

- 8.1 Escreva um programa que receba como entrada um número inteiro e calcula, por meio de uma função, o módulo (valor absoluto) desse número.

```
1 algoritmo "Lista 08_01 – Cálculo do Módulo de um número inteiro"
2
3 var
4     numero, resultado: inteiro
5
6 // Função que recebe um inteiro e retorna seu valor absoluto
7 funcao calcular_modulo(n: inteiro): inteiro
8 inicio
9     se (n < 0) entao
10         retorne -n
11     senao
12         retorne n
13 fimse
14 fimfuncao
15
16 inicio
17     escreva("Digite um número inteiro: ")
18     leia(numero)
19
20 // Chamada da função
21 resultado <- calcular_modulo(numero)
22
23 // Exibição do resultado (sem formatação complexa)
24 escreva("O modulo de ", numero, " é ", resultado)
25 finalgoritmo
```

Função

```
Digite um número inteiro: -6
O modulo de -6 é 6
>>> Fim da execução do programa !
```

8.2 Desenvolva um programa (**pseudocódigo**) que realiza a operação de divisão. O programa (**pseudocódigo**) deve receber como entrada o valor do dividendo e do divisor e, por meio de uma sub-rotina, calcula e imprime o resultado da operação.

```
1  algoritmo "Lista 08_02 – Divisão utilizando subrotina"
2
3  var
4      valor1, valor2: real
5
6  // Sub-rotina (procedimento) para calcular e imprimir a divisão
7  procedimento realizar_divisao(dividendo, divisor: real)
8  inicio
9      se (divisor <> 0) entao
10         escreva("O resultado da divisao e: ", dividendo / divisor)
11     senao
12         escreva("Erro: Divisao por zero nao e permitida.")
13     fimse
14 fimprocedimento
15
16 inicio
17     escreva("Digite o dividendo: ")
18     leia(valor1)
19     escreva("Digite o divisor: ")
20     leia(valor2)
21
22 // Chamada da sub-rotina
23     realizar_divisao(valor1, valor2)
24 fimalgoritmo
```

Procedimento

```
Digite o dividendo: 10
Digite o divisor: 2
O resultado da divisao e: 5
>>> Fim da execução do programa !
```

```
Digite o dividendo: 10
Digite o divisor: 0
Erro: Divisao por zero nao e permitida.
>>> Fim da execução do programa !
```

8.3 Escreva um programa que receba como entrada o valor de um ângulo em graus e apresente seu valor em radianos, fazendo uso de uma função para o cálculo. A fórmula para conversão de graus para radianos é: $\text{valor em radianos} = \text{valor em graus} * \pi / 180$. Considere o valor de π como sendo 3.1416

```
1 algoritmo "Lista 08_03 – Conversor de Graus em Radianos"
2 var
3   anguloGraus, anguloRad: real
4
5   // Função para realizar o cálculo da conversão
6   // Não utilizado pi como variável, pois dentro do visualg ele já representa PI e com isso
7   // apresenta erro em definir como variável. No Visualg o programador não cria constantes.
8
9   funcao converterParaRad(g: real) : real
10  var
11    NPI: real
12  inicio
13    NPI <- 3.1416
14    retorne g * pi / 180
15  fimfuncao
16
17 inicio
18   escreval("--- CONVERSOR DE ÂNGULOS ---")
19   escreva("Digite o valor do ângulo em graus: ")
20   leia(anguloGraus)
21
22   // Chamada da função e atribuição do retorno à variável
23   anguloRad <- converterParaRad(anguloGraus)
24
25   escreval("-----")
26   escreval("O valor em radianos é: ", anguloRad)
27   escreval("-----")
28
29 fimalgoritmo
```

Função

```
--- CONVERSOR DE ÂNGULOS ---
Digite o valor do ângulo em graus: 360
-----
O valor em radianos é: 6.2832
-----
>>> Fim da execução do programa !
```

8.4 Elabore um algoritmo que calcule e apresente o valor de um determinado termo da sequência de Fibonacci. O programa lê um número i , inserido pelo usuário, e imprime o i -ésimo termo da sequência. O programa deve contemplar uma função específica que receba o número i e retorne o valor do termo correspondente para o programa principal.

