ALGORITMOS

Prof. Nilton

Aula de hoje

- Acumuladores e contadores
- Estruturas de Repetição
  - Laços Condicionais
  - Laços Contados

Acumuladores e Contadores

Um **acumulador** é uma variável que ocorre em ambos os lados de uma atribuição e que, antes de ser usada pela primeira vez, é iniciada com um valor específico.

total <- **total** + soma

Um **contador** é um tipo de acumulador cujo valor aumenta, ou diminui, de 1 em 1 ou de um valor constante.

contador <- **contador** + 1

Estruturas de Repetição

São muito comuns as situações em que se deseja repetir um determinado trecho de um programa certo número de vezes. Essas situações não são repetidas eternamente, mas se encerram quando o objetivo é atingido.

Exemplo:

- Apertar um parafuso
- Chamada feita por um professor

Existem outras situações que podem ser quantificadas com antecedência. Ex. O aluno de castigo que precisa escrever 100 vezes no quadro “vou me dedicar à disciplina de algoritmo”

Estruturas de Repetição

Todas as repetições tem uma característica comum: o fato de haver uma verificação de condição que pode ser representada por um valor lógico, para determinar se a repetição prossegue ou não.

As estruturas de repetição são muitas vezes chamadas de Laços ou, também, de Loop

Estruturas de Repetição

A classificação das estruturas de repetição é feita de acordo com o conhecimento prévio do número de vezes que o conjunto de comandos será executado. Assim, os laços dividem-se em:

- **laços condicionais**, quando não se conhece de antemão o número de vezes que o conjunto de comandos no interior do laço será repetido, pelo fato de o mesmo estar amarrado a uma condição sujeita à modificação pelas instruções do interior do laço.

- **laços contados**, quando se conhece previamente quantas vezes o comando composto no interior da construção será executado.

Estruturas de Repetição

Laços Condicionais

Laços condicionais são aqueles cujo conjunto de comandos em seu interior é executado até que uma determinada condição seja satisfeita. Ao contrário do que acontece nos laços contados, nos laços condicionais não se sabe de antemão quantas vezes o corpo do laço será executado.

As construções que implementam laços condicionais mais comuns nas linguagens de programação modernas são:

- Enquanto (Pre-Condição)
- Repita (Pós-Condição)

Enquanto

Vamos ver um problema do mundo real: elevador

Um elevador residencial tem um comportamento que pode ser descrito de forma algorítmica. Vejamos seu funcionamento:

Na subida: sobe cada andar, verificando se está em um andar selecionado dentro do elevador. Isso é feito até chegar ao andar mais alto selecionado dentro ou fora do elevador.

Enquanto

Enquanto não chegar ao andar mais alto selecionado **faça**

Inicio

    suba um andar

    se o andar foi selecionado **então**

    inicio

        pare;

        abra a porta;

        feche a porta;

    fim

fim

Enquanto

Na descida: desce cada andar, verificando se está em um andar selecionado. Isso é feito até chegar ao andar mais baixo selecionado.

Enquanto não chegar ao andar mais baixo selecionado **faça**
Inicio
    desça um andar
    se o andar foi selecionado **então**
    inicio
        pare;
        abra a porta;
        feche a porta;
    fim
fim

Enquanto..faça

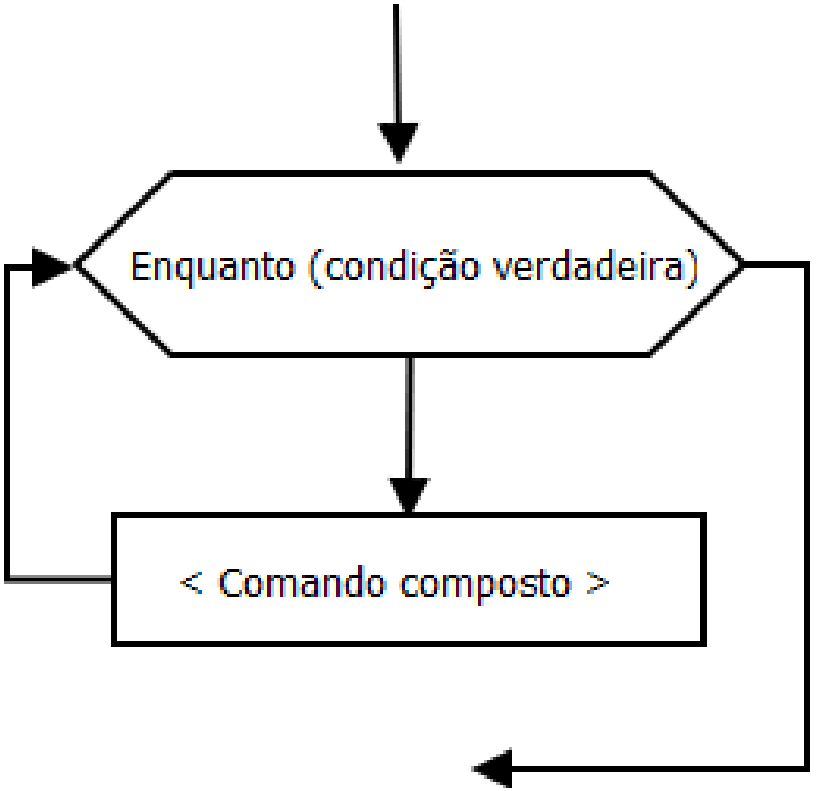
O comando enquanto caracteriza-se por uma verificação de encerramento de atividades antes de se iniciar (ou reiniciar) a execução de seu bloco de instruções.

