Estruturas de Repetição

Eldrey Galindo

Lembre-me:

- ✓ **Chamada** está disponível no Classroom
- ✓ Aula está sendo gravada e estará disponível para os alunos que solicitarem na secretaria.
- ✓ Abra a câmera

Estruturas de repetição

Diferente do ser humano, o computador não se cansa de realizar operações repetitivas.

Diante disto, podemos incluir nos algoritmos as estruturas de repetição.

Introdução

O que é uma repetição?

- Também chamada a laço ou loop
- É uma instrução que permite a execução de um trecho de algoritmo várias vezes seguidas.

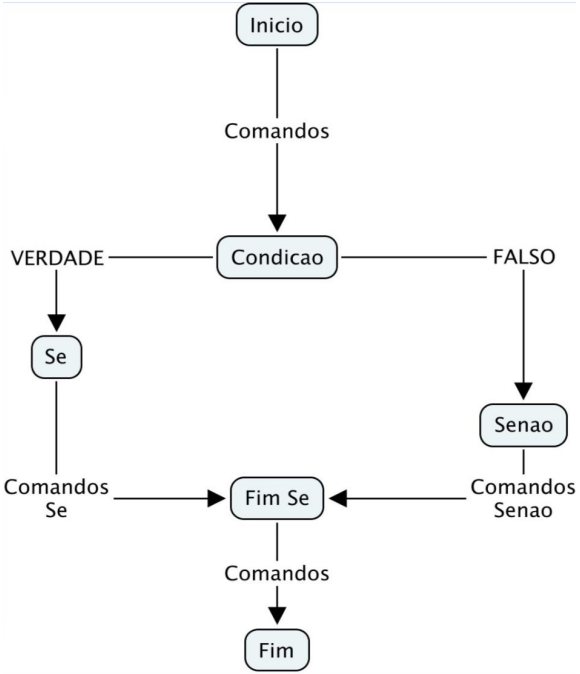
Com estruturas de repetição é possível:

- Repetir determinado trecho de código;
- Poupar linhas de código;
- Tornar o programa mais robusto, legível.

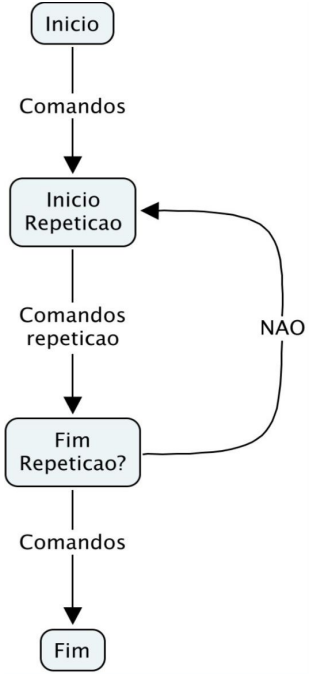
Estruturas de repetição

Sequencial

Condicional



Repetição



Tipos de estruturas

Existem três tipos de estruturas de repetição:

- Com teste ao início

  enquanto condicao { ... }

- Com teste ao final

  faca { ... } enquanto condicao

- Com variável de controle

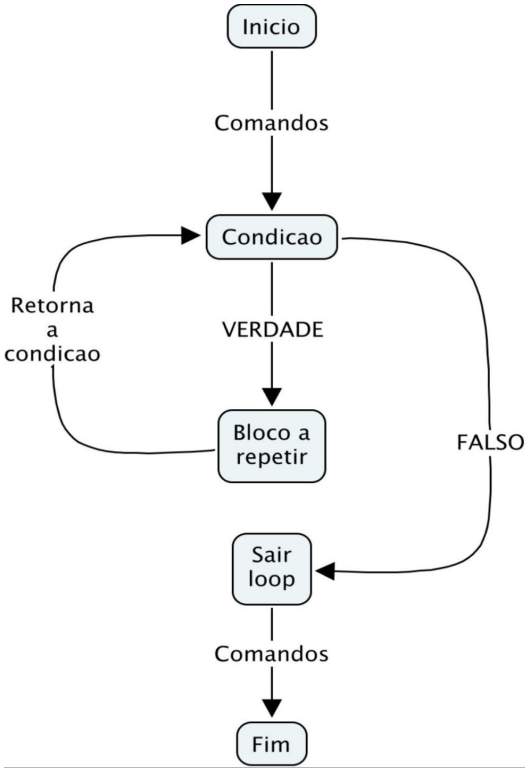
  para (variavel_controle;condicao;alteracao_variavel_controle) { ... }

Enquanto

A instrução enquanto repete uma determinada instrução enquanto uma determinada condição for verdadeira.

