



LÓGICA DE PROGRAMAÇÃO PARA ENGENHARIA

ESTRUTURAS DE REPETIÇÃO SIMPLES

Prof. Dr. Daniel Caetano

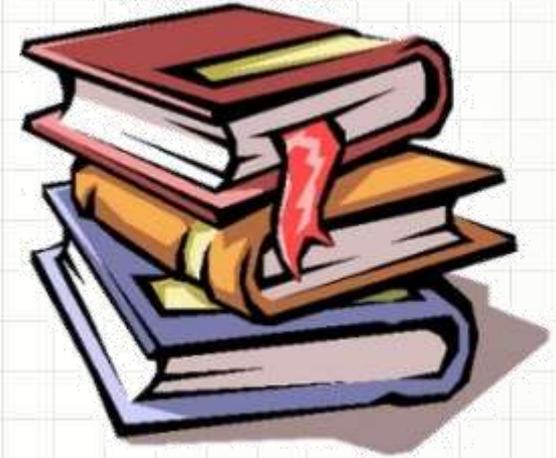
2012 - 2

Objetivos

- Entender o que é uma estrutura de repetição
- Compreender como implementar as repetições
- Capacitar para a criação de algoritmos que envolvam repetição
- **PARA CASA**
 - Lista de Exercícios 2!



Material de Estudo



Material

Acesso ao Material

Notas de Aula

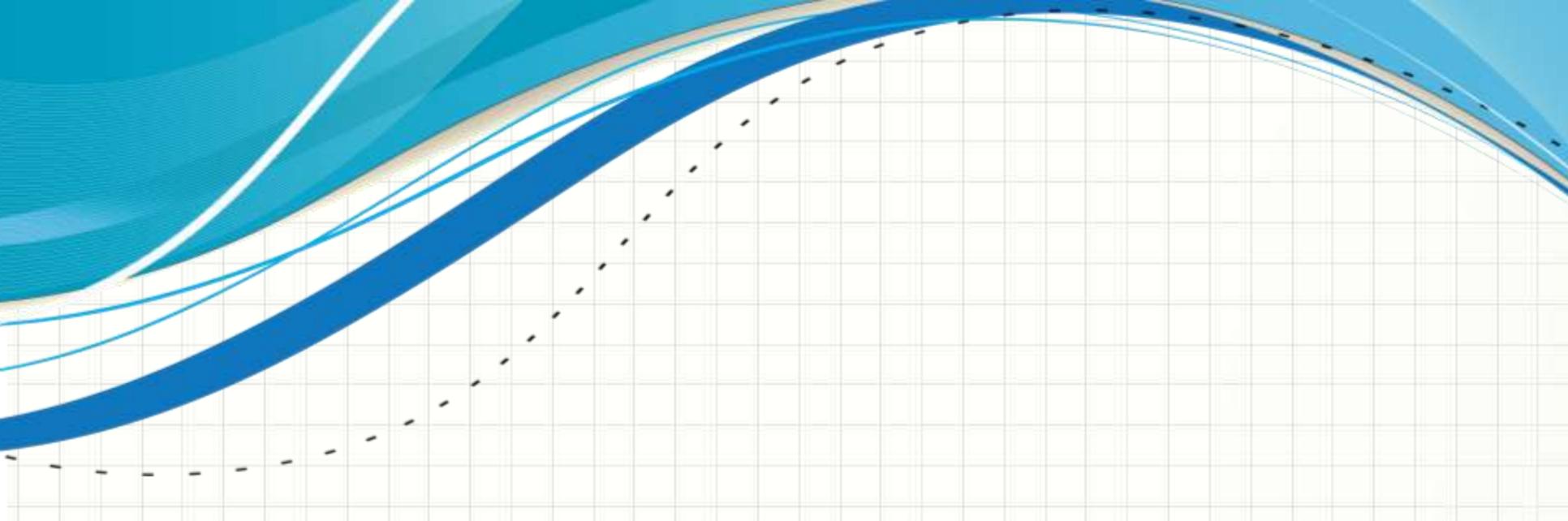
<http://www.caetano.eng.br/>
(Aula 12)

