Disciplina

R para Ciência de Dados
Aula 2

Introdução à Lógica de Programação e Algoritmos

Alan Rodrigo Panosso

Departamento de Ciências Exatas
(UNESP/Jaboticabal)
alan.panosso@unesp.br

Algoritmos

O que é Algoritmo?

Algoritmo

=

Sequência lógica e não ambígua de **instruções** que levam à solução de um problema num tempo finito.

- Características básicas de um algoritmo
  - Sequência lógica
    - As instruções devem ser definidas em uma ordem correta.
  - Não ambígua
    - A sequência lógica e as instruções não devem dar margem à dupla interpretação.
  - Solução de um problema
    - A sequência lógica deve resolver exatamente (nem mais e nem menos) o problema identificado.
  - Tempo finito
    - A sequência lógica não deve possuir iterações infinitas.

O Problema em ser Programador

Minha mãe disse:
- Filho, vá até o mercado e compre 1 garrafa de leite, se eles tiverem ovos, traga 6.
Eu voltei para casa com 6 garrafas de leite

Ela disse:
-Porque você comprou 6 garrafas de leite?

Eu respondi:
-Porque eles tinham ovos.

- Exemplos de algoritmos
  - Receitas de comidas
  - Coreografia
  - Manuais de instalação
- Contra-exemplos de algoritmos
  - Listas de compras
  - Dança informal
  - Índice remissivo de um livro

- Atenção:
  - Um algoritmo é "**uma** solução" e não "**a** solução" de um problema.
    - Um problema pode ser resolvido por mais de um algoritmo! ******* SEMPRE ******
  - Tarefas que possuem "padrão de comportamento" podem ser descritas por um algoritmo.
    - Ex: Qual será o próximo número da sequência

    0, 1, 4, 9, 16, 25 ?

O que é Lógica de Programação?

Lógica de Programação

=

É o encadeamento lógico de **instruções** para o desenvolvimento de programas.

O que é Programa?

Programa

=

É a implementação das **instruções**

(codificação + tradução + execução)

de um algoritmo em uma

linguagem de programação.

O que é
Instrução?

Instrução

=

Informação que representa uma **ação elementar** que deve ser executada.

- Exemplos de instruções
  - Girar a chave;
  - Desligar interruptor da luz;
  - Acordar;

- Contra-exemplos e instruções
  - Entrar em casa (girar a chave + ...)
  - Trocar uma lâmpada (desligar interruptor + ...)
  - Ir para o trabalho (acordar + ...)

Por que aprender algoritmos ?

- Para desenvolver o raciocínio lógico e conceber uma solução a um dado problema, independente de uma linguagem de programação. (Ex: Fortran, Pascal, C e Phyton, R)

- Porque, a partir do algoritmo desenvolvido, fica mais fácil implementar o respectivo programa.

- Segundo grandes pesquisadores ...
  - “A noção de algoritmo é básica para toda a programação de computadores”. KNUTH
    - KNUTH - Professor da Universidade de Stanford e autor da coleção “The art of computer programming”.
  - “O conceito central da programação e da ciência da computação é o conceito de algoritmo”. WIRTH
    - WIRTH - Professor da Universidade de Zurique, autor de diversos livros na área e responsável pela criação de linguagens de programação como ALGOL, PASCAL e MODULA -2.

Estruturas de Controle

Estruturas de Controle

Na criação de algoritmos, utilizamos os conceitos de bloco lógico, entrada e saída de dados, variáveis, constantes, atribuições, expressões lógicas, relacionais e aritméticas, bem como comandos que traduzam esses conceitos de forma a representar o conjunto de ações.

Para que esse conjunto de ações se torne viável, deve existir uma perfeita relação lógica intrínseca ao modo pelo qual essas ações são executadas, ao modo pelo qual é regido o **fluxo de execução** do algoritmo.

Por meio das estrutura básicas de controle do fluxo de execução – **sequencial**, **seleção**, **repetição** e da **combinação delas** – poderemos criar algoritmos para solucionar nossos problemas.

