Lógica de Programação para Engenharia

Outras Estruturas de Repetição

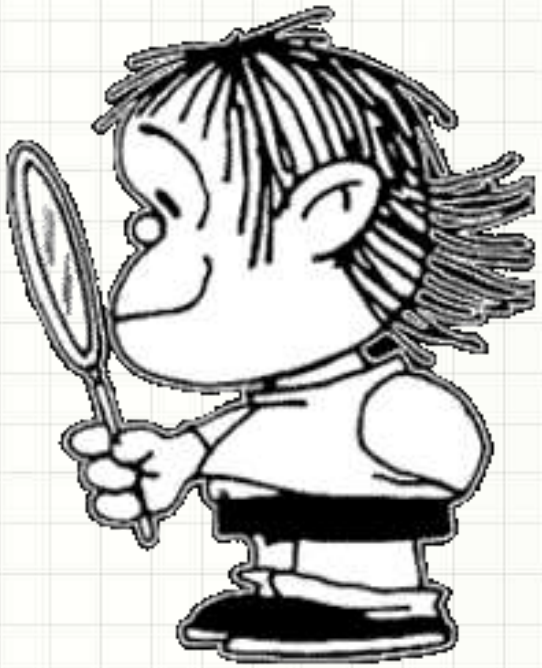
Prof. Dr. Daniel Caetano

2018 - 1

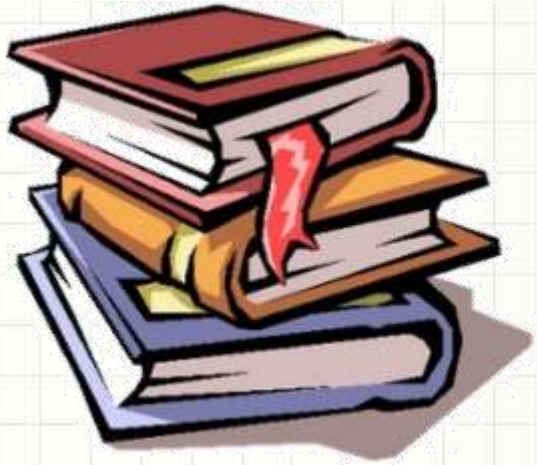
Objetivos

- Conhecer outras estruturas de repetição da linguagem C/C++
- Compreender o uso de cada uma delas
- Capacitar para a criação de algoritmos com repetição

- **Atividades Aula 11 – SAVA!**



Material de Estudo

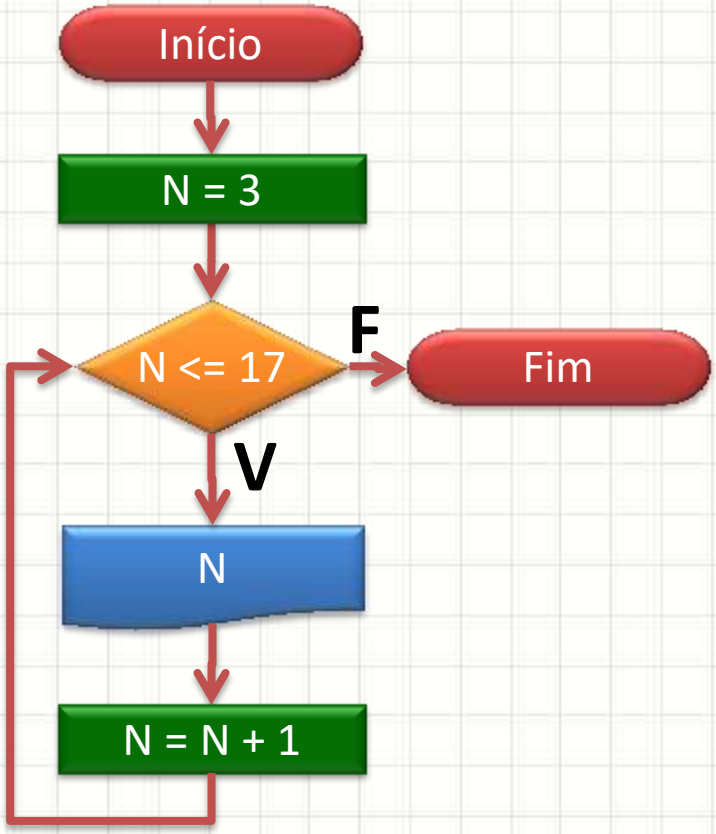


Material	Acesso ao Material
Notas de Aula e Apresentação	http://www.caetano.eng.br/ (Lógica de Programação para Eng. – Aula 11)
Material Didático	Lógica de Programação, págs 124 a 149.
Aula Online	Aula 7,8 e 9
Biblioteca Virtual	“Lógica de Programação – Fundamentos da Programação de Computadores”, págs 93 a 144.

