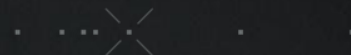




FIAP



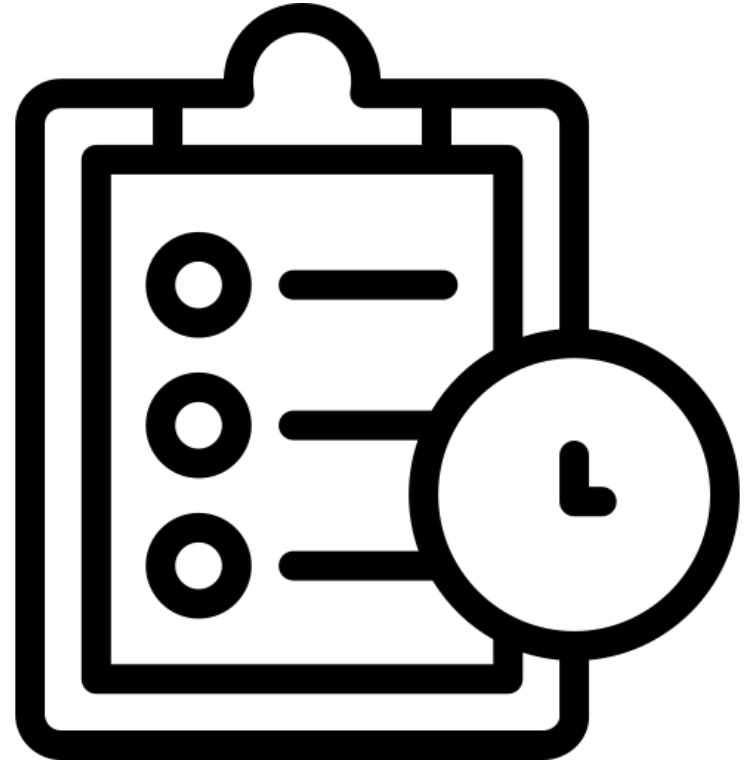
Engenharia de Software

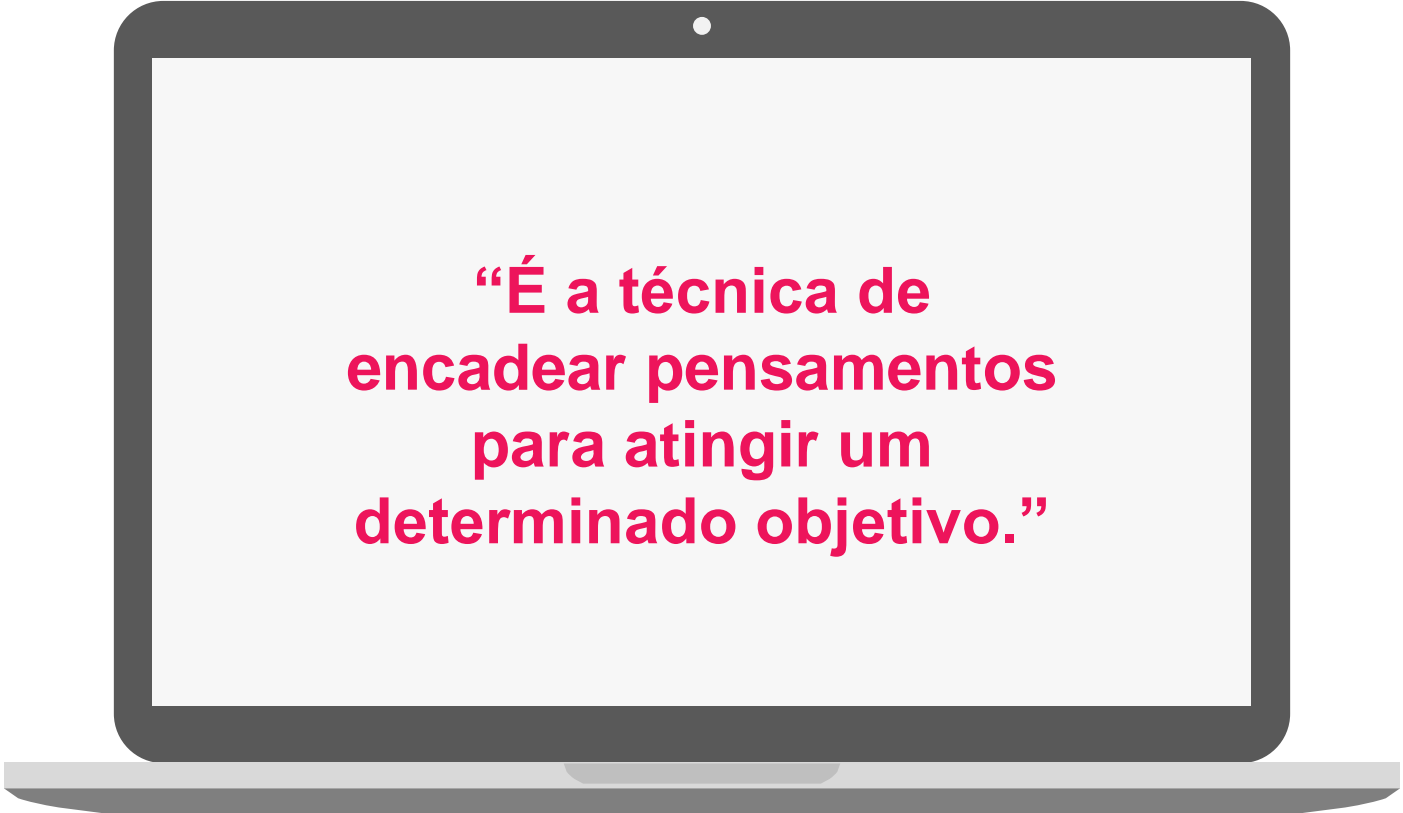
EDGE COMPUTING & COMPUTER SYSTEMS

03 – Lógica de Programação

Agenda

- Lógica de Programação;
- Fluxogramas;
- Variáveis e Constantes;
- Operadores Aritméticos;
- Operadores Relacionais;
- Estruturas Condicionais;
- Estruturas de Repetição;
- Laboratório;
- Exercícios;





“É a técnica de encadear pensamentos para atingir um determinado objetivo.”

