



# Algoritmos e Programação de Computadores

## Repetição

Prof. Lucas Boaventura  
lucas.boaventura@unb.br





# Introdução

- Vimos que as condicionais são utilizadas para determinar múltiplos fluxos de execução no seu código
- Desta um programa pode determinar se um conjunto de instruções deve ou não ser executado
- Essa estrutura de repetição é básica na programação e chamada de **laço**





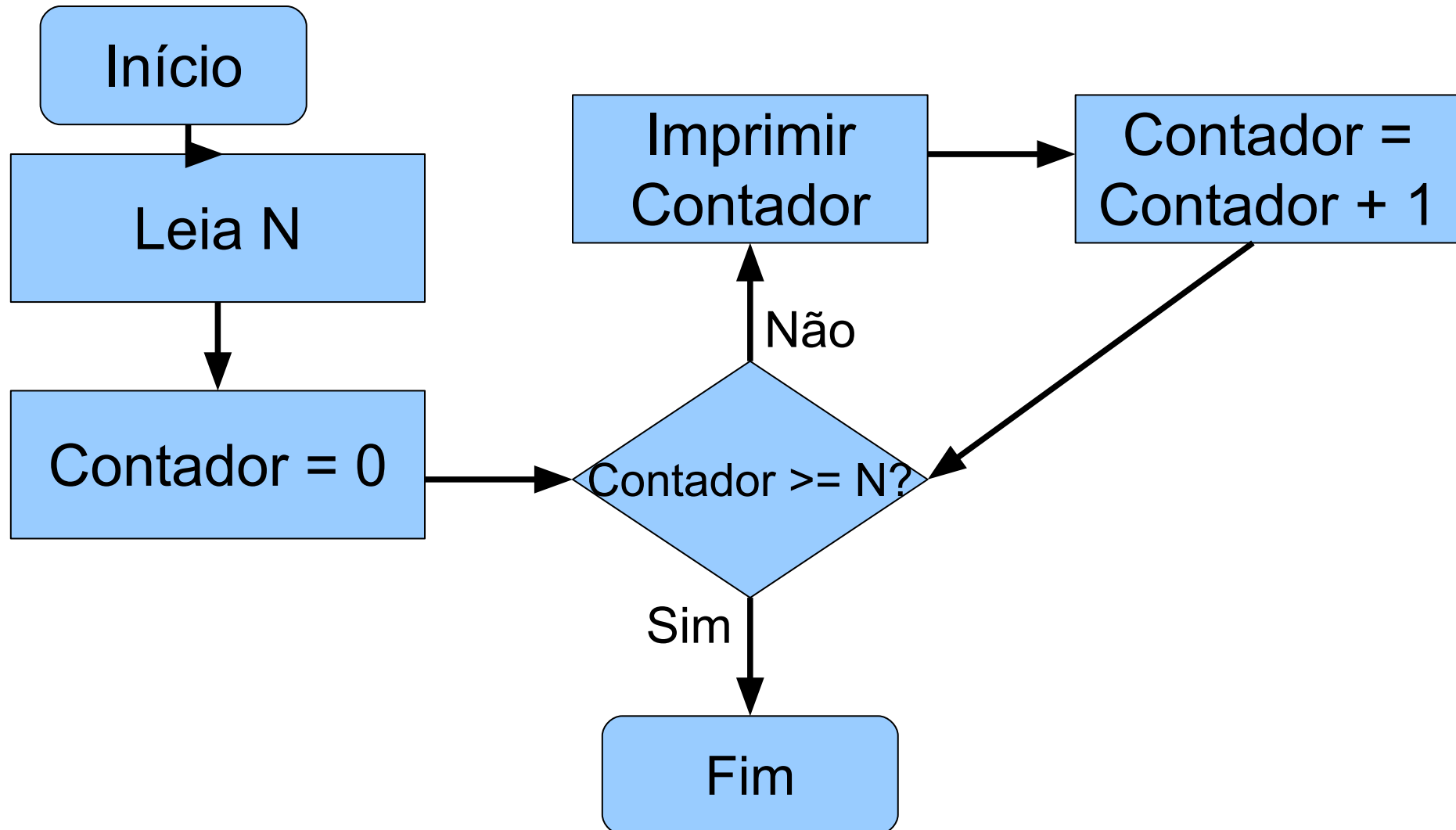
# Introdução

- Durante a construção de programas é comum repetirmos as instruções que serão precisas ser executadas
- Por exemplo: dado um número  $N$ , imprimir todos os números de 0 até  $N$