- Estruturas básicas de um algoritmo:
  - **Sequência** – Início/Fim
    - Define uma estrutura onde as instruções serão executadas na ordem que aparecem.
  - **Seleção** – Se-Então/Senão
    - Define uma estrutura condicional que, dada a sua avaliação (V ou F), determina qual "caminho" do algoritmo será executado.
  - **Repetição** – Repita, Enquanto, Faça-Enquanto ou Para
    - Define uma estrutura de iteração condicional (V ou F) ou contada (predefinida) de instruções.

Exemplo: Faça um Algoritmo na forma textual cujo objetivo é trocar uma lâmpada.

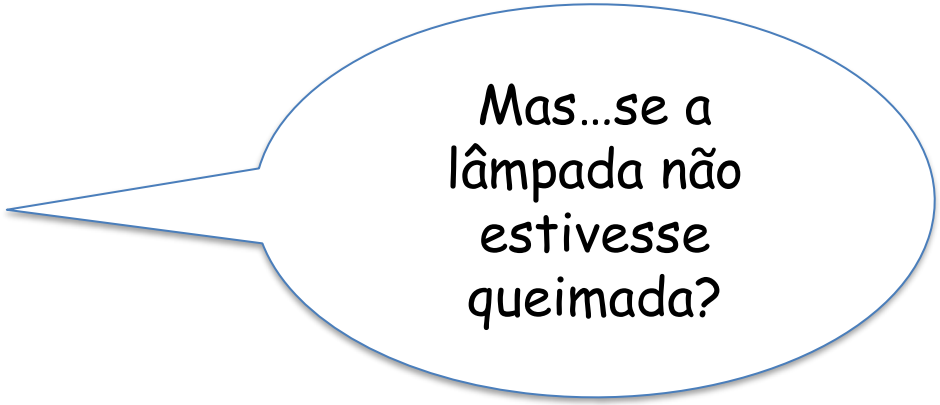
Algoritmo

Início

- **Pegar escada;**
- **Posicionar escada embaixo da lâmpada;**
- **Buscar nova lâmpada;**
- **Subir escada;**
- **Retirar lâmpada velha;**
- **Colocar a lâmpada nova;**

Final

Voltando ao exemplo da lâmpada, alguém poderia perguntar:



Poderíamos efetuar um teste de decisão, utilizando a estrutura SE, onde temos um teste lógico, o que acontece se o teste for VERDADEIRO e, se necessário, o que acontece se o teste for FALSO.

Algoritmo

Início

- **Pegar escada;**
- **Posicionar escada embaixo da lâmpada;**
- **Buscar nova lâmpada;**
- **Acionar o interruptor;**
- **SE a lâmpada não acender, ENTÃO**

  Início do bloco
  - **Subir escada;**
  - **Retirar lâmpada velha;**
  - **Colocar a lâmpada nova;**

  Fim do bloco

  Bloco de instruções
- **Fim do SE**

Final

Observe que se a lâmpada acender, ou seja, a condição "a lâmpada não acende" for FALSA as ações relativas à troca não acontecem.

Mas...eu preciso mesmo...!?!?
Pegar escada;
Posicionar escada embaixo da lâmpada;
Buscar nova lâmpada;

Algumas ações podem ser evitadas, com a simples otimização do *Algoritmo*, ou seja, poderíamos fazer o teste *"a lâmpada não acende"* no início das ações.

Algoritmo
Início

- Acionar o interruptor;
- SE a lâmpada não acender, então

  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;

  Fim do bloco

  Bloco de instruções
- Fim do SE

Final

Estrutura de Seleção

Portanto, essa estrutura permite a escolha de um grupo de ações (bloco) a ser executado quando determinadas *condições*, representadas por expressões lógicas e/ou relacionais, são satisfeitas ou não. Os tipo de seleção apresentados serão: *Simples*, *Composta* e *Encadeada*.