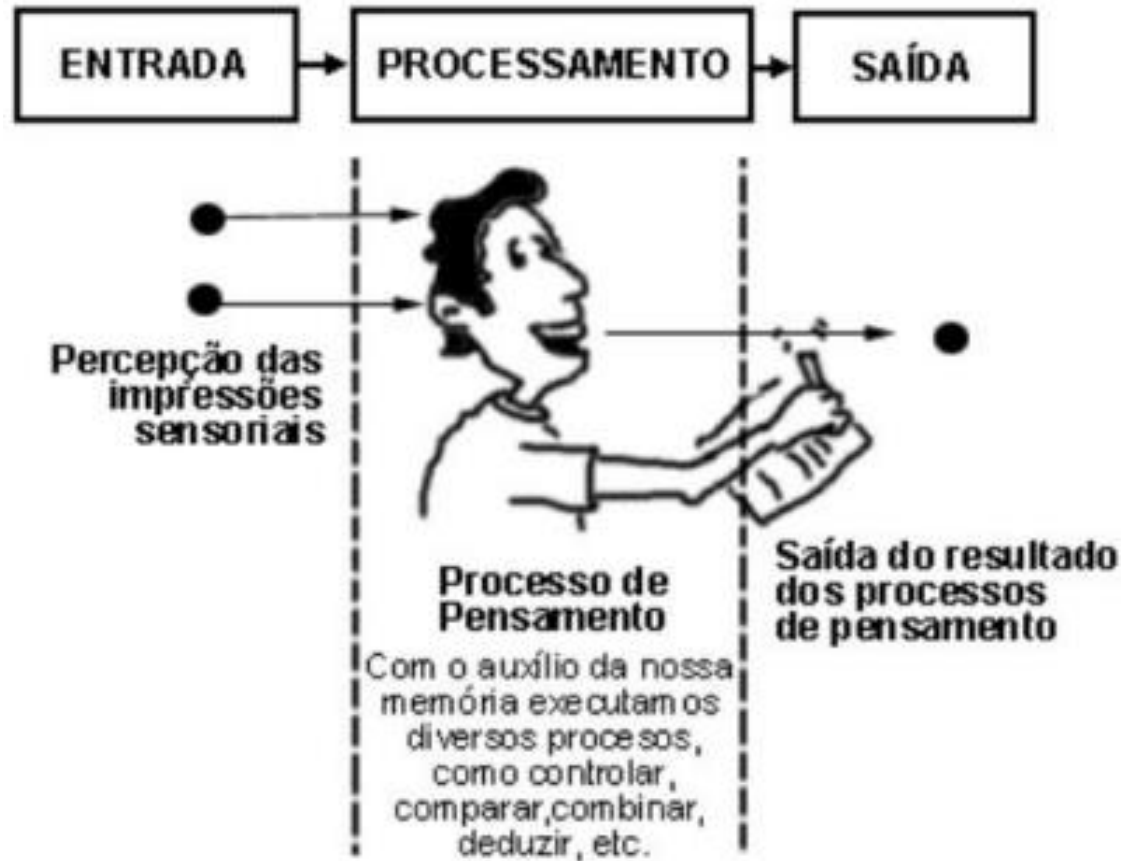
Lógica de Programação

Quais são os passos necessários para se trocar uma lâmpada queimada?