Sintaxe

Enquanto < valor booleano) **faça**

< bloco de instruções >

Fim_enquanto



Enquanto..faça

PSEUDOCÓDIGO

```
Algoritmo Licao_Aluno
Var contador: inteiro
Inicio
  contador ← 0;
  enquanto (contador < 100) faca
     escreval (‘Vou me dedicar a disciplina de algoritmos’);
     contador ← contador + 1;
  fimenquanto
fim
```

Repita..até

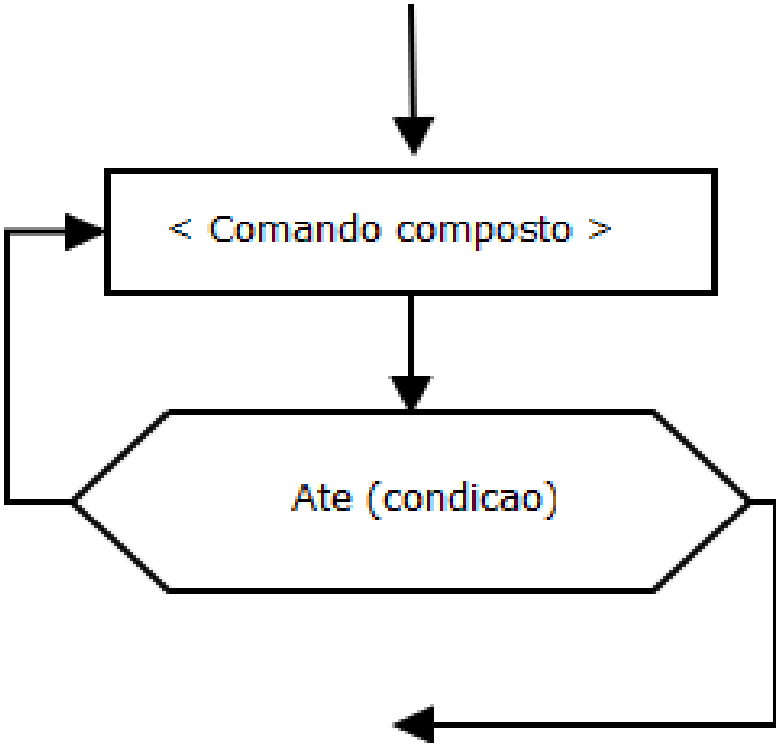
O comando **repita** funciona de forma similar ao comando **enquanto**, exceto pelo fato de que a condição de controle só é testada após a execução do bloco de comandos, e não antes, como é o caso do comando **enquanto.**

Sintaxe

repita

< bloco de instruções >

até < valor booleano >



Repita..até

Assim podemos utilizar o comando repita sempre que tivermos certeza de que o bloco de instruções será executado ao menos uma vez, sem a necessidade de teste na entrada do bloco.

Repita..até

PSEUDOCODIGO

```
Algoritmo Pergunta
Var resp : caracter;
Inicio
  resp ← 'S';
  repita
    escreva('Deseja continuar?');
    leia(resp);
  até ( resp = 'N');
fim
```

Estruturas de Repetição

Laços Contados

Os laços contados são úteis quando se conhece previamente o número de vezes que se deseja executar um determinado conjunto de comandos. Então, este tipo de laço nada mais é que uma estrutura dotada de mecanismos para contar o número de vezes que o corpo do laço (ou seja, o comando composto em seu interior) é executado.

