Lógica de Programação para Engenharia

Outras Estruturas de Repetição

Prof. Dr. Daniel Caetano

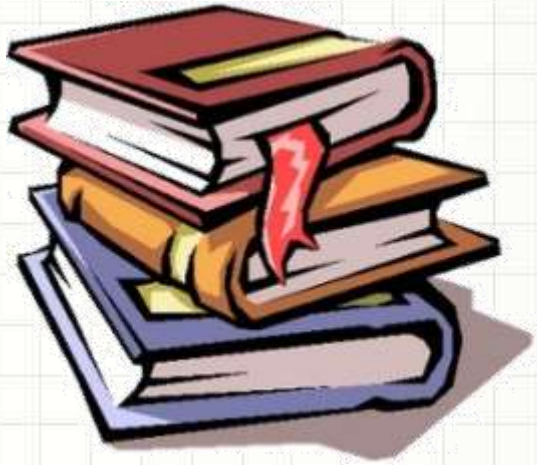
2012 - 1

Objetivos

- Conhecer as várias estruturas de repetição da linguagem C/C++
- Compreender o uso de cada uma destas estruturas
- Capacitar para a criação de algoritmos que envolvam repetição

- **PARA CASA**
  - **Lista de Exercícios 2 está ONLINE!**

Material de Estudo

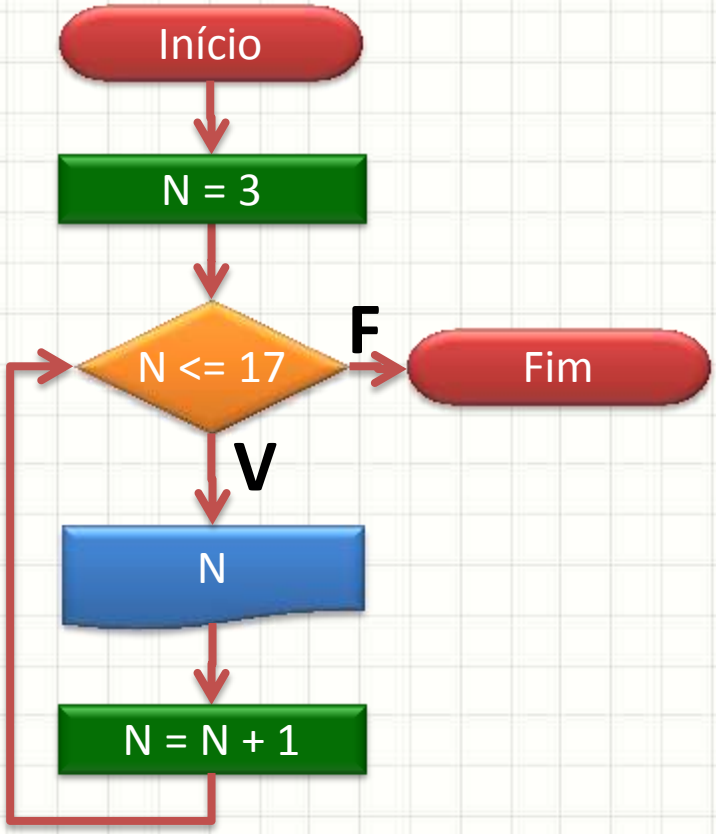


Material	Acesso ao Material
Notas de Aula	http://www.caetano.eng.br/aulas/lpe/ (Aula 13)
Apresentação	http://www.caetano.eng.br/aulas/lpe/ (Aula 13) – PARCIAL / COMPLETO
Material Didático	Fundamentos da Programação de Computadores – Parte 2 – Páginas 93 a 144.