Apresentação

<http://www.caetano.eng.br/>
(Aula 12) – PARCIAL / COMPLETO

Material Didático

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



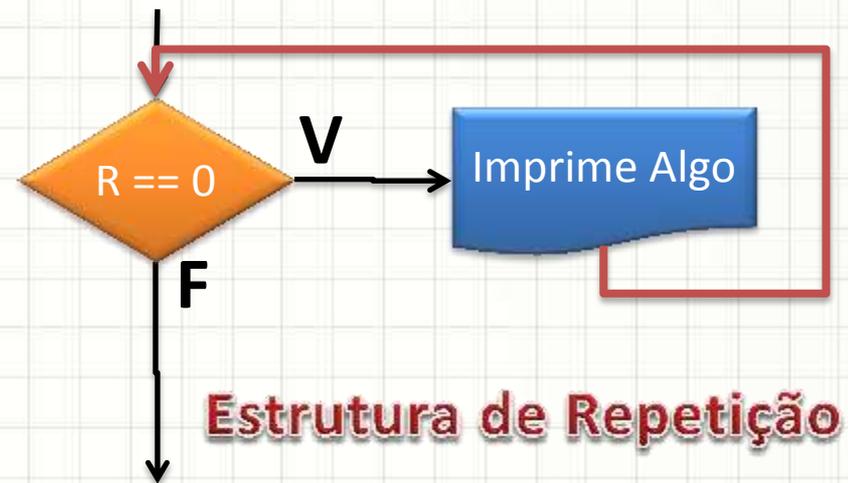
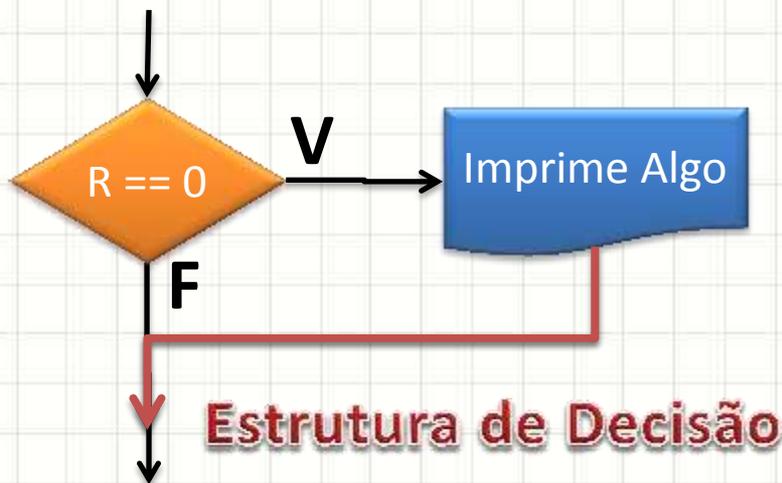
O QUE É ESTRUTURA DE REPETIÇÃO SIMPLES?

O que são Estruturas de Repetição?

- Repetir continuamente um código
 - Solicitação de entradas do usuário
 - Procedimentos repetitivos
 - Esperar que alguma coisa ocorra
- **MUITO** usadas!

O que são Estruturas de Repetição?

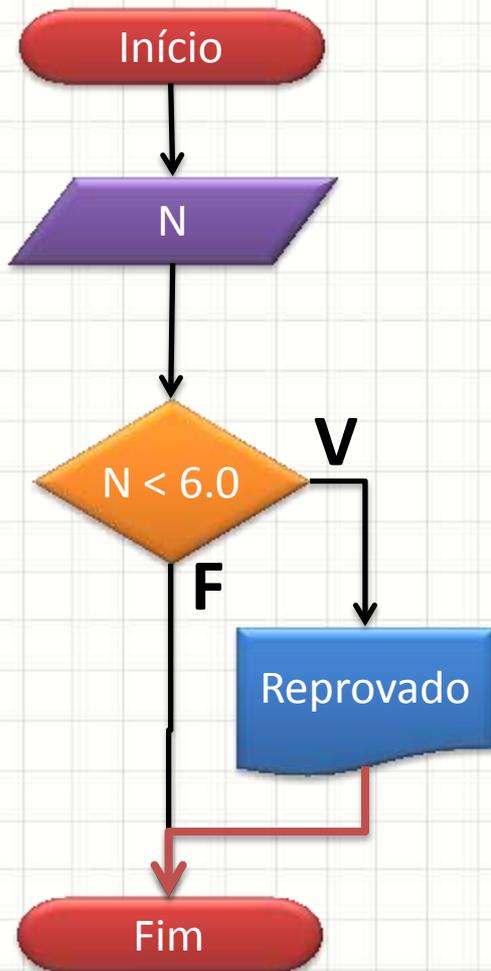
- Estrutura de Decisão: **se** executo um código
- Estrutura de Repetição é parecida...
 - Decidir até quando um código será executado



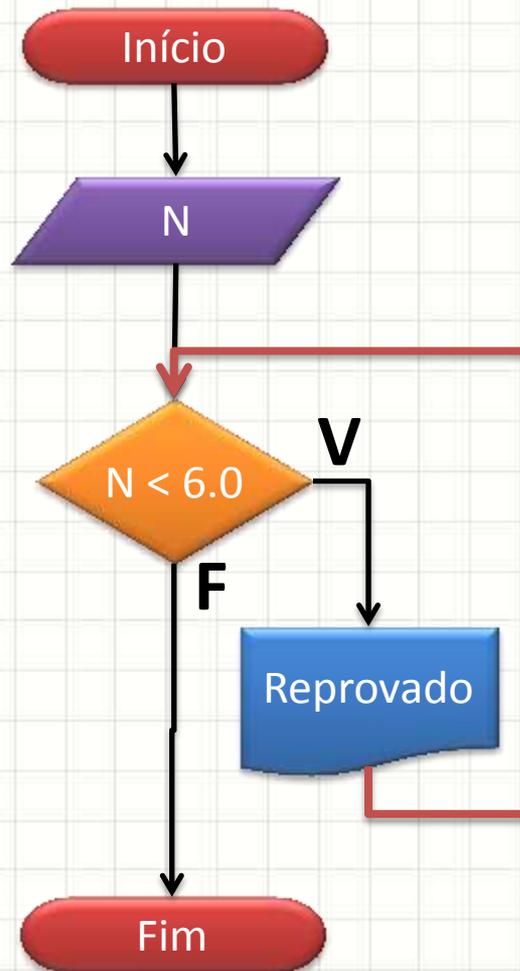
- Diferença: para onde vai a execução depois?