Recordando o While

Recordando o While

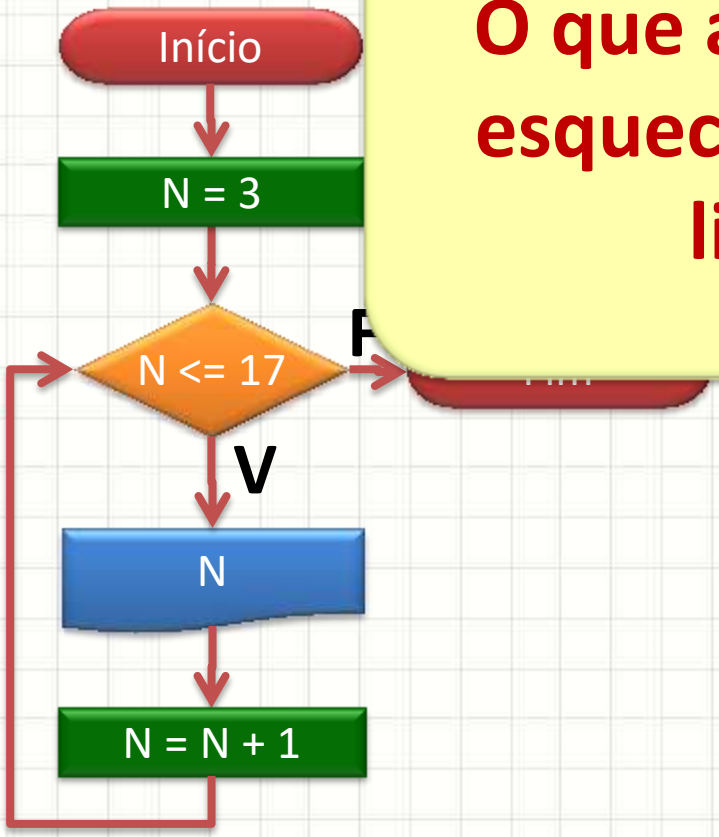
- Estrutura de repetição **while**
  - **O que faz?**




