

A8

Arduino

if e loop

Voltemos a receita de Pão de Queijo

Atenção às instruções!

MODO DE PREPARO

Misture o polvilho e o sal e reserve.

Ferva o leite e o óleo e em seguida escale o polvilho com o sal.

Deixe esfriar, acrescente os ovos e o queijo.

Quando a massa estiver homogênea, faça bolinhas.

Leve para assar em forno quente até dourar.

Instruções são ordens
sequenciais dada ao sistema.

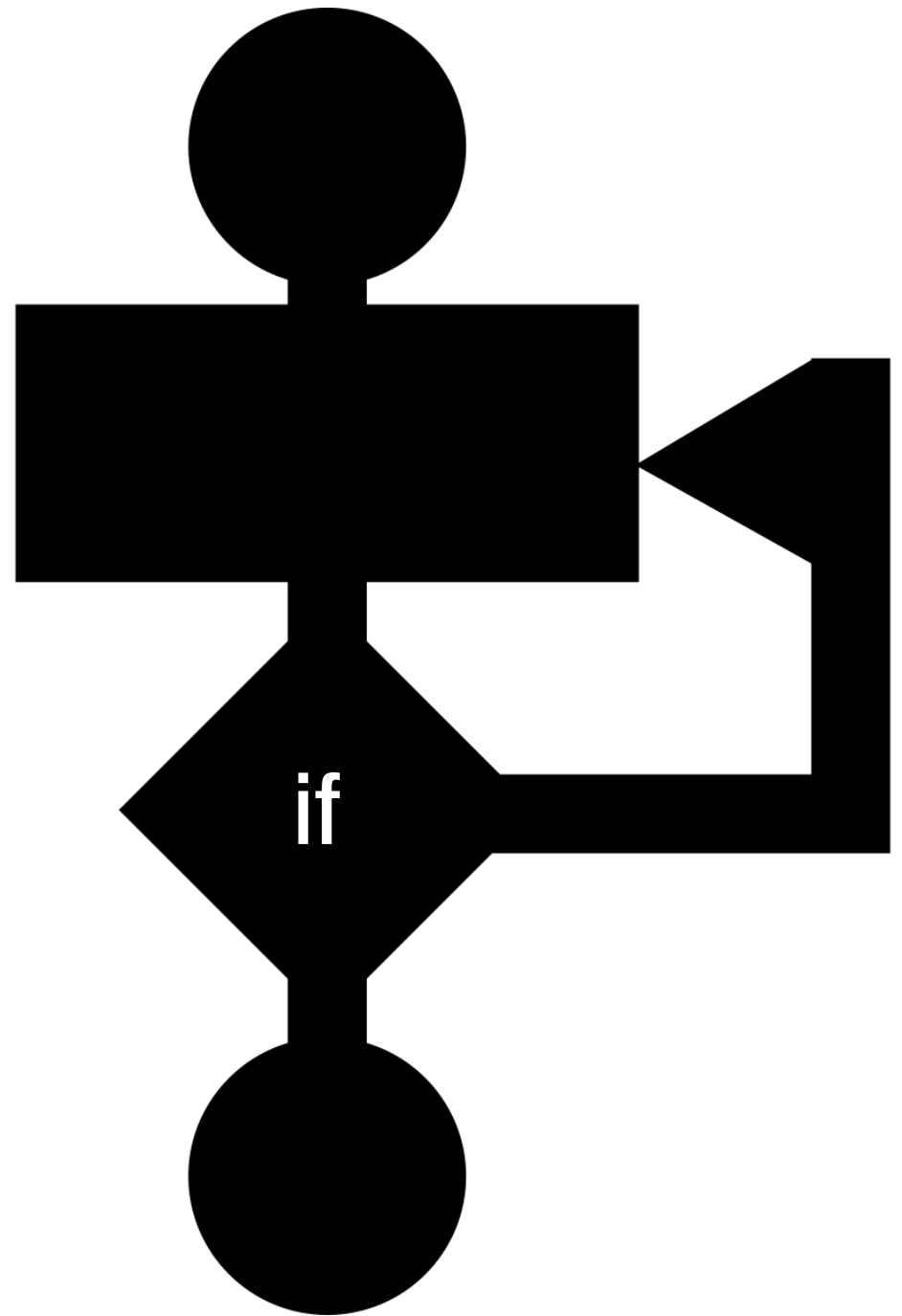
Há duas grandes categorias
de instrução em computação.

