Desafio FAST - Lógica de Programação

Olá,

Esse é o Desafio FAST para os Bootcamps de Automação de Testes e Machine Learning. Leiam com atenção todos os pontos abaixo antes de iniciar.

Este desafio terá início às 13h da sexta-feira, 22/10, e será encerrada às 23:59 do domingo, 24/10.

Este desafio possui 9 (nove) questões, sendo 7 (sete) valendo 1,0 (um ponto) e 2 (duas) valendo 1,5 (um ponto e meio)

Atenção!!

\* Não existe uma única forma correta para construir um código.

\* No campo de e-mail, informe o mesmo e-mail utilizado na inscrição.

\* Verifique todas as questões antes de enviar seu desafio. Após enviado, não será possível alterá-lo.

Bom Desafio!

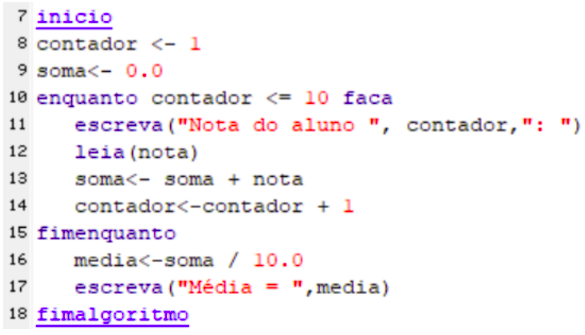
**Questionário**

* E-mail: [lhaislla@gmail.com](mailto:lhaislla@gmail.com)
* Declaro ter lido as instruções anteriores e ciente de que após o envio do formulário, não será possível alterá-lo. : Estou ciente.
* Nome Completo: Lhaíslla Eduarda Cavalcanti Rodrigues da Silva

**Perguntas**

1. [1,0] "As estruturas de repetição são usadas para controlar a execução de códigos, fazendo ele repetir um determinado bloco de ações até que uma condição seja verdadeira. Uma estrutura de repetição inviabiliza o uso de estruturas de decisão. São chamadas também de LOOPS ou LAÇOS". Podemos dizer que essa afirmação sobre estrutura de repetição é: \*
   1. Falsa (ok)
   2. Verdadeira

Explicação: Uma **estrutura de repetição** permite que um conjunto de comandos sejam executados **enquanto uma condição for verdadeira**. As estruturas de repetição também são chamadas de **laços** ou **loops**.Cada vez que o conjunto de ações do laço é repetido, acontece uma **iteração**.

1. Observe o pseudocódigo abaixo e escolha a alternativa CORRETA:   
   
   1. O pseudocódigo imprime a média de uma turma de 40 alunos.
   2. O pseudocódigo imprime a média de um aluno.
   3. O pseudocódigo imprime a nota de um aluno.
   4. O pseudocódigo imprime a média de uma turma de 10 alunos.
   5. N.D.A  
      Explicação:

cont =1

soma = 0.0

while cont <= 10:

print(“Nota de aluno”, cont, “:”)

leia = input(nota)

soma = soma +nota

cont = cont +1

break

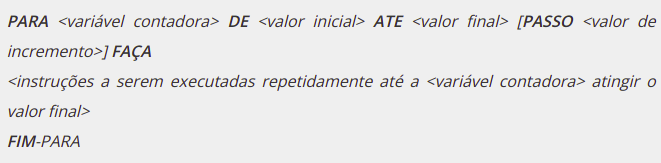
media = soma/ 10.0

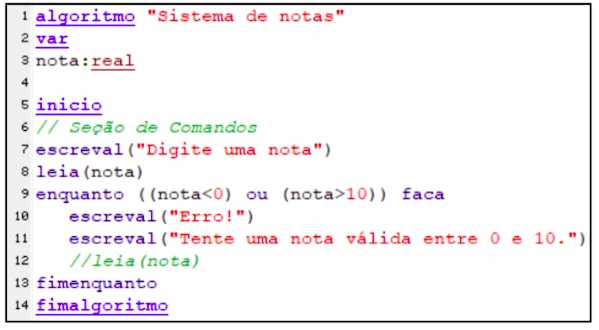
print(“Média =”,media)