- Sintaxe:

```
enquanto condicao {
        comando a ser executado
        comando a ser executado
}
```



Enquanto

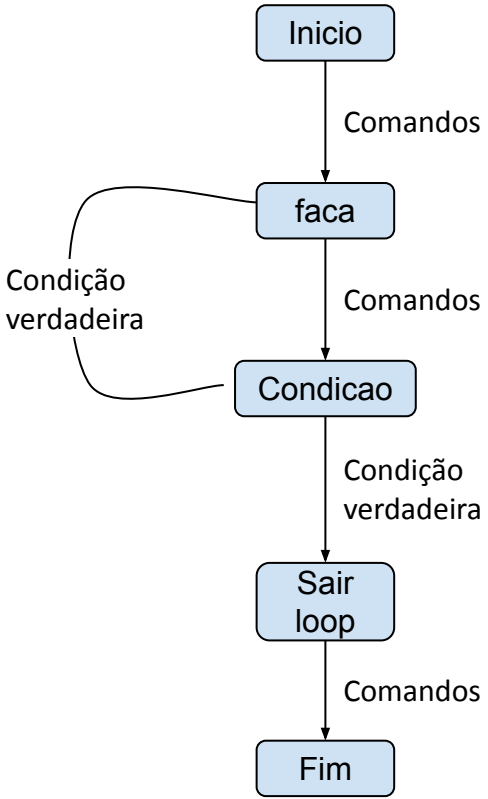
- Funciona semelhante ao se
  - Primeiro, avalia a CONDICAO
  - Se verdadeiro, executa os comandos do escopo do enquanto
  - Se falso, nada do escopo do enquanto é executado.
- Ao final da execução de todos os comandos do enquanto, retorna à CONDICAO e a reavalia
  - Se ainda verdadeiro, repete o processo
  - Se falso, sai do laço e continua o programa com os comandos após o enquanto.
- Esta estrutura é chamada de repetição com teste no início
  - A decisão entre repetir e parar o laço é feita no início do comando.
  - Se o teste for falso desde o início, o laço não será executado.

Faca - Enquanto

A instrução faca - enquanto repete uma determinada instrução até uma determinada condição for verdadeira.

- Sintaxe:

```
faca {
        comando a ser executado
        comando a ser executado
}enquanto(condicao)
```



Faça ... Enquanto

- O comando faca ... enquanto indica que todos os comandos entre a palavra faca e a palavra enquanto encontram-se dentro de um laço e devem ser executados.
- Após a palavra enquanto deve haver uma expressão lógica que:
  - Se verdadeira, indica que o laço deve ser executado novamente;
  - Se falsa, indica que o laço acabou e o algoritmo deve continuar sendo executado.
- Esta estrutura é chamada de repetição com teste no final
  - A decisão entre repetir e parar o laço é feita ao final do comando.
  - É garantido que o laço será executado aos menos uma vez.

Comandos enquanto e faca ... enquanto

Exemplo:

- Fazer um algoritmo para ler diversos números informados pelo usuário, e após cada leitura exibir se o número é par ou ímpar. Considere que ao fornecer o valor 0 o usuário deseja encerrar e entrada de dados.

Questões

- Quantos dados serão fornecidos?
- Quantas variáveis serão necessárias?
- Temos que identificar o trecho que será repetido e adicioná-lo dentro de um comando de repetição, declarando apenas as variáveis necessárias para uma repetição.

Comandos enquanto e faca ... enquanto

```
funcao inicio()
{
    inteiro num = 1

    enquanto(num != 0){
        escreva("\nInforme um número (0 - para sair)\n")
        leia(num)
        se(num % 2 == 0){
            escreva("o número informado é PAR\n")
        }senao{
            escreva("O número informado é IMPAR\n")
        }
    }

    escreva("Espero ter ajudado, TCHAU")
}
```

```
funcao inicio()
{
    inteiro num
    faca{
        escreva("\nInforme um número (0 - para sair)\n")
        leia(num)
        se(num % 2 == 0){
            escreva("o número informado é PAR\n")
        }senao{
            escreva("O número informado é IMPAR\n")
        }
    }enquanto(num != 0)

    escreva("Espero ter ajudado, TCHAU")
}
```

Variáveis contadoras

- Uma variável é chamada de contadora quando armazena dentro de si um número referente a uma certa quantidade de elementos ou iterações.
- Este tipo de variável é muito comum em estruturas de repetição, dada as suas diversas aplicações em problemas que envolvem contagens de valores.