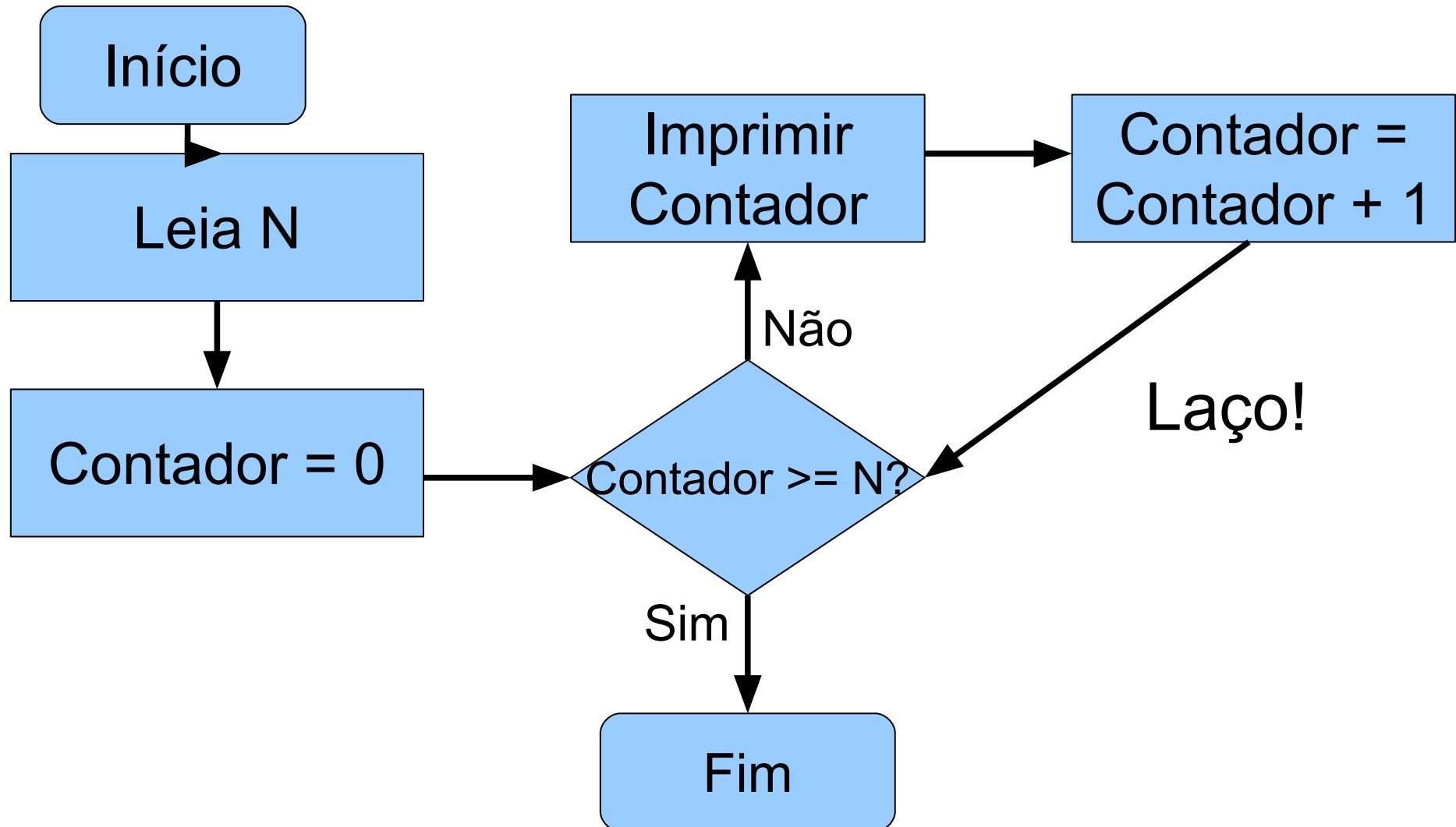


# Fluxograma





# Fluxograma





# While

- Para construir esse código em C, podemos utilizar a palavra-chave: **while** (enquanto)
- Sintaxe:
  - `while (condição) { ... }`
- As instruções `{ ... }` serão executadas enquanto a condição for verdadeira





