Funções

Parte 2

Passagem de parâmetros por valor

Sumário

- Conceito de passagem de parâmetros **por valor**

Parâmetros de funções

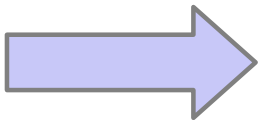
- De modo geral, funções precisam de valores para efetuar alguma tarefa:
  - `x = pow (b,e);`
  - `y = sqrt (n);`

- As variáveis dentro dos parênteses (**argumentos**) são valores que a função precisa para executar seus comandos e retornar um resultado.

- Estes valores que a função espera receber ao ser chamada são os **parâmetros de entrada** da função
  - especificados no **cabeçalho** de **definição** da função

Parâmetros de funções

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

int lerCodigo ( ) {
        int code;

        cin >> code;
        while ( code <= 0 )
            cin >> code;
        return code;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {
      op = lerCodigo ( );

      if ( op == 1 )          x = sqrt (n);
      else if ( op == 2 )   x = cbrt (n);
      else if ( op == 3 )   x = exp (n);

      cout << x << endl;
      cin >> n;
    }
    return 0;
}
```



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

int lerCodigo ( ) {
        int code;
        cin >> code;
        while ( code <= 0 )
            cin >> code;
        return code;
}

float calcOp ( float v, int op ) {
      float res;

      if ( op == 1 )          res = sqrt ( v );
      else if ( op == 2 )   res = cbrt ( v );
      else if ( op == 3 )   res = exp ( v );
      return res;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {
      op = lerCodigo ( );

      x = calcOp ( n, op );

      cout << x << endl;
      cin >> n;
    }
    return 0;
}
```

Definição de função

- Formato geral:

```
tipo_retorno nomeFuncao ( lista_parâmetros )
{
    declarações de variáveis;
    comandos;
    return valor_retorno;
}
```

  - **tipo_retorno:** indica o tipo de **valor_retorno**
    - resultado que a função deve retornar ao final de sua execução
  - **lista_parâmetros:** lista de **declarações de variáveis** usadas para passagem de valores para a função.
    - variáveis separadas por **vírgula**, cada variável com seu **tipo**.

      tipo1 param_1 , tipo2 param_2 , ...

    - Em funções simples que não recebem parâmetros, esta lista é **vazia**

Definição de função

- Exemplo:

```
float calcOp ( float n, int op )
```

  - ao ser chamada, esta função espera receber 2 valores nas duas variáveis:

    `n` → um valor real
    `op` → um valor inteiro

  - estas variáveis serão usadas dentro do bloco de comandos da função

- Os parâmetros são **variáveis locais** da função

  - Recebem um valor inicial quando a função é **chamada** durante a execução do programa

Protótipos de funções com parâmetros

- O **protótipo** consiste em escrever o mesmo cabeçalho de definição da função seguido apenas de **ponto-e-vírgula**.

- O importante é que ou a **definição** ou **protótipo** de uma função estejam escritas no código-fonte ANTES da PRIMEIRA VEZ que ela é **referenciada** no código-fonte de todo o programa.

- No **protótipo**, pode-se apenas colocar os tipos dos parâmetros, sem colocar os nomes dos parâmetros:

```
float calcOp ( float n, int op );

OU

float calcOp ( float, int );
```

Definição de função

- Os parâmetros são **variáveis locais** da função
  - **Não devem** ser re-declaradas dentro do bloco da função

```
int somaValor ( int qt )
{
        int  res, i, qt ;   // ERRO !!! Não se re-declara parâmetro de função

        i=0;
        res = 0;
        while (i < qt) {
           res = res + i;
           i = i+1;
        }

        return res;
}
```

Chamada de uma função

- Como o programa usa ou executa uma função?
  - Através da **chamada:**

```
nomeFuncao ( lista_argumentos )
```

    - **lista_argumentos:** lista de variáveis / valores / expressões cujo **valor** será atribuído aos parâmetros da função
      - **argumentos** são separados por **vírgula**

arg_1, arg_2, ...