Recordando o While

Recordando o While

- Aula passada: estrutura de repetição **while**
  - **O que faz?**

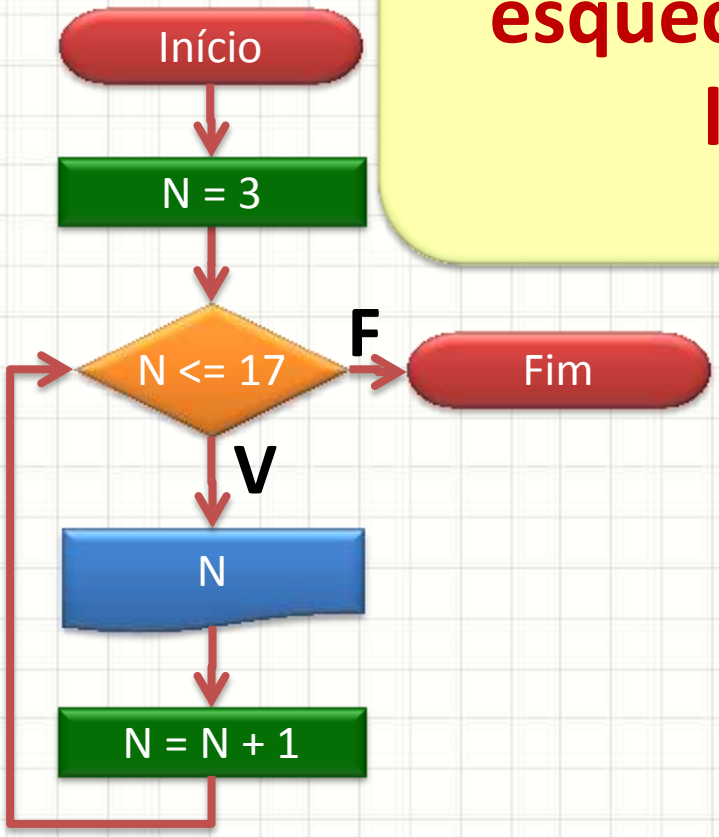



```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int N;

      N = 3;
      while ( N <= 17 ) {
          cout << N << endl;
          N = N + 1;
          }

getchar();
}
```

Recordando o While

- Aula passada: estrutura de repetição **while**
  - **O que fa**

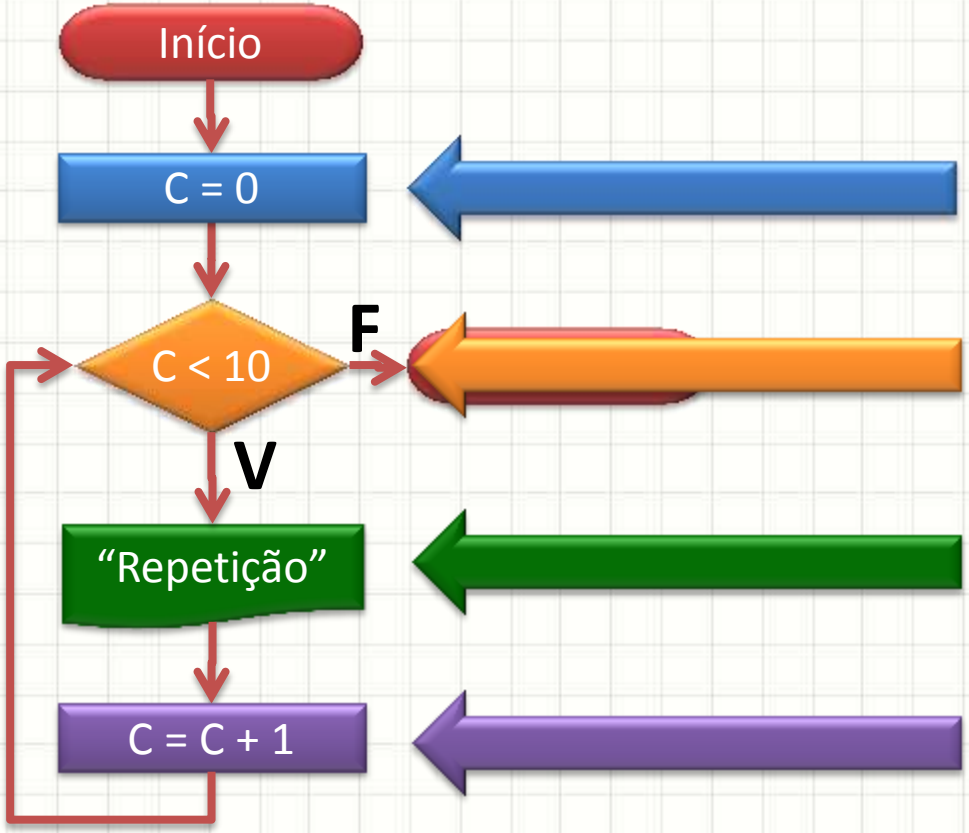
O que acontece se esquecermos essa linha?




```
o>
ream>
e std;
{
N =
while ( N <= 17 ) {
  cout << N << endl;
  N = N + 1;
}

getchar();
}
```

Recordando o While

- Observe:
  - O que faz?



INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

BLOCO

ATUALIZAÇÃO

Recordando o While

- No código...

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```

le

Fácil esquecer um deles!

```
#i
#i
usi
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```

INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

BLOCO

ATUALIZAÇÃO

A Estrutura de Repetição FOR

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      CONT = 0;
      while ( CONT < 10 ) {
         cout << "Isso é uma repetição" << endl;
         CONT = CONT + 1;
         }

getchar();
}
```

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

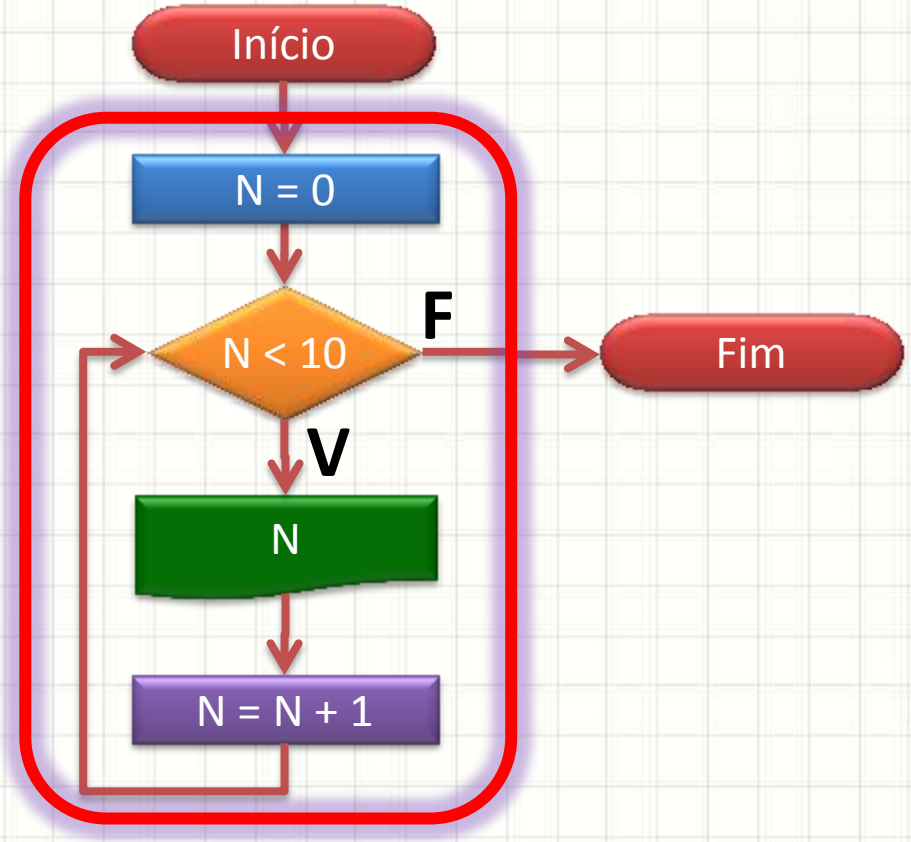
```
#include <stdio>
#include <iostream>
using namespace std;
int main(void) {

      int CONT;

      for (CONT = 0; CONT < 10; CONT = CONT + 1) {
         cout << "Isso é uma repetição" << endl;
         }

getchar();
}
```

O que é a estrutura **for**

- Todos os elementos em uma única linha
  - Só o bloco fica “isolado”

Forma Geral do **for**

```
for ( inicialização; condição de repetição; atualização)  {
    Executa enquanto a proposição for verdadeira
}
```



Leitura do **for**

```
for ( X = 0 ; X < 7 ; X = X + 2 ) {
    cout << X << endl;
}
```

Faça, a partir de X = 0, enquanto X < 7 e contando de 2 em 2, a impressão de X.