CONDIÇÃO
&
REPETIÇÃO



Condição

As sentenças condicionais são usadas quando o sistema tem que tomar uma decisão.



Dentro do bloco de texto loop();

```
// a função de loop roda repetidamente, para sempre
void loop() {
  // escrever o valor atual de brilho no pino 9:
  analogWrite(led, brilho);

  // atualizar o valor do brilho para ser usado no próximo loop
  brilho = brilho + mudancaNoBrilho;

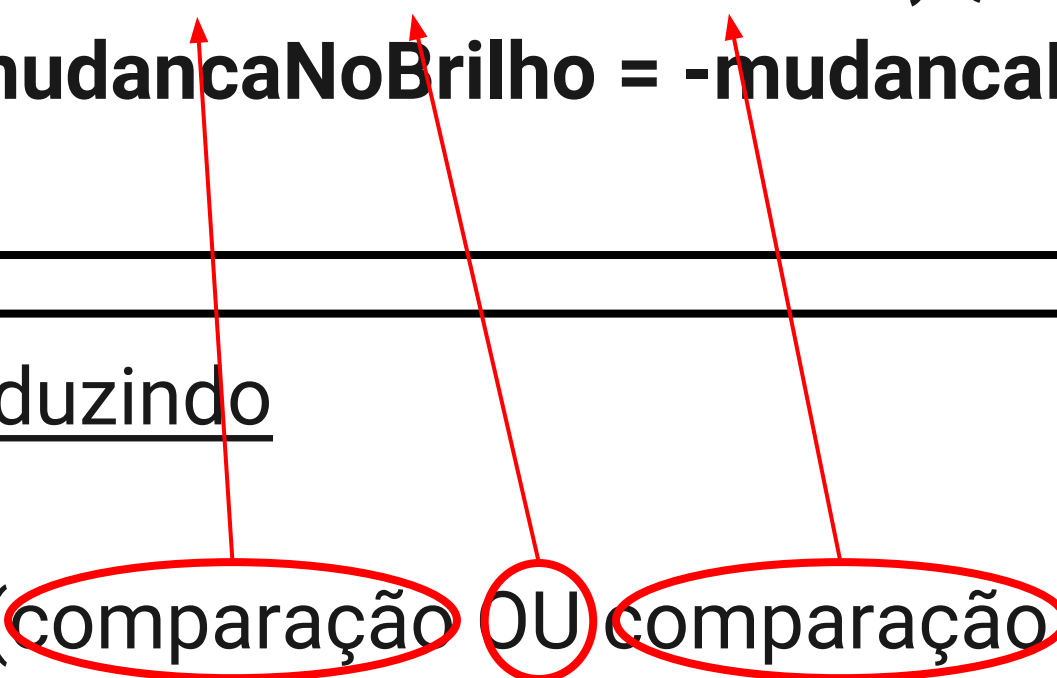
  // muda a direção do fade se o LED está completamente ligado
  if (brilho == 0 || brilho == 255) {
    mudancaNoBrilho = -mudancaNoBrilho ;
  }

  // espera 30 millisegundos para ver o efeito do fade:
  delay(30);
}
```

```
if (brilho == 0 || brilho == 255) {  
    mudancaNoBrilho = -mudancaNoBrilho ;  
}
```

Traduzindo

Se (comparação OU comparação) forem verdadeiras) siga a seguinte ordem.



A condição tem de ser VERDADEIRA para executar a ordem que está contida nela.

A condição para executar este `if()` é composta por duas comparações.

brilho == 0

A primeira comparação é: o valor da variável **brilho** é igual a zero?

brilho == 255

A segunda comparação é: o valor da variável **brilho** é igual a 255?

