Lógica de Programação para Engenharia

Modularização e Organização de Código

Prof. Dr. Daniel Caetano

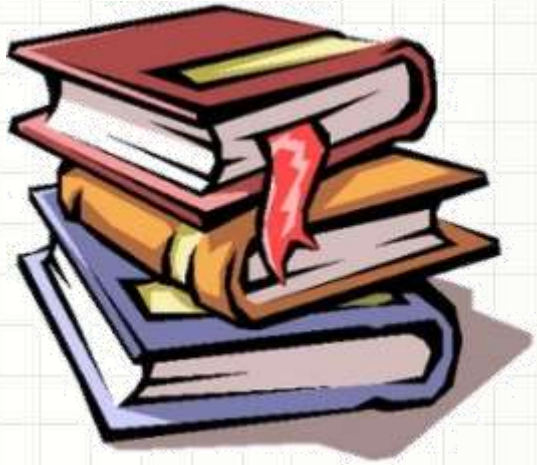
2018 - 1

Objetivos

- Entender a utilidade das funções
- Compreender o escopo das variáveis
- Capacitar o aluno para criar suas próprias funções
- Compreender a função main

- **Estudar para Prova!**

Material de Estudo



Material	Acesso ao Material
Notas de Aula e Apresentação	http://www.caetano.eng.br/ (Lógica de Programação para Eng. – Aula 7)
Material Didático	Lógica de Programação, págs 173 a 187.
Aula Online	Aula 4
Biblioteca Virtual	“Lógica de Programação – Fundamentos da Programação de Computadores”, págs 7 a 47.

Funções Simples

Funções Simples

- Situação: precisamos imprimir 8x o seguinte texto:

  Sistema de Impressão v.1.0, (c) Daniel Caetano

- Um jeito de fazer seria usar vários “cout”:

```
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
```

Funções Simples

- Será que não tem um jeito mais simples?
- Não seria legal dar um nome **imprime**...
  - E quando quiser imprimir só escrever **imprime**?
- Isso é possível!

```
void imprime(void) {
   cout << “Sistema de Impressão v.1.0, (c) Daniel Caetano”;
   }
```

Funções Simples

- Agora posso imprimir 8x aquele texto assim:

```
imprime();
imprime();
imprime();
imprime();
imprime();
imprime();
imprime();
imprime();
```

- Bem mais fácil, não?
- Como fica o programa inteiro?

Arq.: imprime.cpp

```
#include <iostream>
using namespace std;
void imprime(void) {
   cout << "Sistema de Impressão v.1.0, (c) Daniel Caetano";
}

main()  {
   imprime();              /* Repita quanto quiser! */
   imprime();
   imprime();
}
```

Arq.: imprime.cpp

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

void imprime(void) {
   cout << "Sistema de Impressão v.1.0, (c) Daniel Caetano";
}

main()  {
   imprime();              /* Repita quanto quiser! */
   imprime();
   imprime();
}
```

FUNÇÃO

Arq.: imprime.cpp

```
#include <iostream>
using namespace std;
void imprime(void) {
    cout << "Sistema de In                    tano";
}

main()  {
    imprime();          /* Repita quanto quiser! */
    imprime();
    imprime();
}
```

Declaração da Função: define o **nome** da função

Arq.: imprime.cpp

```
#include <iostream>
using namespace std;
void imprime(void) {
   cout << "Sistema de Impressão v.1.0, (c) Daniel Caetano";
}

main()  {
   imprime();
   imprime();
   imprime();
}
```

Para executar a função, indicamos o **nome** seguido de parênteses

Arq.: imprime.cpp

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

void imprime(void) {
   cout << "Sistema de Impressão v.1.0, (c) Daniel Caetano";
}

main()  {
   imprime();          /* Repita
   imprime();
   imprime();
}
```

Uma função precisa ser declarada antes de ser usada

Outro Exemplo de Função

- Queremos uma função que imprima a assinatura de nosso e-mail:

  Atenciosamente,

  Prof. Daniel Caetano

  prof@caetano.eng.br

```
cout << “Atenciosamente” << endl;
cout << “Prof. Daniel Caetano” << endl;
cout << “prof@caetano.eng.br” << endl;
```

  - Mas toda função precisa de um nome...!