Seleção Simples

Quando precisamos testar uma certa condição antes de executar uma ação.

```
se <condição> então // condição é uma expressão lógica ou relacional
        Início do bloco
        C; // ação primitiva que ocorre se a expressão lógica for verdadeira
        fim do bloco
Fim se; // fim da seleção
```

OBS: Quando houver somente uma ação primitiva, a estrutura pode ser:

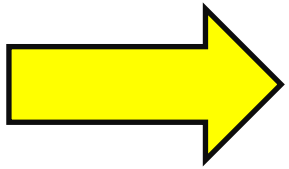
```
se <condição> então
        C;
Fim se;
```

Estrutura de Seleção

Seleção Simples

Pseudocódigo:

se <condição> **então**
Início do bloco
C1;
C2;
fim do bloco
Fim se;



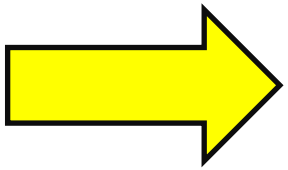
R

```
if  (condição)
{
      C1
      C2
}
```

Seleção Simples para uma ação primitiva apenas

Pseudocódigo

se <condição> **então**
C1;
Fim se;



R

```
if  (condição)
      C1
```

 # ou

```
if  (condição) C1
```

Estrutura de Seleção

Seleção Composta

Utilizadas em situações em que duas alternativas dependem de uma mesma condição: uma da condição VERDADEIRA, e outra da condição FALSA.

```
se <condição> então
Início do bloco
        C1; // ação se verdadeiro
fim do bloco
senão
Início do bloco
        C2; // ação se falso
fim do bloco
Fim se; // fim da seleção
```

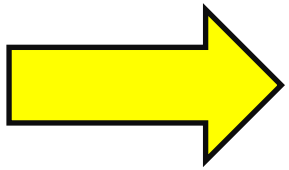
OBS: Quando houver somente uma ação primitiva, para cada situação, a estrutura pode ser:

```
se <condição> então
                C1;
senão
                C2;
Fim se;
```

Estrutura de Seleção
Seleção Composta

Pseudocódigo

```
se <condição> então
Início do bloco
        C1;
        C2;
fim do bloco
senão
Início do bloco
        C3;
        C4;
fim do bloco
Fim se;
```



R

```
if  (condição) {
        C1
        C2
}else{
        C3
        C4
}
```

Estrutura de Seleção

Seleção Encadeada

É o agrupamento de várias seleções, ocorre quanto uma determinada ação, ou bloco deve ser executado se um grande conjunto de possibilidades ou combinações de situações for satisfeito.

```
se <condição 1>  então
        se <condição 2>  então
                se <condição 3>  então
                        C1    // ação se condições 1, 2 e 3 forem Verdadeiras
                senão
                        C2    // 1 e 2  forem Verdadeiras mas 3 for Falsa
                fim se
        fim se
Fim se
```

Estrutura de Seleção

Seleção Encadeada

Exemplo: Dados A, B e C, verificar se eles formam um triângulo (cada um dos lados é menor que a soma dos outros dois), se formarem, verificar se compõe um triângulo equilátero (3 lados iguais), isósceles (2 lados iguais) ou escaleno (3 lados diferentes).

Escrevendo as condições em expressões lógicas

É triângulo: (A < B+C) e (B < A+C) e (C<A+B)
É equilátero: (A = B) e (B = C)
É isósceles: (A=B) ou (A=C) ou (B=C)
É escaleno: (A <> B) e (A <> C) e (B <>C)

Calma...é fácil !
Teste uma condição por vez

Algoritmo

```
início
    real: A, B, C;
    leia (A, B, C);
    se ((A < B+C) e (B<A+C) e (C<A+B))  então
        se ((A = B) e (B=C)) então
            escreva ("Triângulo Equilátero");
        senão
            se ((A = B) ou (A=C) ou (B=C))   então
                escreva ("Triângulo Isósceles");
            senão
                escreva ("Triângulo Escaleno");
            fim se;
        fim se;
    senão
        escreva ("Não formam triângulo");
    fim se;
fim.
```

Algoritmo

```
início
    real: A, B, C;
    leia (A, B, C);
    se ((A < B+C) e (B<A+C) e (C<A+B))  então
    senão
        escreva ("Não formam triângulo");
    fim se;
fim.
```

Primeiro SE irá testar a condição:
"É triângulo: (A < B+C) e (B < A+C) e (C<A+B)"

Algoritmo

```
início
    real: A, B, C;
    leia (A, B, C);
    se ((A < B+C) e (B<A+C) e (C<A+B))  então
        se ((A = B) e (B=C)) então
            escreva ("Triângulo Equilátero");
        senão
        fim se;
    senão
        escreva ("Não formam triângulo");
    fim se;
fim.
```