Operadores de Comparação

$x == y$ (x é igual a y ?)
 $x != y$ (x é diferente de y ?)
 $x < y$ (x é menor que y ?)
 $x > y$ (x é maior que y ?)
 $x <= y$ (x é menor ou igual a y ?)
 $x >= y$ (x é maior ou igual a y ?)

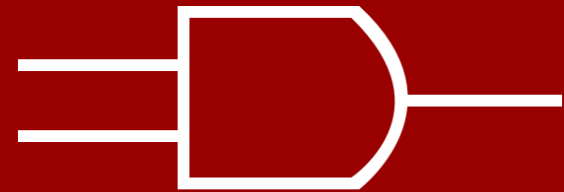


As comparações são associadas
por um conector lógico. ãããã??



Operadores Lógicos

x AND y
só é verdadeiro se **x** e **y**
forem verdadeiros.



x OR y
é verdadeiro se **x** ou **y** for
verdadeiro. Um deles pode
ser falso.

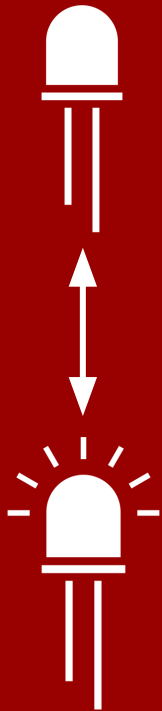


Então, se o valor da variável **brightness** for igual a zero OU igual a 255, a condição se torna VERDADEIRA.

O sistema toma a decisão de executar a ordem:

mudancaNoBrilho = -mudancaNoBrilho;

E o que
significa esta
ordem?



A **mudancaNoBrilho** é uma variável que armazena o valor a ser incrementado entre o apagado e o ligado do LED.

Esta ordem inverte o sentido dos passos do fade. Se ele está apagado, segue para aumentar o brilho. Se está ligado, segue no inverso para começar a apagar.

Vamos olhar o bloco loop() novamente

```
// a função de loop roda repetidamente, para sempre
void loop() {
    // escrever o valor atual de brilho no pino 9:
    analogWrite(led, brilho);

    // atualizar o valor do brilho para ser usado no próximo loop
    brilho = brilho + mudancaNoBrilho;

    // muda a direção do fade se o LED está completamente ligado
    if (brilho == 0 || brilho == 255) {
        mudancaNoBrilho = -mudancaNoBrilho ;
    }
    // espera 30 millisegundos para ver o efeito do fade:
    delay(30);
}
```

O loop() é um bloco de REPETIÇÃO


Vamos traduzir...

1. EscrevaAnalógico na PORTA em que se encontra o LED o valor do brilho.

2. Atribua à variável **brilho** o valor dela mesma e mais o valor contido na variável **mudancaNoBrilho**.

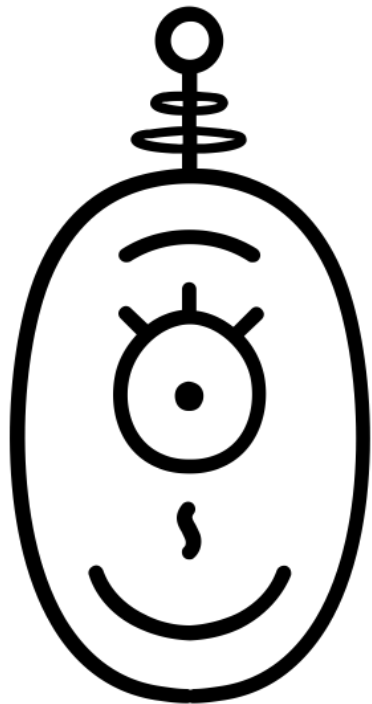
3. Verifique se o LED está apagado ou ligado. Se apagado ou ligado, mude a direção da mudança do brilho do LED.

4. Espere 30 milisegundos.



**O bloco loop() segue fazendo isso
ininterruptamente**

Pronto, você está preparado para o
próximo passo!



A9