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

Não se usa ;

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```







```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9 8
0x7f..9 c

Valor

LIXO
LIXO

Variável


Saída:





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    → int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	LIXO	N
0x7f..9 c	LIXO	

Saída:





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    → int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	LIXO	N
0x7f..9 c	0	contador

Saída:





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

➡ 

```
scanf("%d", &N);
```

```
while (contador <= N)
```

```
{
```

```
    printf("%d\n", contador);
```

```
    contador = contador + 1;
```

```
}
```

```
return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	0	contador

Saída:





```
#include <stdio.h>
```

```
int main()  
{  
    int N;  
    int contador = 0;  
  
    scanf("%d", &N);
```

```
    while (contador <= N)  
    {  
        printf("%d\n", contador);  
        contador = contador + 1;  
    }  
    return 0;  
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	0	contado r

Saída:





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        → printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9 8
0x7f..9 c

Valor

3
0

Variável

N
contador

Saída:

0





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        → contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9 8
0x7f..9 c

Valor

3
1

Variável

N
contador

Saída:

0





```
#include <stdio.h>
```

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);
```

```
    while (contador <= N)
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    return 0;
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	1	contado r

Saída:  
0







```
#include <stdio.h>
```

```
int main()
```

```
{
```

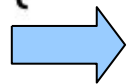
```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```



```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9

8

0x7f..9

c

Valor

3

1

Variável

N

contado  
r

Saída:

0

1





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

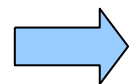
```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```



```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	2	contado r

Saída:

0

1





```
#include <stdio.h>
```

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);
```

```
    while (contador <= N)
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    return 0;
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	2	contado r

Saída:

0

1





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        → printf("%d\n", contador);  
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9 8
0x7f..9 c

Valor

3
2

Variável

N
contador

Saída:

0

1

2





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

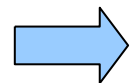
```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```



```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	3	contado r

Saída:

0

1

2





```
#include <stdio.h>
```

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);
```

```
    while (contador <= N)
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    return 0;
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	3	contador

Saída:

0  
1  
2



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        → printf("%d\n", contador);  
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9

8

0x7f..9

c

Valor

3

3

Variável

N

contado  
r

Saída:

0

1

2

3





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

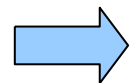
```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```



```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

0x7f..9

8

0x7f..9

c

Valor

3

4

Variável

N

contado  
r

Saída:

0

1

2

3







```
#include <stdio.h>
```

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);
```

```
    while (contador <= N)
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    return 0;
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9 8	3	N
0x7f..9 c	4	contado r

Saída:

0  
1  
2  
3



