Funções

Parte 3

Passagem de parâmetros por referência

Sumário

- Conceito de passagem de parâmetros **por referência**
- Diferença da passagem de parâmetro **por valor**

Trocando valores ... (passagem **por valor)**

Fazer uma função chamada **troca()** que receba dois parâmetros inteiros **X** e **Y** e retorne as variáveis com os conteúdos trocados. Isto é, se **X=8** e **Y=5**, após a função, **X=5** e **Y=8**.

```
/* Programa 'trocaValores' */
#include <iostream>
#include <cmath>
using namespace std;

void troca ( int a, int b ) {
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
}
int main ( ) {
    int x, y ;
    x = 8;   y = 5;
    troca (x, y);
    cout << x << " " << y << endl;
}
```

Não ocorre a troca!!!

As variáveis são **locais** e **distintas.**

A troca realizada na **função** não se reflete nas variáveis usadas como argumentos da função em **main()**

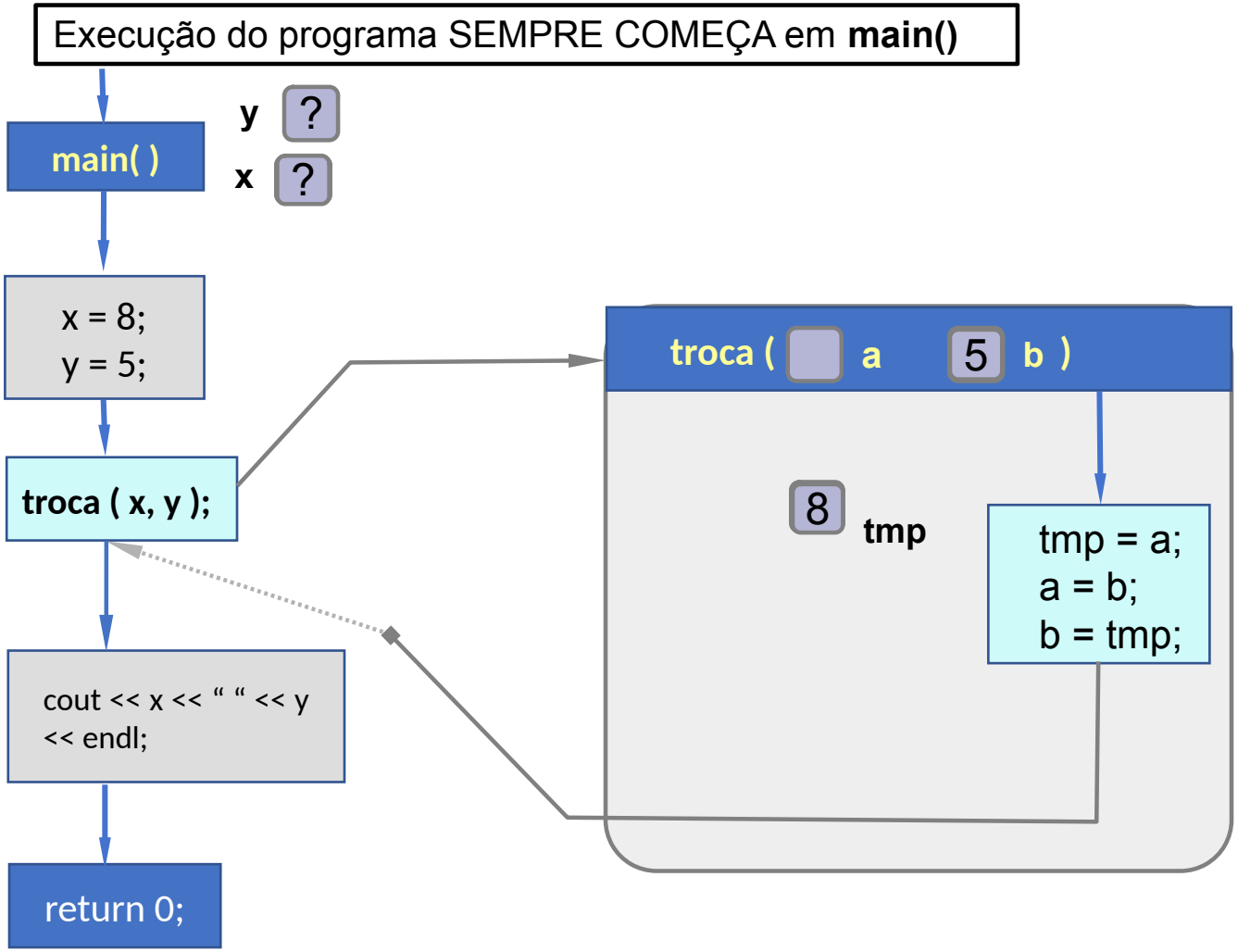
Trocando valores ... (passagem **por valor)**

```
/* Programa 'trocaValores' */
#include <iostream>
#include <cmath>
using namespace std;

void troca ( int a, int b ) {
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
}


int main ( ) {
    int x, y ;
    x = 8;   y = 5;
    troca (x, y);
    cout << x << " " << y << endl;
    return 0;
}
```