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

Recordando o While

- Estrutura de repetição **while**
  - **O que faz?**

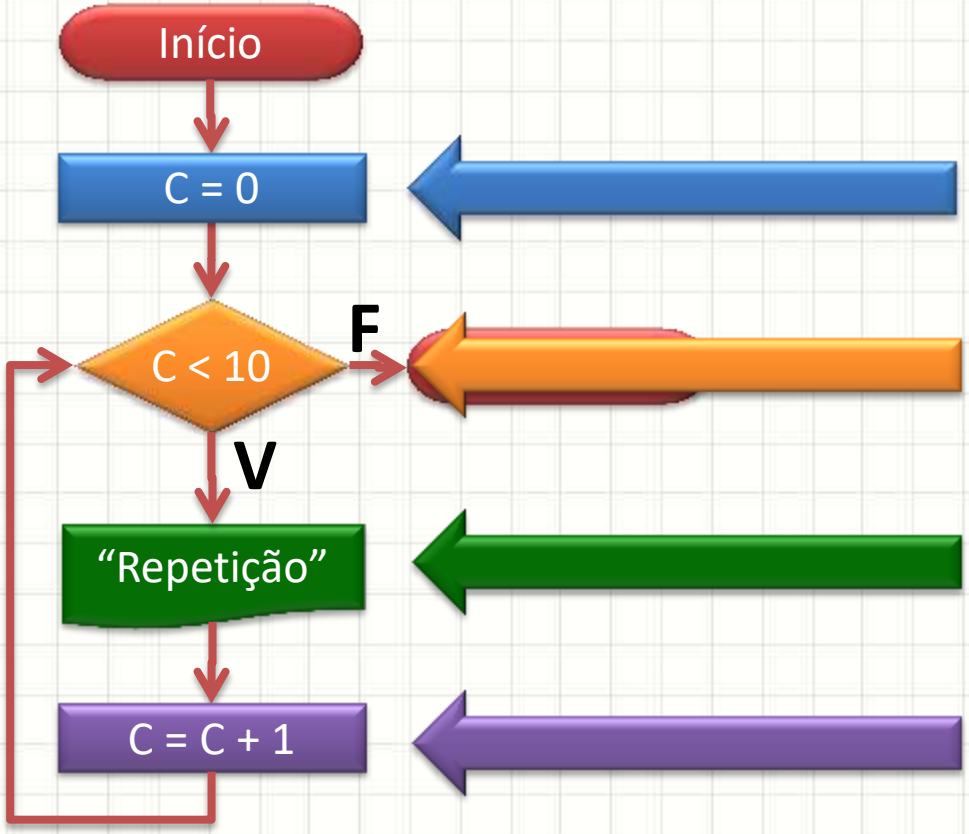


O que acontece se esquecermos essa linha?

```
#include <iostream>
... std;
...
while( N <= 17 )
{
    cout << N << endl;
    N = N + 1;
}
}
```

Recordando o While

- Observe:
  - O que faz?



INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

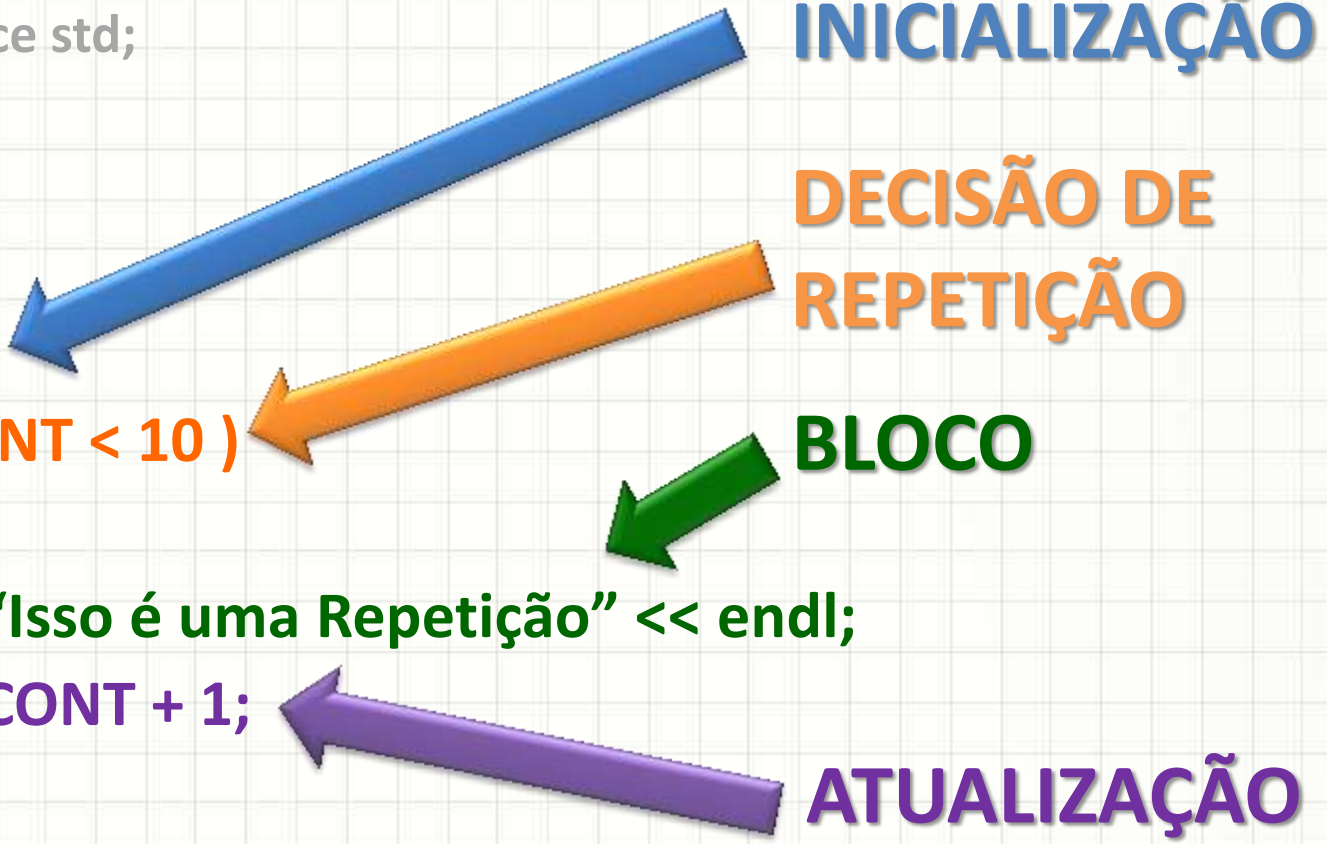
BLOCO

ATUALIZAÇÃO

Recordando o While

- No código...

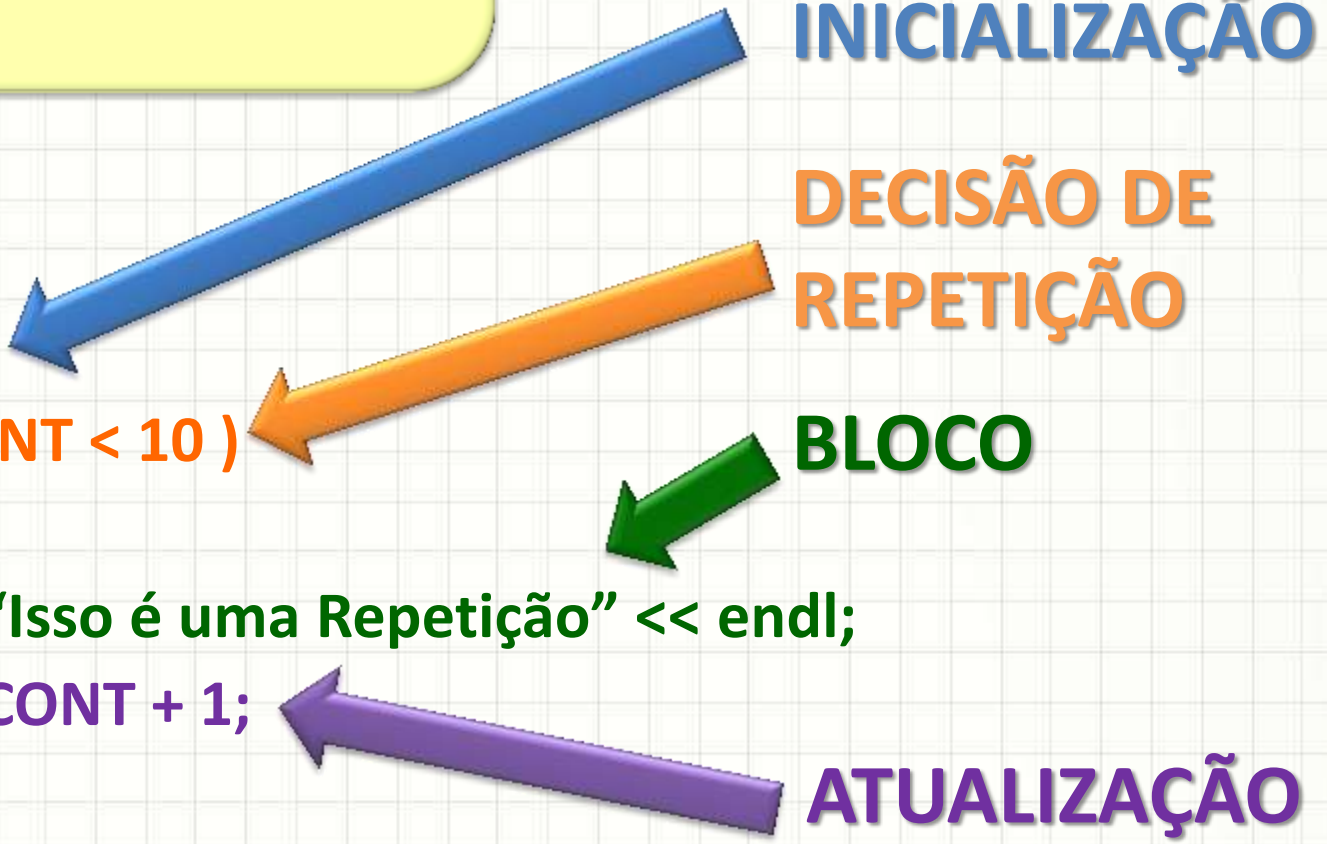
```
#include <iostream>
using namespace std;
main()
{
    int CONT;
    CONT = 0;
    while ( CONT < 10 )
    {
      cout << "Isso é uma Repetição" << endl;
      CONT = CONT + 1;
    }
}
```