Outro Exemplo de Função

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

assina()
{
    cout << “Atenciosamente,” << endl;
    cout << “Prof. Daniel Caetano” << endl;
    cout << “prof@caetano.eng.br” << endl;
}

main()
{
    assina();
}
```

Erro?

Outro Exemplo de Função

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

void assina()
{
    cout << "Atenciosamente," << endl;
    cout << "Prof. Daniel Caetano" << endl;
    cout << "prof@caetano.eng.br" << endl;
}

main()
{
    assina();
}
```

void : indica função sem retorno!

main não precisa ter retorno?!?

Funções com Retorno

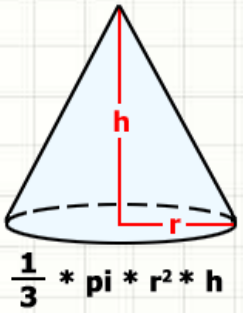
Funções com Retorno

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

float pi()
{
    return 3.14159;
}

main()
{
    cout << pi();
}
```

E se mudar o **float** por **int**?

Funções com Parâmetros

Funções com Parâmetros



- Calcular o volume de um cone

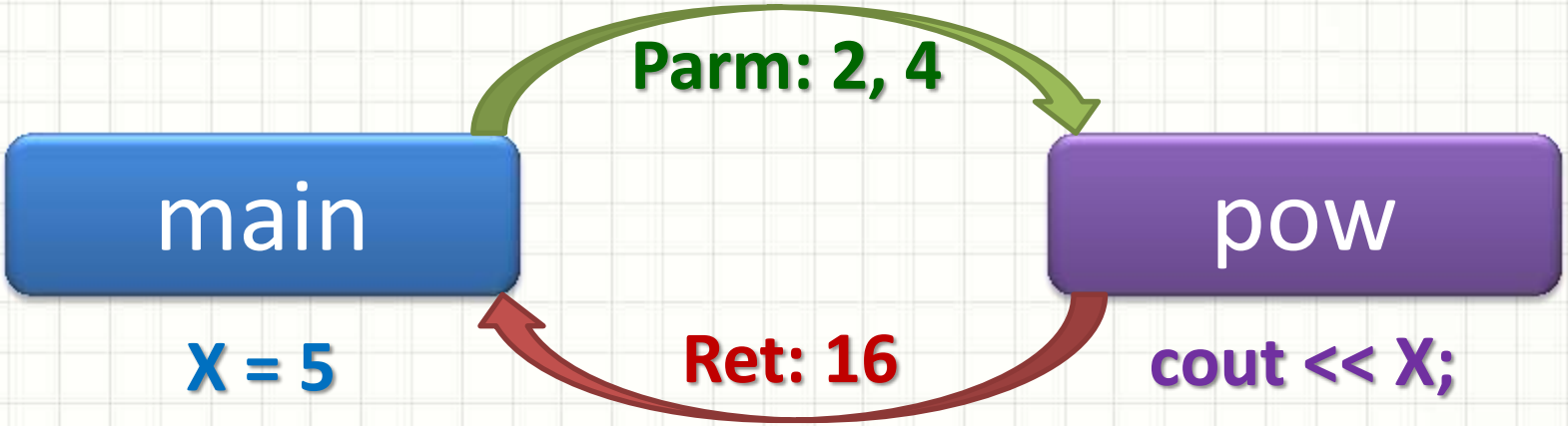
```
#include <iostream>
using namespace std;

float volumeCone(float r, float h)
{
    return (1.0/3.0)*3.14*r*r*h;
}

main()
{
    cout << volumeCone(10,2);
}
```

O que ocorre se mudarmos esses valores?

Escopo de Variáveis

Escopo de Variáveis

- Escopo = Vale Onde?
- Variáveis das funções: **locais**
  - **Só valem dentro da própria função**
- Passar dados de uma função para outra?
  - Na chamada: parâmetros
  - Receber respostas: return



Escopo de Variáveis

- Variáveis das funções: **locais**
  - Valem apenas dentro da função

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

float volumeCone(float r, float h) {
    float x;
    x = (1.0/3.0)*3.14*r*r*h;
    return x;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

ERRO!

Escopo de Variáveis

- Variáveis **globais**
  - Valem no programa todo

Cuidado!

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

void volumeCone(float r, float h) {
    x = (1.0/3.0)*3.14*r*r*h;
}
main() {
    volumeCone(10,2);
    cout << x;
}
```