Segundo SE irá testar a condição:
"É equilátero: (A = B) e (B = C)"

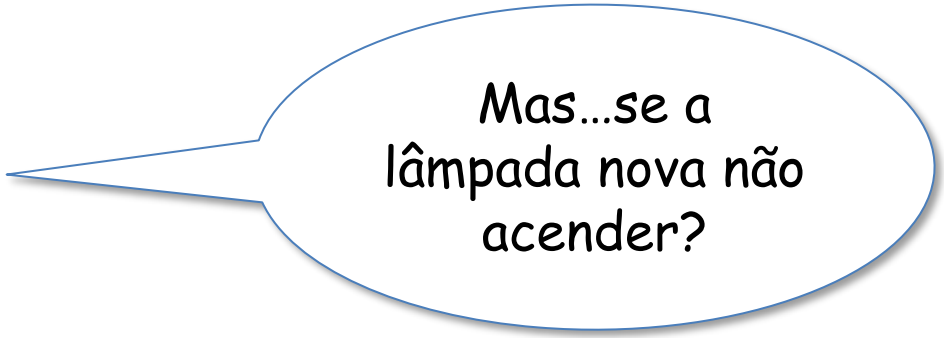
Algoritmo

```
início
    real: A, B, C;
    leia (A, B, C);
    se ((A < B+C) e (B<A+C) e (C<A+B))  então
        se ((A = B) e (B=C)) então
            escreva ("Triângulo Equilátero");
        senão
            se ((A = B) ou (A=C) ou (B=C))   então
                escreva ("Triângulo Isósceles");
            senão
                escreva ("Triângulo Escaleno");
            fim se;
        fim se;
    senão
        escreva ("Não formam triângulo");
    fim se;
fim.
```

Terceiro SE irá testar a condição:
É isósceles: (A=B) ou (A=C) ou (B=C) senão,
só pode ser escaleno

RESOLVER A LISTA 03

Até agora vimos que existem várias formas de resolver um problema, entretanto, voltando ao exemplo da lâmpada...



Poderíamos adicionar vários testes, pois, se a condição "a lâmpada não acende" for verdadeira, temos a opção de retirarmos a lâmpada e colocarmos outra, assim, Algoritmo ficará:

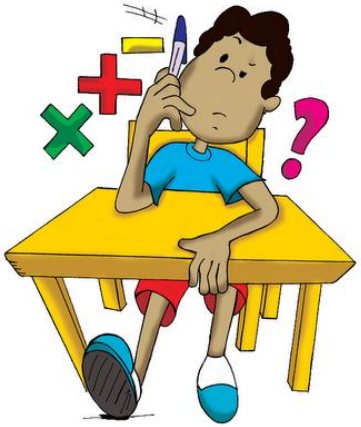
Algoritmo

Início

- Acionar o interruptor;
- SE a lâmpada não acender, então
  - Início do bloco
    - Pegar escada;
    - Posicionar escada embaixo da lâmpada;
    - Buscar nova lâmpada;
    - Subir escada;
    - Retirar lâmpada velha;
    - Colocar a lâmpada nova;
    - SE a lâmpada não acender, então
      - Início do bloco
        - Retirar lâmpada;
        - Colocar a lâmpada nova;
      - fim do bloco
  - Fim do bloco
- Fim do SE

Final

Estruturas de Controle



Mas...se a nova lâmpada também não acender?

Algoritmo

Início

- Acionar o interruptor;
- SE a lâmpada não acender, então

  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
  - SE a lâmpada não acender, então

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco
  - SE a lâmpada não acender, então

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

  Fim do bloco
- Fim do SE

Final

Mas... e se essa lâmpada também não acender?

Algoritmo

Início

- Acionar o interruptor;
- SE a lâmpada não acender, então

  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
  - SE a lâmpada não acender, então

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco
  - SE a lâmpada não acender, então

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco
  - SE a lâmpada não acender, então

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

  Fim do bloco
- Fim do SE

Final

Mas... e se a lâmpada não acender de novo?