Dado: A sequência de Fibonacci é a seguinte: 1 1 2 3 5 8 13... inicia-se em 1, repete o 1 no segundo termo e, a partir do terceiro termo, o valor de qualquer termo é obtido pela soma dos dois termos anteriores. Ou seja, $n = (n-1) + (n-2)$.

```
1  algoritmo "Lisdta 08_04 - FibonacciTermo"
2
3  var
4      n, resultado: inteiro
5
6  // Função que calcula o i-ésimo termo da sequência
7  funcao fibonacci(i: inteiro): inteiro
8  var
9      cont, t1, t2, proximo: inteiro
10 inicio
11     se (i <= 2) entao
12         retorne 1
13     senao
14         t1 <- 1
15         t2 <- 1
16         // Começamos do terceiro até o i-ésimo termo
17         para cont de 3 ate i faca
18             proximo <- t1 + t2
19             t1 <- t2
20             t2 <- proximo
21         fimpara
22         retorne t2
23     fimse
24 fimfuncao
25
26 inicio
27     escreva("Digite a posicao (i) do termo desejado: ")
28     leia(n)
29
30     // Chamada da função passando a posição i
31     resultado <- fibonacci(n)
32
33     escreva("O ", n, "o termo da sequencia de Fibonacci e: ", resultado)
34
35 fimalgoritmo
```

Função

```
Digite a posicao (i) do termo desejado: 6
O 6o termo da sequencia de Fibonacci e: 8
>>> Fim da execução do programa !
```

8.5 Faça um programa que leia o valor do salário atual de um militar e imprima o valor do salário reajustado. O programa principal deve solicitar ao usuário para digitar o salário do militar e chamar a função de nome `Calcula_Aumento`, receber o retorno desta função, imprimir o novo valor do salário e perguntar se o usuário deseja digitar outro salário ou não. Se o usuário responder que sim, o ciclo deve se repetir até que o usuário responda que não. Se o usuário digitar um valor diferente de “sim” ou “não”, o programa deverá apresentar uma mensagem de entrada inválida e solicitar uma nova entrada, até que o usuário digite uma entrada válida (sim ou não). **A função `Calcula_Aumento`** deve seguir a seguinte regra para calcular o novo salário: se o salário atual for menor ou igual a R\$ 1.000,00, deverá aplicar um aumento de 10%; se for maior que R\$ 1.000,00 e menor ou igual a R\$ 2.000,00, o aumento será de 8%; já se for maior que R\$ 2.000,00, o aumento deverá ser de 5%. Essa função deverá calcular o novo salário e retornar o valor para o programa principal.

```
1  algoritmo "Lista 08_05 – Reajuste Salarial de Militar"
2
3  var
4      salario_atual, salario_novo: real
5      resposta: caractere
6      continuar: logico
7
8  // Função que calcula o novo salário com base nas faixas de aumento
9  funcao Calcula_Aumento(salario: real): real
10 inicio
11     se (salario <= 1000) entao
12         retorne salario * 1.10 // 10% de aumento
13     senao
14         se (salario <= 2000) entao
15             retorne salario * 1.08 // 8% de aumento
16         senao
17             retorne salario * 1.05 // 5% de aumento
18     fimse
19 fimse
20 fimfuncao
21
22 inicio
23     continuar <- verdadeiro
24
25     enquanto (continuar) faca
26         // Entrada de dados
27         escreva("Digite o salario atual do militar: R$ ")
28         leia(salario_atual)
29
30         // Chamada da função e recebimento do retorno
31         salario_novo <- Calcula_Aumento(salario_atual)
32
33         // Saída do resultado
34         escreval("Novo salario reajustado: R$ ", salario_novo:2:2)
35         escreval("")
36
37         // Ciclo de validação da resposta (Sim/Não)
38         repita
```

```

39     escreva("Deseja digitar outro salario? (sim/nao): ")
40     leia(resposta)
41
42     se (resposta = "sim") entao
43         continuar <- verdadeiro
44     senao
45         se (resposta = "nao") entao
46             continuar <- falso
47         senao
48             escreval("Entrada invalida! Digite apenas 'sim' ou 'nao'.")
49         fimse
50     fimse
51     ate (resposta = "sim") ou (resposta = "nao")
52
53     escreval("-----")
54     fimenquanto
55
56     escreval("Programa encerrado.")
57
58 fimalgoritmo

```

```

Digite o salario atual do militar: R$ 1500
Novo salario reajustado: R$ 1620.00

Deseja digitar outro salario? (sim/nao): s
Entrada invalida! Digite apenas 'sim' ou 'nao'.
Deseja digitar outro salario? (sim/nao): sim
-----
Digite o salario atual do militar: R$ 2500
Novo salario reajustado: R$ 2625.00

Deseja digitar outro salario? (sim/nao): nao
-----
Programa encerrado.

>>> Fim da execução do programa !

