Nexaas

TECNOLOGIA DA INFORMAÇÃO
Lógica de Programação

Paulo S R Rios
paulinhorios@gmail.com

Lógica de Programação

Estruturas de Repetição: São blocos de códigos que podem ser repetidos um determinado número de vezes ou até que uma determinada condição verificada a cada repetição seja satisfeita.

para (contador ; sentença; incremento) { … }

enquanto { ... }

faca { … } enquanto (sentença)

Lógica de Programação

para (contador ; sentença; incremento) { … }:
Este tipo de repetição define um ponto de começo e uma quantidade definida de repetições com base em incrementos:

```
para (contador = 1; contador <= 10; contador++)
{

}
```

Lógica de Programação

```
Programa {
funcao inicio()   {
            inteiro numero, resultado, contador
            escreva("Informe um número para ver sua tabuada: ")
            leia(numero)
            limpa()
            para (contador = 1; contador <= 10; contador++)
            {
            resultado = numero * contador
            escreva (numero, " X ", contador, " = ", resultado , "\n")
            }
            }
}
```

Lógica de Programação

Exercício 1: Faça um algoritmo um numero inteiro e o programa exibe as tabuadas de 1 a 10.

Exemplo de saída de dados:

1 x 1 = 1	1 x 2 = 2	1 x 3 = 3	...
2 x 1 = 2	2 x 2 = 4	2 x 3 = 6	...
3 x 1 = 3	3 x 2 = 6	3 x 3 = 9	...
4 x 1 = 4	4 x 2 = 8	4 x 3 = 12	...
5 x 1 = 5	5 x 2 = 10	5 x 3 = 15	...
6 x 1 = 6	6 x 2 = 12	6 x 3 = 18	...
7 x 1 = 7	7 x 2 = 14	7 x 3 = 21	...
8 x 1 = 8	8 x 2 = 16	8 x 3 = 24	...
9 x 1 = 9	9 x 2 = 18	9 x 3 = 27	...
10 x 1 = 10	10 x 2 = 20	10 x 3 = 30	...

Enquanto { … }

Este tipo de repetição faz uma checagem inicial antes de iniciar a repetição. Caso a sentença seja verdadeira ele inicia a repetição.

```
contador = 0
enquanto(contador <= 10)
{
        escreva(“Repetição = ”, contador)
        contador = contador + 1
 }
```

Enquanto { … }

Caso a sentença seja falsa logo na primeira verificação da sentença o bloco de repetição não é iniciado.

```
contador = 11
enquanto(contador <= 10)
{
        escreva(“Repetição = ”, contador)
        contador = contador + 1
}
```

Lógica de Programação

Exercício 2: Faça um algorítimo onde o usuário informa a senha de autorização, enquanto a senha for incorreta o sistema deve exibir a mensagem “Senha Incorreta” até a senha correta ser informada exibindo a mensagem “Autorizado”.

Faca { … } enquanto (sentença)

Neste modelo de repetição o bloco de repetição é iniciado e ao final é feita a verificação. É útil quando é necessário que o código seja executado pelo menos 1 vez antes da checagem para uma nova repetição.

```
faca
{
        escreva(“1 – para continuar \n”)
        escreva(“2 – para sair \n”)
        leia (opc)
} enquanto (opc <> 2)
```

Lógica de Programação

Exercício 3: Um menu de opções pode ser simulado utilizando uma estrutura de repetição e uma estrutura de decisão para realizar blocos de códigos.

Faça um algoritmo onde o usuário informa a opção desejada e o algoritmo realiza a operação aritmética entre dois numerou informados pelo usuário.

1 – Realizar uma Soma
2 – Realizar uma subtração
3 – Realizar uma Multiplicação
4 – Realizar uma Divisão
0 – Sair do Programa

Lógica de Programação

Exercício 4: Faça um algoritmo onde o usuário informa a quantidade de dias e o total em GB de dados trafegado na rede naquele dia. O programa deve continuar com a leitura de cada dia até exibir o total de dados ao finalizar o ultimo lançamento.

Exemplo:
Total de dias de análise = 3
Dia 1 = 50
Dia 2 = 90
Dia 3 = 30

Total do período = 170GB de dados trafegados.

Lógica de Programação

Exercício 5: Empresas de cartão de crédito possuem uma checagem de senha que permite o usuário errar apenas 3 vezes a senha antes de bloquear o cartão.

Faça um algoritmo que simule a autorização de venda em uma maquina de cartão.

Lógica de Programação

Exercício 6: Faça um algoritmo que simule um Oráculo onde o usuário pode fazer uma pergunta e o oráculo responde com SIM, NÃO ou TALVEZ.
O algoritmo ao final deve perguntar se deseja fazer mais perguntas e continuar a responder enquanto o usuário responder “sim” ao final.

*dica: adicione as bibliotecas Util. E utilize a biblioteca para gerar números aleatórios entre 1, 2 e 3

```
programa {
  inclua biblioteca Util
…
se(Util.sorteia(0,3)==1) { }
```