UFPR

Trocando valores ... (passagem **por referência)**

Fazer uma função chamada **troca()** que receba dois parâmetros inteiros **X** e **Y** e retorne as variáveis com os conteúdos trocados. Isto é, se **X=8** e **Y=5**, após a função, **X=5** e **Y=8**.

```
/* Programa 'trocaValores' */
#include <iostream>
#include <cmath>
using namespace std;

void troca ( int &a, int &b ) {
    int tmp;

    tmp = x;
    x = y;
    y = tmp;
}
int main ( ) {
    int x, y ;
    x = 8;   y = 5;
    troca (x, y);
    cout << x << " " << y << endl;
}
```

A **definição** da função deve indicar que os parâmetros **se referem** às variáveis usadas nos **argumentos** usados chamada da função

A **definição** da função deve indicar que os parâmetros são **passados por referência**

→ caracter **&** nos parâmetros de **definição** da função

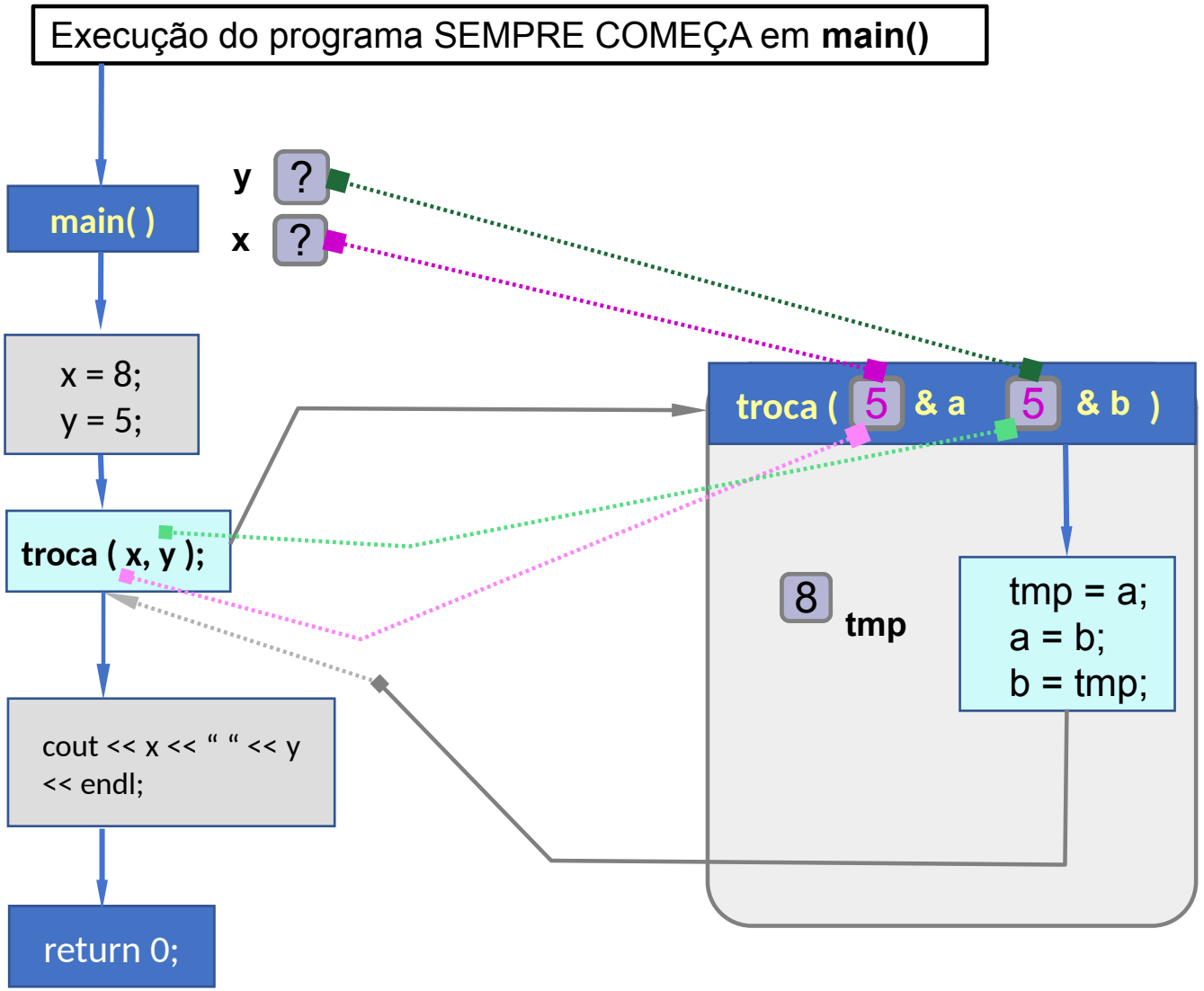
Trocando valores ... (passagem **por referência)**

```
/* Programa 'trocaValores' */
#include <iostream>
#include <cmath>
using namespace std;

void troca ( int & a, int & b ) {
    int tmp;

    tmp = a;
    a = b;
    b = tmp;
}


int main ( ) {
   int x, y ;
   x = 8;   y = 5;
   troca (x, y);
   cout << x << " " << y << endl;
   return 0;
}
```



Passagem de parâmetros **por referência**

- Ao chamar a função, esta recebe no parâmetro uma referência (link) direta ao argumento correspondente usado na chamada da função:
  - Não há cópia de valores de argumento para parâmetro, como na passagem **por valor**