Chamada de uma função

- **Não se colocam os tipos em argumentos na chamada**
  - **Apenas** o nome da função chamada, seguida da lista de **argumentos** entre parênteses, **sem especificar tipos**

```
int somaValor ( int qt ) {
        int  res, i;
        I=0;
        res = 0;
        while (i < qt) {
            res = res + i;   i=i+1;
        }
        return res;
}

int main() {
      int v, x, n;
      …..
      n = somaValor ( v );          // Forma de chamada da função → OK.
      …….
      x = int somaValor ( int v);   // ERRO !!! Não se colocam tipos na chamada
      ……
      return 0;
}
```

Chamada de funções

- Ao **chamar** uma função, cada **argumento** deve ser uma variável/valor/expressão do **mesmo tipo** do respectivo **parâmetro** na **definição** da função.

- Associação argumento-parâmetro é feita pela **posição**:
  - **Valor** do 1º argumento é copiado para o 1º parâmetro,
  - **Valor** do 2º argumento é copiado para o 2º parâmetro,
  - e assim por diante

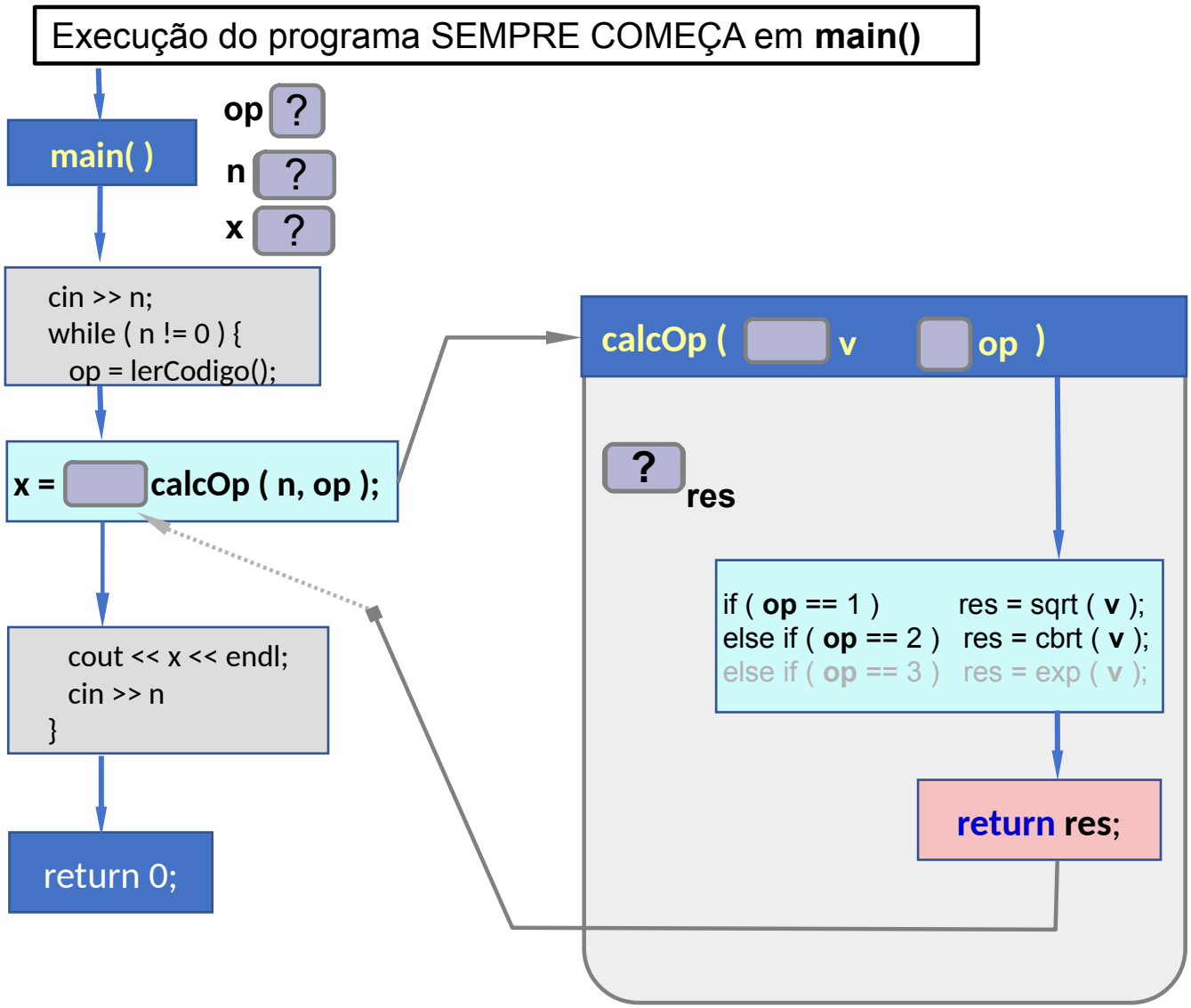
```
float calcOp ( float n, int op ) { … }

int main () {
    float x, n;
    int opt;
    x = calcOp ( n, opt );
    x = calcOp ( n + x, 2 );
    x = calcOp ( 1.73, 3 );
    …
}
```