```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int N;
```

```
    int contador = 0;
```

```
    scanf("%d", &N);
```

```
    while (contador <= N)
```

```
    {
```

```
        printf("%d\n", contador);
```

```
        contador = contador + 1;
```

```
    }
```

```
    return 0;
```

```
}
```

Endereço  
de  
Memória

Valor

Variável

0x7f..9

8

3

N

0x7f..9

c

4

contado  
r

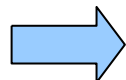
Saída:

0

1

2

3





# Do / While

- Também é possível usar a estrutura **do/while**
- Sintaxe:
  - `do { ... } while (condição)`
- As instruções `{ ... }` serão executadas enquanto a condição for verdadeira
- Qual seria a diferença para o **while**?





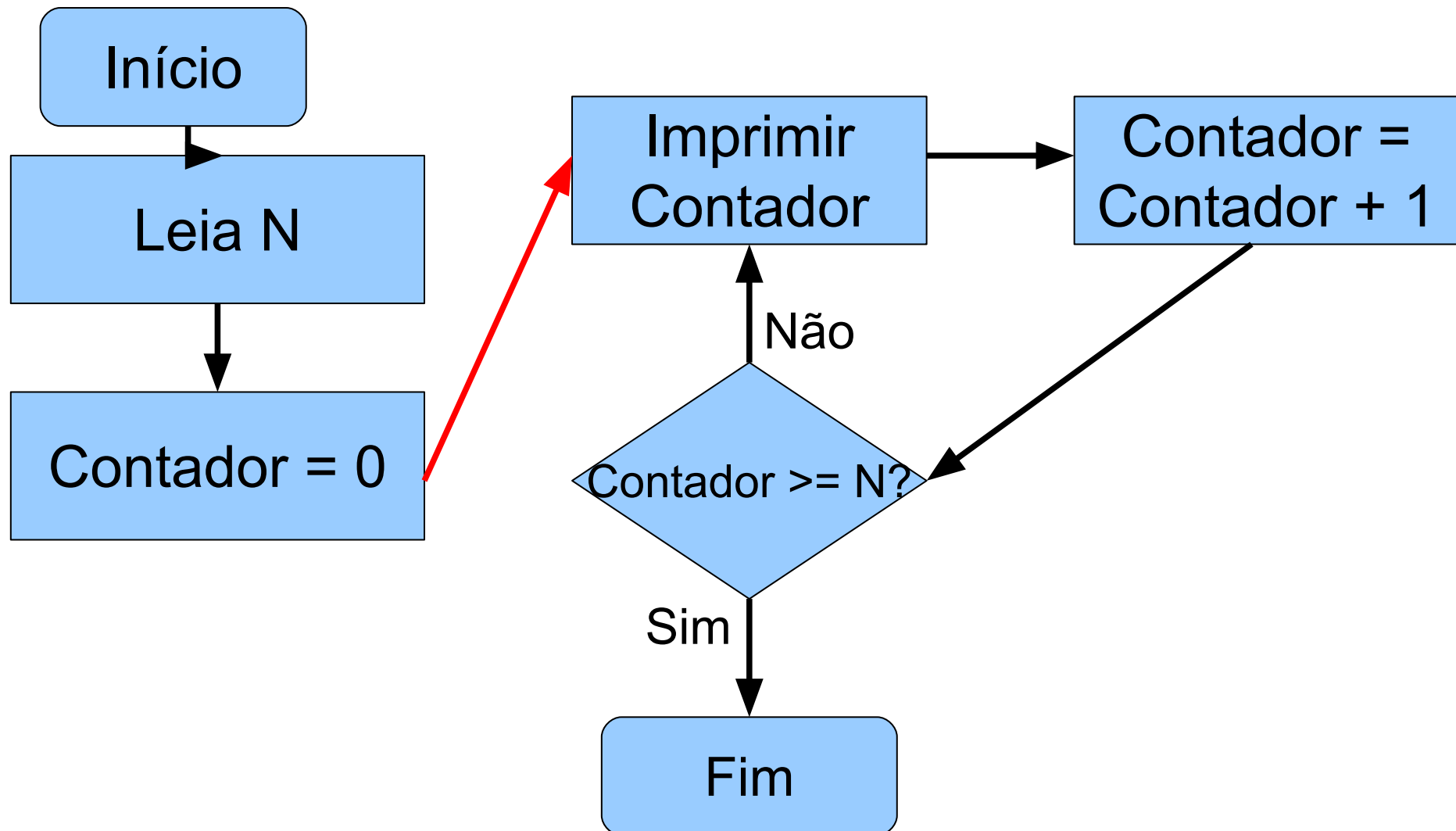
# Comparação while <-> do / while

- while (condição) { ... }
- Verifica a condição -> Executa instruções
  - do { ... } while (condição)
- Executa instruções -> Verifica condição
- O do/while executa pelo menos uma vez as instruções, mesmo que a (condição) seja sempre falsa!

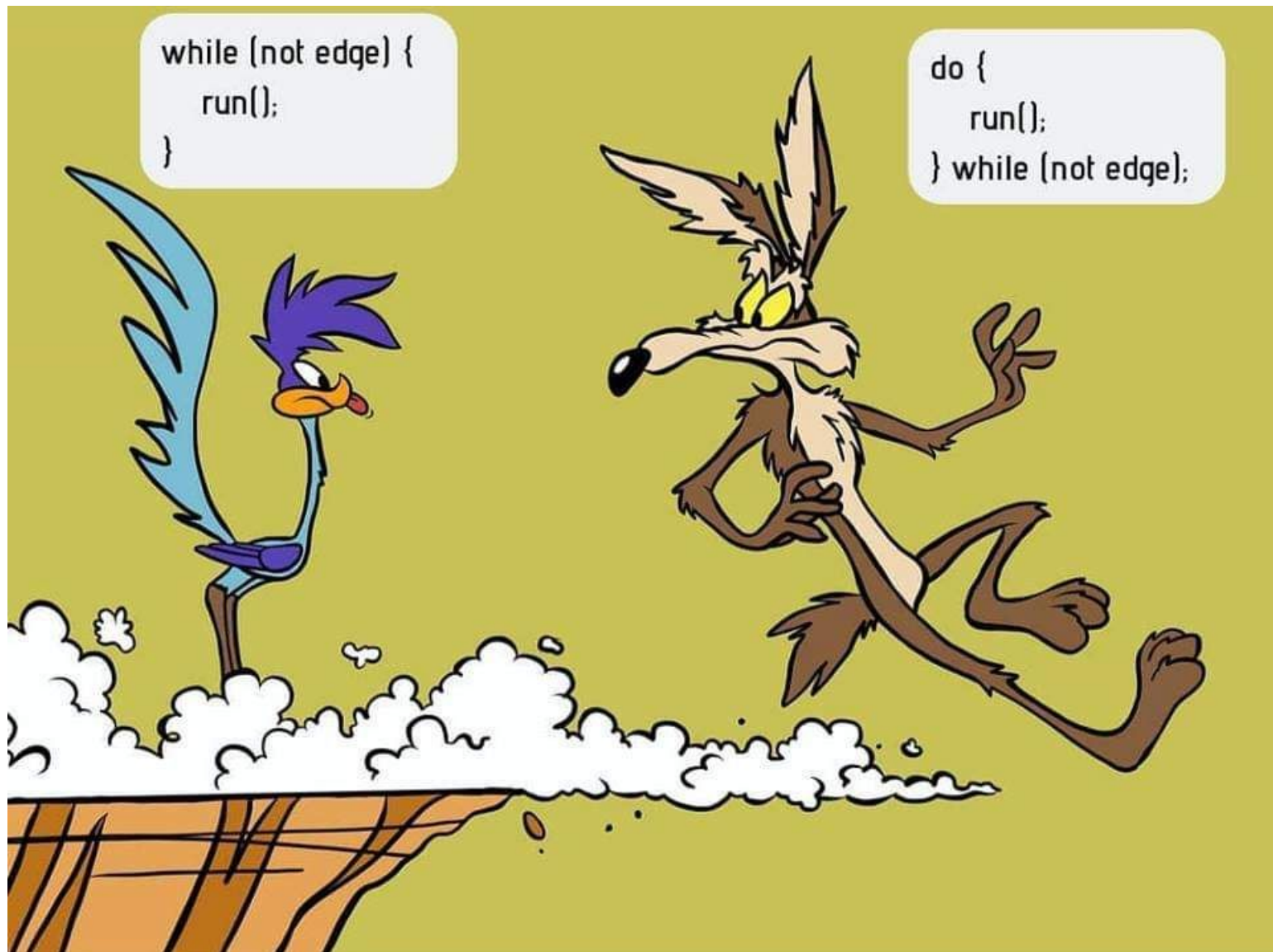




# Comparação while $\leftrightarrow$ do / while



# Comparação while <-> do / while





# Comparação while <=> do / while

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);

    do
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    while (contador <= N);

    return 0;
}
```