le

Fácil esquecer um deles!

```
main()
{
    int CONT;
    CONT = 0;
    while ( CONT < 10 )
    {
      cout << "Isso é uma Repetição" << endl;
      CONT = CONT + 1;
    }
}
```



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”

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

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

INICIALIZAÇÃO

DECISÃO DE REPETIÇÃO

ATUALIZAÇÃO

BLOCO

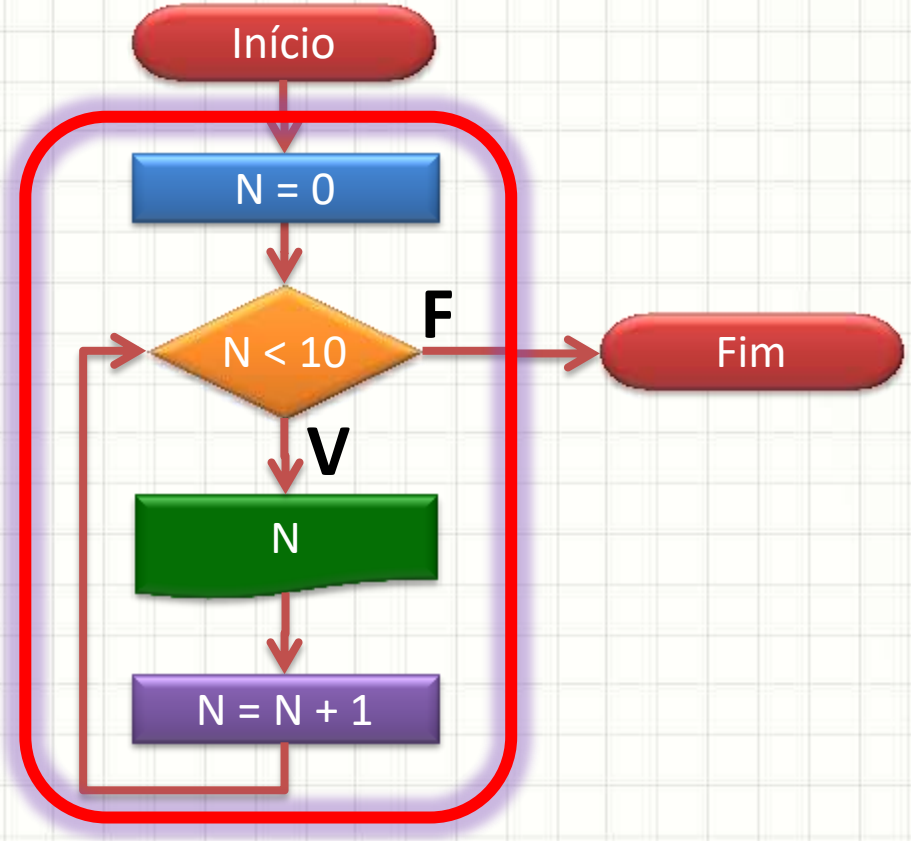
Forma Geral do **for**

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;
}
```

Para X variando de 0 até o um valor < 7, contando de 2 em 2, imprima 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

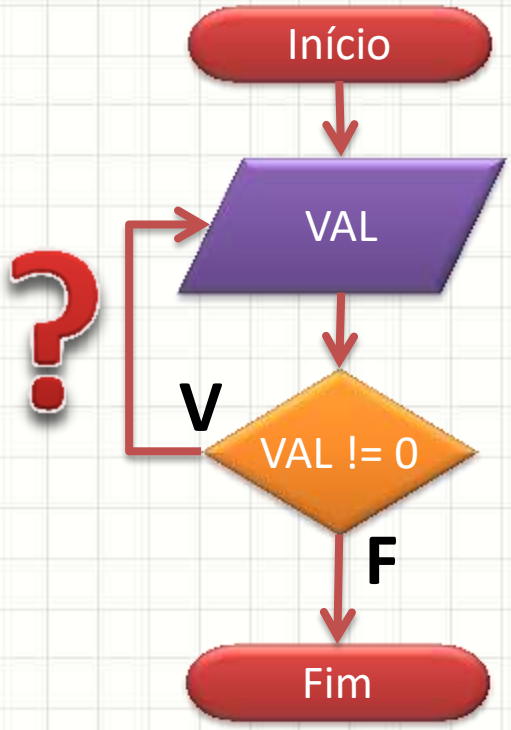
C) Modifique para que imprima só números divisíveis por 5.

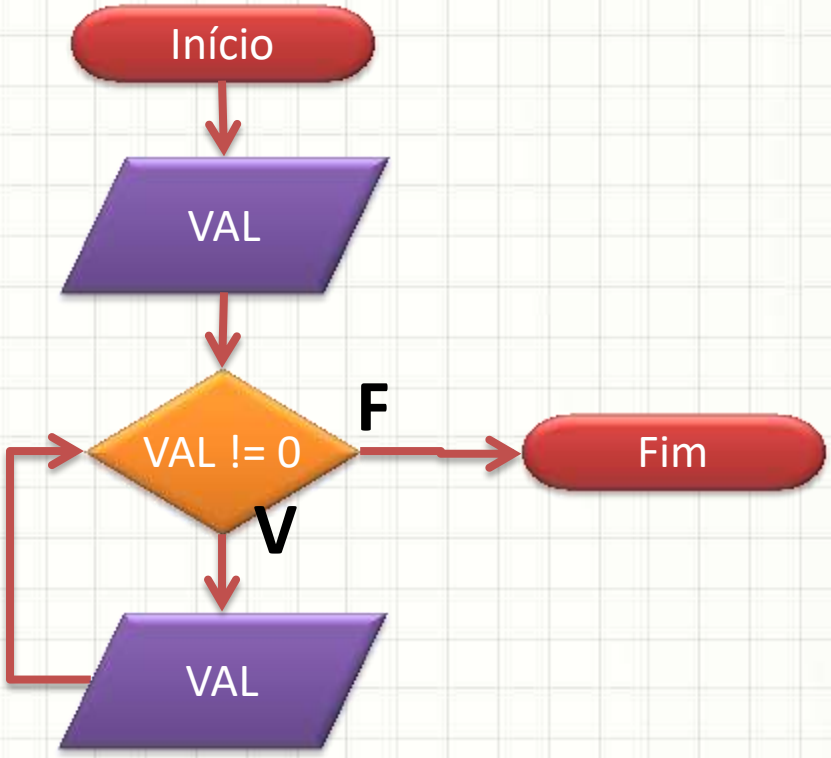
```
#include <iostream>
using namespace std;
main()
{
    int C;