Erros comuns em chamada de funções

```
int somaValor ( int qt, int p ) {
        int  res, i;
        i=0;
        res = p;
        while (i < qt) {
            res = res + i;   i=i+1;
        }
        return res;
}

int main() {
      int quant, x, n;
      float y;
      …..
      x = somaValor ( quant, n );        // Forma de chamada da função → OK.
      …….
      x = somaValor ( y , n );           // ERRO !!! Tipo do 1º argumento (float)
      ……                                 // não é o mesmo do 1º parâmetro da função (int)
      x = somaValor ( n  ,  quant );       // ATENÇÃO !!! Os dois argumentos são do tipo certo
      ……                                 // mas a ordem pode estar errada.
      return 0;                          // O programa terá erro ao executar !!
}
```

Mecanismo da passagem de parâmetros **por valor**

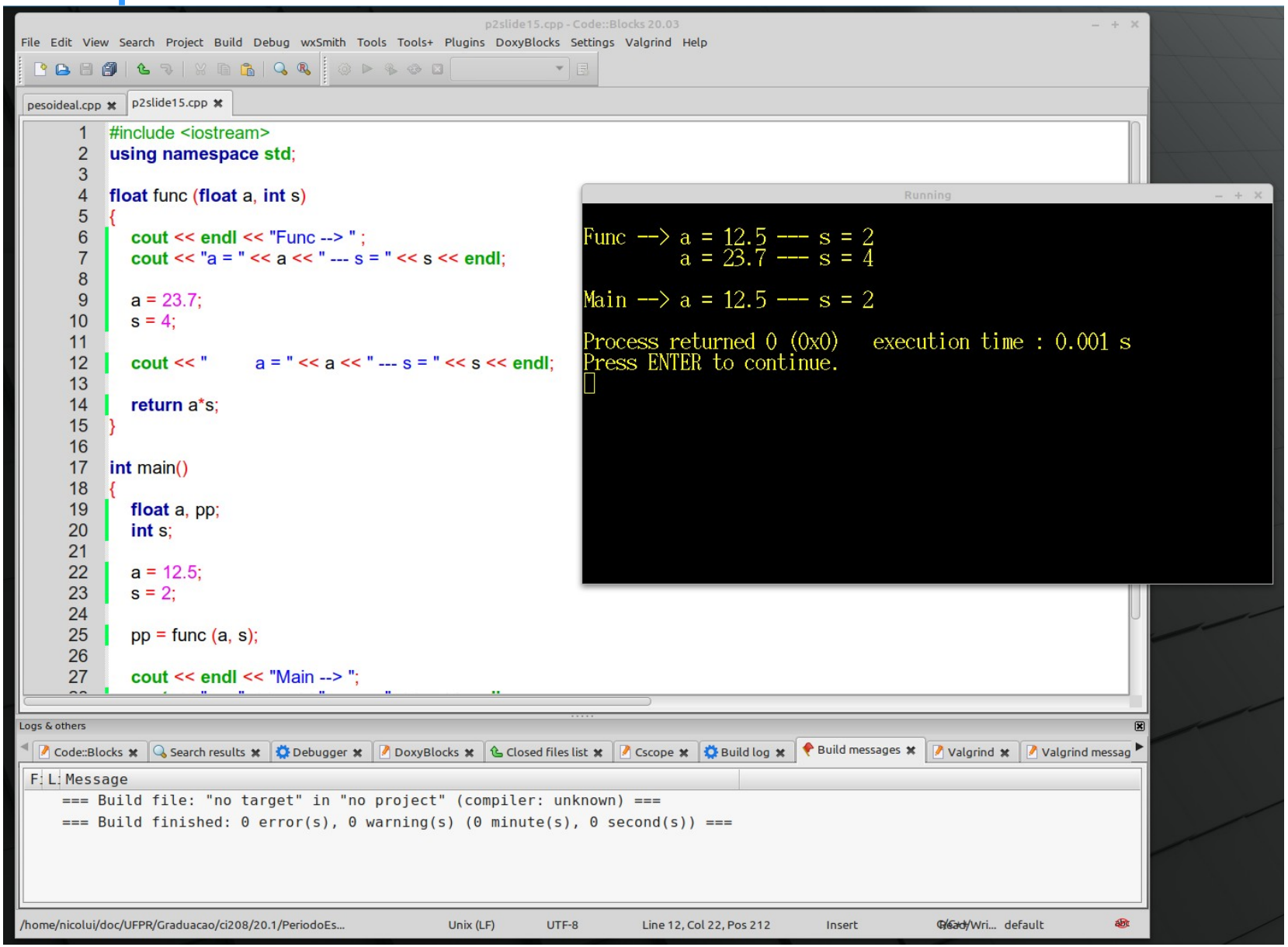
```
/* Programa 'menuOpcoes' */
#include <iostream>
#include <cmath>
using namespace std;

int lerCodigo ( ) { … }

float calcOp ( float v, int op ) {
      float res;

      if ( op == 1 )        res = sqrt ( v );
      else if ( op == 2 )   res = cbrt ( v );
      else if ( op == 3 )   res = exp ( v );

      return res;
}

int main( ) {
    int op;  float n, x;

