Desenho de algoritmos - pseudocódigo

Para cada um dos exercícios referido a seguir, segue o conjunto de comandos em pseudocódigo que, ao serem colocados pela ordem correta (pode existir mais do qe uma combinação possível), traduzem um algoritmo para resolver o exercício em questão. Assim sendo, coloque os comandos pela ordem que achar correta.

A. Instruções/acções de atribuição e leitura/escrita

1. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): introduza dois números inteiros, calcule a soma e a diferença entre eles e, por fim, mostre os dois resultados.

parametros de saida: S, D (inteiros)
escrever: "Soma = ", S
fim_algoritmo
escrever: "Diferenca = ", D
S <-- A + B
algoritmo somaDiferenca
ler: A
D <-- A - B
ler: B
parametros de entrada: A, B (inteiros)

5. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): introduza o preço base de um produto (valor real) e uma taxa de IVA (valor inteiro), calcule o preço final do produto e, por fim, mostre o preço final obtido.

escrever: "Preço Final = ", PF
ler: PB
parametros de saida: PF (real)
algoritmo precoProduto
ler: IVA
parametros de entrada: PB (real), IVA (inteiro)
fim_algoritmo
PF <-- PB + IVA / 100 x PB

9. Construa um algoritmo que que realize as seguintes ações (pela ordem indicada): introduza um número inteiro (associado a um tempo em segundos), converta este valor no formato HH:MM:SS (HH horas, MM minutos e SS segundos) e, por fim, mostre os 3 números em separado (primeiro as horas, depois os minutos e por fim os segundos).

```
SS <-- restoDivisão(T, 60)
fim_algoritmo
algoritmo tempoHHMMSS
HH <-- divisão(T2, 60)
parametros de entrada: T (inteiro)
T2 <-- divisão(T, 60)
ler: T
escrever: "Segundos = ", SS
MM <-- restoDivisão(T2, 60)
escrever: "Minutos = ", MM
parametros de saida: HH, MM, SS (inteiros)
escrever: "Horas = ", HH
```

10. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): introduza um número inteiro positivo com três dígitos, determine os dígitos que formam este número e, por fim, mostre estes dígitos separadamente e por ordem (centenas, dezenas e unidades). Exemplo: 937 é composto pelos algarismos 9, 3 e 7.

```
dezenas <-- restoDivisão(num, 10)
fim_algoritmo
ler: num
unidades <-- restoDivisão(num, 10)
escrever: "Algarismo das unidades: ", unidades
centenas <-- divisão(num, 10)
algoritmo algarismosNumero
escrever: "Algarismo das dezenas: ", dezenas
parametros de saida: centenas, dezenas, unidades (inteiros)
num <-- divisão(num, 10)
parametros de entrada: num (inteiro)
escrever: "Algarismo das centenas: ", centenas
```

B. Instruções/acções condicionais

2. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): peça ao utilizador que introduza três números inteiros, determine o maior deles e depois mostre o maior obtido.

```
parâmetros de entrada: A, B, C (inteiros)
se (C > maior) então
ler: A, B, C
maior <-- C
escrever: "Maior dos três números = ", maior
fim_se
senão
maior <-- B
se (A > B) então
algoritmo maiorDeTres
parâmetros de saida: maior (inteiro)
maior <-- A
fim_algoritmo
escrever: "Introduza três números inteiros:"
fim_se
```

3. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): peça ao utilizador que introduza dois números inteiros positivos não nulos (> 0), calcule o resto da divisão inteira do número maior pelo número menor e, por fim, mostre o resultado obtido.

```
fim_se
fim_algoritmo
algoritmo restoDivisaoMaior
se (M > N) então
parâmetros de entrada: M, N (inteiros)
escrever: "Resto = ", resto
escrever: "Introduza dois números inteiros (> 0):"
resto <-- restoDivisão(N, M)
ler: M, N
parâmetros de saida: resto (inteiro)
resto <-- restoDivisão(M, N)
senão
```

5. Construa um algoritmo que realize as seguintes ações (pela ordem indicada): peça ao utilizador que introduza dois números inteiros positivos não nulos (M e N), verifique se M é múltiplo de N e, depois, mostre uma mensagem a informar esta situação (exemplo: 20 é múltiplo de 5).

escrever: M, "não é múltiplo de ", N
se (restoDivisão(M, N) = 0) então
fim_algoritmo
escrever: M, "é múltiplo de ", N
fim_se
escrever: "Introduza dois números inteiros (> 0):"
ler: M, N
parâmetros de saida: --
parâmetros de entrada: M, N (inteiros)
algoritmo multiplos
senão

C. Instruções/acções de repetição

3. Construa um algoritmo para calcular e mostrar a soma dos 100 primeiros números naturais.

soma <-- soma + k
enquanto (k <= 100) fazer
escrever: "Soma dos 100 primeiro naturais = ", soma
parâmetros de entrada: --
soma <-- 0
fim_algoritmo
algoritmo soma100PrimeirosNaturais
k <-- k + 1
k <-- 1
parâmetros de saida: soma (inteiro)
fim_enquanto

5. Construa um algoritmo para ler uma sequência de inteiros positivos (ou seja, termina com a introdução de um valor negativo) e calcule a sua soma.

fim_enquanto
escrever: "Soma de positivos = ", soma
ler: num
escrever: "Introduza um inteiro (negativo para terminar):"
parâmetros de saida: soma (inteiro)
algoritmo somaInteirosPositivos
fim_algoritmo
soma <-- 0
ler: num
escrever: "Introduza um inteiro (negativo para terminar):"
soma <-- soma + num
enquanto (num >=0) fazer
parâmetros de entrada: num (inteiro)

7. Construa um algoritmo que permita determinar o maior número de uma sequência de N ($N \geq 1$) números inteiros dados pelo utilizador. Se $N < 1$ deverá ser pedido novamente até obter um valor válido. A introdução de números deverá terminar quando forem inseridos N números. Nessa altura a aplicação deverá mostrar o resultados obtido (maior valor introduzido).

escrever: "Introduza um inteiro >= 1):"
ler: num
escrever: "Introduza o primeiro numero inteiro:"
k <-- 1
maior <-- num
escrever: "Maior = ", maior
maior <-- num
fazer
fim_algoritmo
escrever: "Introduza um numero inteiro:"
algoritmo maiorNumero
parâmetros de entrada: N, num (inteiros)
se (num > maior) então
parâmetros de saida: maior (inteiro)
enquanto (k <= N-1) fazer
enquanto (N < 1)
fim_enquanto
fim_se
ler: num
ler: N
k <-- k + 1

9. Construa um algoritmo para determinar o produto e a soma dos números inteiros positivos não nulos (> 0) pares entre N1 e N2. A aplicação deverá dar erro se N2 ≤ N1 e voltar a pedir estes valores. Deverá no final apresentar o produto e a soma calculados.

```
fim_algoritmo
algoritmo produtoSomaPares
fim_enquanto
fim_se
escrever: "Introduzir um inteiro > 0:"
fazer
escrever: "Produto e soma = ", produto, soma
ler: N1
ler: N2
se (restoDivisão(A, 2) = 0) então
parâmetros de entrada: N1, N2 (inteiros)
enquanto (N1 <= 0)
enquanto (A <= N2) fazer
enquanto (N2 <= N1)
fazer
soma <-- soma + A
produto <-- 1
A <-- N1
escrever: "Introduzir um inteiro > N1:"
A <-- A + 1
parâmetros de saida: produto, soma (inteiros)
soma <-- 0
produto <-- produto x A
```