# Comparação while <=> do / while

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);

    do
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    while (contador <= N); Usa-se ;

    return 0;
}
```







# Comparação while <-> do / while

- Os dois códigos que mostramos anteriormente, possuem o mesmo comportamento com **while** e **do / while** para muitas entradas
- Nem sempre isso não é verdade

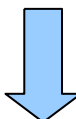




# Comparação while <=> do / while

```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);
    while (contador < N)
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    return 0;
}
```

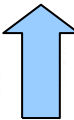


```
int main()
{
    int N;
    int contador = 0;

    scanf("%d", &N);

    do
    {
        printf("%d\n", contador);
        contador = contador + 1;
    }
    while (contador <= N);

    return 0;
}
```





# Comparação while <-> do / while

- Neste exemplo, mesmo com a modificação, a execução é semelhante
- Se o usuário digitar o número 0 (ou negativo) serão diferentes
- Digitando 0:
  - Código while: não imprime nada
  - Código do/while: imprime 0





# For

- Outra palavra chave muito importante é o **for** (para cada)
- Ele é especialmente utilizado para contadores, mas pode ser usado genericamente para vários laços complexos





- Veja o seguinte código:

Nota:  
Programadores  
começam a contar no 0  
normalmente...

```
#include <stdio.h>

int main()
{
    int i;

    i = 0;
    while (i < 10)
    {
        printf("%d\n", i);

        i++;
    }

    return 0;
}
```





```
#include <stdio.h>
```

```
int main()
```

```
{
```

```
    int i;
```

```
    i = 0;
```

```
    while (i < 10)
```

```
{
```

```
        printf("%d\n", i);
```

```
        i++;
```

```
    }
```

```
    return 0;
```

```
}
```

- Esse código pode ser dividido nas partes:

- Inicialização

- Condição

- Laço

- Incremento





# For

- Essa estrutura de repetição se repete diversas vezes, que pode ser simplificado utilizando um laço **for**

```
int main()  
{  
    for (int i = 0; i < 10; i++)  
    {  
        printf("%d\n", i);  
    }  
    return 0;  
}
```





# For

Inicialização

Condição

Incremento

```
int main()  
{  
    for (int i = 0; i < 10; i++)  
    {  
        printf("%d\n", i);  
    }  
    return 0;  
}
```

Laço







# For

- No **for**, é opcional fazer a declaração da variável no laço, ou utilizar uma que já existe

```
int main()
{
    for (int i = 0; i < 10; i++)
    {
        printf("%d\n", i);
    }
    return 0;
}
```

```
int main()
{
    int i;
    for (i = 0; i < 10; i++)
    {
        printf("%d\n", i);
    }
    return 0;
}
```



# For

- Podemos utilizar alguns membros vazios
- Para ser vazio, continuamos usando “;” dentro do for, mas não colocamos nada

```
int main()
{
    int i;

    scanf("%d", &i);
    for ( ; i < 10; i++)
    {
        printf("%d\n", i);
    }
    return 0;
}
```





# For

- Também é possível utilizar múltiplos campos, separando-os por vírgulas:

```
int main()
{
    for (int i = 0, j = 10 ; i < 10; i++, j --)
    {
        printf("%d - %d\n", i, j);
    }
    return 0;
}
```

Saída do código:

0 - 10  
1 - 9  
...  
9 - 1





# Laço Infinito

- O loop infinito é um problema causado quando a condição de um loop nunca é falsa

```
int main()
{
    int i = 0;
    int j = 5;

    while (i < j)
    {
        printf("i e' menor do que j\n");
    }
    return 0;
}
```





# Laço Infinito

- Neste exemplo, o programa irá imprimir indefinidamente a mensagem na tela e nunca irá parar por conta própria (return 0 na main)





# Aninhamento

- Os loops também pode ser aninhados
- Os exemplos que vimos em sala são simples, mas códigos complexos podem demandar vários loops dentro do outro!





# Aninhamento

```
int main()
{
    for (int i = 1; i < 10; i++)
    {
        for (int j = 0; j < i; j++)
        {
            printf("%d ", j);
        }
        printf("\n");
    }
    return 0;
}
```





# Aninhamento

- Saída do código anterior: 0  
0 1  
0 1 2  
0 1 2 3  
0 1 2 3 4  
0 1 2 3 4 5  
0 1 2 3 4 5 6  
0 1 2 3 4 5 6 7  
0 1 2 3 4 5 6 7 8







# Exercício

- Faça um código que leia N duplas de número e imprime a soma deles. A primeira linha contém o número de casos

3

1 2

2 3

3 3





# Exercício

```
#include <stdio.h>

int main()
{
    int N;
    int x, y;

    scanf("%d", &N);

    for (int i = 0; i < N; i++)
    {
        scanf("%d %d", &x, &y);
        printf("%d\n", x + y);
    }
    return 0;
}
```





# Dúvidas?

- [lucas.boaventura@unb.br](mailto:lucas.boaventura@unb.br)