1. O uso de vetores permite:
   1. Executar blocos de código quando uma condição é verdadeira. F
   2. Armazenar vários valores de um mesmo tipo em uma única estrutura; V
   3. Receber do usuário um único valor, dividido em várias partes; F
   4. Armazenar informações de outros programas; F
   5. Armazenar vários valores de diferentes tipos em uma única estrutura; F

Explicação: Um **vetor** é uma sequência de variáveis de mesmo tipo e referenciadas por um nome único.

1. As afirmações apresentadas abaixo podem ser verdadeiras. Leia com atenção e assinale a alternativa correta:

* O REPITA-ATE possibilita a execução repetitiva de comandos ATÉ que a condição de controle de repetição seja FALSA. F  
    
   Explicação: Na estrutura **REPITA-ATÉ**, as instruções do bloco são executadas repetidamente enquanto a expressão booleana resultar FALSO. A partir do momento que a expressão booleana resultar VERDADEIRO, o fluxo do algoritmo sairá do LOOP.
* O PARA-ATE-FACA utiliza uma variável de controle do tipo real para fazer com que a estrutura tenha uma condição de parada. V  
   Explicação: 
* Uma condição é feita com operadores lógicos e/ou relacionais. Logo, o resultado sempre será VERDADEIRO ou FALSO. V  
   Explicação: O valor de uma expressão lógica é ou VERDADEIRO ou FALSO.
* O comando ENQUANTO-FACA permite a execução repetitiva de comandos ENQUANTO a condição de controle de repetição for VERDADEIRA.V  
   Explicação: realiza a condição enquanto for verdadeira, caso não ele sai da condição e continua o código.

1. Observe o código abaixo e marque a alternativa que melhor descreve este algoritmo:
2. 
   1. Esse código não rodará nenhuma vez.
   2. Enquanto o valor de nota não for um número entre zero e dez o algoritmo dará um erro, mas esse loop irá parar quando o usuário digitar um valor válido. V
   3. O algoritmo entra em loop infinito imprimindo os comandos das linhas 10 e 11.
   4. Se o valor de nota for menor que zero ou maior que dez, o algoritmo executa os comandos das linhas 10 e 11 apenas uma vez.
   5. N.D.A
3. Sobre passagem de parâmetros por referência em uma função, é correto afirmar:
   1. Se dentro de uma função modificarmos o valor de uma variável passada por referência, essa mudança ocorre também na variável “original”.
   2. Os valores passados por parâmetro precisam ser definidos, obrigatoriamente, pelo usuário.
   3. O valor de uma variável passada por referência como parâmetro para uma função não pode ser modificado. V
   4. Se um dos parâmetros for passado por referência, todos os demais também precisam ser.
   5. A função que recebe parâmetros passados por referência não pode ter nenhum tipo de retorno.

