Funções III

INF110 – Programação I

Prof. André Gustavo
DPI/UFV – 2019/1

UFV

Nas aulas anteriores...

```
1  #include <iostream>
2  using namespace std;
3
4  double media (double a, double b) {
5      double resp = (a + b) / 2;
6      return resp;
7  }
8
9  int main() {
10     double x, y, m;
11
12     cin >> x >> y;
13     m = media(x,y);
14     cout << "Media: " << m << endl;
15     return 0;
16 }
```

a, b, resp ⇒ variáveis locais da função media

x, y, m ⇒ variáveis locais da função main

Relembrando...

- Variáveis declaradas dentro de uma função têm escopo local
  - Só podem ser acessadas dentro da função onde estão declaradas
  - E isto vale também para os parâmetros da função
- A comunicação de valores é feita na chamada e no retorno
  - Na chamada, valores podem ser passados da função que chama para a função chamada
  - No retorno, um valor pode ser passado da função chamada para a função que a chamou
  - **São passados valores, não variáveis!**
    Podem ser usadas expressões quaisquer, de tipo compatível

Relembrando...

- Apesar do programa da esquerda passar `x` e `y` e retornar `resp`, seus **valores** é que são passados, não as variáveis
  - Assim como no da direita, que não passa nem retorna variável

```cpp
#include <iostream>
using namespace std;

double media (double a, double b) {
    double resp = (a + b) / 2;
    return resp;
}

int main() {
    double x, y, m;

    cin >> x >> y;
    m = media(x,y);
    cout << "Media: " << m << endl;
    return 0;
}
```

```cpp
#include <iostream>
using namespace std;

double media (double a, double b) {
    return (a + b) / 2;
}

int main() {
    double m;

    m = media(4.5,3.1);
    cout << "Media: " << m << endl;
    return 0;
}
```

Passagem por referência

E se quisermos passar a variável?

- Como fazer a função `troca` no seguinte programa?

```
int main() {
    int a, b;

    cin >> a >> b;          //Le dois valores
    if (a>b)                //Se o primeiro for maior
        troca(a,b);         //  troca os valores
    cout << "Em ordem: " << a << " " << b << endl;
    return 0;
}
```

E se quisermos passar a variável?

```
1  #include <iostream>
2  using namespace std;
3
4  void troca(int x, int y) {  //NAO FUNCIONA!!!
5      int temp = x;
6      x = y;
7      y = temp;
8      return;
9  }
10
11 int main() {
12     int a, b;
13
14     cin >> a >> b;          //Le dois valores
15     if (a>b)                //Se o primeiro for maior
16         troca(a,b);         //  troca os valores
17     cout << "Em ordem: " << a << " " << b << endl;
18     return 0;
19 }
```

Troca localmente os valores de x e y

Mas isso não troca a e b, que era o desejado

E se quisermos passar a variável?

```
1  #include <iostream>
2  using namespace std;
3
4  void troca(int &x, int &y) {
5      int temp = x;
6      x = y;
7      y = temp;
8      return;
9  }
10
11 int main() {
12     int a, b;
13
14     cin >> a >> b;          //Le dois valores
15     if (a>b)                //Se o primeiro for maior
16         troca(a,b);         //  troca os valores
17     cout << "Em ordem: " << a << " " << b << endl;
18     return 0;
19 }
```

Usando & a passagem é por referência, `x` e `y` aqui são `a` e `b` da `main`

Ao trocar `x` e `y` de fato são trocados `a` e `b`

Passagem por referência

- Usando & antes do nome do parâmetro a passagem será por **referência**
  - Não é passado o valor da variável, mas uma referência a ela
  - Desta forma, a função pode acessá-la e modificá-la diretamente

- Observações
  - O & cria um *alias* (um "apelido")
  - O acesso é feito pelo nome do parâmetro, não da variável original
  - A variável continua sendo local, seu nome é usado só em sua função
  - Mas pode ser acessada via referência pelo nome do parâmetro

Exemplo #1

- Função que recebe referências a duas variáveis int e coloca seus valores em ordem crescente
  - `void ordena (int &, int &);`

```cpp
#include <iostream>
using namespace std;

//Coloca os valores em ordem, por referência
void ordena(int &x, int &y) {
    if(x>y) {
        int temp = x;
        x = y;
        y = temp;
    }
    return;
}

int main() {
    int a, b;

    cin >> a >> b;          //Le dois valores
    ordena(a,b);            //Ordena os valores
    cout << "Em ordem: " << a << " " << b << endl;
    return 0;
}
```

