# PROGRAMAÇÃO BÁSICA

NATANAEL ANTONIOLI

FÁBRICA DE NOOBS

# 3 LÓGICA DE PROGRAMAÇÃO

# Programação Básica – Lógica de Programação

Por: Fábrica de Noobs

*Índice*

## 1) Prefácio

Por um tempo, questionei se um curso só sobre lógica de programação seria realmente necessário. Afinal, é um assunto relativamente simples, que muitos já o conhecem e dominam sem tê-lo estudado.

Porém, é um tema essencial para poder estudar qualquer linguagem realmente de programação. Sem uma noção de lógica, a compreensão de um script (ou rotina) torna-se impossível.

Sendo assim, decidi escrever esse curso. Talvez você já tenha algum – ou todo – o conhecimento necessário para entender lógica de programação. Nesse caso, sinta-se livre para pular quantos capítulos achar necessário, ou até o curso todo. Mas se você nunca trabalhou, ou ao menos se interessou por compreender um script, é melhor estudar essa apostila.

## 2) Introdução à Lógica de Programação

A lógica de programação pode ser definida pelo estudo e compreensão de como linguagens de programação funcionam. Parece meio estúpido, e talvez realmente seja, já que alguns – pelo menos comigo foi assim – sequer precisam estudar tal tema, uma vez que já o dominam por experiência própria.

Em nosso dia-a-dia, mesmo sem perceber, a maioria de nossas ações envolve uma série de procedimentos. Por exemplo, para ligar um computador, deve-se: pressionar o botão, aguardar, inserir a senha de login etc.

Na computação também é assim. Todo programa funciona por meio de uma rotina, de uma sequência de instruções, que indicam para o computador o que ele deve executar.

Essa rotina também pode envolver uma tomada de decisões, onde o computador deverá analisar uma informação e tomar uma escolha entre outras conforme sua análise.

Basicamente, a lógica se resume a transformar uma ideia em uma série de procedimentos lógicos.

Esse conjunto de procedimentos, ao ser colocado em sequência, é chamado de algoritmo. Podemos criar um algoritmo para diversas coisas, por exemplo, a sequência para se somar dois números seria:

- Receber valor A
- Receber valor B
- Realizar A + B
- Retornar resultado

## 3) Construção de Algoritmos

Quando estamos trabalhando em alguma linguagem de programação, devemos utilizar uma sintaxe própria para escrever nossa rotina. Veja abaixo a mesma rotina em Batch e em Phyton, respectivamente:

```
@echo Off
echo.
SET /p a=primeiro valor:
SET /p b=segundo valor:
SET /a r= %a% + %b%
echo.
echo O resultado e %r%.
pause
```

```
num1 = int(input("Insira o Primeiro Numero:"))
num2 = int(input("Insira o Segundo Número:"))

soma = num1 + num2

print(soma)
```

Ambos os códigos realizam o mesmo procedimento que o algoritmo que escrevemos no capítulo anterior. Porém, são compilados (escritos) em sintaxes diferentes. Isso varia conforme a linguagem de programação escolhida, mas o princípio lógico é sempre o mesmo.

Dentro da lógica de programação, não temos uma linguagem específica para se escrever os algoritmos. Sendo assim, os escrevemos no chamado pseudocódigo.

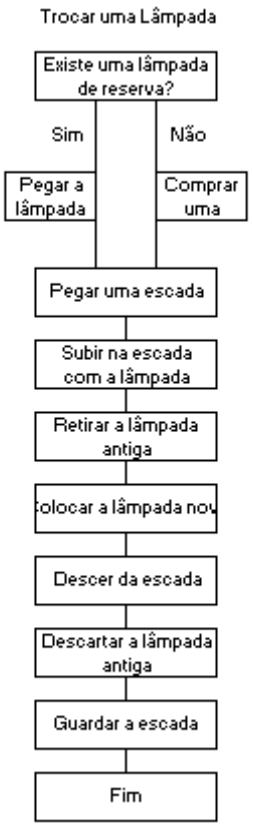
Pseudocódigo é uma definição ampla, referente ao processo anterior a implementação do código em si. Portanto, não existem tantas regras e sintaxes como em uma linguagem de programação. Logo, as únicas regras que deve-se seguir para escrever um algoritmo são:

- Imaginar que o código é escrito para alguém que não necessariamente entende de informática.
- Usar frases curtas, simples e objetivas.
- Apenas um verbo por frase.
- Não usar palavras com sentido dúbio.

