Parâmetros em uma função

- 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.

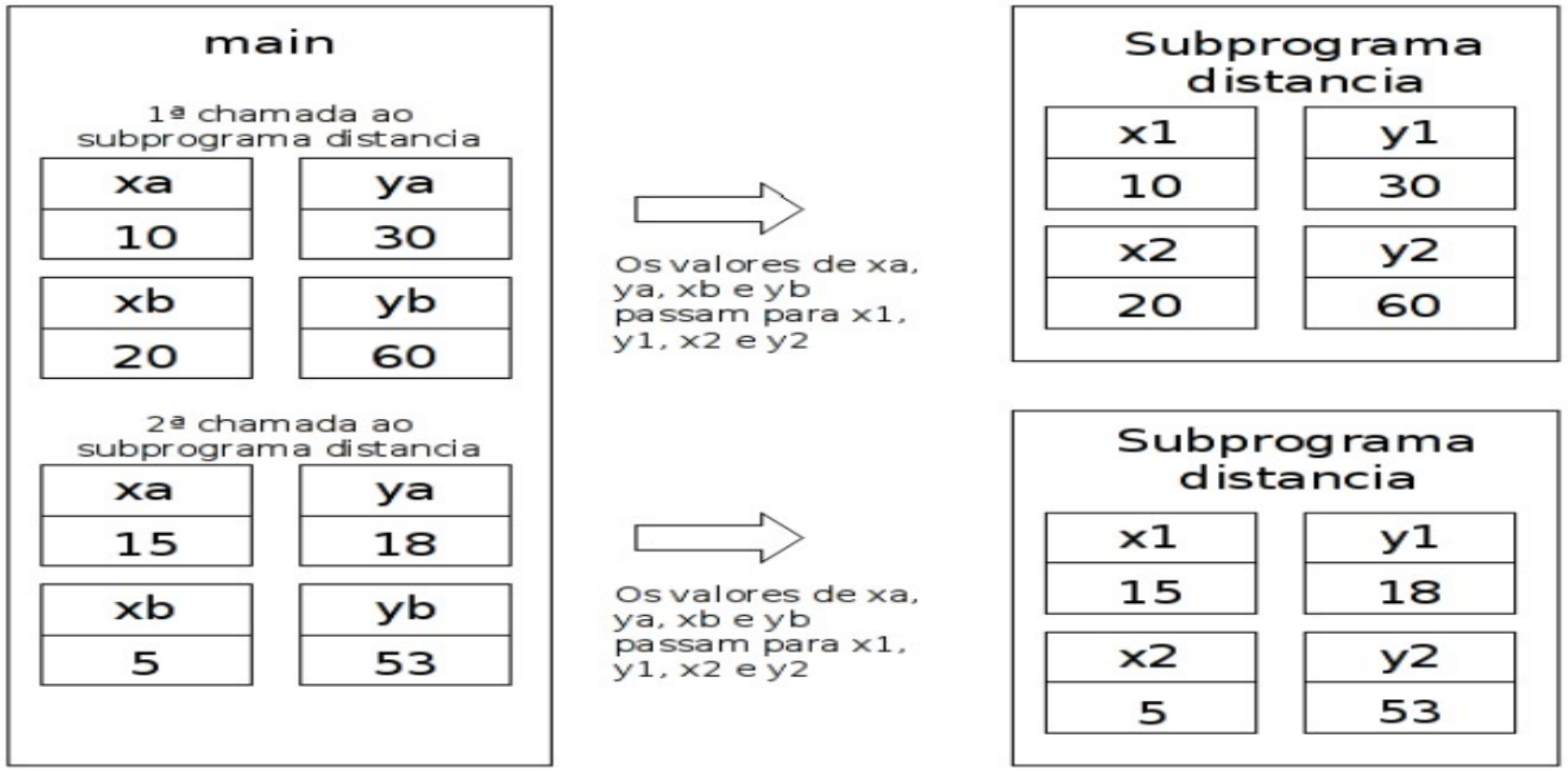


Figura 3.4: Passagem de parâmetro.

Troca de valores entre variáveis

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

```
#include <stdio.h>
void troca(float, float);

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);
}
```

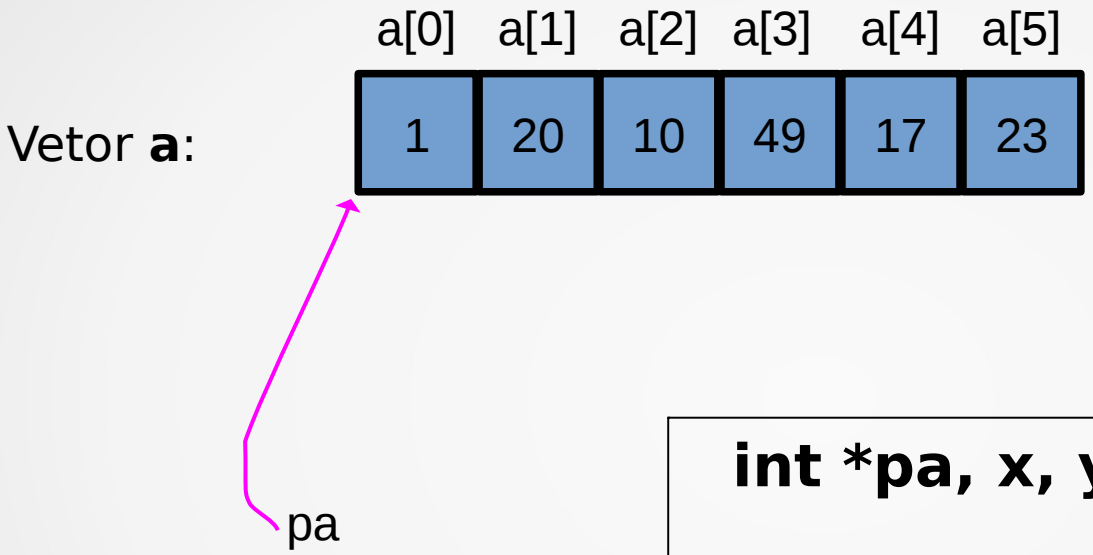
Uma função só pode devolver um valor para aquela que faz a chamada. No entanto, a função troca precisa devolver dois valores. Como podemos resolver isso?

Passagem de referência como parâmetro de uma função

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

```
#include <stdio.h>

void troca(float *, float *);

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);
}
```

Apontadores e vetores




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

pa = &a[0];

x = *pa;

y = (pa+1) == &pa[1];

z = *(pa+1) == a[1];
```

Apontadores e vetores

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

  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)
- 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.

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++;
```

Exemplos

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

pa = a;

a = pa; (erro!)

pa++;

a++; (erro!)
```

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

*pa += 2;

*pa++;

2º  1º
```

Vetor como parâmetro de função

```
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;
}
```