Entendendo melhor:

Funções com Parâmetros e Retorno de Valores

Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```

Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```

Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```

Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```

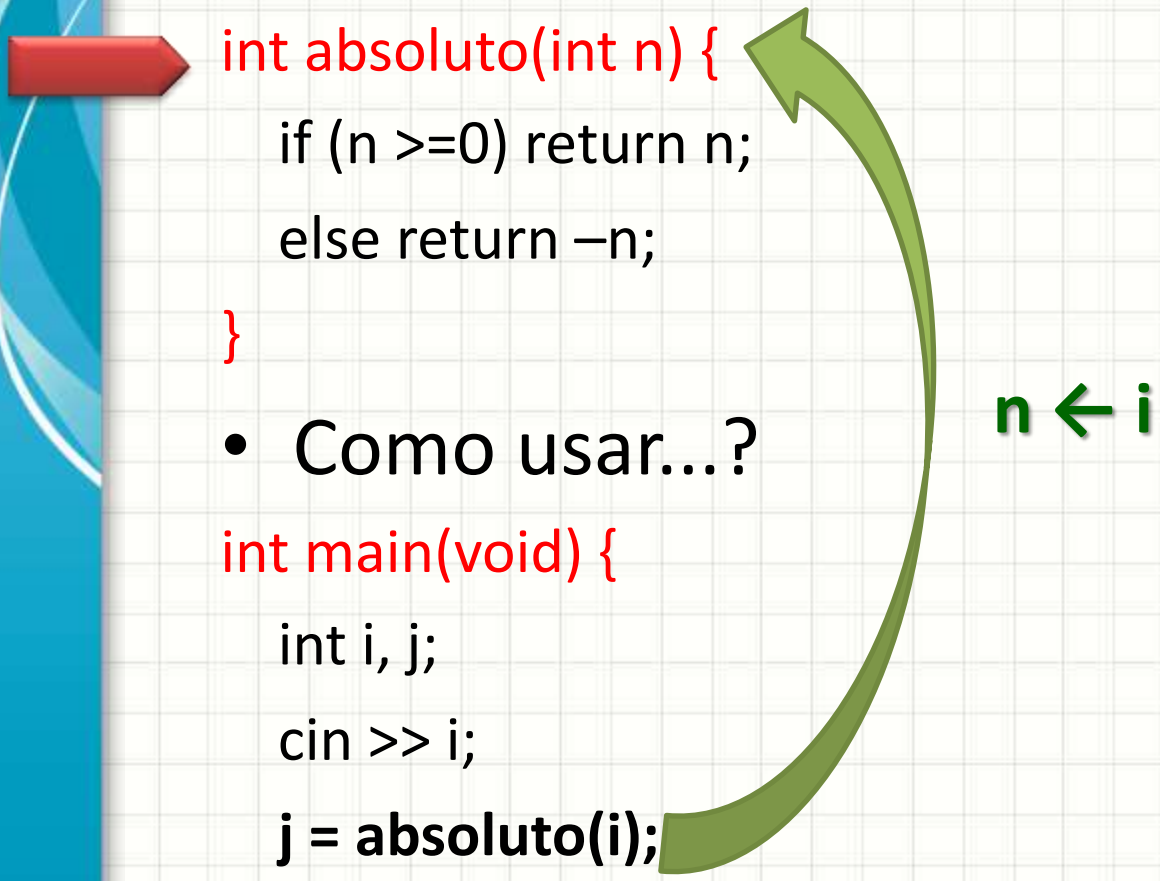
Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```



