



LÓGICA DE PROGRAMAÇÃO PARA ENGENHARIA

OUTRAS ESTRUTURAS DE REPETIÇÃO

Prof. Dr. Daniel Caetano

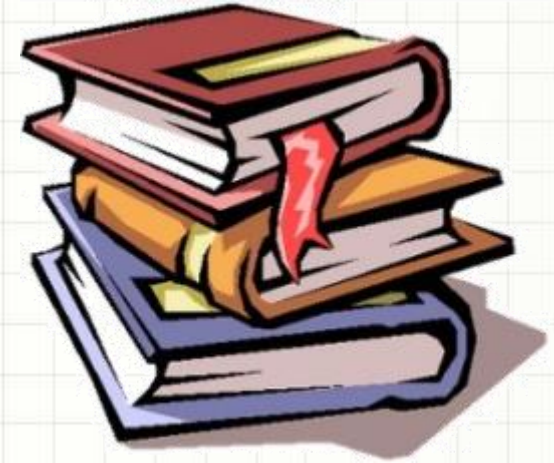
2013 - 1

Objetivos

- Conhecer outras estruturas de repetição da linguagem C/C++
- Compreender o uso de cada uma delas
- Capacitar para a criação de algoritmos com repetição
- **PARA CASA**
 - Lista Aula 14!



Material de Estudo



Material

Acesso ao Material

Notas de Aula

<http://www.caetano.eng.br/>
(Aula 14)

Apresentação

<http://www.caetano.eng.br/>
(Aula 14) – PARCIAL / COMPLETO

Material Didático

Fundamentos da Programação de Computadores –
Parte 2 – Páginas 93 a 144.

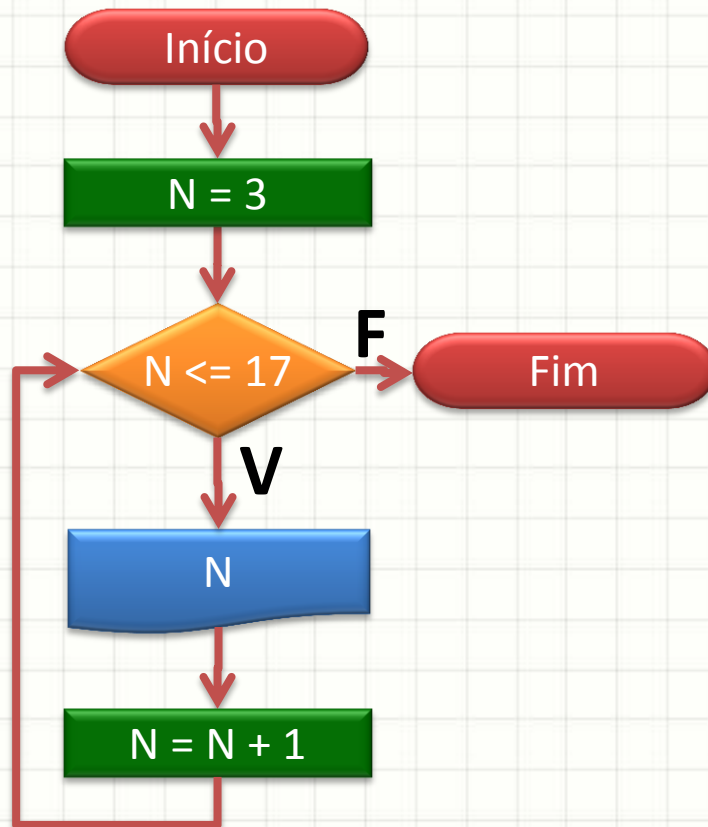


**RECORDANDO O
WHILE**

Recordando o While

- Aula passada: estrutura de repetição **while**

– O que faz?



```
#include <iostream>
using namespace std;
main()
{
    int N;
    N = 3;
    while ( N <= 17 )
    {
        cout << N << endl;
        N = N + 1;
    }
}
```