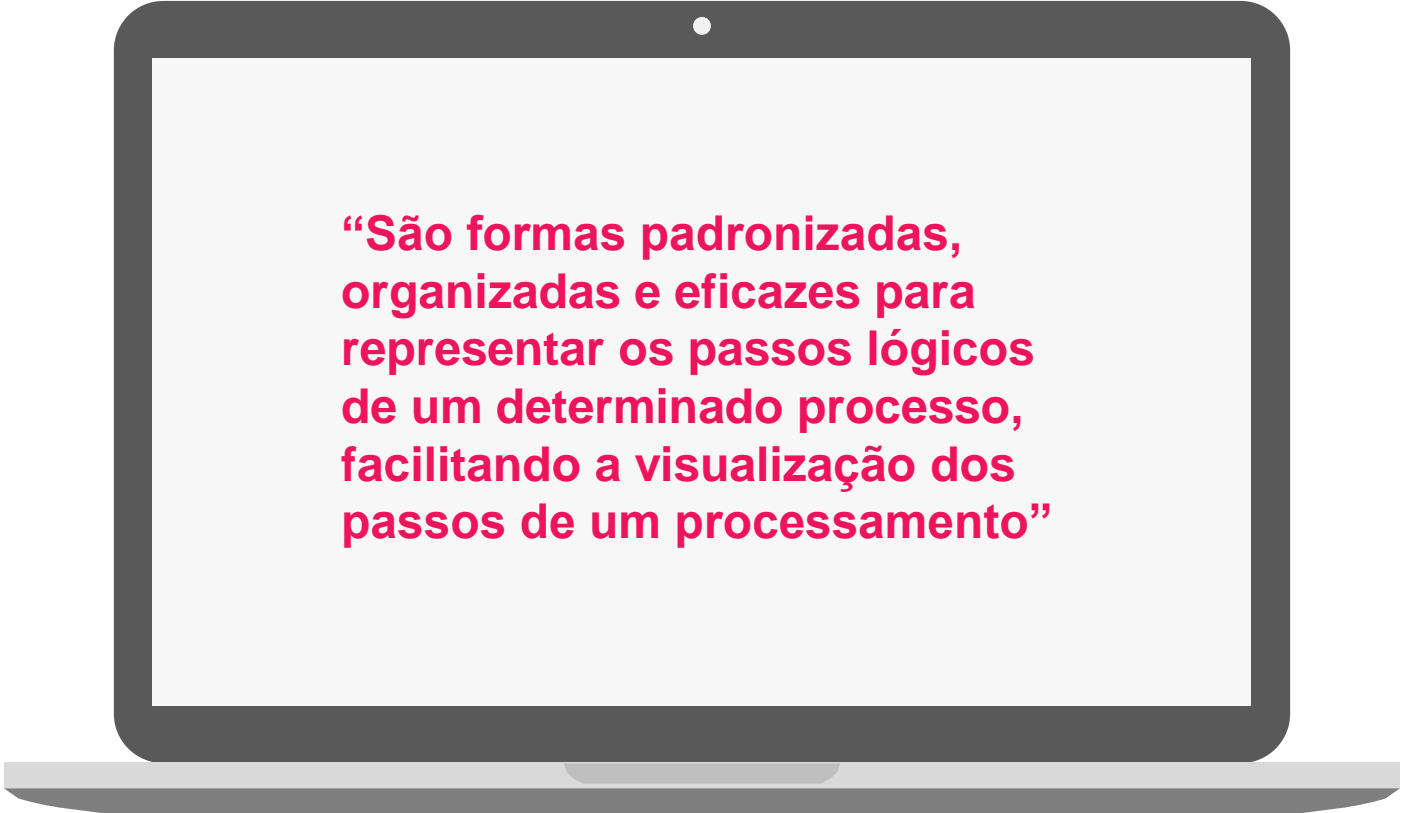


- 1 Comprar uma lâmpada nova.
- 2 Pegar uma escada.
- 3 Desligar o interruptor.
- 4 Pegar a lâmpada nova.
- 5 Subir na escada.
- 6 Remover a lâmpada queimada.
- 7 Instalar a lâmpada nova.
- 8 Descer da escada.
- 9 Descartar a lâmpada queimada.
- 10 Acionar o interruptor.

Processo básico de um algoritmo



Fluxogramas



“São formas padronizadas, organizadas e eficazes para representar os passos lógicos de um determinado processo, facilitando a visualização dos passos de um processamento”

Fluxogramas

 São basicamente diagramas de blocos

Início ou fim

Teste
condicional

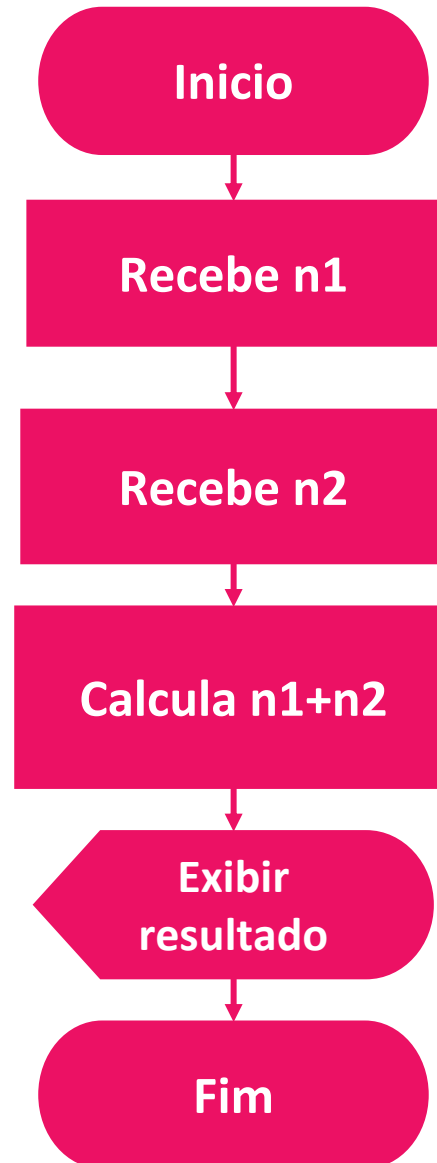
Processamento geral. Neste
curso, será usado
também para entrada ou
saída de dados