Algoritmo
Início

- Acionar o interruptor;
- SE a lâmpada não acender, então
  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco

  Fim do bloco
- Fim do SE

Final

Observe que os SEs repetem-se várias vezes, portanto, deve haver uma estrutura de controle que repita esse processo, para o código não ficar muito grande, sujeito à erros e de difícil manutenção.

Assim, não é necessário escrever várias vezes essas ações. Poderemos alterar o fluxo de execução utilizando uma Estrutura de Repetição:

- **ENQUANTO** a lâmpada não acender **faça**
  - Retirar lâmpada;
  - Colocar a lâmpada nova;

Algoritmo

Início

- Acionar o interruptor;
- SE a lâmpada não acender, então
  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco
    - SE a lâmpada não acender, então
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco

  Fim do bloco
- Fim do SE

Final

Algoritmo

Início

- Acionar o interruptor;
- SE a lâmpada não acender, então
  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
    - Enquanto a lâmpada não acender faça
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco

  Fim do bloco
- Fim do SE

Final

Estrutura de Repetição

Repetição com teste no Início

É uma estrutura da controle do fluxo de execução que permite repetir diversas vezes um mesmo trecho do algoritmo, porém, sempre verificando ANTES de cada execução se é 'permitido' executar o mesmo trecho. O Enquanto permite que um determinado comando (ou bloco) seja repetido enquanto uma determinada <*condição*> for VERDADEIRA.

Pseudocódigo

```
Enquanto <condição> faça
Início do bloco
        C1;
        C2;
        C3;
        C4;
Fim do bloco
Fim Enquanto
```

R

```
while(condição)
{
        C1
        C2
        C3
        C4
}
```

Estrutura de Repetição

Repetição com teste no Início

OBS: Quando o resultado da <condição> for FALSO o comando de repetição é abandonado. Se na primeira vez o resultado for FALSO, os comandos NÃO SÃO EXECUTADOS.

Para exemplificar, construa um algoritmo que enquanto o usuário não digitar um número menor que 10, seja novamente pedido para o usuário digitar um número menor que 10.

```
Algoritmo
início
        inteiro: n
        leia (n);
        Enquanto (n>=10) faça
        Início do bloco
                leia (n);
        Fim do bloco
        Fim enquanto
Fim
```

Estrutura de Repetição

Repetição com teste no Início

OBS: Muitas vezes precisamos estabelecer um modo de contagem (contador), ou seja, uma variável (i, j ou k, por exemplo) com um dado valor inicial que é incrementado a cada repetição.

Exemplo de contador

```
Inteiro: cont; // declaração do contador
         cont← 0; // inicialização do contador
         cont ← cont+1; // incrementar o contador em uma unidade
```

Construa um algoritmo que pergunte o nome do usuário e permita a escolha entre um número (n) de 1 a 10 e escreva n vezes o nome na tela.

Estrutura de Repetição

Repetição com teste no Início

```
Algoritmo
início
        caractere: nome;
        inteiro: n, i;
        leia (nome, n);
         i ← 0;
        Enquanto (contador<n) faça
        Início do bloco
                escreva (nome);
                i ← i + 1;
        Fim do bloco
        fim Enquanto
Fim
```

Estrutura de Repetição

Repetição com teste no Final

Uma das estruturas com teste no final é a repita...se, que permite que um comando, ou bloco de comandos sejam executados se uma determinada <*condição*> seja FALSA.

Verificamos que, devido a sua sintaxe os comandos dentro do bloco *São Executados Pelo Menos Uma Vez*, independentemente da validade da condição. Isso ocorre pois a inspeção da <*condição*> ocorre no FINAL da estrutura.

Pseudocódigo

```
Repita
Início do bloco
        C1;
        C2;
        C3;
        C4;
Fim do bloco
se <condição> interrompa;
```

R

```
repeat
{
        C1
        C2
        C3
        C4
if(condição){break}
}
```

Estrutura de Repetição

Repetição com teste no Final

OBS: Quando o resultado da <condição> for FALSO o comando de repetição é abandonado. Entretanto, os comandos SÃO EXECUTADOS PELO MENOS UMA VEZ.