Recordando o While

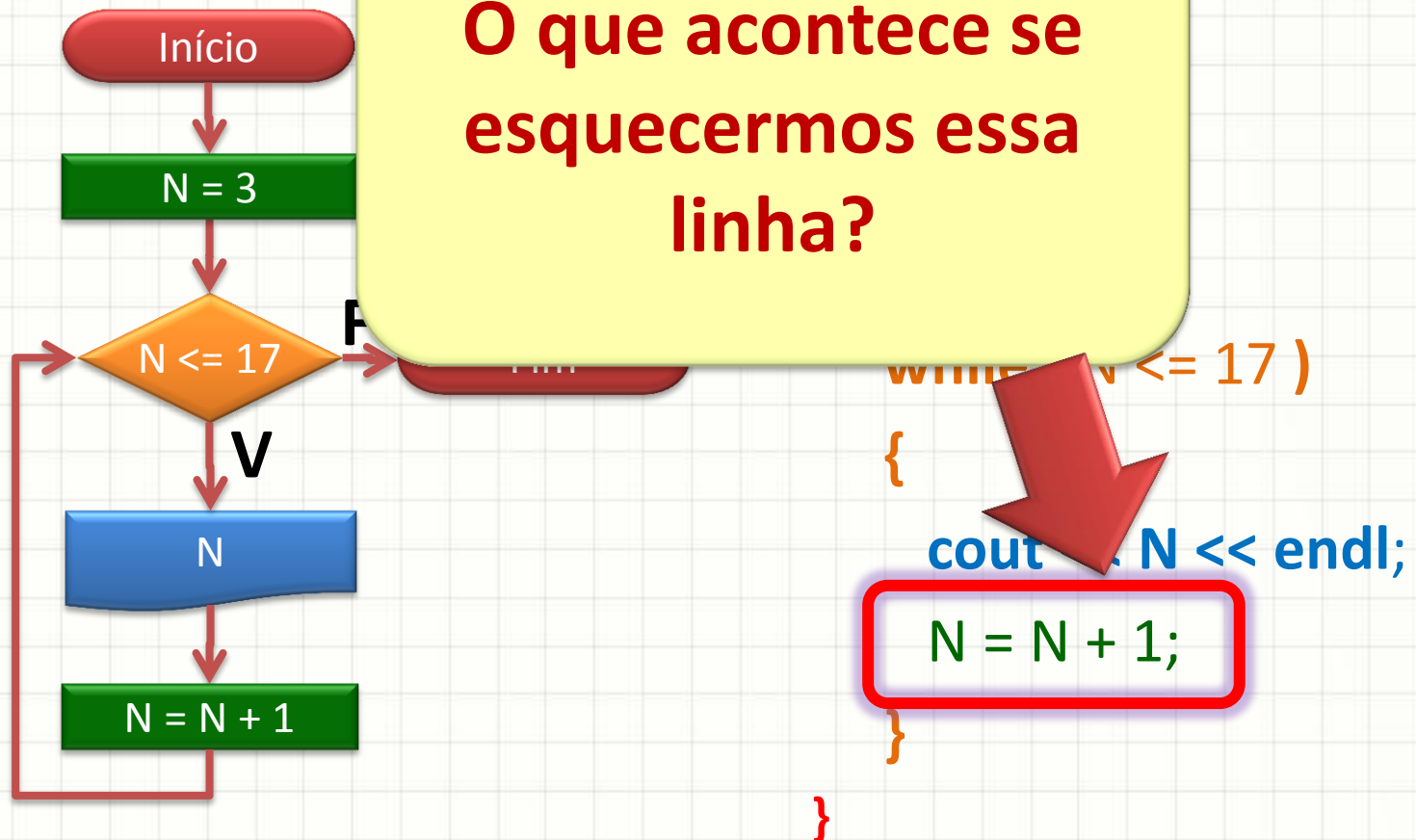
- Aula passada: estrutura de repetição **while**

