Computação e Linguagem de Programação

Aula 5 parte 4

Professores
Sandro Fonseca de Souza
Dilson de Jesus Damião

Aula Anterior

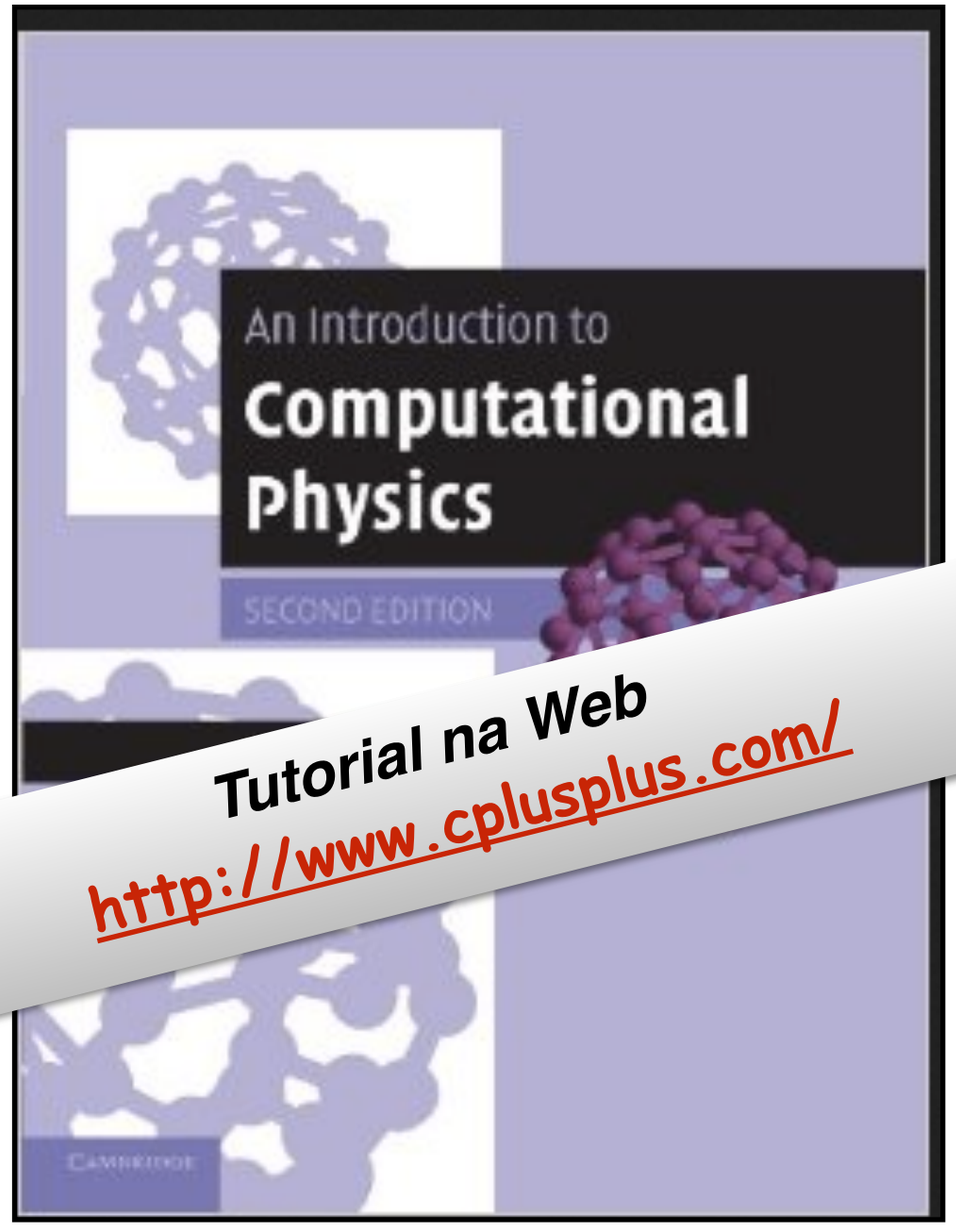
- **Operadores relacionais e lógicos**
- **Expressões boleadas**
- **Estrutura if**
- **Estrutura if .. else**
- **Estrutura if .. else if .. else**
- **Loop while**
- **Loop do.....while**
- **Loop for**
- **break e continue**
- **loops infinitos**
- **loops aninhados**
- **problemas resolvidos**