Repetição Simples na Prática

Estrutura de
Decisão



Estrutura de
Repetição



- Repetição: decisão do tipo **“enquanto isso for verdadeiro, continue repetindo!”**
- O que ocorre no código ao lado?

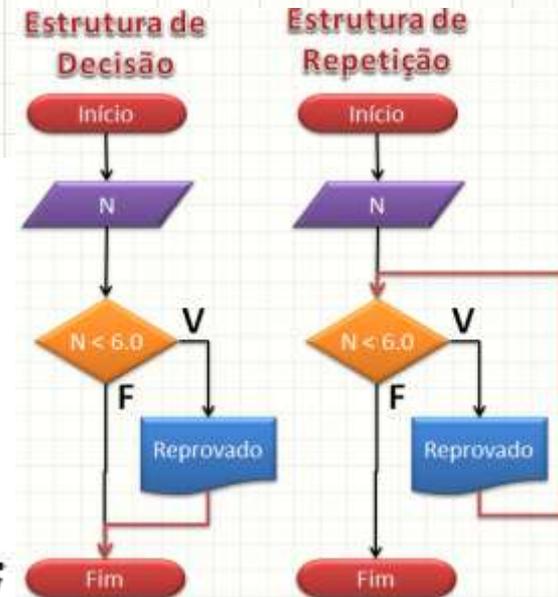
Repetição Simples na Prática

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

```
    float N;
```

```
    cout << "Digite a nota: ";
    cin >> N;
```

```
    if ( N < 6.0 ) {
        cout << "Aluno Reprovado" << endl;
    }
}
```



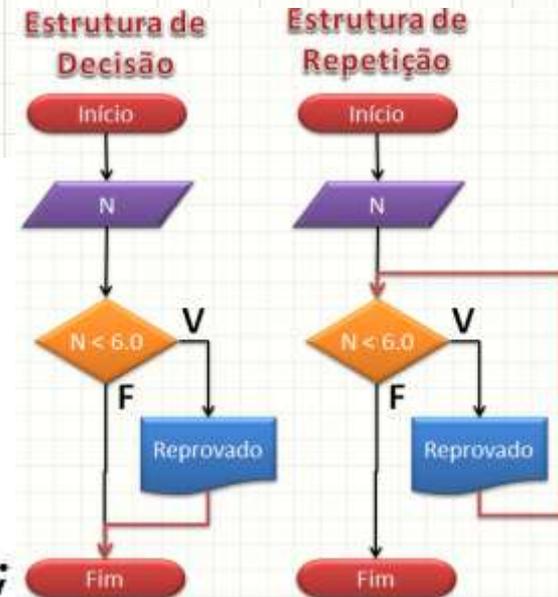
Repetição Simples na Prática

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

```
    float N;
```

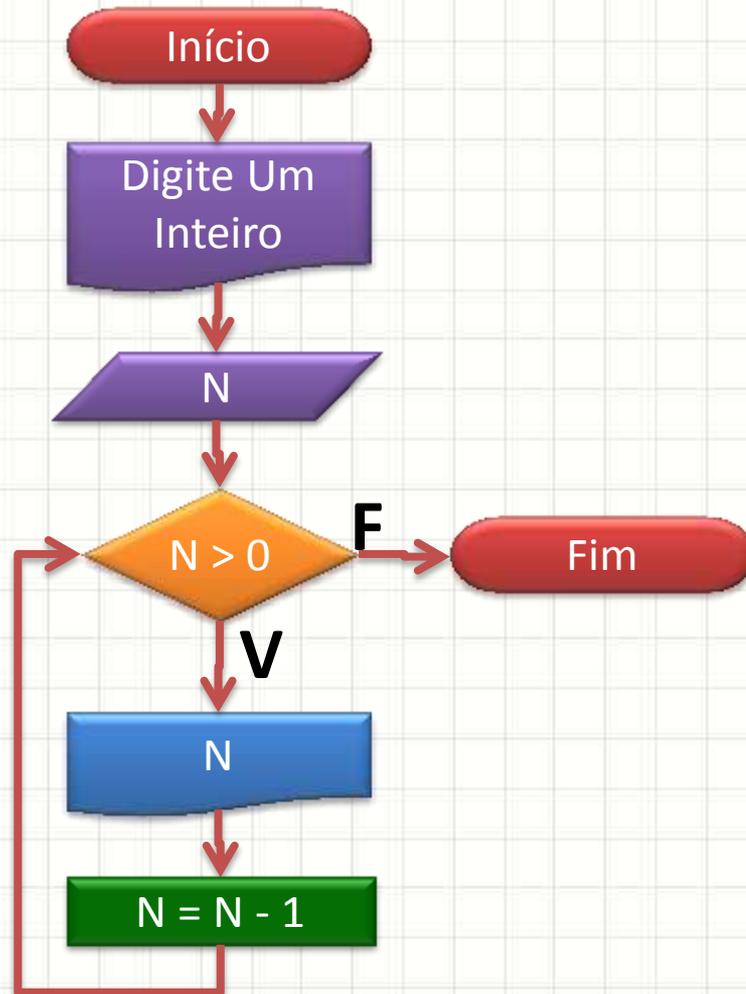
```
    cout << "Digite a nota: ";
    cin >> N;
```

```
    while ( N < 6.0 ) {
        cout << "Aluno Reprovado" << endl;
    }
}
```