EXERCÍCIO

A) Faça um programa que apresente os números de 50 a 75.

EXERCÍCIO

B) Modifique o programa anterior para que ele conte de 2 em 2.

EXERCÍCIO

C) Modifique o programa para que imprima só números divisíveis por 5.

Repetição com Do~While

Repetição com Do~While

- Algumas vezes queremos que um procedimento seja executado “pelo menos uma vez”.

- Quando?
  - Esperar que um dado específico seja digitado...
    - É preciso ler a entrada **antes** de testá-la
  - Esperar um valor específico de um sensor
    - É preciso ler o sensor **antes** de testar o valor
  - Etc.

Repetição com Do~While

- Observe:

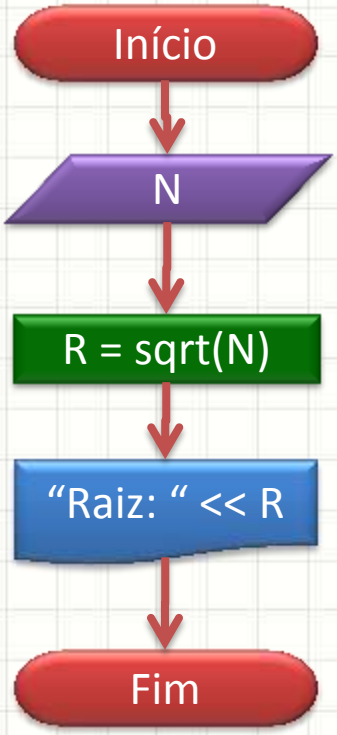
```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite um número positivo: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

E se o usuário digitar um número negativo?

Não seria legal poder repetir a pergunta?

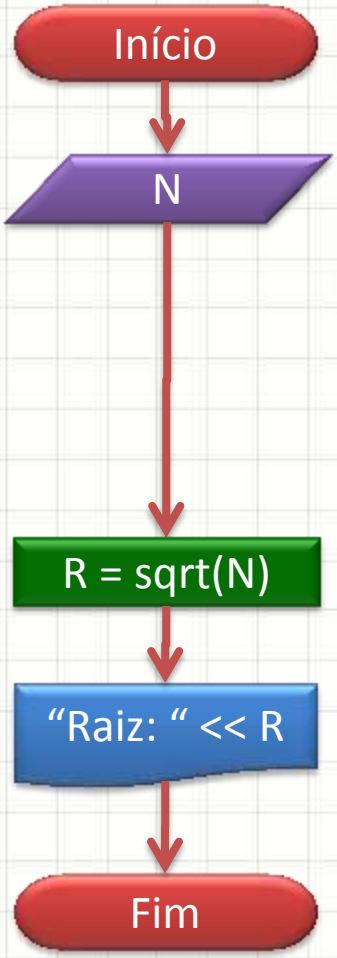
Repetição com Do~While

```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite no. > 0: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```