– O que faz?

```
#include <iostream>
```

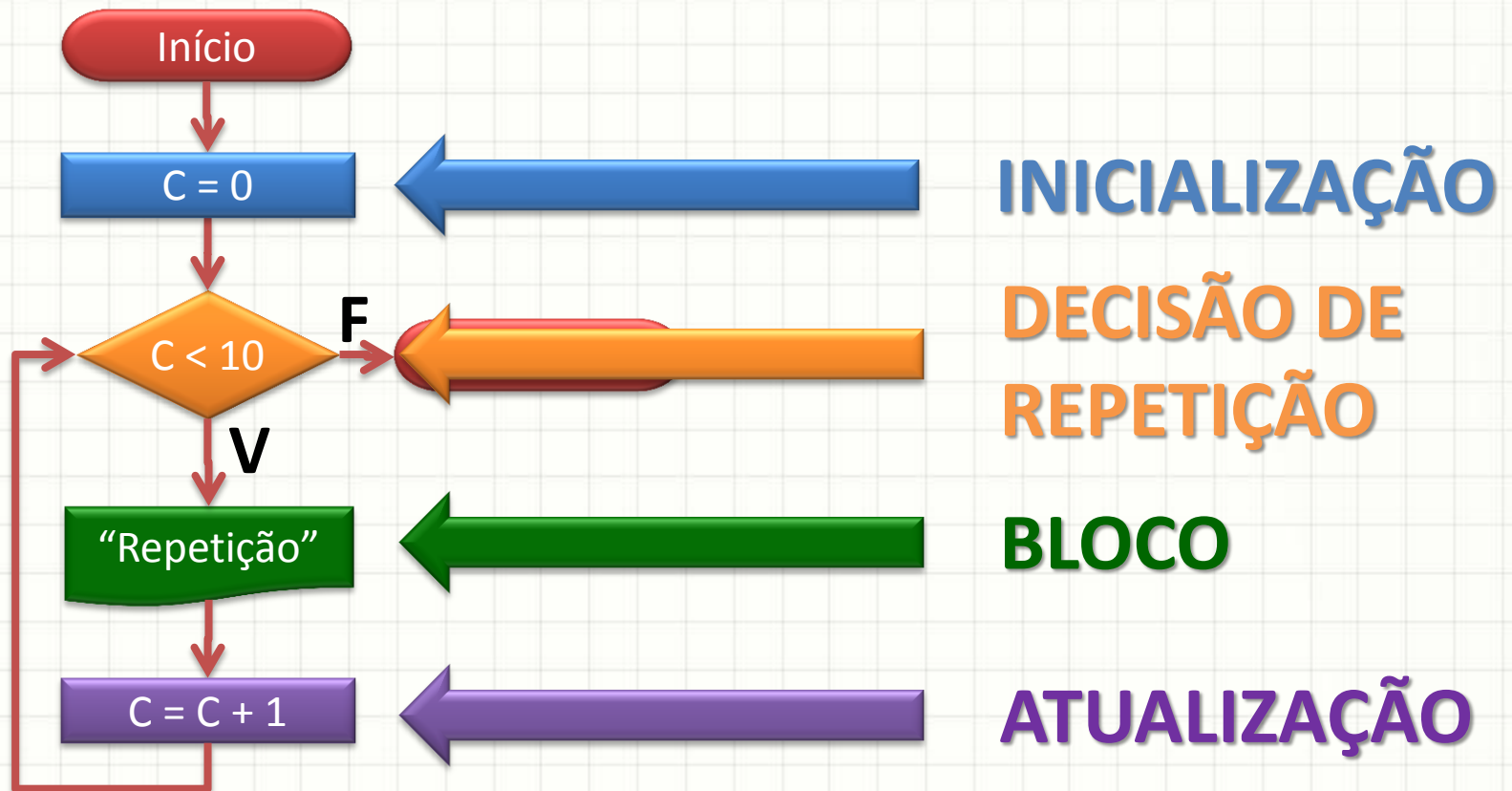
```
using namespace std;
```

O que acontece se esquecermos essa linha?



Recordando o While

- Observe:
 - O que faz?



Recordando o While

- No código...

```
#include <iostream>
using namespace std;
main()
```

```
{
```

```
int CONT;
```

```
CONT = 0;
```

```
while ( CONT < 10 )
```

```
{
```

```
cout << "Isso é uma Repetição" << endl;
```

```
CONT = CONT + 1;
```

```
}
```

```
}
```

INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

BLOCO

ATUALIZAÇÃO

Fácil esquecer um deles!

le

```
main()
```

```
{
```

```
int CONT;
```

```
CONT = 0;
```

```
while ( CONT < 10 )
```

```
{
```

```
cout << "Isso é uma Repetição" << endl;
```

```
CONT = CONT + 1;
```

```
}
```

```
}
```