Repetindo Código N Vezes

- Observe o fluxograma



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

```
int main(void) {
```

```
    int N;
```

```
    cout << "Digite um Inteiro: ";
```

```
    cin >> N;
```

```
    while ( N > 0 ) {
```

```
        cout << N << endl;
```

```
        N = N - 1;
```

```
    }
```

```
}
```

Repetindo Código N Vezes

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

```
int main(void) {
    int N;
    cout << "Digite um Inteiro: ";
    cin >> N;
    while ( N > 0 ) {
        cout << N << endl;
        N = N - 1;
    }
}
```

1. Digite Este Programa

2. Experimente executá-lo com diferentes valores. Exemplo: 5, 1, 0, -10

3. Experimente modificar o **while** para que a condição seja **N >= 0**.

4. O que aconteceu / mudou em cada caso?

Repetindo Código N Vezes

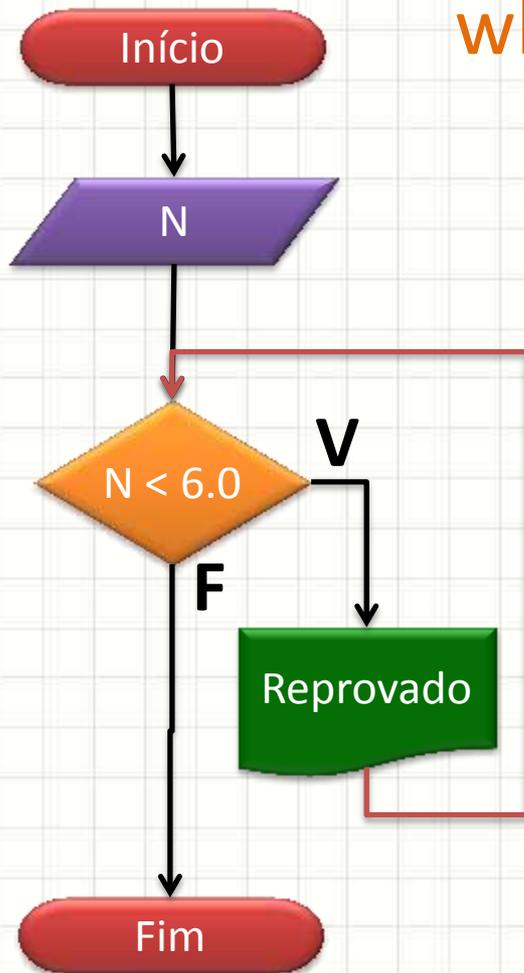
```
#include <iostream>
using namespace std;
```

```
int main(void) {
    int N;
    cout << "Digite um Inteiro: ";
    cin >> N;
    while ( N > 0 ) {
        cout << N << endl;
        N = N - 1;
    }
}
```

ATENÇÃO

1. Cuidado ao construir as proposições de repetição!
2. É responsabilidade do programador garantir que a condição de finalização seja atendida!
 - 2.1. Experimente modificar a atualização para $N = N + 1$!

Forma Geral do While



```
while ( proposição_lógica ) {
```

Executa enquanto a proposição for verdadeira

```
}
```

ATENÇÃO

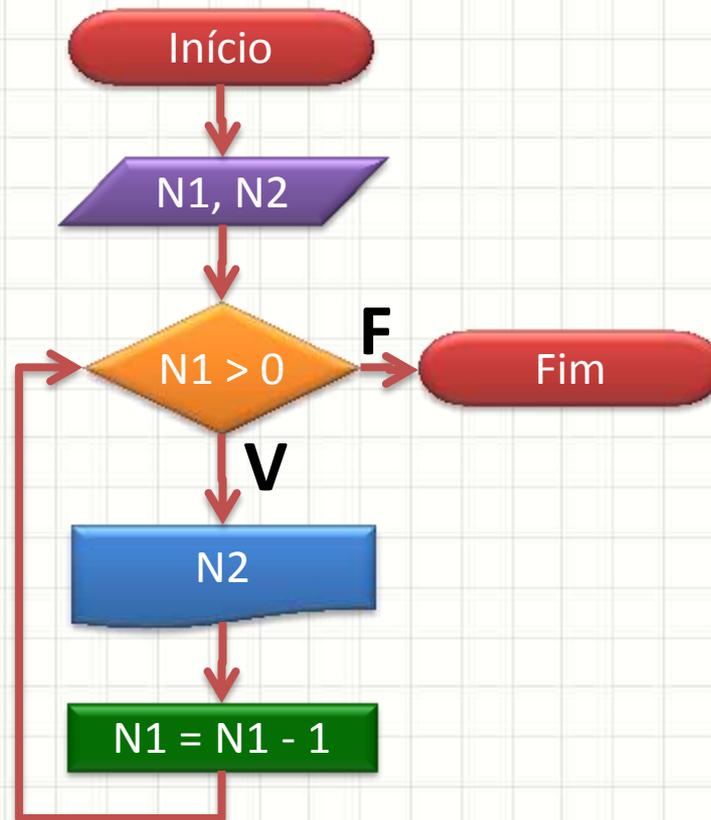
No WHILE não
existe ELSE!