```

8.6 Modifique o algoritmo do exercício 6.11 da lista de exercícios referente ao capítulo 6 (Condicionais) da apostila de Algoritmos, de modo que o cálculo do valor de delta, o cálculo do valor das raízes (quando houverem) e a impressão dos resultados, sejam realizadas por meio de sub-rotinas, que serão chamadas pelo programa principal.

1	Algoritmo "Lista 08_06 – Equação do Segundo Grau com Subrotinas"	
2		
3	Var	
4	a, b, c, valorDelta: Real	
5		
6	// 1. Função para calcular o valor de Delta	
7	Funcao CalcularDelta (a, b, c: Real): Real	Função
8	Inicio	
9	Retorne $(b * b) - (4 * a * c)$	
10	FimFuncao	
11		
12	// 2. Procedimento para calcular e mostrar as raízes	
13	Procedimento CalcularRaizes (a, b, delta: Real)	Procedimento
14	Var	
15	x1, x2: Real	
16	Inicio	
17	Se $(delta < 0)$ entao	
18	Escreval("A equação não possui raízes reais.")	
19	Senao	
20	Se $(delta = 0)$ entao	
21	x1 $\leftarrow (-b) / (2 * a)$	
22	Escreval("A equação possui uma raiz real: ", x1)	
23	Senao	
24	x1 $\leftarrow (-b + RaizQ(delta)) / (2 * a)$	
25	x2 $\leftarrow (-b - RaizQ(delta)) / (2 * a)$	
26	Escreval("A equação possui duas raízes: ", x1, " e ", x2)	
27	FimSe	
28	FimSe	
29	FimProcedimento	
30		
31	// -- Programa Principal --	
32	Inicio	
33	Escreva("Informe o valor de a: ")	
34	Leia(a)	
35		
36	// O valor de 'a' não pode ser zero em uma equação do segundo grau	
37	Se $(a = 0)$ entao	
38	Escreval("O valor de 'a' deve ser diferente de zero.")	
39	Senao	
40	Escreva("Informe o valor de b: ")	
41	Leia(b)	
42	Escreva("Informe o valor de c: ")	
43	Leia(c)	
44		
45	// Chamada da função para obter o Delta	
46	valorDelta $\leftarrow$ CalcularDelta (a, b, c)	
47		

```
48 // Chamada do procedimento para processar o resultado
49 Escreval("-----")
50 CalcularRaizes(a, b, valorDelta)
51 FimSe
52 fimalgoritmo
```

```
Informe o valor de a: -1
Informe o valor de b: 2
Informe o valor de c: 5
-----
A equação possui duas raízes: -1.44948974278318 e
3.44948974278318
>>> Fim da execução do programa !
```