Variáveis contadoras

```
funcao inicio()
{
    inteiro num, count = 0
    faca{
        escreva("\nInforme um número (0 - para sair)\n")
        leia(num)
        se(num % 2 == 0){
            escreva("o número informado é PAR\n")
        }senao{
            escreva("O número informado é IMPAR\n")
        }

        count ++ // count = count +1  OU count += 1

    }enquanto(num != 0)

    escreva("Espero ter ajudado todas as ", count ," vezes, TCHAU")
}
```

Loop Infinito

Mostrar os números de 1 a 100 na tela

```
funcao inicio()
{
	inteiro numero
	numero = 1
	enquanto(numero <= 100) {
		escreva(numero)
	}
}
```

Loop Infinito

O exemplo anterior realiza um loop infinito:

- Esse programa nunca termina
- O valor da variável numero inicia com 1
- A condição para repetir é que numero seja menor ou igual a 100
- Mas numero não muda
- Deve-se aumentar valor de numero ao final do bloco de comandos do enquanto
- Condição de parada “manual”

Variáveis contadoras

Ler 30 números inteiros fornecidos pelo usuário, e exibir quantos números ímpares foram informados.

- Solução
  - Serão necessárias 30 leituras. Podemos criar uma variável contadora para controlar este laço.
  - Precisaremos de uma outra variável para contar a quantidade de números ímpares.
    - Não podemos deixar para contar “no final”, pois cada número fornecido apaga o anterior. Logo precisamos ir contando após cada entrada, incrementando uma nova variável contadora.
    - Esta nova variável contadora só é incrementada se o número informado for ímpar.

Variáveis contadoras

- Solução

```
funcao inicio()
{
	inteiro cont, num, qtd_impar
	cont = 1
	qtd_impar = 0
	enquanto(cont <= 30) {
		escreva("Digite um número: ")
		leia(num)
		se(num % 2 != 0) {
			qtd_impar++
		}
		cont++
	}
	escreva("O total de ímpares foi ", qtd_impar)
}
```

Variáveis acumuladoras

- Uma variável é chamada de acumuladora quando tem por característica armazenar dentro de si o resultado acumulado de uma série de valores.

- Quando armazenamos a soma de uma quantidade pequena de números, a atribuição é direta. Numa repetição devemos armazenar a soma de diversos números sucessivos, e para isto utilizamos uma variável acumuladora.

Variáveis acumuladoras

Calcular a soma de diversos números reais informados pelo usuário. A entrada de dados termina com o número -999.

- Solução

```
funcao inicio()
{
    inteiro num, soma
    soma = 0
    escreva("Para sair informe -999")
    escreva("\nEntre com um número: ")
    leia(num)
    enquanto(num != -999) {
        soma = soma + num //soma += num
        escreva("Entre com um número: ")
        leia(num)
    }
    escreva("A soma foi ", soma)
}
```

Comando para

- É muito comum a existência de repetições que fazem uso de variáveis contadoras, especialmente para contagens de **N em N**.
- Para facilitar a construção deste tipo de laço, pode-se utilizar um outro comando de repetição complementar chamada para.
- Sintaxe:

```
para (variavel_controle; condicao; alteracao_variavel_controle) {
        comando a ser executado
        comando a ser executado
}
```

Comando para

```
funcao inicio()
{
    inteiro num = -1, count = 0
    para(count; num != 0; count++){
        escreva("\nInforme um número (0 - para sair)\n")
        leia(num)
        se(num % 2 == 0){
            escreva("o número informado é PAR\n")
        }senao{
            escreva("O número informado é IMPAR\n")
        }

    }

    escreva("Espero ter ajudado todas as ", count ," vezes, TCHAU")
}
```

Repetições encadeadas

- Da mesma forma que é permitido o encadeamento de instruções de decisão, também é possível encadear comandos de repetição.

- Um encadeamento de repetições ocorre quando há necessidade de efetuar um laço dentro de outro.

- Neste tipo de situação, o algoritmo possui repetições controladas por um teste interno e outro externo.

Repetições encadeadas

Imprimir as tabuadas de multiplicação dos números 3, 4, 5 e 6.

- Solução

```
funcao inicio()
{
	inteiro num, mult, cont

	num = 3
	enquanto(num <= 6) {
		escreva("\nTabuada de ", num)
		cont = 1
		enquanto(cont <= 10) {
			mult = num * cont
			escreva("\n", num, "x", cont, " = ", mult)
			cont++
		}
		num++
	}
}
```

Estruturas de repetição

Vamos praticar?

1. A empresa Programadores Felizes possui alguns funcionários. faça um algoritmo que deverá ler a quantidade de funcionários da empresa e o valor do salário de cada um deles e por fim, informar a média salarial da empresa.
2. Altere o algoritmo anterior para que, além da média, ele apresente o maior e o menor salário informados.
3.

Estruturas de repetição