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 <iostream>
using namespace std;
main()
{
    int CONT;
    CONT = 0;
    while ( CONT < 10 )
    {
        cout << “Isso é uma Repetição” << endl;
        CONT = CONT + 1;
    }
}
```

O que é a estrutura **for**

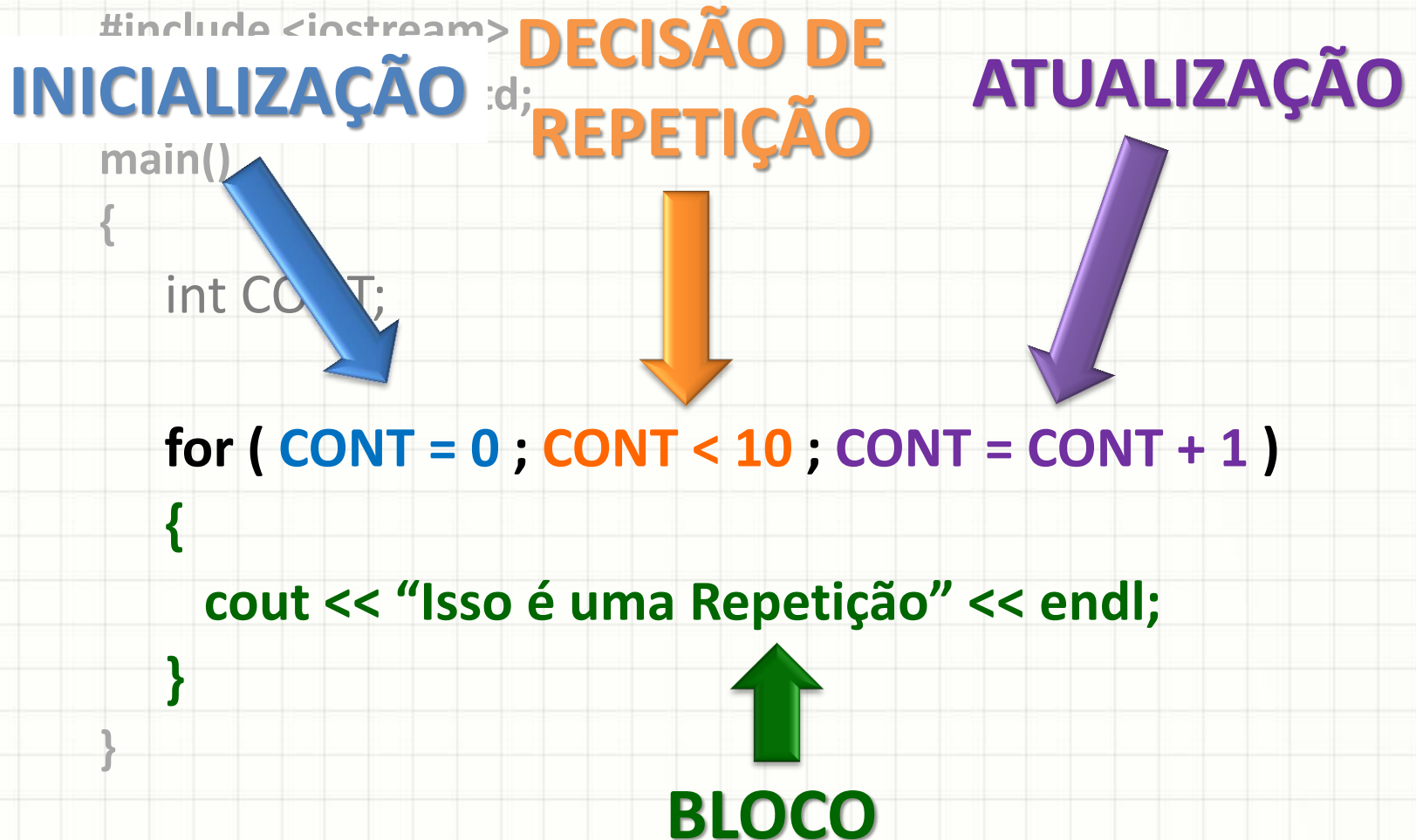
- Todos os elementos em uma única linha
 - Só o bloco fica “isolado”

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

    for ( CONT = 0 ; CONT < 10 ; CONT = CONT + 1 )
    {
        cout << “Isso é uma Repetição” << endl;
    }
}
```