- 8.7 Escrever um programa que receba como entrada dois números inteiros e apresente os seguintes resultados: maior valor digitado (obtido por uma função chamada Calcula_Maior); resultado da multiplicação dos dois números digitados (obtido por uma função chamada Multiplica_Numeros); se cada número digitado é positivo ou negativo (obtido pela função Verifica_Sinal).

```
1  algoritmo "Lista 08_07 Funcoes aritméticas e relacionais"
2  var
3      n1, n2: inteiro
4
5  // Função 1: Retorna o maior entre dois números
6  funcao Calcula_Maior(a, b: inteiro) : inteiro
7  inicio
8      se (a > b) entao
9          retorne a
10         senao
11             retorne b
12         fimse
13 fimfuncao
14
15 // Função 2: Retorna o produto de dois números
16 funcao Multiplica_Numeros(a, b: inteiro) : inteiro
17 inicio
18     retorne a * b
19 fimfuncao
20
21 // Função 3: Retorna se o número é positivo ou negativo
22 funcao Verifica_Sinal(n: inteiro) : caractere
23 inicio
24     se (n >= 0) entao
25         retorne "Positivo"
26     senao
27         retorne "Negativo"
28     fimse
29 fimfuncao
30
31 inicio
32     // Entrada de dados
33     escreva("Digite o primeiro número inteiro: ")
34     leia(n1)
35     escreva("Digite o segundo número inteiro: ")
36     leia(n2)
37     // Saída de informações utilizando as chamadas das funções
38     escreval("-----")
39     escreval("MAIOR VALOR: ", Calcula_Maior(n1, n2))
40     escreval("MULTIPLICAÇÃO: ", Multiplica_Numeros(n1, n2))
41     escreval("SINAL DO 1º NÚMERO: ", Verifica_Sinal(n1))
42     escreval("SINAL DO 2º NÚMERO: ", Verifica_Sinal(n2))
43     escreval("-----")
44 fimalgoritmo
```

Função 1

Função 2

Função 3

```

Digite o primeiro número inteiro: 2
Digite o segundo número inteiro: 3
-----
MAIOR VALOR: 3
MULTIPLICAÇÃO: 6
SINAL DO 1º NÚMERO: Positivo
SINAL DO 2º NÚMERO: Positivo
-----
>>> Fim da execução do programa !

```

- 8.8 Desenvolva um algoritmo que receba como entrada um caractere e, por meio de uma sub-rotina, verifica e imprime a informação que o caractere digitado trata-se de uma vogal ou consoante. Deve-se partir do princípio que o usuário digitará uma letra e não um número ou qualquer outro caractere especial.

1	algoritmo "Lista 08_08 – Verifica se letra é Vogal ou Consoante"
2	
3	// Sub-rotina que verifica e imprime o tipo do caractere
4	procedimento verificar_letra(letra: caractere)
5	inicio
6	se (letra = "a") ou (letra = "e") ou (letra = "i") ou (letra = "o") ou (letra = "u") ou
7	(letra = "A") ou (letra = "E") ou (letra = "I") ou (letra = "O") ou (letra = "U") entao
8	escreval("O caractere '", letra, "' e uma vogal.")
9	senao
10	escreval("O caractere '", letra, "' e uma consoante.")
11	fimse
12	fimprocedimento
13	var
14	letra: caractere
15	
16	inicio
17	escreva("Digite uma letra do alfabeto: ")
18	leia(letra)
19	
20	// Chamada da sub-rotina
21	verificar_letra(letra)
22	
23	fimalgoritmo

```

Digite uma letra do alfabeto: a
O caractere 'a' e uma vogal.
>>> Fim da execução do programa !

```

- 8.9 Escreva um programa que recebe como entrada dois números inteiros positivos e, por meio de uma função, calcula a soma dos números inteiros existentes entre eles (excluindo eles próprios), retornando tal valor para o programa principal. O programa deve validar a entrada, ou seja, ficar apresentando a mensagem de entrada inválida e solicitando um valor correto, até que a entrada seja válida. O programa deve tratar os casos em que não existem números entre os valores digitados, apresentando tal mensagem na tela.

```
1  algoritmo "Lista 08_09 – Soma de Intervalo de Números"
2
3  var
4      num1, num2, resultado, diferenca: inteiro
5
6  // Função que calcula a soma dos inteiros entre dois números
7  funcao calcular_soma_entre(n1, n2: inteiro): inteiro
8  var
9      i, menor, maior, soma: inteiro
10 inicio
11     soma <- 0
12
13     // Lógica manual para identificar o maior e o menor
14     se n1 < n2 entao
15         menor <- n1
16         maior <- n2
17     senao
18         menor <- n2
19         maior <- n1
20     fimse
21
22     // O laço só executa se houver números entre eles
23     para i de (menor + 1) ate (maior - 1) faca
24         soma <- soma + i
25     fimpara
26
27     retorne soma
28 fimfuncao
29
30 inicio
31     // Validação do primeiro número
32     repita
33         escreva("Digite o primeiro número inteiro positivo: ")
34         leia(num1)
35         se num1 <= 0 entao
36             escreval("Entrada inválida!")
37         fimse
38     ate num1 > 0
39
40     // Validação do segundo número
41     repita
42         escreva("Digite o segundo número inteiro positivo: ")
43         leia(num2)
44         se num2 <= 0 entao
```

```

45     escreval("Entrada inválida!")
46     fimse
47     ate num2 > 0
48
49     // Verificação manual de distância (poderia usar a função "abs" e simplificar o código)
50     se num1 > num2 entao
51         diferenca <- num1 - num2
52     senao
53         diferenca <- num2 - num1
54     fimse
55
56     // Se a diferença for 0 (iguais) ou 1 (vizinhos), não há números entre eles
57     se (diferenca <= 1) entao
58         escreval("Não existem números inteiros entre ", num1, " e ", num2, ".")
59     senao
60         resultado <- calcular_soma_entre(num1, num2)
61         escreval("A soma dos inteiros entre eles é: ", resultado)
62     fimse
63
64     fimalgoritmo