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

Repetição com Do~While

Repetição com Do~While

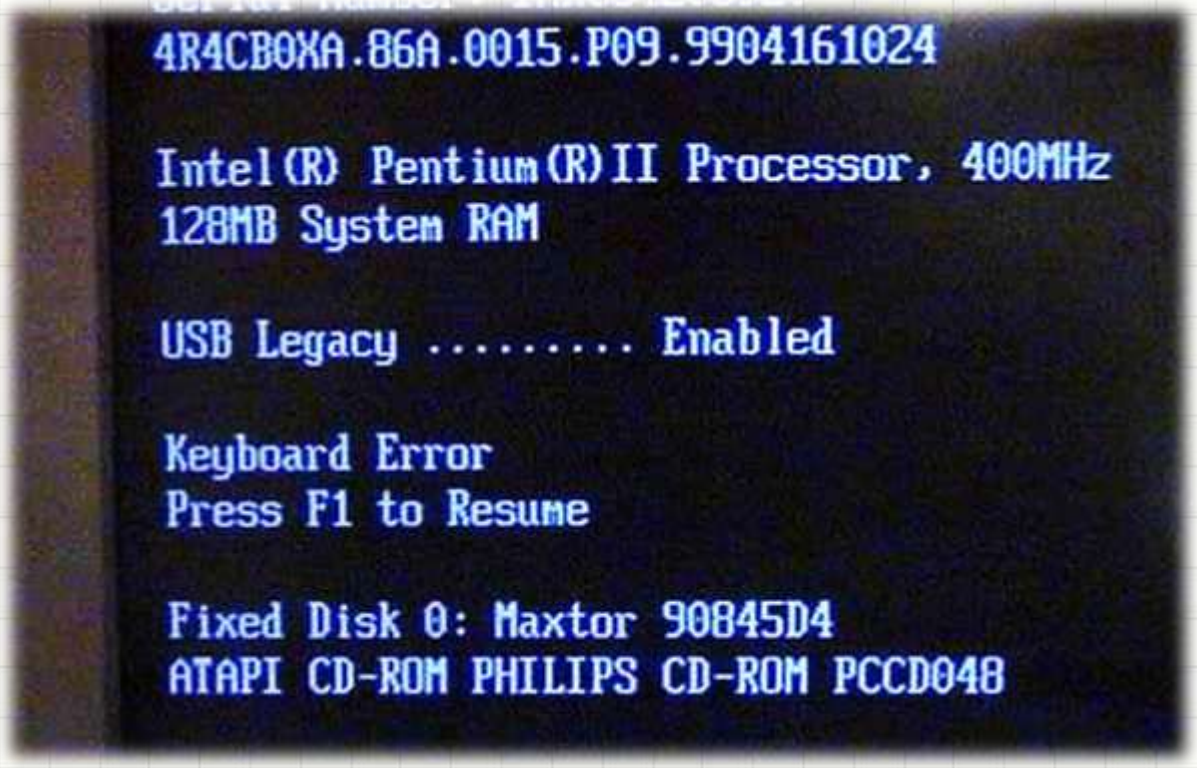
- Algumas vezes queremos repetir um procedimento até que ele resulte em um valor específico:





Repetição com Do~While

- Isso ocorre muito?
  - Ler entrada até que um dado “x” seja digitado...
    - É preciso ler a entrada **antes** de testá-la



Repetição com Do~While

- Isso ocorre muito?
  - Ler sensor até que um valor específico ocorra
    - É preciso ler o sensor **antes** de testar o valor

Repetição com Do~While

- Observe:

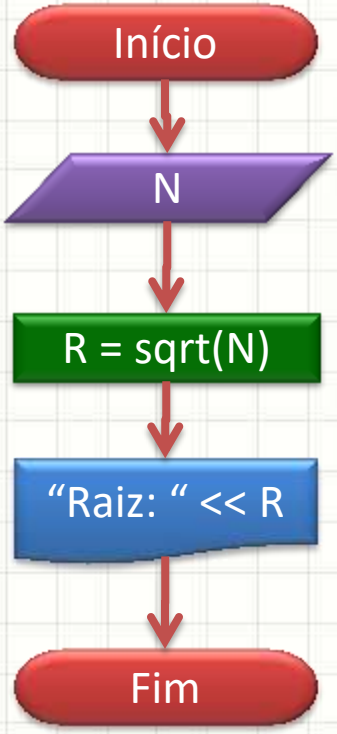
```
#include <math.h>
#include <iostream>
using namespace std;

main()
{
    float N,R;
    cout << "Digite um número positivo: ";
    cin >> N;
    R = sqrt(N);
    cout << "Raiz: " << R;
}
```

Não seria legal poder repetir a pergunta?

E se o usuário digitar um número negativo?