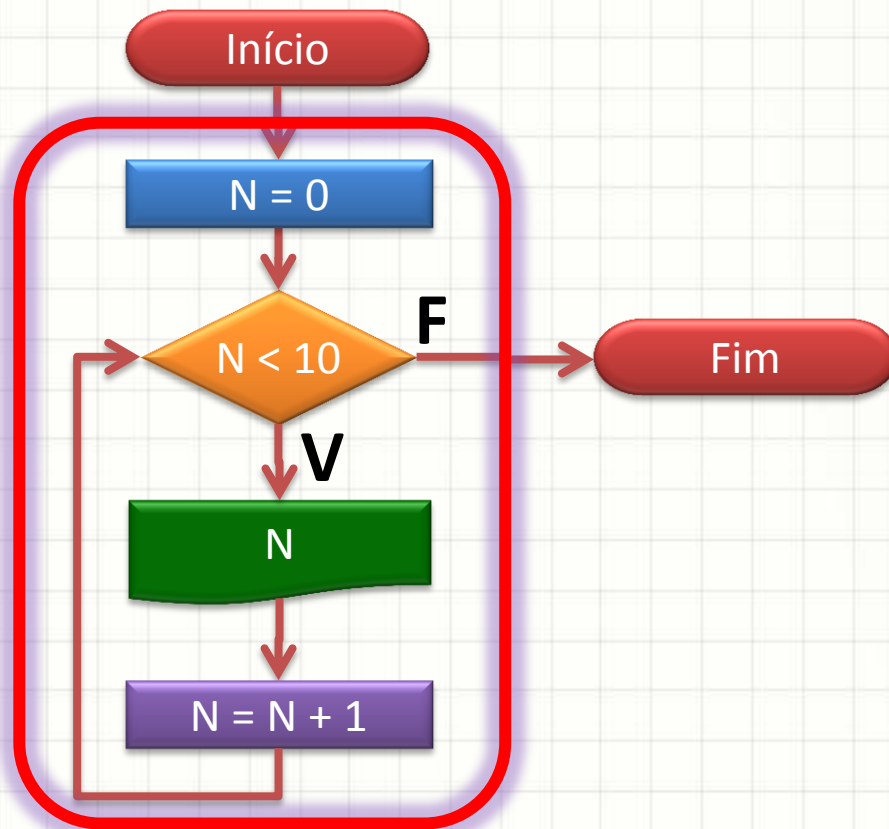
O que é a estrutura **for**

- 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 52 a 75.

EXERCÍCIO

A) Faça um programa que apresente os números de 52 a 75.

Exemplo:

```
#include <iostream>
using namespace std;
main()
{
    int CONT;
    for ( CONT = 0 ; CONT < 10 ; CONT = CONT + 1 )
    {
        cout << "Isso é uma Repetição" << endl;
    }
}
```

EXERCÍCIO

A) Faça um programa que apresente os números de 52 a 75.

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

    for ( C = 52 ; C <= 75; C = C + 1 )
    {
        cout << C << endl;
    }
}
```

EXERCÍCIO

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

EXERCÍCIO

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

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

    for ( C = 52 ; C <= 75; C = C + 1 )
    {
        cout << C << endl;
    }
}
```

Exercício
Anterior!

EXERCÍCIO

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

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

    for ( C = 52 ; C <= 75; C = C + 2 )
    {
        cout << C << endl;
    }
}
```

EXERCÍCIO

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

EXERCÍCIO

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

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

    for ( C = 52 ; C <= 75; C = C + 2 )
    {
        cout << C << endl;
    }
}
```

Exercício
Anterior!