Sumário

- Funções
- Função void
- Polimorfismo

Bibliografia Sugerida

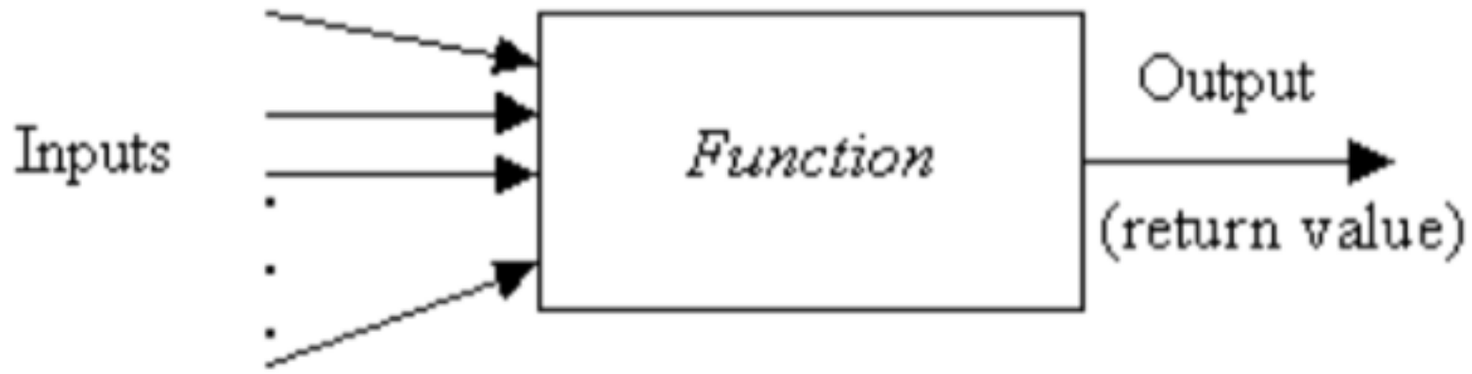
Bibliotecas padrão do C++
http://www.cplusplus.com/reference/

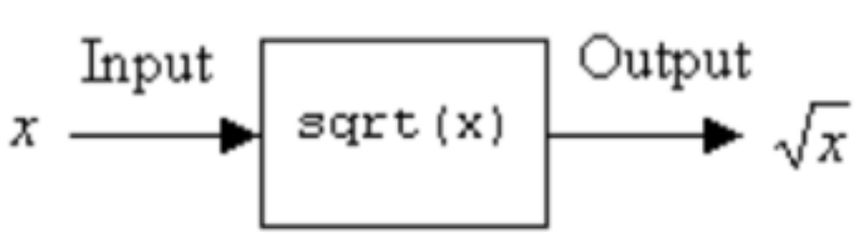


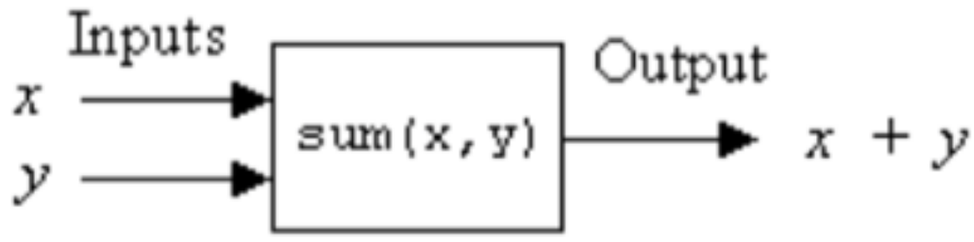
Tutorial na Web
http://www.cplusplus.com/

Funções(subprogramas)

- A function is an object that accepts some inputs and then outputs a result depending on the inputs.







Definindo e usando funções

- General form of a function declearation

```
type name(parameter 1, parameter2, ...){
   ...
      statements
   ...
}
```

- Example function to compute the area of a triangle

```
double area(double base, double height)
{
    double ar;
    ar = base * height / 2.0;
    return ar;
}
```

Definindo e usando funções

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

double area(double base, double height)
{
    double ar;
    ar = base * height / 2.0;
    return ar;
}

int main()
{
  double a, b = 3.0, h = 4.0;

  a = area(b, h);
  cout << "area = " << a << endl;

  return 0;
}
```

```
area = 6
```

Definindo e usando funções

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

// function prototype (fonksiyon örneği veya kalıbı)
double area(double, double);

int main(){
  double a , b = 3.0 ,h = 4.0;

  a = area(b, h); // function call
  cout << "area = " << a << endl;

  return 0;
}

// Function decleration
double area(double base, double height){
   double ar;
   ar = base * height / 2.0;
   return ar;
}
```

```
area = 6
```