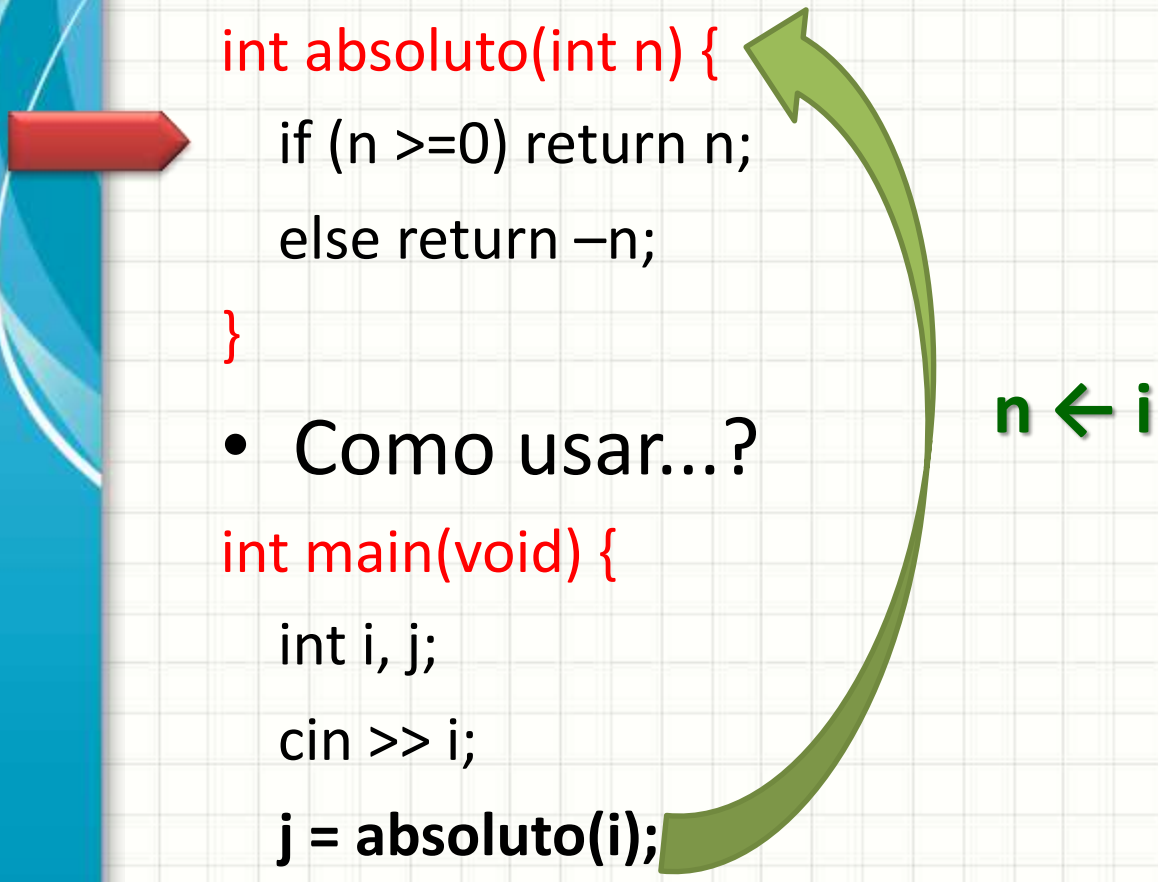
Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```



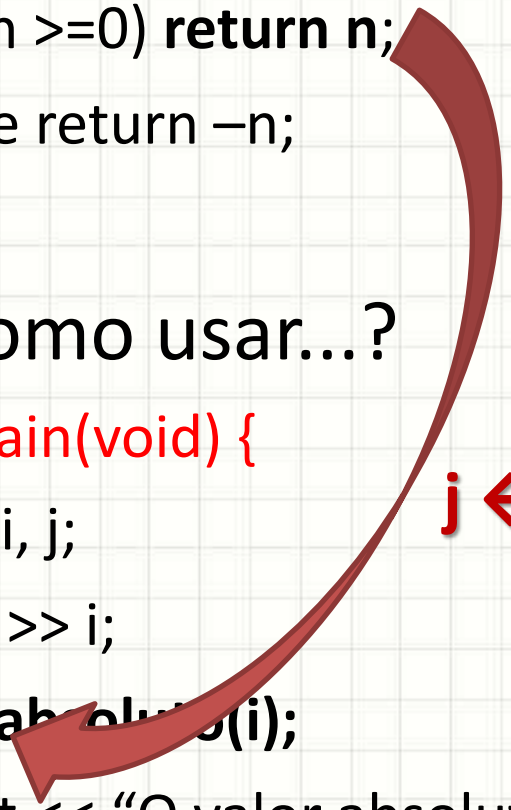
Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```