EXERCÍCIO

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

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

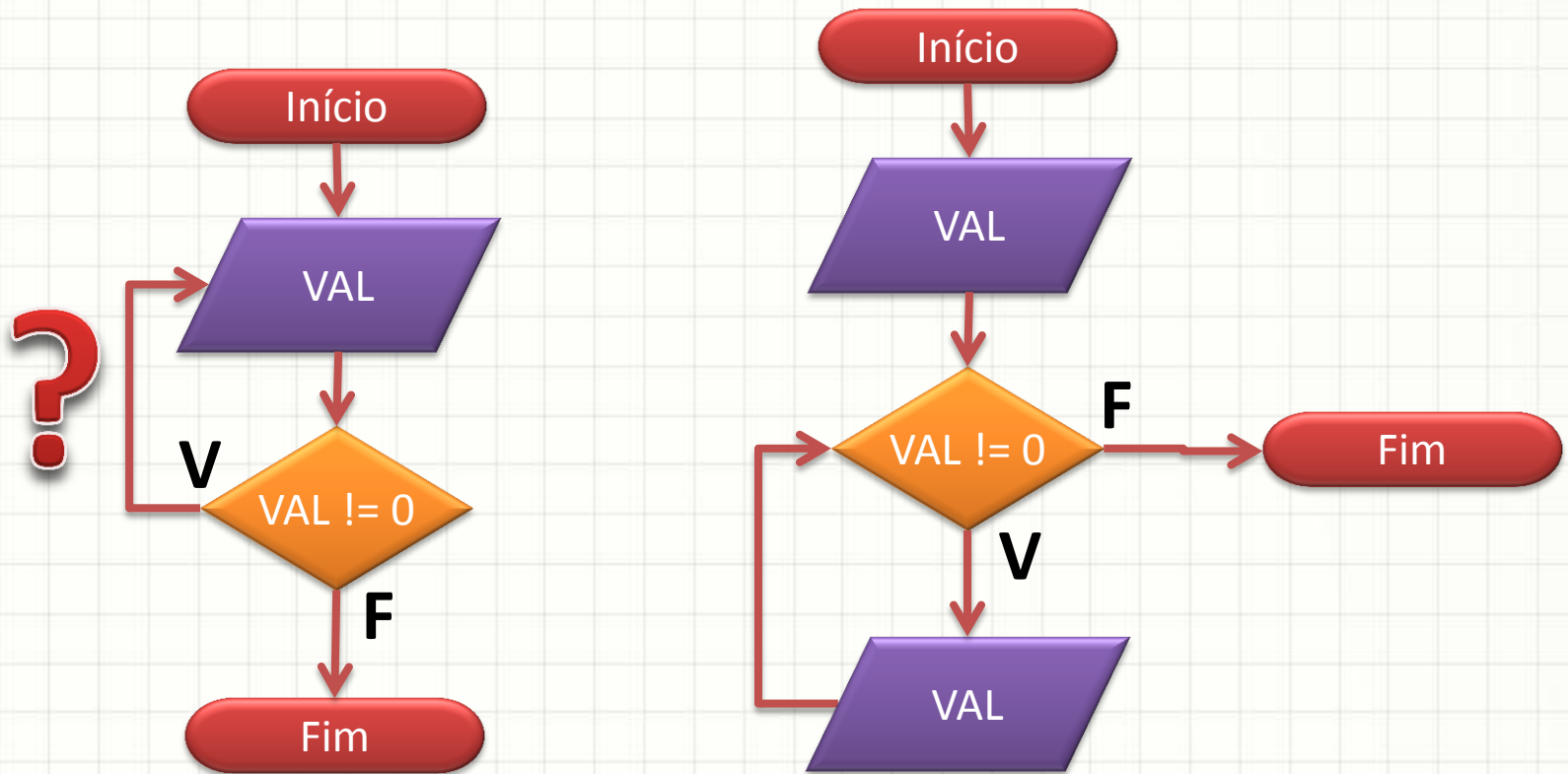
    for ( C = 52 ; C <= 75; C = C + 2 )
    {
        if ( C%5 == 0 )
            cout << C << endl;
    }
}
```




REPETIÇÃO COM DO~WHILE

Repetição com Do~While

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



Repetição com Do~While

- Isso ocorre muito?
 - Esperar que um dado específico seja digitado...
 - É preciso ler a entrada **antes** de testá-la



4R4CB0XA.86A.0015.P09.9904161024

Intel (R) Pentium (R) II Processor, 400MHz
128MB System RAM

USB Legacy Enabled

Keyboard Error
Press F1 to Resume

Fixed Disk 0: Maxtor 90845D4
ATAPI CD-ROM PHILIPS CD-ROM PCCD04B

Repetição com Do~While

- Isso ocorre muito?
 - Esperar um valor específico de um sensor
 - É preciso ler o sensor **antes** de testar o valor



Repetição com Do~While

- Observe:

```
#include <math.h>
#include <iostream>
using namespace std;
```

```
main()
```

```
{
```

```
    float N,R;
```

```
    cout << "Digite um número positivo: ";
```

```
    cin >> N;
```

```
    R = sqrt(N);
```

```
    cout << "Raiz: " << R;
```

```
}
```

E se o usuário digitar um número negativo?

Não seria legal poder repetir a pergunta?

Repetição com Do~While

```
#include <math.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
main()
```

```
{
```

```
    float N,R;
```

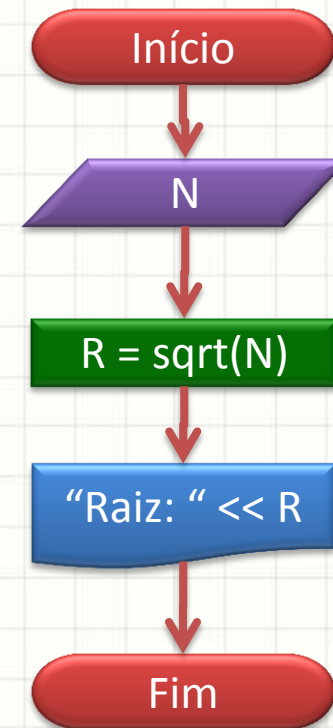
```
    cout << "Digite no. > 0: ";
```

```
    cin >> N;
```

```
    R = sqrt(N);
```

```
    cout << "Raiz: " << R;
```

```
}
```



Repetição com Do~While

```
#include <math.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
main()
```

```
{
```

```
    float N,R;
```

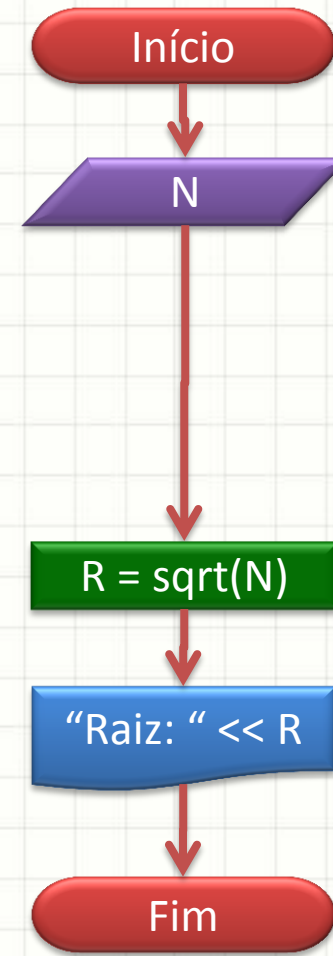
```
    cout << "Digite no. > 0: ";
```

```
    cin >> N;
```

```
    R = sqrt(N);
```

```
    cout << "Raiz: " << R;
```

```
}
```



Repetição com Do~While

```
#include <math.h>
```

```
#include <iostream>
```

```
using namespace std;
```

```
int main()
```

```
{
```

```
float l
```

```
cout << "Digite no. > 0: ";
```

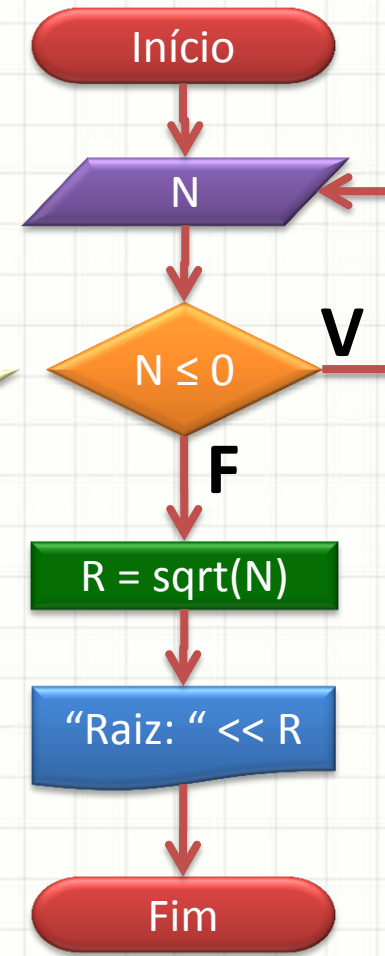
```
cin >> N;
```