Exibir

Fluxogramas



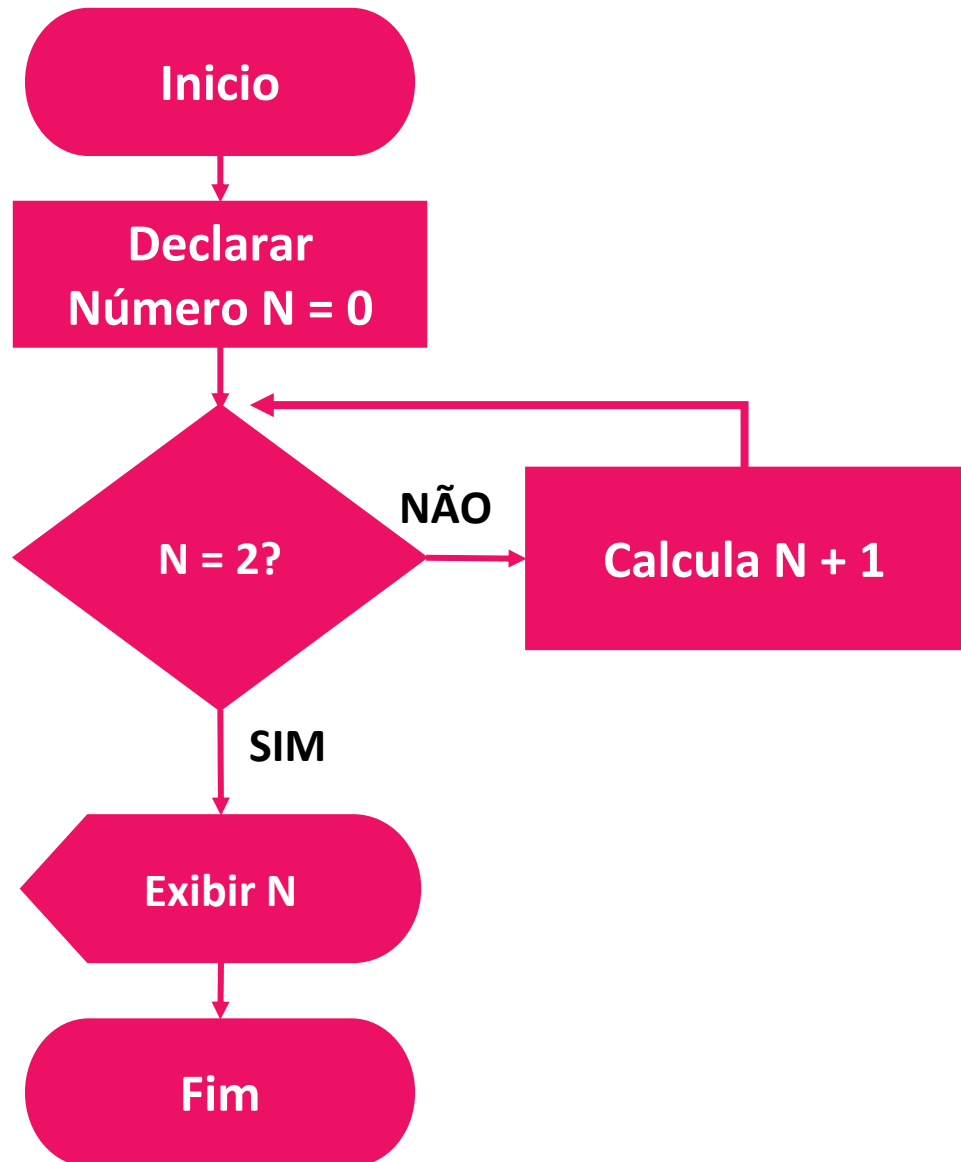
Soma de dois números



Fluxogramas



Contar até dois



Variáveis e Constantes



Constantes

Representa um **valor fixo** durante toda a execução do programa.



Variáveis

Representa um **valor que pode ser alterado** durante a execução do programa.

```
/*  
 * @brief Programa que calcula a média  
 * entre 5 números (n1, n2, n3, n4 e n5)  
 */  
int media(int n1, int n2, int n3, int n4, int n5) {  
    int media = 0;  
    media = (n1+n2+n3+n4+n5)/5;  
    return media/  
}
```

Variáveis e Constantes



Variáveis

Representa um **valor que pode ser alterado** durante a execução do programa.

```
/*
 * @brief Programa que calcula a média
 * entre 5 números (n1, n2, n3, n4 e n5)
 */
int media(int n1, int n2, int n3, int n4, int n5) {
    int media = 0;
    media = (n1+n2+n3+n4+n5)/5;
    return media;
}
```



Constantes

Representa um **valor fixo** durante toda a execução do programa.

```
/*
 * @brief Programa que calcula a média
 * entre 5 números (n1, n2, n3, n4 e n5)
 */
int media(int n1, int n2, int n3, int n4, int n5) {
    int media = 0;
    media = (n1+n2+n3+n4+n5)/5;
    return media;
}
```

Variáveis e Constantes

Numérico: Inteiros (int) ou reais (float), podendo ser usado para cálculos matemáticos;

Exemplo: 1, 0, 3.14

Caracteres (char): Símbolos que não contêm números, como nomes;

Exemplo 'a', 'b', '\$'

Obs: um conjunto de caracteres é chamado de **String:**
“palavra”

Alfanumérico: combinação de números e letras, podendo conter só letras ou só números, mas não pode executar operações matemáticas;

Exemplo: “teste01”

Lógica / booleana: Verdadeiro ou falso;

Exemplo: “true”, “false,” “TRUE”, “FALSE”.

Obs: Algumas linguagens atribuem falso ao valor 0 e verdadeiro a qualquer outro valor diferente de 0

Operadores Aritméticos

Usados para operações de dados numéricos

Operação	Símbolo	Exemplo
Adição	+	Add = $10 + 2$
Subtração	-	Sub = $10 - 2$
Multiplicação	*	Mult = $10 * 2$
Divisão	/	Div = $10 / 2$
Resto (divisão)	%	Rest = $10 \% 2$

Ordem de execução:

1º Parênteses ();

2º Multiplicação ou divisão;

3º Soma ou subtração;

Lógica de Programação: Operadores Aritméticos

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Lógica de Programação: Operadores Aritméticos \n\n");
7
8     int a = 253;
9     int b = 432;
10
11     int soma = a + b;
12     int subtracao = b - a;
13     int multiplicacao = a * b;
14     int divisao = b / a;
15     int resto = b % a;
16
17     printf("soma = %d \n", soma);
18     printf("subtracao = %d \n", subtracao);
19     printf("multiplicacao = %d \n", multiplicacao);
20     printf("divisao = %d \n", divisao);
21     printf("resto = %d \n", resto);
22     return 0;
23 }
```

Ambiente on-line: <https://www.programiz.com/c-programming/online-compiler/>

Lógica de Programação:

Operadores Relacionais

Usados para comparações (decisões), retornando valores lógicos (true ou false).

Operador	Significado	Na prática...
==	Igual	Verifica se uma variável é igual a outra
!=	Diferente	Verifica se uma variável é diferente da outra
<	Menor	Verifica se uma variável é menor que a outra
>	Maior	Verifica se uma variável é maior que a outra
<=	Menor ou Igual	Verifica se uma variável é menor ou igual a outra
>=	Maior ou Igual	Verifica se uma variável é maior ou igual a outra

Lógica de Programação: Operadores Relacionais

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Operadores Relacionais \n\n");
7
8     int a = 253;
9     int b = 432;
10
11     printf("a == b? %d \n", a == b);
12     printf("a != b? %d \n", a != b);
13     printf("a < b? %d \n", a < b);
14     printf("a > b? %d \n", a > b);
15     printf("a <= b? %d \n", a <= b);
16     printf("a >= b? %d \n", a >= b);
17
18     return 0;
19 }
```