Reconfigure algoritmo que enquanto o usuário não digitar um número menor que 10, seja novamente pedido para o usuário digitar um número menor que 10.

```
Algoritmo
início
        inteiro: n
        leia (n);
        Faça
        Início do bloco
                leia (n);
        Fim do bloco
        Enquanto (n>=10) ;
Fim
```

Estrutura de Repetição

Repetição com teste no Final

OBS: Muitas vezes precisamos estabelecer um modo de contagem (contador), ou seja, uma variável (i, j ou k, por exemplo) com um dado valor inicial que é incrementado a cada repetição.

Exemplo de contador

```
Inteiro: cont; // declaração do contador
         cont← 0; // inicialização do contador
         cont ← cont+1; // incrementar o contador em uma unidade
```

Reconfigure algoritmo que pergunta o nome do usuário e permite a entre um número (n) de 1 a 10 e escreva n vezes o nome na tela.

voltando ao exemplo da lâmpada...

Mas...e se inicialmente, ao invés de uma lâmpada, tivéssemos 10 lâmpadas para testar? E trocar se necessário?

A solução mais óbvia seria repetir o algoritmo de uma única lâmpada para os dez soquetes...e adicionar a instrução

- **Ir para a lâmpada 1;**

Estruturas de Controle

Algoritmo
Início

- Ir para a lâmpada 1; // primeira lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então
  Início do bloco
  - Pegar escada;
  - Posicionar escada embaixo da lâmpada;
  - Buscar nova lâmpada;
  - Subir escada;
  - Retirar lâmpada velha;
  - Colocar a lâmpada nova;
    - Enquanto a lâmpada não acender faça
      Início do bloco
      - Retirar lâmpada;
      - Colocar a lâmpada nova;

      fim do bloco

  Fim do bloco
- Fim do SE

Final

Algoritmo
Início

- Ir para a lâmpada 1; // primeira lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 2; // segunda lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 3; // terceira lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 4; // quarta lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 5; // quinta lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 6; // sexta lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 7; // sétima lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 8; // oitava lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 9; // nona lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

- Ir para a lâmpada 10; // décima lâmpada
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
  - Enquanto a lâmpada não acender faça

    Início do bloco
    - Retirar lâmpada;
    - Colocar a lâmpada nova;

    fim do bloco

Fim do bloco

- Fim do SE

Final

Para resolver esse problema podemos utilizar outra estrutura, a PARA.

Algoritmo
Início
- Para lâmpada = 1 até 10

Início Bloco
- Ir para a lâmpada;
- Acionar o interruptor;
- SE a lâmpada não acender, então

Início do bloco
- Pegar escada;
- Posicionar escada embaixo da lâmpada;
- Buscar nova lâmpada;
- Subir escada;
- Retirar lâmpada velha;
- Colocar a lâmpada nova;
- Enquanto a lâmpada não acender faça

Início do bloco
- Retirar lâmpada;
- Colocar a lâmpada nova;

fim do bloco

Fim do bloco
- Fim do SE

Fim Bloco
- Próxima lâmpada

Final

Estrutura de Repetição

Repetição com Variável de Controle

As estruturas **Enquanto** e **repita** ocorrem em casos de difícil determinação do número de vezes que um comando, ou bloco, será executado. A Estrutura **Para** é diferente, já que sempre repete a execução do bloco um número predeterminado de vezes, ***pois ela não prevê uma condição*** e possui **limites fixos**.

Pseudocódigo

```
Para i = 1 até i = n  passo = 1
Início do bloco
    C1;
    C2;
    C3;
    C4;
Fim do bloco
Próximo i
```

C/C++

```
for(i in 1:n)
{
        C1
        C2
        C3
        C4
}
```

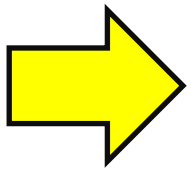
Estrutura de Repetição

Repetição com Variável de Controle

Repetição para uma ação primitiva apenas

Pseudocódigo

```
Para i = 1 até i = n  passo = 1
    C1;
Próximo i
```



```
for(i in 1:n) C1
```

Estrutura de Repetição

Elabore um algoritmo que, utilizando uma das estruturas de repetição imprima a tabuada do número 5.

```
Algoritmo
inicio
        Inteiro: i;
        para i de 1 até 10 passo 1 faça
                escreva (i, " x 5 = ", i*5);
        Próximo i;
fim.
```

RESOLVER A LISTA 04