```

```

Digite o primeiro número inteiro positivo: 5
Digite o segundo número inteiro positivo: 8
A soma dos inteiros entre eles é: 13

>>> Fim da execução do programa !

```

```

Digite o primeiro número inteiro positivo: 2
Digite o segundo número inteiro positivo: 2
Não existem números inteiros entre 2 e 2.

>>> Fim da execução do programa !

```

```

Digite o primeiro número inteiro positivo: 10
Digite o segundo número inteiro positivo: 6
A soma dos inteiros entre eles é: 24

>>> Fim da execução do programa !

```

- 8.10 Desenvolva um algoritmo que, por meio de sub-rotinas, realiza e apresenta o somatório dos N primeiros números primos, sendo que o valor de N é digitado pelo usuário. O programa deverá possuir uma função e um procedimento: a função verifica se determinado número é primo; já o procedimento, calcula o somatório e imprime o resultado. O programa principal deverá receber o valor de “N” digitado pelo usuário e invocar o procedimento que realiza e mostra o somatório dos números primos, que por sua vez chama a função que verifica se determinado número é primo. O programa não deverá permitir que o usuário digite um valor de “N” menor do que 1, mostrando repetidamente a mensagem de valor inválido, até que o usuário digite um número inteiro positivo.

```
1  algoritmo "Lista 08_10 – Soma <n> números primos"
2
3  var
4      n: inteiro
5
6  // Função que verifica se um número é primo. Retorna verdadeiro ou falso
7  funcao eh_primo(num: inteiro): logico
8  var
9      cont, divisores: inteiro
10 inicio
11     se (num < 2) entao
12         retorne falso
13     fimse
14
15     divisores <- 0
16     para cont de 1 ate num faca
17         se (num % cont = 0) entao
18             divisores <- divisores + 1
19         fimse
20     fimpara
21
22     se (divisores = 2) entao
23         retorne verdadeiro
24     senao
25         retorne falso
26     fimse
27 fimfuncao
28
29 // Procedimento que calcula o somatório dos N primeiros primos e imprime
30 procedimento calcular_e_exibir_soma(n_termos: inteiro)
31 var
32     soma, quantidade_encontrada, numero_atual: inteiro
33 inicio
34     soma <- 0
35     quantidade_encontrada <- 0
36     numero_atual <- 2 // Começamos a testar a partir do primeiro primo
37
38     enquanto (quantidade_encontrada < n_termos) faca
39         se (eh_primo(numero_atual)) entao
40             soma <- soma + numero_atual
41             quantidade_encontrada <- quantidade_encontrada + 1
42     fimse
```

Função

Procedimento

```

43     numero_atual <- numero_atual + 1
44     fimenquanto
45
46     escreval("O somatorio dos ", n_termos, " primeiros numeros primos e: ", soma)
47 fimprocedimento
48
49 inicio
50     // Validação da entrada: N deve ser pelo menos 1
51     repita
52         escreva("Digite a quantidade de numeros primos a somar (N): ")
53         leia(n)
54
55         se (n < 1) entao
56             escreval("Valor invalido! Digite um numero inteiro positivo.")
57         fimse
58     ate (n >= 1)
59
60     // Chamada do procedimento principal
61     calcular_e_exibir_soma(n)
62
63 fimalgoritmo

```

```

Digite a quantidade de numeros primos a somar (N): 6
O somatorio dos 6 primeiros numeros primos e: 41

>>> Fim da execução do programa !