Ambiente on-line: <https://www.programiz.com/c-programming/online-compiler/>

Lógica de Programação:

Operadores Relacionais

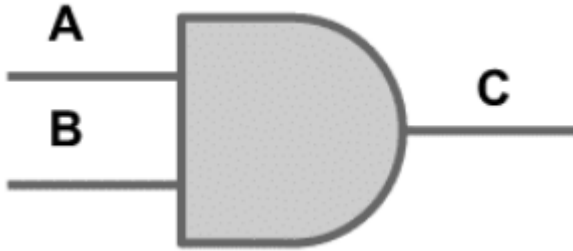
Usados para comparações (decisões), retornando valores lógicos.

Operador	Significado
&&	AND - E
	OR - OU
!	NOT - Inversor

Lógica de Programação:

Operadores Relacionais AND - &&

Equivale a uma operação de multiplicação

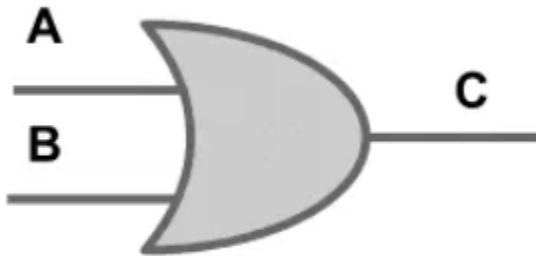


A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

Lógica de Programação:

Operadores Relacionais OR - ||

Equivale a uma operação de soma



A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

Lógica de Programação:

Operadores Relacionais NOT -

FIAP

! Equivale a uma operação de inversão



A	\bar{A}
0	1
1	0

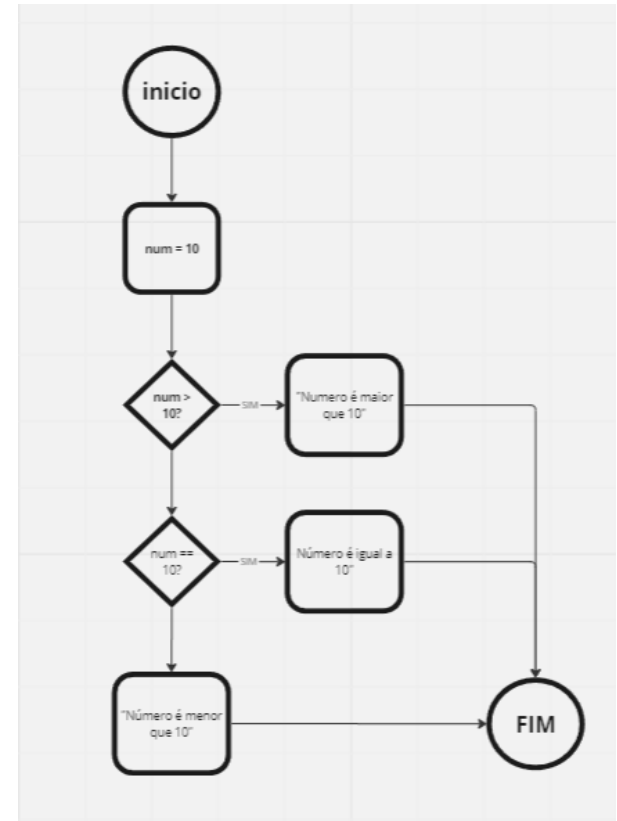
Lógica de Programação: Operadores Relacionais