    cin >> n;
    while ( n != 0 ) {
      op = lerCodigo ( );

      x = calcOp ( n, op );

      cout << x << endl;
      cin >> n;
    }
    return 0;
}
```



Passagem de parâmetros **por valor**

- Ao **chamar** uma função, apenas o **valor** dos argumentos é passado (atribuído) aos respectivos **parâmetros** da função
  - Dizemos que a passagem de parâmetros é **por valor**
  - O parâmetro neste caso é dito ser um **parâmetro de entrada**
- Os **parâmetros** podem ter o mesmo nome de variáveis usadas como **argumentos** na chamada da função
  - **LEMBRE-SE: não são a mesma variável**
  - Estão em locais diferentes na memória

Exemplo



Sobre bom uso de funções

- É importante que cada função tenha uma especificação bem definida:
  - Comentário descritivo antes da **DEFINIÇÃO** da função:
    - Objetivo da função
    - Descrição dos algoritmos usados na função
    - Descrição dos **parâmetros** da função
    - O que será **devolvido** pela função

- ATENÇÃO: Uma função sempre implementa uma PARTE do programa.
  - É em **main()** onde normalmente se define a ordem de chamada das funções, definindo assim a sequência de execução do programa como um todo
  - **Não tente colocar todo o programa em uma função.**