Pode-se escrever um algoritmo através de fluxogramas, também chamados diagramas de blocos. Nele, representamos as operações lógicas de forma gráfica. Veja o mesmo algoritmo escrito em texto e em fluxograma:

“Trocar uma lâmpada”

- Existe uma lâmpada de reserva?
    - Sim
        - Pegar a lâmpada e prosseguir.
    - Não
        - Comprar uma e prosseguir.
- Pegar uma escada.
- Subir na escada com a lâmpada reserva.
- Retirar a lâmpada antiga.
- Colocar a lâmpada nova.
- Descer da escada.
- Descartar a lâmpada antiga.
- Guardar a escada.
- Fim.




Algoritmos em fluxograma são mais úteis quando envolvem tomadas de decisão, como a situação de verificar se havia uma lâmpada reserva, uma vez que permite melhor visualização. Porém, ao escrever um código, devemos usar a linguagem escrita.

Todo algoritmo pode ser dividido em 3 partes: entrada, processamento e saída. A entrada corresponde a coleta de informações. O processamento, ao processo em si. E a saída, ao resultado final.

Após criar um algoritmo, principalmente os que envolvem operações matemáticas, é recomendável segui-lo e testá-lo com diferentes valores, para verificar se ele realmente funciona ou apresenta falhas. Isso é chamado de teste de mesa.

## 4) Variáveis e Constantes

Ao criar um código para uma ação do dia-a-dia, nosso cérebro automaticamente armazena informações e as utiliza depois. No caso da programação, é preciso informa-las durante a execução do código.

Uma constante é um valor fixo, que não se altera durante a execução do código.

Já uma variável, corresponde a um espaço na memória do computador destinado a armazenar determinado valor, que pode mudar durante o processo.

Veja o exemplo abaixo:

$$mediafinal = \frac{p1 + p2 + p3}{3}$$

Nele, mediafinal, p1, p2 e p3 são variáveis, uma vez que elas podem assumir qualquer valor. As 3 últimas dependem do valor inserido pelo usuário, e a final, depende de tais valores.

Já o 3 corresponde a uma constante. O valor não irá se alterar durante a execução do código, independente de quais outros valores sejam inseridos.

Uma variável pode ser de 4 tipos:

- Numéricas: armazenam apenas valores numéricos, podendo ser utilizadas para cálculos.
- Caracteres: armazenam conjuntos de caracteres. São mais usadas para exibir mensagens ou construção de menus.
- Alfanuméricas: podem ser usadas tanto para números quanto para caracteres. Porém, no caso de números, ela não pode ser usada para uma operação matemática.
- Lógicas: armazenam apenas 2 valores: sim ou não, ou verdadeiro e falso.

Um exemplo de uso de variáveis pode ser visto no último algoritmo que analisamos. “Existe uma lâmpada reserva?” é uma variável lógica, que pode ser Sim ou Não.

## 5) Operadores

Utilizamos os operadores para incrementar, subtrair, comparar ou realizar qualquer tipo de operação envolvendo as variáveis. Eles podem ser divididos em aritméticos, relacionais e lógicos.

Operadores aritméticos são utilizados para trabalhar com resultados numéricos. Basicamente, se resumem as operações matemáticas fundamentais:

| | |
|---|---|
| Adição | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |
| Exponenciação | ^ |

Operadores relacionais servem para comparar duas variáveis, que podem ser de qualquer tipo. O resultado de tal comparação pode ser verdadeiro ou falso. São eles:

| | |
|---|---|
| Igual a | = |
| Diferente de | # ou <> |
| Maior que | > |
| Menor que | < |
| Maior ou igual | >= |
| Menor ou igual | <= |

Já os operadores lógicos servem para comparar duas expressões (as quais podem ser feitas utilizando os operadores acima citados), tendo como resultado, também verdadeiro ou falso.. São eles:

| | | | |
|---|---|---|---|
| Ambas são verdadeiras | AND | E | && |
| Uma ou outra é verdadeira | OR | OU | \|\| |
| Ela possui valor oposto | NOT | NÃO | |