```
R = sqrt(N);
```

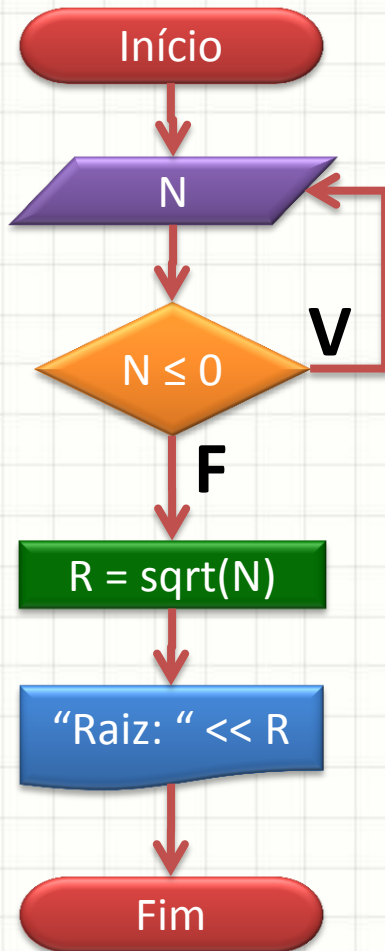
```
cout << "Raiz: " << R;
```

```
}
```

**while
serve?**



Repetição com Do~While



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

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

}

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

- Para cada opção ele deve imprimir um texto que indique a opção selecionada: “Saldo” ou “Extrato”.
- O programa não deve aceitar opções inválidas

EXERCÍCIO

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

- 1- Saldo
- 2- Extrato

```
#include <iostream>
using namespace std;
main()
{
    int N;
    do
    {
        cout << "1- Saldo" << endl;
        cout << "2- Extrato" << endl;
        cout << "Opcao: ";
        cin >> N;
    } while ( N !=1 && N != 2 );

    if ( N == 1 )
        cout << "Saldo" << endl;
    if ( N == 2 )
        cout << "Extrato" << endl;
}
```

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.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.



do~while

Código 2

- a) Leia um número N;
- b) Enquanto N for menor que 32, repita (c)
- c) $N = N * 2$;
- d) Imprima N.



while

EXERCÍCIO

B.2) Uma parte da turma deve implementar o código 1 e outra o código 2:

Use um dos anteriores como base!

Código 1

- a) Leia um número N ;
- b) $N = N * 2$;
- c) Enquanto $N < 32$, volta para o passo (b);
- d) Imprima N .

Código 2

- a) Leia um número N ;
- b) Enquanto $N < 32$, repita (c)
- c) $N = N * 2$;
- d) Imprima N .

EXERCÍCIO

B.2) Implemente os códigos

Código 1

- Leia um número N;
- $N = N * 2$;
- Se N for menor que 32, volta para o passo (b);
- Imprima N.

```
#include <iostream>
using namespace std;
main()
{
    float N;
    cout << "Digite um no.:";
    cin >> N;
    do
    {
        N = N * 2;
    } while ( N < 32 );
    cout << "Resultado:" << N << endl;
}
```

EXERCÍCIO

B.2)
Implemente os
códigos:

Código 2

- Leia um número N;
- Enquanto N for menor que 32, repita (c)
- $N = N * 2$;
- Imprima N.

```
#include <iostream>
using namespace std;
main()
{
    float N;
    cout << "Digite um no.:";
    cin >> N;
    while ( N < 32 )
    {
        N = N * 2;
    }

    cout << "Resultado:" << N << endl;
}
```

EXERCÍCIO

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

0

20

40

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

EXERCÍCIO

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

0

20

40

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

Valor de Entrada:	Resultado Código 1	Resultado Código 2
0	Travou	Travou
20	40	40
40	80	40



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 Aula 14!

Próxima Aula



THE END!



PERGUNTAS?



**BOM DESCANSO
A TODOS!**