Explicação: passagem de parâmetros para uma função ocorre quando alterações nos parâmetros formais, dentro da função, alteram os valores dos parâmetros que foram passados para a função. Este tipo de chamada de função tem o nome de "chamada por referência". Este nome vem do fato de que, neste tipo de chamada, não se passa para a função os valores das variáveis, mas sim suas referências (a função usa as referências para alterar os valores das variáveis fora da função).   
 Quando uma função precisa ser capaz de alterar os valores das variáveis usadas como argumentos, os parâmetros precisam ser explicitamente declarados como tipos ponteiros. Assim, utiliza-se a passagem por nome, quando um ponteiro é passado, ou de passagem por referência, quando referências são usadas. Em ambos os casos um endereço é manipulado. Os endereços são passados por duas razões. A primeira é simplesmente pela eficiência; sabendo-se que a variável será somente lida, então, geralmente é mais rápido passar o endereço quando se usam objetos maiores do que um ponteiro. Existe também um benefício adicional em se passar um endereço. No caso de objetos, passa-se o endereço para um objeto inicializado e o tamanho do endereço é sempre o mesmo, assim não há problema envolvendo cópia e inicialização. A segunda razão para passar um endereço é manipular fisicamente o objeto ou variável para o qual o endereço aponta, isto é, pode-se alterar uma variável ou objeto que é externo a uma função. Normalmente, os resultados de uma chamada de função devem ser expressos pelo valor de retorno da função, mas existem situações em que isso não é satisfatório. Por exemplo, a função pode precisar alterar mais de um objeto. Se os endereços do objeto são passados, a função pode afetar diretamente qualquer número de objetos na função chamadora.

1. Para utilizar uma estrutura de decisão no código de um determinado problema devemos observar:   
   1. O problema pede uma estrutura que possua análise de condições, retornando um valor lógico (verdadeiro ou falso). Caso a condição seja verdadeira o código executa uma determinada ação, caso seja falsa, outro comando será executado.
   2. O problema apresenta questões em que o ser humano deve pensar nas possíveis decisões e implementá-las no código a partir de um conjunto de dados. Com a estrutura de decisão, o programa executa apenas as decisões que são consideradas verdadeiras. As decisões falsas são descartadas do código, eliminando todas as variáveis.
   3. O problema apresenta uma estrutura em que algumas ações são repetitivas e, com a utilização de condicionais, podemos deixar o código mais simples. Além disso, possuem processos pelo qual o indivíduo deve tomar decisões e incrementar essas decisões no código.
   4. O problema não deve ser muito complexo, pois, caso seja, necessitará de estruturas de repetição ao invés de estruturas condicionais. Já que as estruturas de decisão são responsáveis apenas pela tomada de decisões simples.
   5. O problema deve apresentar apenas duas análises de condições, sendo um VERDADEIRO e o outro FALSO. A máquina irá processar as informações de forma booleana e a partir dos dados de entrada, tomará uma decisão.  
        
      Explicação: Uma estrutura de decisão examina uma ou mais condições e decide quais instruções serão executadas dependendo se a condição foi ou não foi.
2. [1,0] Podemos definir uma “variável” em programação como sendo:   
   1. Um objeto que existe no código e que não precisa ser armazenado em memória, pois seu valor pode ser alterado em qualquer parte do código. F
   2. Um objeto que não ocupa espaço em memória, mas que pode representar valores ou expressões. F
   3. É um espaço em memória ao qual podemos armazenar números, textos e caracteres sem precisar declará-los previamente. F
   4. Variável é um espaço em memória capaz de guardar uma informação, seja um valor ou uma expressão. V
   5. É um espaço em memória chamado também de constante, no qual ele não altera o valor durante a execução do código. F  
        
      Explicação: Uma variável é um espaço na memória do computador destinado a um dado que é alterado durante a execução do algoritmo. Para funcionar corretamente, as variáveis precisam ser definidas por nomes e tipos.
3. Sabendo que precisamos de uma estrutura de repetição para que execute o código pelo menos 1 vez antes da condição ser verificada (pós-testada), devemos escolher a:  
   1. Para
   2. Enquanto
   3. Repita até V
   4. Desde
   5. Se/Senão

Explicação: Uma outra estrutura de repetição interessante é a **REPITA-ATÉ** (Repeat-Until). A diferença desta estrutura é que ela é um **LOOP PÓS-TESTADO**, isto é, o teste para verificar se o bloco será executado novamente, acontece no final do bloco. Isso garante que as instruções dentro deste bloco serão executadas ao menos uma vez.

Referências: <https://www.treinaweb.com.br/blog/estruturas-condicionais-e-de-repeticao>