Estruturas de Repetição

Se analisarmos os exemplos de utilização dos laços, percebemos que a maioria deles tem comportamento similar. Uma situação inicial, definida antes do inicio do laço, um teste de controle para a entrada/saída do bloco e uma instrução dentro do laço que, em algum momento fará com que a condição de controle seja atingida e o laço se encerre.

O comando **para** procura resumir essas três características comuns à maioria das implementações de laços em uma só instrução, facilitando assim a construção típica de laços.

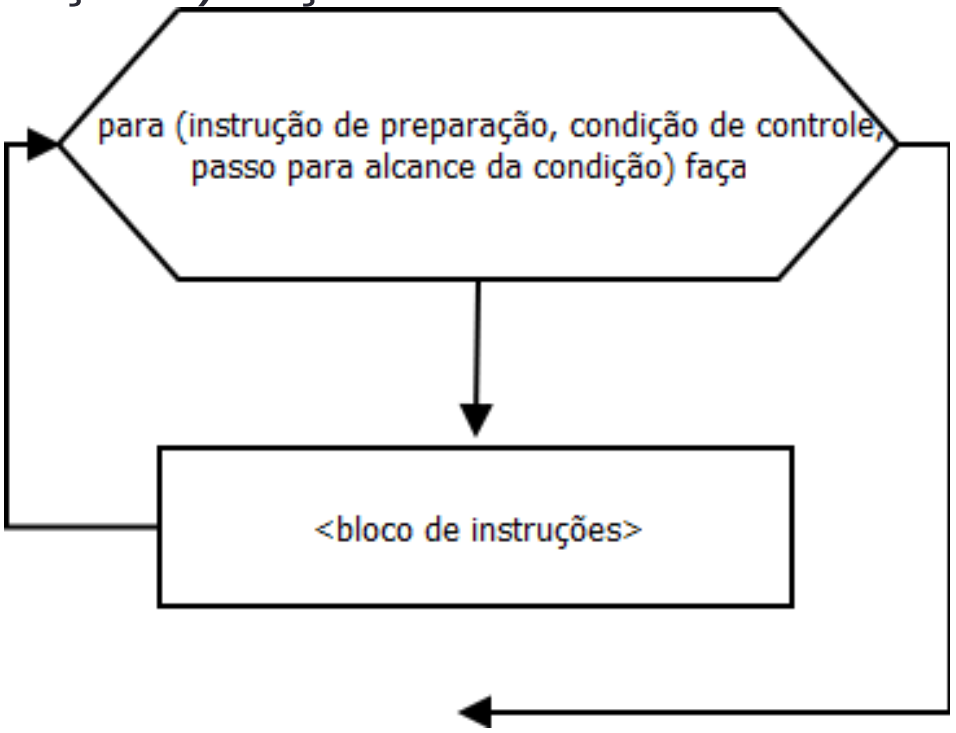
Para..até

Sintaxe

Para (<instrução de preparação>, <condição de controle>, <passo para alcance da condição>) faça

    <comando_composto>

Fim_para

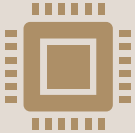


Para..até

PSEUDOCÓDIGO

```
Algoritmo Conte_10
Var cont : inteiro;
Inicio
    para cont de 1 ate 10 faca
        escreval (‘Numero: ’, cont);
    fimpara
fim
```

Exercícios – Enquanto..faça



1) Criar um algoritmo que receba um nome no teclado e imprima-o dez vezes;

2) Criar um algoritmo que leia 50 números e retorne a soma e a média desses valores.

3) Criar um algoritmo que leia as notas de um aluno e calcule a media final. O algoritmo deve continuar lendo as notas até que seja digitado -1.

Exercício Repita..até

4) Altere o exercício 1 do capítulo anterior criando uma nova opção (4 - Sair), fazendo com que o menu de opções seja exibido até que seja digitado a opção de saída.

5) Faça um algoritmo capaz de fazer uma contagem inteligente, ou seja, o usuário informará o numero inicial e final, o algoritmo identificará se a contagem é progressiva ou regressiva e contará os números desse intervalo.

Exercícios – Para..até

6) Criar um algoritmo que **leia os limites inferior e superior** de um intervalo e imprima todos os números no intervalo aberto e sua somatório. Suponha que os dados digitados são para um intervalo crescente;

7) Faça um algoritmo que conte de 1 a 100 e a cada múltiplo de 10 emita uma mensagem: “Múltiplo de 10”.

8) Altere o exercício anterior fazendo que seja impresso apenas os números pares do intervalo;