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Operadores Relacionais \n\n");
7
8     int a = 1;
9     int b = 0;
10    int c = 1;
11
12    printf("a && b? %d \n", a && b);
13    printf("a && c? %d \n", a && c);
14    printf("a || b? %d \n", a || b);
15    printf("a || c? %d \n", a || c);
16    printf("!a? %d \n", !a);
17    printf("!b? %d \n", !b);
18
19    return 0;
20 }
```

Ambiente on-line: <https://www.programiz.com/c-programming/online-compiler/>

Estruturas Condicionais

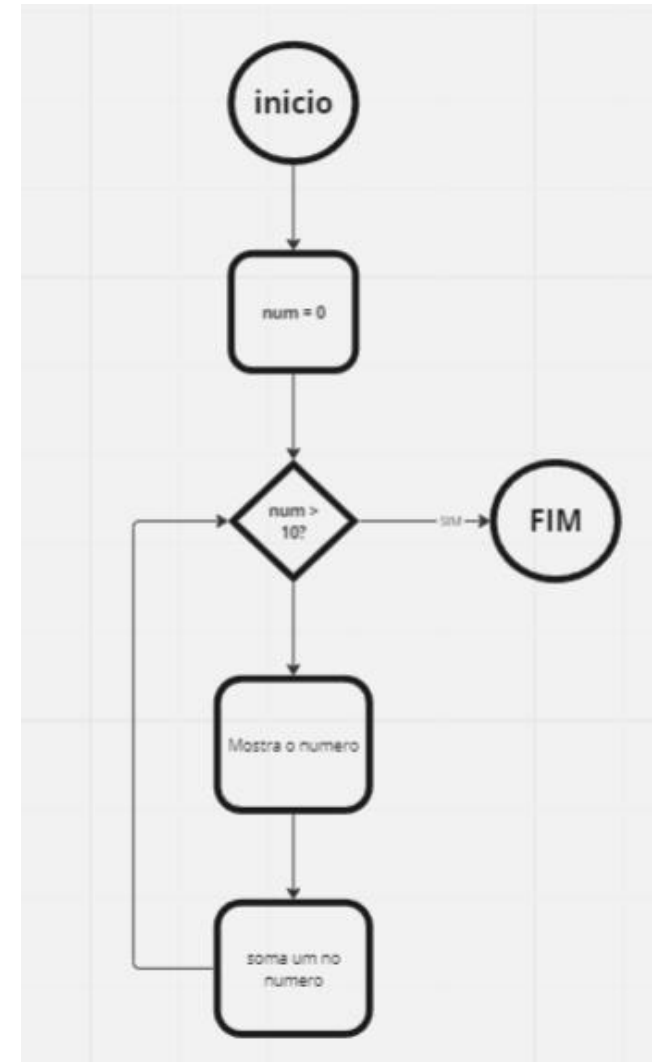
```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Estruturas Condicioanais \n\n");
7
8     int numero = 10;
9
10    if(numero > 10){
11        printf("numero é maior que 10");
12    }
13    else if (numero == 10){
14        printf("numero é igual a 10");
15    }
16    else{
17        printf("numero é menor a 10");
18    }
19
20    return 0;
21 }
```



Estruturas de Repetição

While

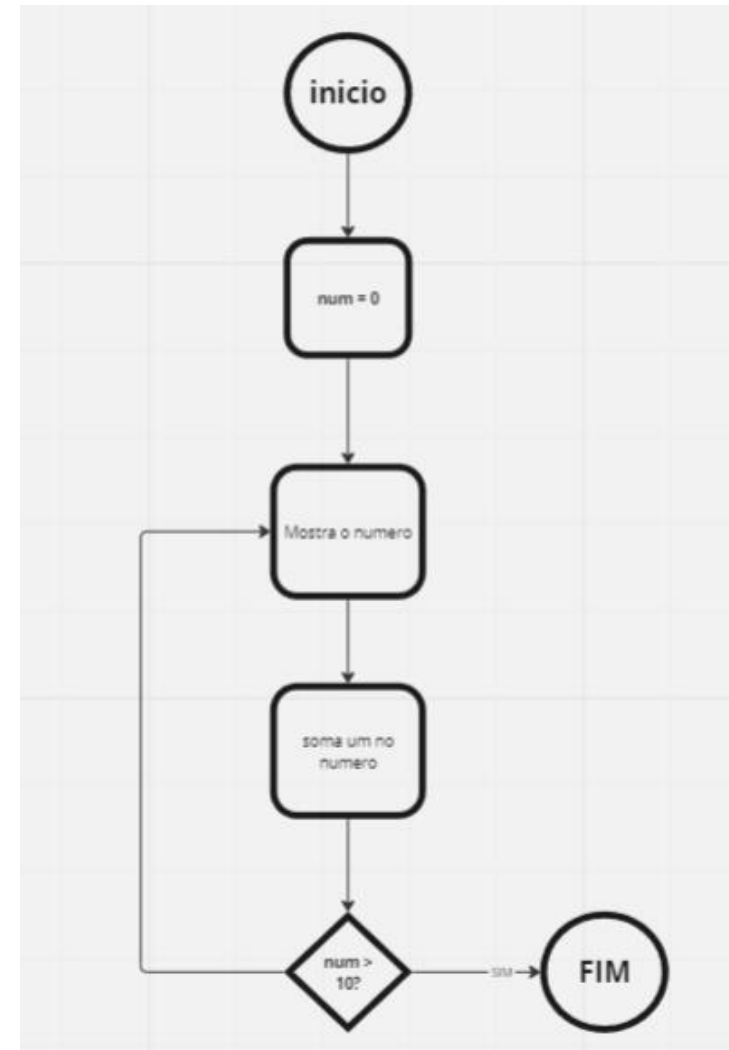
```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Estruturas De Repeticao \n\n");
7
8     int numero = 0;
9
10    while(numero < 10){
11        printf("numero = %d \n", numero);
12        numero ++;
13    }
14
15    return 0;
16 }
```



Estruturas de Repetição

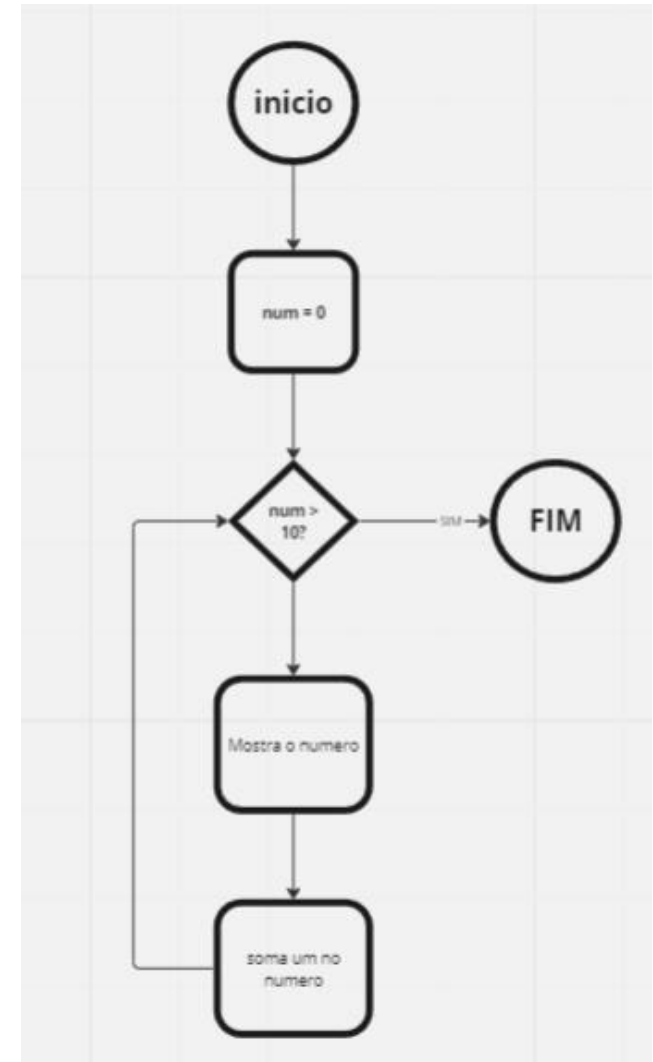
Do While

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Estruturas De Repeticao \n\n");
7
8     int numero = 10;
9     do{
10         printf("numero = %d \n", numero);
11         numero ++;
12     }while(numero < 10);
13
14     return 0;
15 }
```