Veja um exemplo de aplicação de operadores lógicos:

| 1ª Expressão | 2ª Expressão | O. Lógico | Resultado |
|---|---|---|---|
| 2 + 2 = 4 | 5 + 5 = 10 | AND | VERDADEIRO |
| 2 + 5 # 7 | 5 > 7 | OR | FALSO |
| 6 + 8 = 11 | 7 > 5 | OR | VERDADEIRO |
| 5 + 5 = 10 | 2 + 2 = 4 | OR | VERDADEIRO |
| 7 + 1 = 8 | 3 # 2 | AND | FALSO |
| 7 + 1 = 9 | | NOT | VERDADEIRO |

Aqui, temos 2 expressões, ambas feitas com operadores lógicos e aritméticos, e comparadas com um operador lógico.

A sintaxe mostrada nas tabelas acima representa os sinais mais comumente utilizados no pseudocódigo. Dependendo da linguagem de programação, ela pode ser diferente.

**6) Tomadas de decisão**

Após obtermos o resultado através de um operador, podemos especificar rotinas para serem seguidas dependendo do seu resultado. Por exemplo:

**Se** possuir lâmpada = verdadeiro
    Trocar lâmpada
**Se** possuir lâmpada = falso
    Comprar uma

Ou então:

**Se** possuir lâmpada = verdadeiro
    Trocar lâmpada
**Se não**
    Comprar uma

Nesse código, se e se não foram os valores usados para orientar a nova rotina conforme o valor apresentado pela variável lógica possuir lâmpada. Esses comandos são chamados comandos de decisão, e podem ser escrito em diversas formas dependendo da linguagem. Em pseudocódigo, são também representados da forma if/else. Para mostrar a continuidade da rotina, usa-se o then. Veja no exemplo:

**If** possuir lâmpada = true **then** trocar lâmpada
**Else then** comprar uma

Vejamos agora esse outro exemplo:

Receber nota
Definir variável a = nota
**If** a < 0, exibir "A nota inserida é negativa" **else** continuar
**If** a > 11, exibir "Insira uma nota de 0 a 10" **else** continuar
**If** a >= 6, exibir "Nota azul" **else** exibir "Nota vermelha"

Aqui, primeiro recebemos uma nota. Depois, definimos a variável a pelo valor de tal nota. Em seguida, fazemos 3 comparações:

- A nota é negativa?
- A nota é maior que 10?
- A nota é menor ou igual a 6?

Repare que nos 2 primeiros if, mandávamos o script continuar a rotina caso a comparação fosse falsa. Já no último, exibimos diretamente a mensagem de "Nota vermelha". Isso ocorre porque se a nota não se adequa a todos as outras condições, então ela só pode ser vermelha. É desnecessário inserir mais um else continuar para terminarmos a rotina.

**7) Comandos de Repetição**

Vamos imaginar um algoritmo criado para, partindo de uma variável definida pelo usuário, ir somando 1 até a variável se igualar a 40. Sua forma escrita seria mais ou menos essa:

Receber variável a
If a < 40, prosseguir else exibir "o Valor apresentado não é aceitável.
Realizar a + 1
If a < 40, prosseguir else terminar
If a < 40, prosseguir else terminar
.....?
If a < 40, prosseguir else terminar

Como pode-se perceber, não há uma forma de fazer tal algoritmo sem repetir a mesma linha quantas vezes for preciso para igualar nossa variável a 40. Isso sem considerar que esse número de linhas seria diferente para cada valor de a.

A solução para o problema é o uso de comandos de repetição. Eles dizem ao programa, por exemplo "Realize tal procedimento até que a seja igual a 40". Temos dois possíveis comandos para utilizar nesse caso.

Podemos dizer ao programa “Enquanto tal condição for verdadeira, faça isso”. Para tanto, usamos a sintaxe Do While... Loop. Assim, quando a condição se tornar falsa, o programa irá parar. Por exemplo:

```
Do a + 1 while a < 40
Loop
```

Outra opção é dizer “Faça isso até que tal condição seja verdadeira”. Para tanto, usamos a sintaxe Do Until... Loop. Por exemplo:

```
Do a + 1 until a < 40
Loop
```

## 8) Conclusão

Assim terminamos nosso curso de lógica da programação. Entendendo os conceitos acima mostrados, você provavelmente não terá dificuldade para escrever suas primeiras linhas de código.