  - Para todos os efeitos, ao alterar o valor de um parâmetro por referência, a função está alterando **diretamente** outra variável, que foi declarada em outro local.
- Na definição da função, em **`lista_parâmetros`**:
  - Parâmetro por referência tem **`&`**:

    ```
    tipo1 & param_1, …
    ```
  - Exemplo: `void troca (int & x, int & y) { … }`

Passagem de parâmetros **por referência**

- Na **chamada** da função, nada muda:
  - Em **`lista_argumentos`**, coloca-se a variável, como no caso de passagem por valor:

```
troca (x, y);
```

  - A relação de referência **NÃO SE DÁ PELO NOME**
    - se dá pelo uso de **&** na lista de parâmetros, na **definição** da função

- Quando um **parâmetro** é passado por referência, o **argumento** correspondente na chamada da função **DEVE SER** uma **variável**
  - **Não pode ser** uma expressão ou constante.

Passagem de parâmetros **por referência**

- Parâmetros passados por referência são úteis em situações onde as funções precisam produzir como resultado **dois ou mais** valores.
- Uma função com parâmetros passados por referência TAMBÉM pode devolver um valor via **return**:
  - Usado para indicar condição de erro na função
  - Usado para indicar que resultados nos **parâmetros passados por referência** contém resultados válidos
- Uma função pode ter parâmetros **por valor** e **por referência**

Como retornar dois ou mais valores?