ATIVIDADE

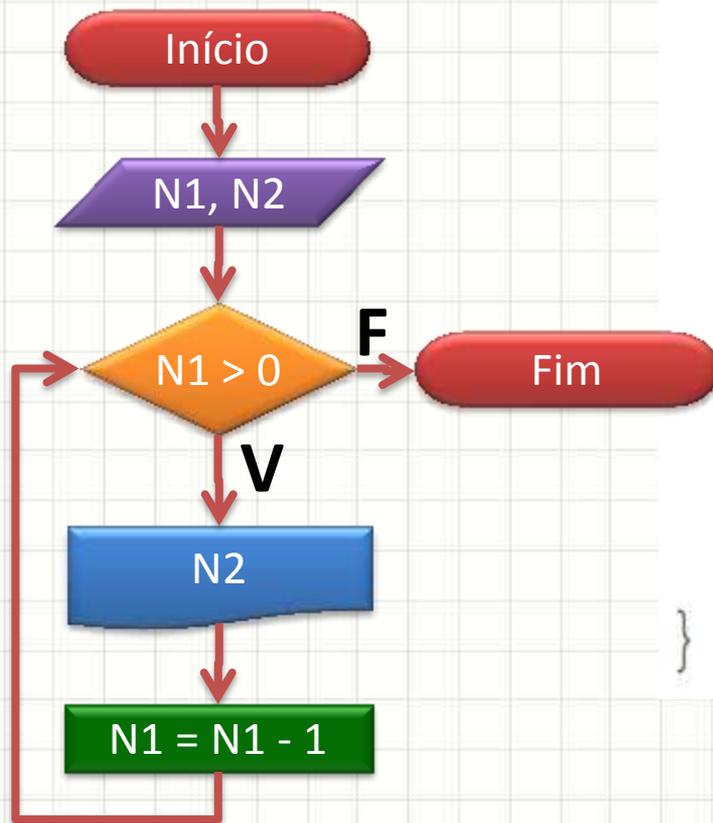
Exercício A

- Faça um programa que
 - a) Leia dois números N1 e N2
 - b) Imprima N1 vezes o valor de N2.



Exercício A

- Faça um programa que
 - a) Leia dois números N1 e N2
 - b) Imprima N1 vezes o valor de N2.



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

```
int N1, N2;
```

```
cout << "Digite um Número: ";
cin >> N1;
cout << "Digite outro Número: ";
cin >> N2;
```

```
while ( N1 > 0 ) {
    cout << N2 << endl;
    N1 = N1 - 1;
}
```

```
}
```

Exercício B

- Modifique o programa anterior para que...
 - a) Além de imprimir N2 em cada passo, imprima também o valor de N1, no seguinte formato:

N1: N2

Exemplo,

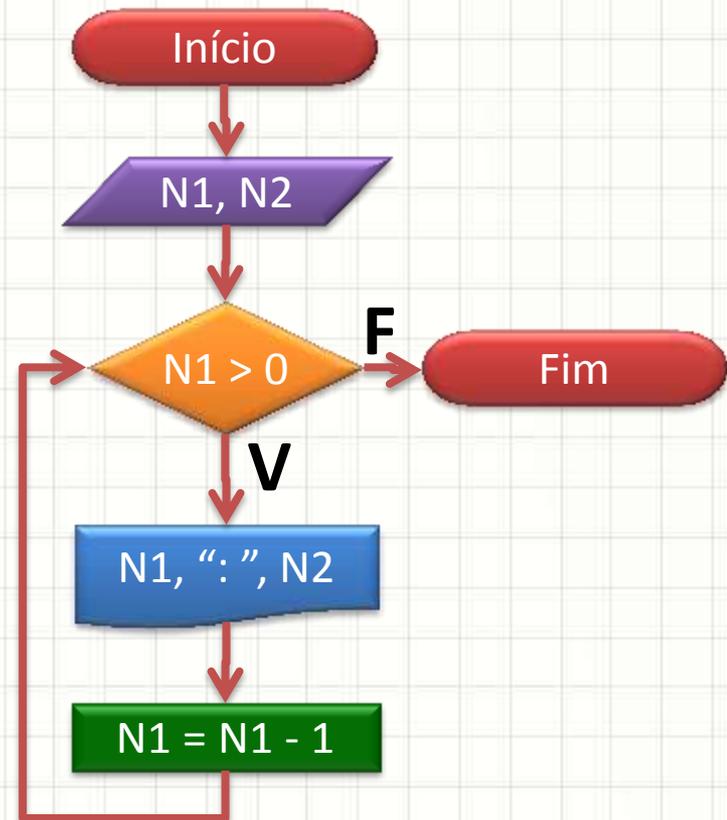
se N1=4 e N2=3

4: 3

3: 3

2: 3

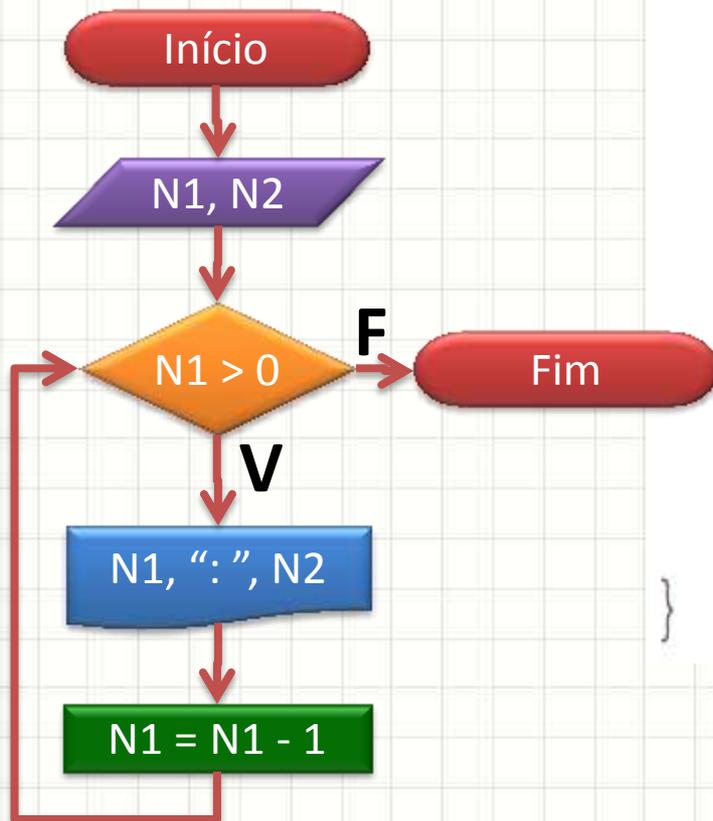
1: 3



Exercício B

- a) Além de imprimir N2 em cada passo, imprima também o valor de N1, no seguinte formato:

N1: N2



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

```
int N1, N2;
```

```
cout << "Digite um Número: ";
cin >> N1;
cout << "Digite outro Número: ";
cin >> N2;
```

```
while ( N1 > 0 ) {
    cout << N1 << " : " << N2 << endl;
    N1 = N1 - 1;
}
}
```

Exercício C

- Modifique o programa anterior para que...
 - a) Além de imprimir N2 em cada passo, imprima também o valor de N1 e o produto $R = N1 * N2$, no seguinte formato:

$$N1 * N2 = R$$

Exemplo,

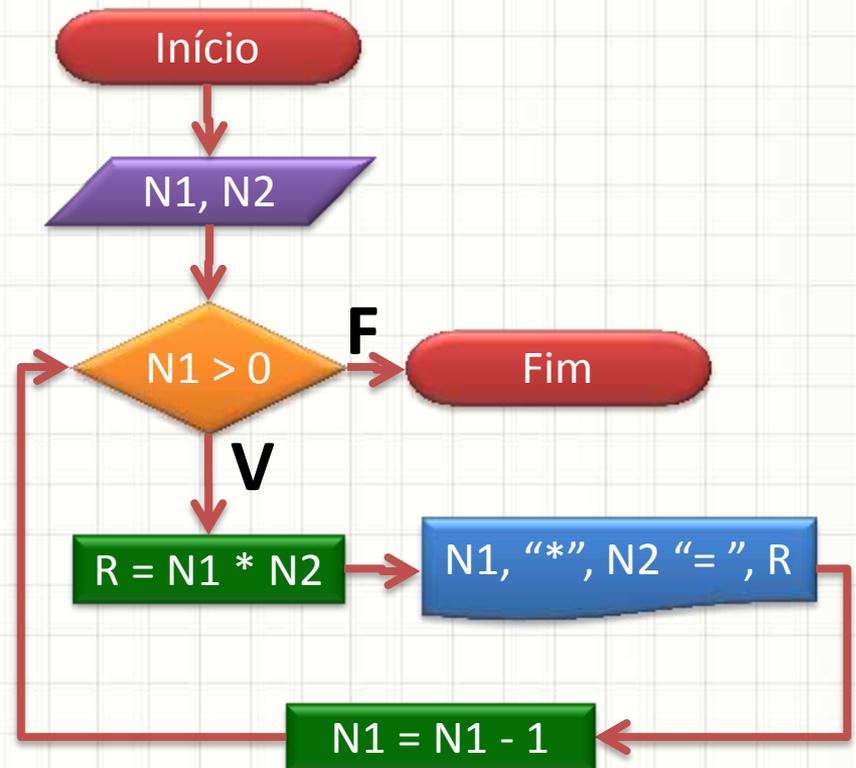
se $N1=4$ e $N2=3$

$$4 * 3 = 12$$

$$3 * 3 = 9$$

$$2 * 3 = 6$$

$$1 * 3 = 3$$



Exercício C

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

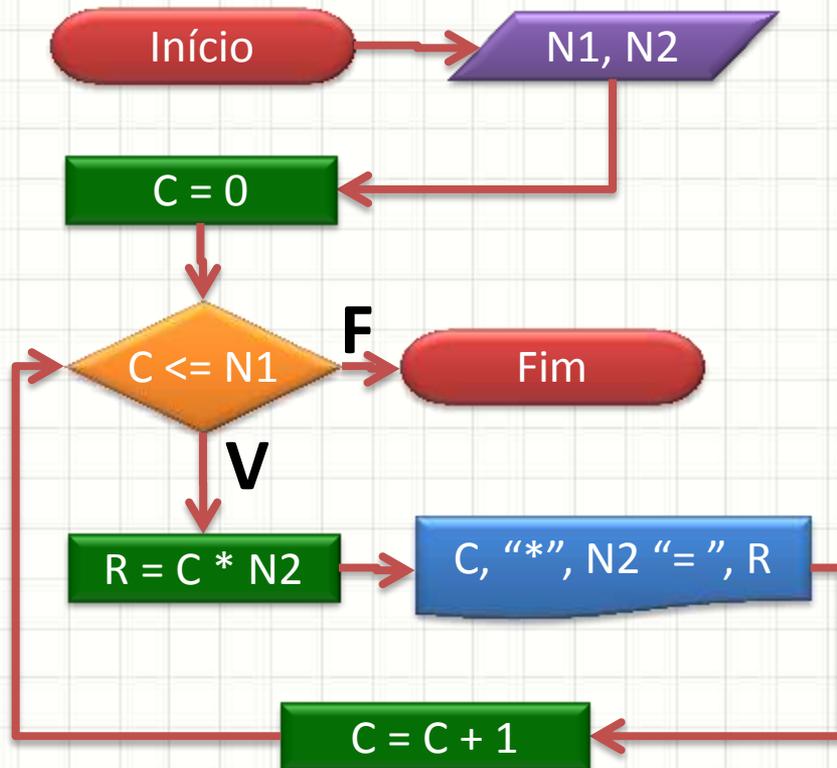
    int N1, N2, R;

    cout << "Digite um Número: ";
    cin >> N1;
    cout << "Digite outro Número: ";
    cin >> N2;

    while ( N1 > 0 ) {
        R = N1 * N2;
        cout << N1 << "*" << N2 << "= " << R << endl;
        N1 = N1 - 1;
    }
}
```

Exercício D

- Modifique o programa anterior para que...
 - a) Ele imprima até o valor 0
 - b) A contagem seja em ordem crescente



Exercício D

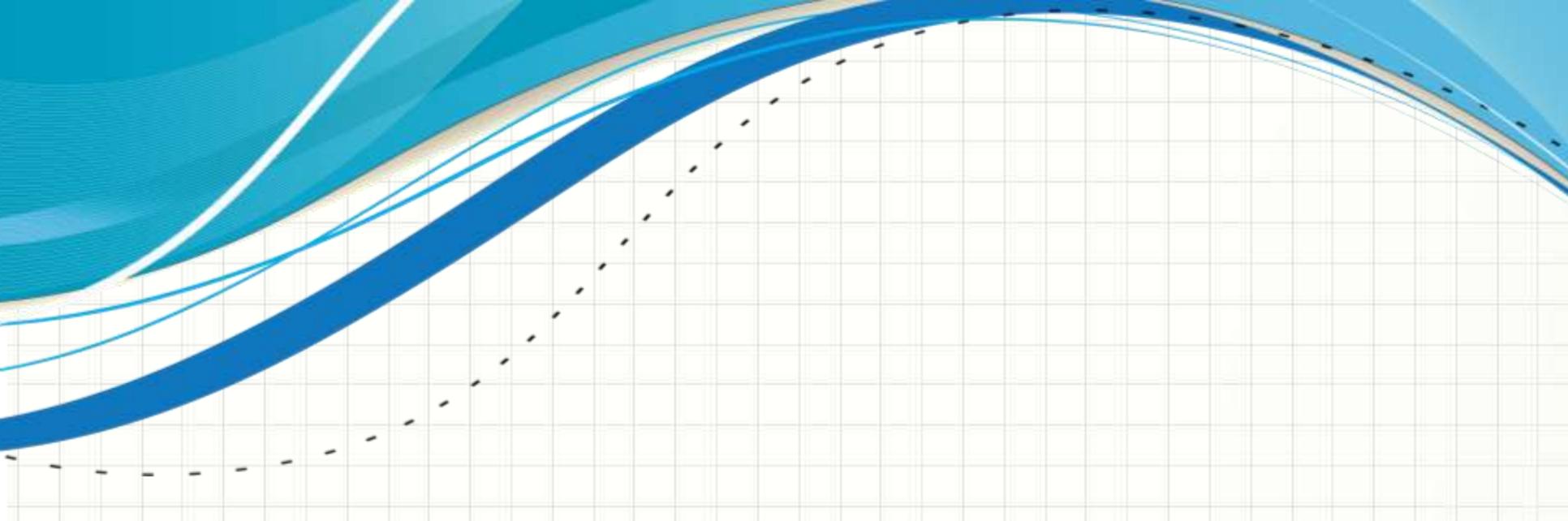
```
#include <iostream>
using namespace std;
int main(void) {

    int N1, N2, R, C;

    cout << "Digite um Número: ";
    cin >> N1;
    cout << "Digite outro Número: ";
    cin >> N2;