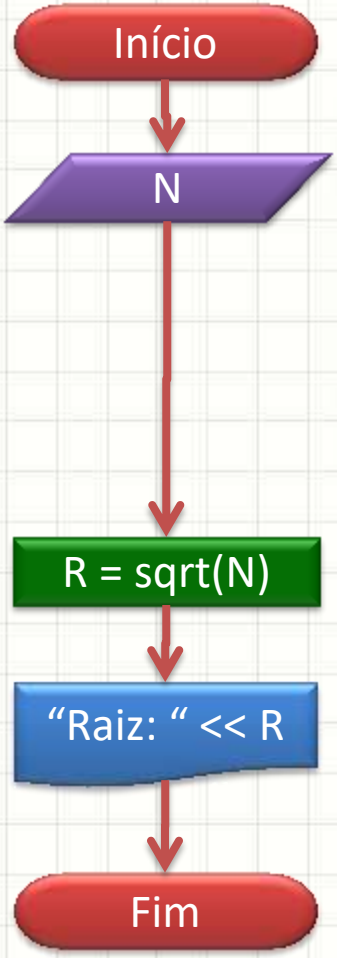
Repetição com Do~While

```
#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

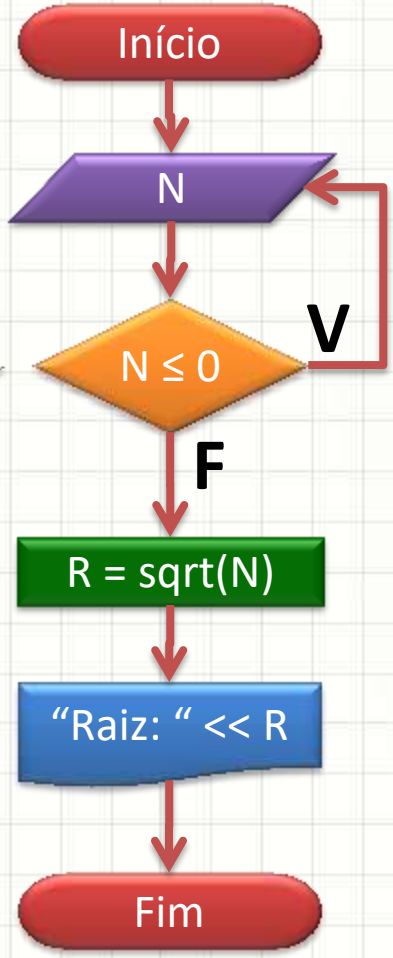
```
#include <math.h>
#include <iostream>
using namespace std;

main()
{
    float N,R;
    cout << “Digite no. > 0: ”;
    cin >> N;
    R = sqrt(N);
    cout << “Raiz: ” << R;
}
```



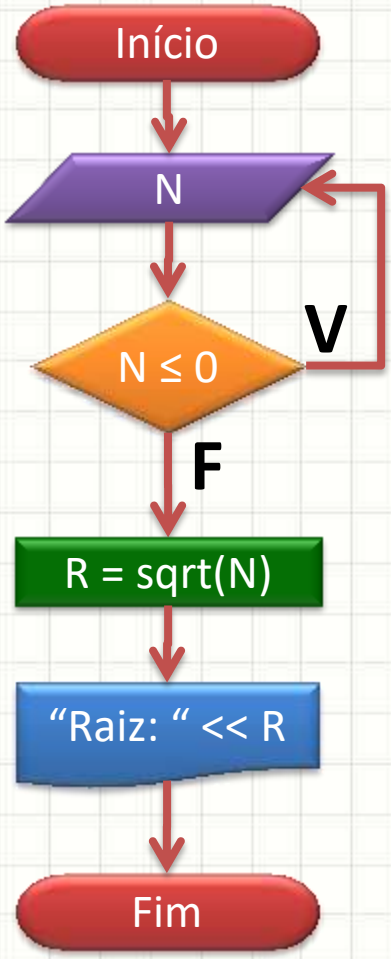
Repetição com Do~While

```
#include <math.h>
#include <iostream>
using nar

main()
{
    float
    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

Comparação – Digitem!

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

    cout << “Res.: ” << N;
}
```

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

    cout << “Res.: ” << N;
}
```

EXERCÍCIO

B.2) 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
- Dependendo da situação, cada uma delas é mais apropriada!
- **TAREFA: Atividade Aula 11!**

THE END!

PERGUNTAS?