Escrever a função **eq2grau()** que recebe os coeficientes reais **a**,**b**,**c** de uma equação de 2º grau e deve devolver como resultado as duas raízes reais da equação.
A função não deve mostrar mensagem alguma. Assuma que os coeficientes sempre levarão a raízes reais.

```
/* Função 'eq2grau' */
#include <iostream>
#include <cmath>
using namespace std;

float eq2grau ( float a, float b, float c )
{
    float x1, x2;


    x1 = (-b + sqrt(b*b – 4*a*c)) / (2*a);
    x2 = (-b - sqrt(b*b – 4*a*c)) / (2*a);

    return x1;
    return x2;

}
```

Errado!!!
Função termina no **primeiro** return, devolvendo **APENAS x1**

Como retornar dois ou mais valores?

Escrever a função **eq2grau()** que recebe os coeficientes reais **a**,**b**,**c** de uma equação de 2º grau e deve devolver como resultado as duas raízes reais da equação.
A função não deve mostrar mensagem alguma. Assuma que os coeficientes sempre levarão a raízes reais.

```
/* Função 'eq2grau' */
#include <iostream>
#include <cmath>
using namespace std;

void eq2grau ( float a, float b, float c,
               float & x1, float & x2 )
{
        x1 = (-b + sqrt(b*b – 4*a*c)) / (2*a);
        x2 = (-b - sqrt(b*b – 4*a*c)) / (2*a);
}


int main ( ) {
    float a, b, c, x, y;
    cin >> a >> b >> c;
    eq2grau ( a, b, c, x, y ) ;
    cout << x << " " << y << endl;
}
```

A **definição** da função deve indicar que há parâmetros que receberão os resultados e que serão **passados por referência**

→ caracter **&** nos parâmetros de **definição** da função
→ parâmetros **a**, **b** e **c** apenas recebem os coeficientes da equação
→ são passados por valor e não tem **&**

x e **y** receberão diretamente os valores das raízes da equação com coeficientes **a**, **b**, e **c**.

Funções que não retornam valores

- Uma função pode não devolver valor usando **return**
  - Tipo da função é **void**.
    - indica que função não retorna valor **usando return**
    - útil quando se quer uma função que apenas imprima valores
    - ocorre quando **todos** os resultados da função estão associados a parâmetros **passados por referência**
  - Como não tem valor de retorno, uma chamada de função **void** não pode ser usada em expressões ou atribuições.

```
void somaMenoresQt ( float & res , int qt ) {  // soma a res inteiros entre 0 e qt
      int i;
      i=0;
      while (i < qt){  res = res + i; I = i+1 }  // o resultado é devolvido pelo parâmetro res
      // não tem return → tipo de retorno é void
}
int main() {
     int v, x;
     float n;
     cin >> n >> v;
     somaMenoresQt (n, v);          // OK.
     x = somaMenoresQt (n, v);     // ERRO !!!
     cout << somaMenoresQt (n, v)  // ERRO !!!  → NÃO ESTÁ imprimindo o valor de n
     return 0;
}
```

Exercícios para aula *online*

Após assistir todas as vídeo-aulas da semana, procure trabalhar na **Lista de exercícios** do Tópico **Funções**, na sala virtual da disciplina na UFPR Virtual.

Estes exercícios serão usados nas aulas *online* para esclarecer e consolidar os conceitos abordados até aqui.

Leitura complementar

Acesse as **Leituras complementares** do Tópico **Funções**, na sala virtual da disciplina da UFPR Virtual.

Elas são importantes e auxiliam na compreensão dos temas abordados até aqui.

Créditos: O conteúdo original deste documento é de autoria do Prof. Armando Luiz N. Delgado, para uso na disciplina *Programação de Computadores* (CI208, CI180, CI183). Agradecemos à Profª Myriam Regattieri (UTFPR), ao Prof. Ricardo Lüders (UTFPR) e ao Prof. Luciano Silva (UFPR) pelas sugestões.