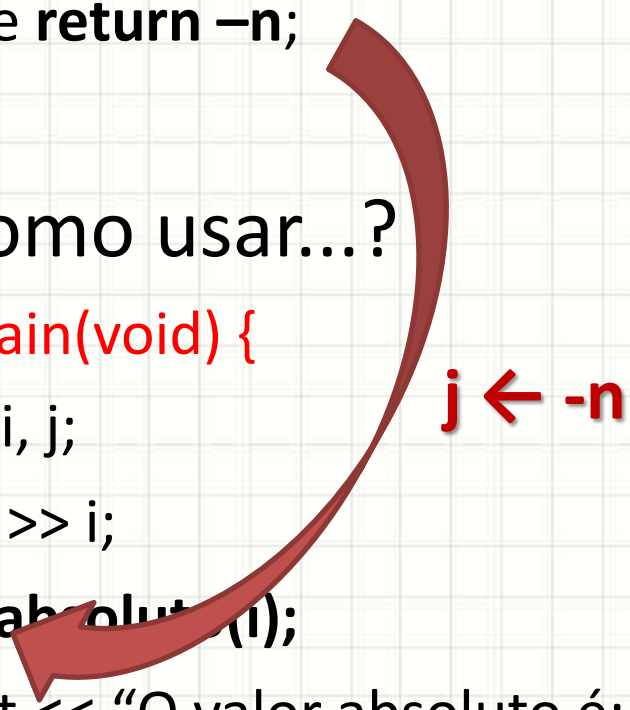
Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
   cout << “O valor absoluto é: ” << j;
}
```



Função com Parâmetros e Retorno

- Vejamos como funciona...

```
int absoluto(int n) {
    if (n >=0) return n;
    else return –n;
}
```

- Como usar...?

```
int main(void) {
    int i, j;
    cin >> i;
    j = absoluto(i);
    cout << “O valor absoluto é: ” << j;
}
```

Funções na Prática

Criando Funções

- **Passo 1**: criar um programa que calcule e imprima o perímetro de um círculo de raio 2
  - $P = 2 \cdot \pi \cdot R$
  - $\pi = 3,141592$

Criando Funções

- **Passo 2**: transformar o cálculo em uma função chamada **perimetro**

Criando Funções

- **Passo 2**: transformar o cálculo em uma função chamada **perimetro**
  - As variáveis criadas dentro da função só existem dentro desta função
  - Elas são chamadas **variáveis locais**
  - Não é possível acessar uma variável local a não ser de dentro da própria função
  - Os valores das variáveis locais **são destruídos** quando a função finaliza

Criando Funções

- **Passo 3**: modificar a função **perimetro** para que ela para que ela retorne o resultado, ao invés de imprimi-lo

Criando Funções

- **Passo 3**: modificar a função **perimetro** para que ela para que ela retorne o resultado, ao invés de imprimi-lo
  - **return** deve sempre retornar um valor do tipo correto
  - **return** pode ser usado sem nenhum valor em funções cujo retorno é do tipo “void”

Criando Funções

- **Passo 4**: modificar a função **perimetro** para que ela receba o raio do círculo como parâmetro

Criando Funções

- **Passo 4**: modificar a função **perimetro** para que ela receba o raio do círculo como parâmetro
  - Os parâmetros funcionam como variáveis locais
  - O valor fornecido como parâmetro (o raio) é **copiado** para essa "variável local"

Criando Funções

- **Passo 5**: mova a função para o fim do arquivo

Criando Funções

- **Passo 5**: mova a função para o fim do arquivo
  - A declaração inicial é chamada:
    - protótipo de função
    - declaração de função
    - assinatura de função

A Função Main

Função Main

- Você já deve ter reparado...
- Main é uma função!
- Seu nome é **main** porque é lá que o computador inicia a execução
- Observe sua definição:

```
int main(void) {

}
```

Função Main

- Observe sua definição:

```
int main(void) {

}
```

- Retorna **int**: código de erro para o Windows
- **void** indica que ela não usa parâmetros

Conclusões

Resumo

- O uso de funções simplifica o reaproveitamento de código
- As variáveis possuem um escopo
- As funções podem receber parâmetros e podem retornar resultados
- “Main” é uma função

- **Estudar para a AV1!**

PERGUNTAS?