Exemplo #2

- Função que recebe referência a uma variável int e atribui à variável um valor lido na entrada
  - `void le (int &);`

```cpp
#include <iostream>
using namespace std;

//Le e grava um valor inteiro em x
void le(int &x) {
    cout << "Digite um numero inteiro: ";
    cin >> x;
    return;
}


int main() {
    int a,b,c;

    //Le 3 valores
    le(a);
    le(b);
    le(c);

    //Imprime a soma dos valores
    cout << "Total: " << a+b+c << endl;

    return 0;
}
```

Exemplo #2b

- Como no anterior, lê um inteiro e o retorna por referência, mas assegurando que esteja no intervalo [int,sup], sendo estes limites passados por valor
  - `void le (int &, int inf, int sup);`

```cpp
#include <iostream>
using namespace std;

//Le e retorna por referencia um valor inteiro no intervalo [inf,sup]
void le(int &x, int inf, int sup) {
    cout << "Digite um numero entre " << inf << " e " << sup << ": ";
    cin >> x;
    while(x<inf || x>sup) {
        cout << "Valor invalido!\n";
        cout << "Digite um numero entre " << inf << " e " << sup << ": ";
        cin >> x;
    }
    return;
}


int main() {
    int a,b,c;

    //Le 3 valores
    le(a,1,10);     //valor entre 1 e 10
    le(b,1,100);    //valor entre 1 e 100
    le(c,0,8);      //valor entre 0 e 8

    //Imprime a soma dos valores
    cout << "Total: " << a+b+c << endl;

    return 0;
}
```

Exemplo #3

- Função que recebe um horário (horas e minutos) e retorna o valor convertido para minutos
  - Ex.: 1 hora e 40 minutos ⇒ 100 minutos
- Função que faz o contrário: recebe um valor em minutos e o converte para horas e minutos
  - Ex.: 100 minutos ⇒ 1 hora e 40 minutos

```
//Converte h horas e m minutos para minutos
int conv_minutos(int h, int m) {
    return 60*h+m;
}

//Converte min minutos para h horas e m minutos
void conv_horaminutos(int min, int &h, int &m) {
    h = min/60;
    m = min%60;
    return;
}
```

- Note que a segunda função foi feita com passagem por referência, pois precisava “retornar” dois valores

```
int main() {
    int inicioh, iniciom;
    int fimh, fimm;
    int durh, durm;
    char c;

    //Le horario inicial e final de um evento
    cout << "Horario inicial (ex: 1:40): ";
    cin >> inicioh >> c >> iniciom;
    cout << "Horario final (ex: 2:15): ";
    cin >> fimh >> c >> fimm;

    //Converte horarios para minutos
    int inicio = conv_minutos(inicioh,iniciom);
    int fim = conv_minutos(fimh,fimm);

    //Converte a duracao do evento em horas e minutos
    conv_horaminutos(fim-inicio,durh,durm);
    cout << "Duracao: " << durh << ":" <<durm << endl;

    return 0;
}
```

- Exemplo de programa que usa as funções de conversão

Exemplo #4

- Função que recebe por valor os coeficientes A, B, C de uma equação do 2º grau e retorna por referência suas raízes

```
#include <iostream>
#include <cmath>
using namespace std;

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

int main() {
    double a, b, c, x1, x2;
    cin >> a >> b >> c;
    eq2grau(a,b,c,x1,x2);
    cout << x1 << " " << x2 << endl;
    return 0;
}
```

- A função foi feita para ilustrar passagem por referência; uma função completa deveria testar (e retornar) se há raiz real

Erros comuns

- Não passar variável

```
#include <iostream>
using namespace std;

void troca(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
    return;
}

int main() {
    int a, b, c;
    ...
    troca(a,5);   //y nao pode referenciar 5
    troca(a+b,c); //x nao pode referenciar a+b
    ...
}
```

Erros comuns

- Passar variável do tipo errado

```
#include <iostream>
using namespace std;

void troca(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
    return;
}

int main() {
    double a, b;
    ...
    troca(a,b);  //int & nao pode referenciar double
    ...
}
```

Cuidado!

- Usando & está dando acesso à variável
- Perigo de efeitos colaterais

```
//Retorna h horas m minutos em minutos
int conv_minutos(int &h, int &m) {
    h = h*60;
    return h+m;
}
```

  - Na função acima, `h` e `m` são parâmetros por referência (sem necessidade, já que é necessário passar apenas os valores)
  - A função altera o valor de `h` para facilitar o cálculo
  - Isso altera a variável referenciada na função que chamou!!
  - Sem o & seria passagem por valor, poderia alterar `h` à vontade