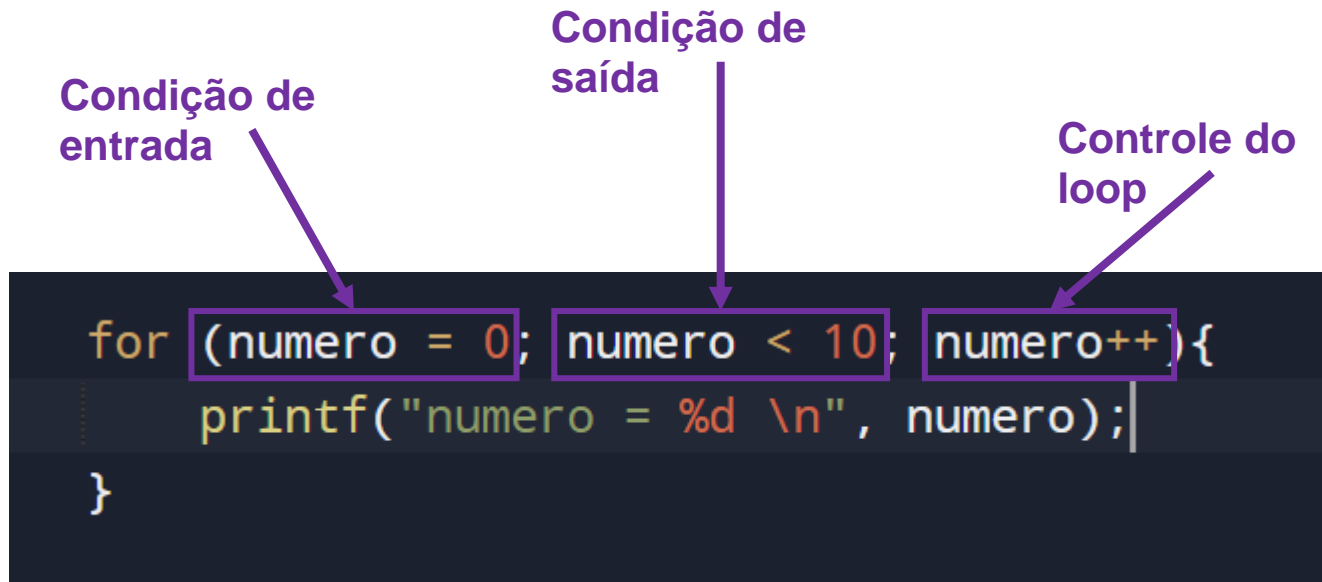


Estruturas de Repetição For

```
1 // Online C compiler to run C program online
2 #include <stdio.h>
3
4 int main() {
5     // Write C code here
6     printf("Logica de programação: Estruturas De Repeticao \n\n");
7
8     int numero = 0;
9
10    for (numero = 0; numero < 10; numero++){
11        printf("numero = %d \n", numero);
12    }
13
14    return 0;
15 }
```



Estruturas de Repetição



The diagram shows a C code snippet for a loop. Three purple arrows point from labels above to parts of the code: 'Condição de entrada' points to '(numero = 0;', 'Condição de saída' points to 'numero < 10;', and 'Controle do loop' points to 'numero++'. The code is as follows:

```
for (numero = 0; numero < 10; numero++){  
    printf("numero = %d \n", numero);  
}
```

- Uma forma de ler esse trecho de Código é: **para** (for) meu numero começando em zero (numero = 0;) **faça** esse loop **enquanto** meu numero for menor que 10 (numero < 3), **sendo que a cada loop** o meu numero incrementa uma unidade (numero++)

Exercícios

1. Usando o ambiente online apresentado, implemente todos os códigos mostrados nessa aula e avaliem as saídas.
2. Mude os valores das variáveis e avalie os resultados.