    C=0;

    while ( C <= N1 ) {
        R = C * N2;
        cout << C << "*" << N2 << "= " << R << endl;
        C = C + 1;
    }
}
```



CONCLUSÕES

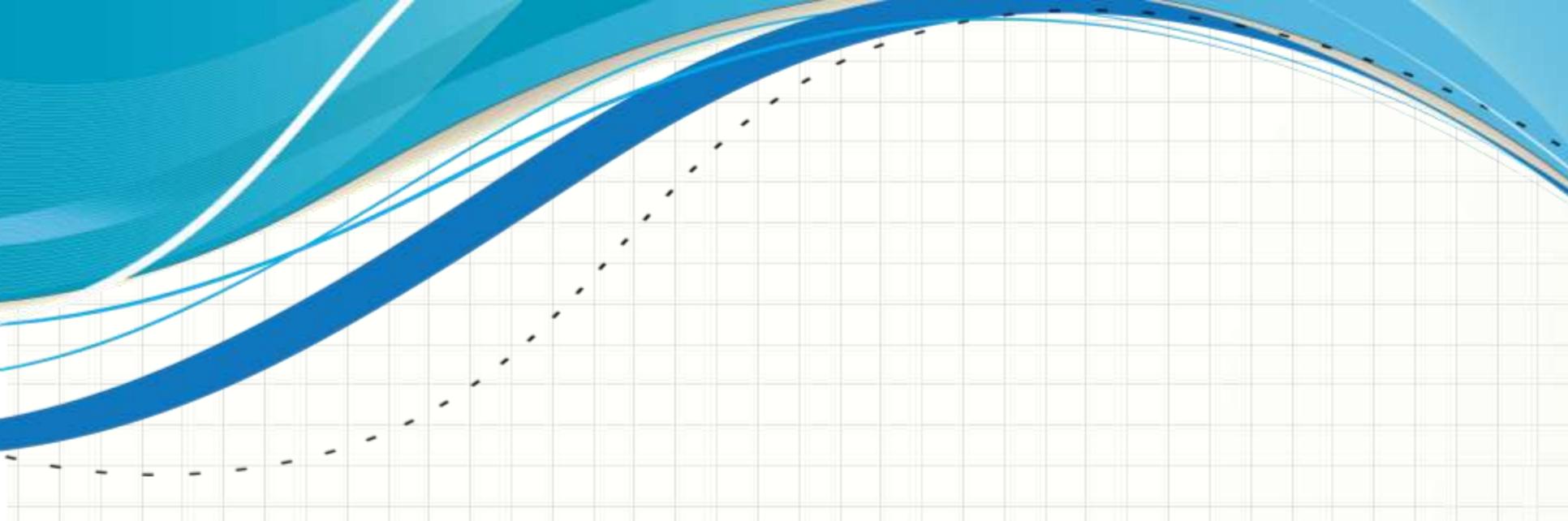
Resumo

- As estruturas de repetição aumentam muito a flexibilidade do computador
- Basicamente é uma estrutura de decisão que verifica “se continua repetindo”.
- Não deixe de praticar!
- **TAREFA!**
 - Lista de Exercícios 2!

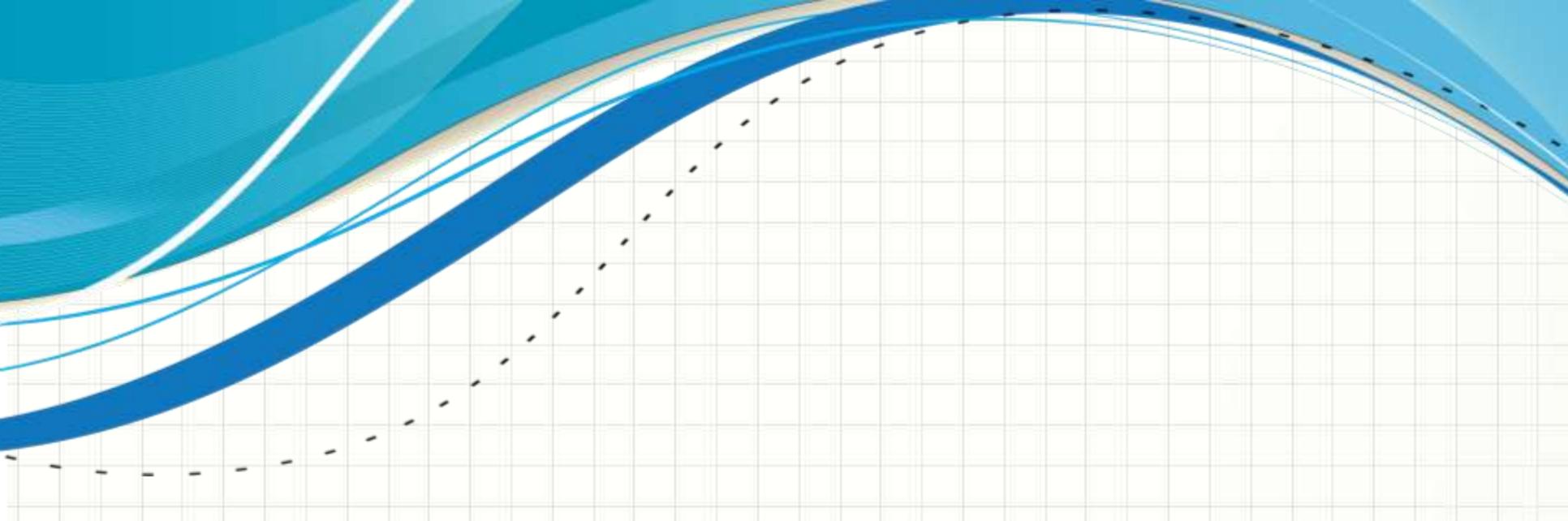
Próxima Aula



- Só existe um tipo de estrutura de repetição?
 - Não!
- Mas antes de ver outras...
 - Vamos treinar!



PERGUNTAS?



**BOM DESCANSO
A TODOS!**