Definindo e usando funções

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

// function prototype (fonksiyon örneği veya kalıbı)
double area(double, double);

int main(){
  double a , b = 3.0 ,h = 4.0;

  a = area(b, h); // function call
  cout << "area = " << a << endl;

  return 0;
}


// Function decleration
double area(double base, double height){
   return base * height / 2.0;
}
```

```
area = 6
```

Função void

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

// no value is returned
void printDouble(int a)
{
  cout << "Double of a:" << 2*a;
}

int main()
{
  printDouble(5);
}
```

```
Double of a: 10
```

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

// no value is returned
void Message(void)
{
 cout << "I am a function";
}

int main()
{
  Message();
}
```

```
I am a function
```

Passando argumentos através & ou ref

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

// arg. Pass by value
void Decrease(int a, int b){
    a--;
    b--;
}

int main()
{
  int x = 3, y = 8;

  cout << x << "   " << y << endl;

  Decrease(x, y);

  cout << x << "   " << y << endl;
}
```

```
3    8
3    8
```

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

// arg. Pass by reference
void Decrease(int& a, int& b){
    a--;
    b--;
}

int main()
{
  int x=3, y=8;

  cout << x << "   " << y << endl;

  Decrease(x,y);

  cout << x << "   " << y << endl;
}
```

```
3    8
2    7
```

Passando argumentos através & ou ref

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

void Convert(float, int& ,float&);

int main()
{
  float rx, x = 3.2;
  int   ix;

  Convert(x, ix, rx);

  cout << " x = " << x  << endl;
  cout << " ix= " << ix << endl;
  cout << " rx= " << rx << endl;
}

void Convert(float num, int& ip, float& rp)
{
  ip = num;
  rp = num - int(num);

}
```

Argumentos Padrão

C++ allows a function to have a variable number of arguments.
Consider the second order polynomial function: $a + bx + cx^2$

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

// -- optional parameters must all be listed last --
double p(double x, double a, double b =0, double c =0)
{
  return a + b*x + c*x*x;
}


int main(){
  double x = 1.0;

  cout <<  p(x, 7)        << endl;
  cout <<  p(x, 7, 6)     << endl;
  cout <<  p(x, 7, 6, 3) << endl;

  return 0;
}
```

```
7

13

16
```

Sobrecarga de Funções (polimorfismo)

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

int topla(int x, int y){
   return x + y;
}

int topla(int x, int y, int z){
  return  x + y + z;
}

double topla(double x, double y){
 return x + y;
}

int main(){
  cout <<"topla(9,7)     = " << topla(9, 7)      << endl;
  cout <<"topla(3,6,2)   = " << topla(3, 6, 2)   << endl;
  cout <<"topla(3.1,4.7)= " << topla(3.1, 4.7) << endl;
  return 0;
}
```

```
topla(9,7)     =  16
topla(3,6,2)   =  11
topla(3.1,4.7)=  7.8
```

Funções Macro

A macro function, which is actually not a function, is defined by using `#define` preprocessor command.

The following macro function definitions are valid in C/C++:

```
#define square(x)  (x)*(x)
#define hypotenus(x,y)  sqrt((x)*(x)+(y)*(y))
#define delta(a,b,c)  ((b)*(b)-4*(a)*(c))
#define max(a,b)  ((a>b)  ?  a:b)
```

Funções Macro

```
// Finding Pythagorian Triples
#include <iostream>
#include <cmath>
using namespace std;

// macro function definition
#define hypotenus(x,y) sqrt((x)*(x) + (y)*(y))
// symbolic constant definition
#define N 50

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

  cout << "Pythagorean Triples less than N = 50" << endl;

  for (a=1; a<=N; a++)
  for (b=a; b<=N; b++)
  for (c=1; c<=N; c++)
      if( c == hypotenus(a,b) )
          cout << "("
               << a << ", " << b << ", " << c
               << ")" << endl;
  return 0;
}
```

Funções Macro

```
Pythagorean Triples less than N = 50
(3, 4, 5)
(5, 12, 13)
(6, 8, 10)
(7, 24, 25)
(8, 15, 17)
(9, 12, 15)
(9, 40, 41)
(10, 24, 26)
(12, 16, 20)
(12, 35, 37)
(14, 48, 50)
(15, 20, 25)
(15, 36, 39)
(16, 30, 34)
(18, 24, 30)
(20, 21, 29)
(21, 28, 35)
(24, 32, 40)
(27, 36, 45)
(30, 40, 50)
```

Próxima Aula

- Arrays
- Ponteiros e Referencias
- Arrays e Funções
- Gerenciamento de memória dinâmica
- Vetores em C++