Repetição com Do~While

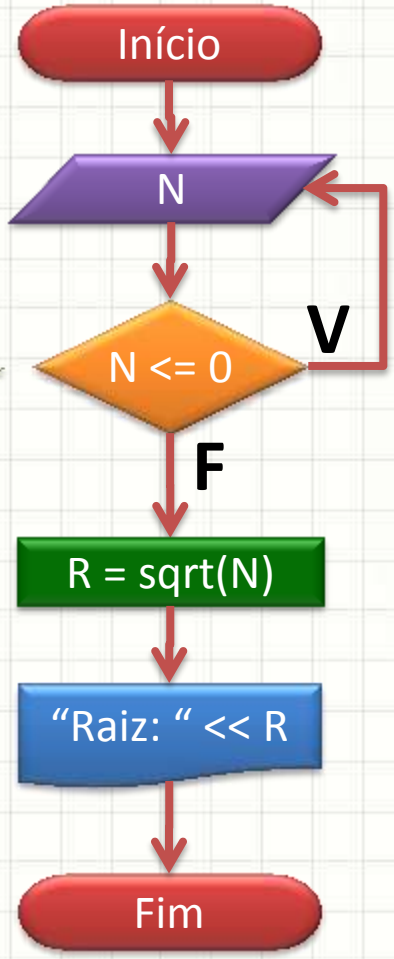
```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    cout << “Digite no. > 0: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```



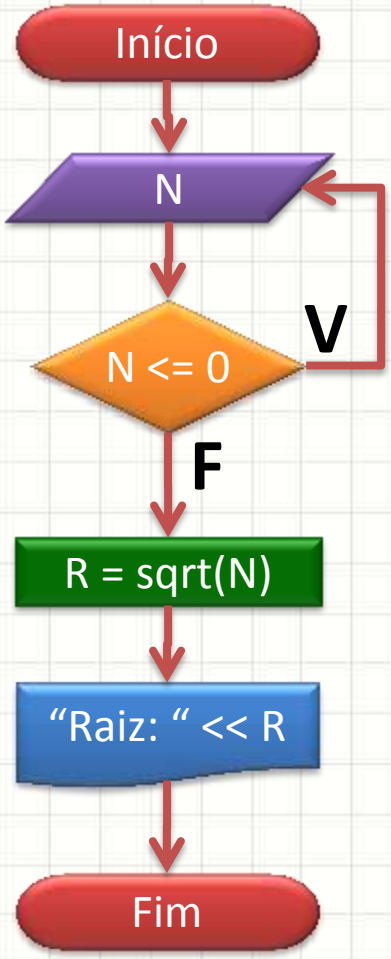
Repetição com Do~While

```
#include <stdio>
#include <math>
#include <iostream>
using nam
int main(
    float
    cout <
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

while serve?



Repetição com Do~While




```
#include <stdio>
#include <math>
#include <iostream>
using namespace std;
int main(void) {
    float N,R;
    do {
        cout << “Digite no. > 0: ”;
        cin >> N;
        } while ( N <= 0 );
    R = sqrt(N);
    cout << “Raiz: ” << R;
getchar();
}
```

Forma Geral do **do~while**

```
do {
    Executa enquanto a proposição for verdadeira
} while ( condição de repetição );
```

- Qual a diferença com relação ao **while** ?

```
while ( condição de repetição ) {
    Executa enquanto a proposição for verdadeira
}
```

EXERCÍCIO

A) Crie um menu para que ele contenha as seguintes opções:

1- Saldo

2- Extrato

3- Saque

4- Depósito

Para cada opção ele deve imprimir um texto que indique a opção selecionada... e ele não deve aceitar opções inválidas

EXERCÍCIO

B.1) Analise os códigos e descubra qual é mais adequado para **while** e qual para **do~while**:

Código 1

a) Leia um número N;

b) N = N * 2;

c) Se N for menor que 32, volta para o passo (b);

d) Imprima N.

Código 2

a) Leia um número N;

b) Enquanto N for menor que 32, repita (c)

c) N = N * 2;

d) Imprima N.

EXERCÍCIO

B.2) Implemente ambos os códigos (pode ser no mesmo programa):

Código 1

a) Leia um número N;

b) N = N * 2;

c) Se N for menor que 32, volta para o passo (b);

d) Imprima N.

Código 2

a) Leia um número N;

b) Enquanto N for menor que 32, repita (c)

c) N = N * 2;

d) Imprima N.

EXERCÍCIO

B.3) Execute ambos os códigos para as entradas:

0

20

40

E responda: os resultados são sempre iguais? Por quê?

Conclusões

Resumo

- Existem diversos tipos de estruturas de decisão
- Elas são intercambiáveis, isto é, tudo que se faz com um uma, é possível fazer com outra
- Dependendo da situação, cada uma delas é mais apropriada!

- **TAREFA!**
  - **Lista de Exercícios 2!**

Próxima Aula

- Chegamos ao fim do curso de lógica...
  - Treinar... treinar... e treinar!
  - Aulas finais: exercícios diversos!

PERGUNTAS?

Bom Descanso a Todos!