Exemplo

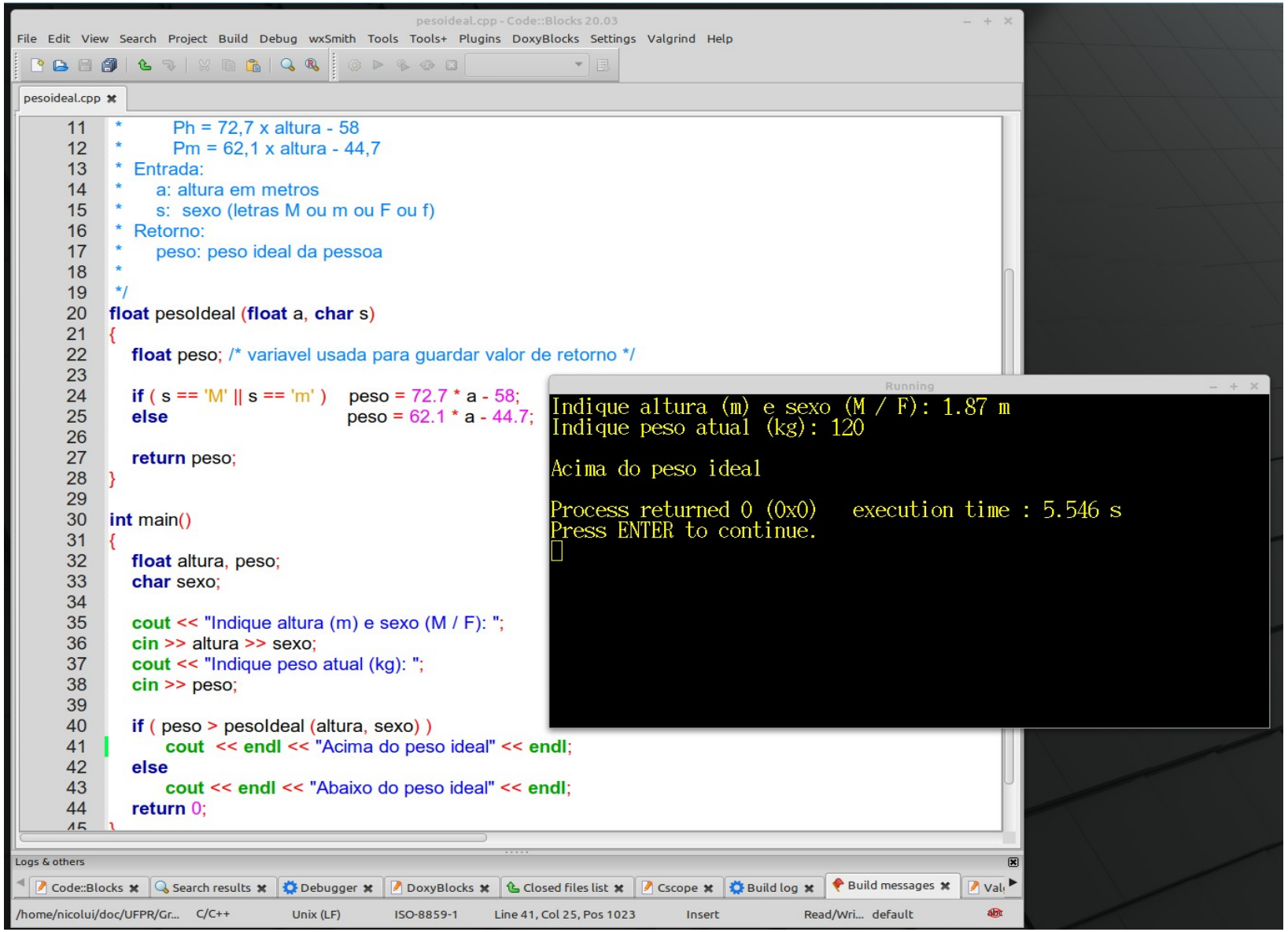
Fazer uma função chamada **pesoldeal()** que recebe como parâmetros a altura e o sexo de uma pessoa (**M** masculino e **F** feminino) e retorna o seu peso ideal, utilizando as seguintes fórmulas:

- para sexo masculino: (72.7 * altura) – 58
- para sexo feminino: (62.1 * altura) – 44.7

Faça também o programa principal que recebe do usuário a sua altura, peso atual e sexo, e imprima na tela mensagem dizendo apenas se o usuário está acima ou abaixo de seu peso ideal, este calculado pela função **pesoldeal()**.

OBS.: Em **enunciados de exercícios**, quando se coloca um identificador seguido de **()** é apenas uma forma de indicar que o nome é uma função. O enunciado dirá se a função deve ou não ter parâmetros

Exemplo



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.