```

- 8.11 Utilizando os conceitos de procedimentos e funções, desenvolva um algoritmo que calcule, a partir de 2 números inteiros positivos P e M, a combinação de M elementos tomados P a P, conforme a expressão: $C = M! / (P!(M - P)!)$. Os valores dos diversos fatoriais existentes na expressão deverão ser obtidos por meio de uma função para isso destinada.

```
1  Algoritmo "Lista 08_11 – Cálculo de Combinação"
2
3  // Função para calcular o fatorial de um número
4  Funcao Fatorial(n: Inteiro): Inteiro
5  Var
6    i, fat: Inteiro
7  Inicio
8    fat <- 1
9    Para i de 1 ate n faca
10     fat <- fat * i
11  FimPara
12  Retorne fat
13  FimFuncao
14
15  Var
16    M, P: Inteiro
17    Combinacao: Real
18
19  Inicio
20    Escreva("Digite o valor de M (total de elementos): ")
21    Leia(M)
22    Escreva("Digite o valor de P (elementos tomados P a P): ")
23    Leia(P)
24
25    // Validação básica
26    Se (M < P) entao
27      Escreval("Erro: M deve ser maior ou igual a P.")
28    Senao
29      // Cálculo da combinação utilizando a função Fatorial
30      Combinacao <- Fatorial(M) / (Fatorial(P) * Fatorial(M - P))
31
32      Escreval("O resultado da combinação é: ", Combinacao)
33    FimSe
34  fimalgoritmo
```

Função

```
Digite o valor de M (total de elementos):
10
Digite o valor de P (elementos tomados P a P): 2
O resultado da combinação é: 45

>>> Fim da execução do programa !
```

- 8.12 Elabore um algoritmo que possua uma função que lê um número positivo do teclado. Caso o usuário não forneça um número inteiro positivo, a função repete a leitura até que um número positivo seja lido. O programa principal deve chamar a função que lê um número do teclado 3 vezes, para a leitura de 3 números distintos. No final, o programa deverá imprimir os números digitados na ordem decrescente.

```
1  algoritmo "Lista 08_12 – Ordenação Decrescente"
2  var
3      n1, n2, n3, troca: inteiro
4
5  // Função que lê e valida se o número é positivo. Não existem parâmetros para esta função.
6  funcao ler_positivo(): inteiro
7  var
8      num: inteiro
9  inicio
10     repita
11         escreva("Digite um número inteiro positivo: ")
12         leia(num)
13
14     se (num <= 0) entao
15         escreval("Entrada inválida! O número deve ser positivo.")
16     fimse
17     ate (num > 0)
18
19     retorne num
20 fimfuncao
21
22 inicio
23     // Chama a função 3 vezes para ler os números
24     escreval("--- Leitura dos Numeros ---")
25     n1 <- ler_positivo()      // A função ler_positivo não tem passagem de argumentos
26     n2 <- ler_positivo()
27     n3 <- ler_positivo()
28
29     // Lógica de ordenação decrescente (Bubble Sort manual para 3 variáveis)
30     // Objetivo: n1 (maior), n2 (meio), n3 (menor)
31
32     se (n1 < n2) entao
33         troca <- n1
34         n1 <- n2
35         n2 <- troca
36     fimse
37     se (n1 < n3) entao
38         troca <- n1
39         n1 <- n3
40         n3 <- troca
41     fimse
42     se (n2 < n3) entao
43         troca <- n2
44         n2 <- n3
45         n3 <- troca
```

```
46 fimse
47
48 // Impressão do resultado
49 escreval("-----")
50 escreval("Numeros em ordem decrescente: ", n1, ", ", n2, ", ", n3)
51
52 finalgoritmo
```

```
--- Leitura dos Numeros ---
Digite um numero inteiro positivo: -1
Entrada invalida! O numero deve ser positivo.
Digite um numero inteiro positivo: 3
Digite um numero inteiro positivo: 0
Entrada invalida! O numero deve ser positivo.
Digite um numero inteiro positivo: 1
Digite um numero inteiro positivo: 9
-----
Numeros em ordem decrescente:  9,  3,  1

>>> Fim da execução do programa !
```