Ambiente on-line: <https://www.programiz.com/c-programming/online-compiler/>

Laboratório – Código Morse

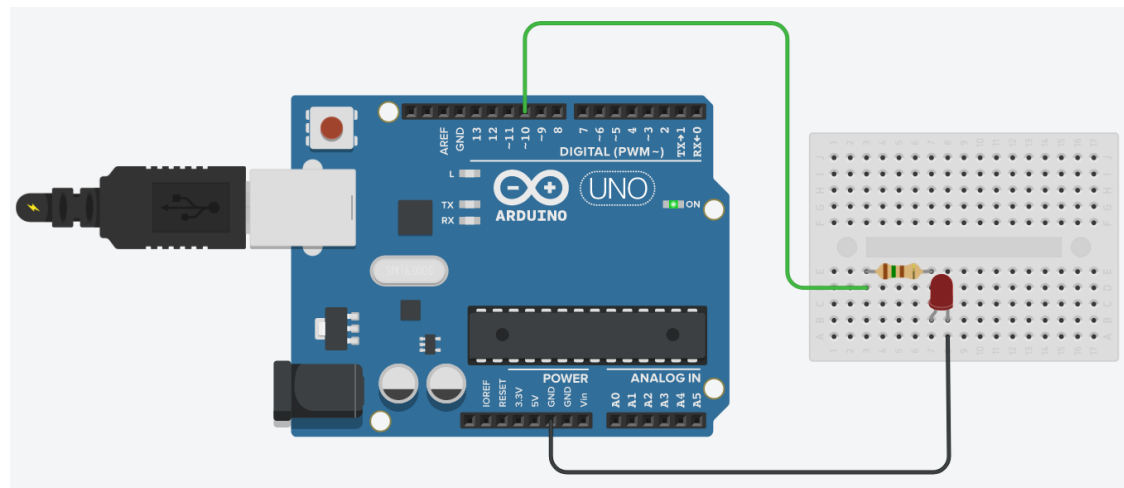
O objetivo desse laboratório é entender os loops de repetição e controlar o tempo em que o led fica aceso e apagado. Considere que um ponto é o led aceso por 1 segundo e apagado por 1 segundo, e que um traço é o led aceso por 2 segundos e apagado por 1 segundo.



A	· —	N	— ·	1	— — — —	?	· · — —
B	· · · —	O	— — —	2	· — — —	!	· — — —
C	· — · —	P	— · — ·	3	· · — —	,	· — · —
D	— · · —	Q	— — · —	4	· — —	;	— — — ·
E	· — — —	R	— · — ·	5	· · · —	:	— — — ·
F	· · — —	S	— · —	6	— · — —	+	— — — ·
G	— · — ·	T	— — —	7	— — — ·	=	— · — —
H	· · · —	U	— · —	8	— — — ·	-	— · — —
I	· — —	V	— · — ·	9	— — — ·	/	— · — —
J	· — — —	W	— · — —	0	— — — —		
K	— · —	X	— · — ·				
L	— · —	Y	— · — ·				
M	— — —	Z	— — · —				

Material necessário:

- 1 Arduino;
- 1 Resistor de 150 ohms (marrom, verde, marrom);
- 1 Led (qualquer cor);
- 1 Protoboard;
- Jumpers cables.



Link: [Projeto 02 – SOS Morse Code Signaler](#)

Exercício Desafio

Vamos aplicar o que vimos nessa aula e montar no simulador um **Semáforo**.

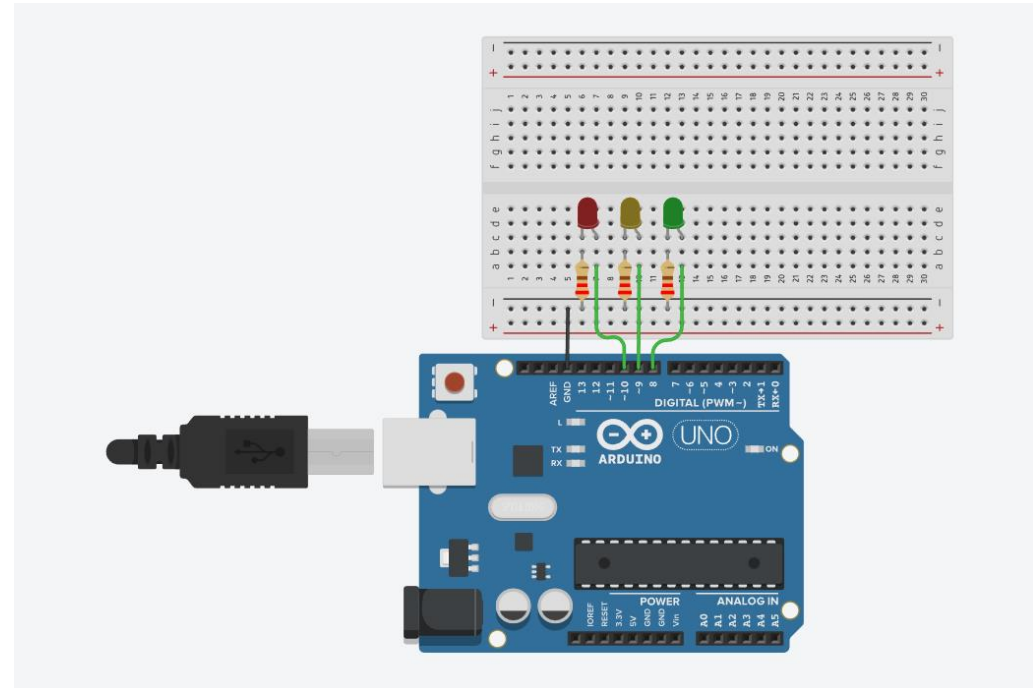
1. Desenhe um fluxograma de como o semáforo deve funcionar, seguindo o seguinte racional:

- Acende o Vermelho;
- Espera 5 segundos;
- Acende o Verde;
- Espera 5 segundos;
- Acende o Amarelo;
- Espera 2,5 segundos;
- Repete o loop;

2. Implemente o projeto no TinkerCad;

3. Monte o projeto na prática;

Link: [Projeto 03 – Traffic Lighs](#)



Material necessário:

- 1 Arduino;
- 3 Resistores de 220 Ohms;
- 1 Led Verde;
- 1 Led Amarelo;
- 1 Led Vermelho;
- 1 Protoboard;
- Jumpers cables



Copyright © 2023

Prof. **Airton** / Prof. **Fabio** / Prof. **Lucas** / Prof. **Yan**

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).