

- 6.1 Desenvolva um algoritmo, nomeado `Estrutura_Decisao_Simples`, que solicita ao usuário que digite um número no teclado e, caso o número digitado seja par, mostre na tela o seguinte texto: "O número digitado é par".

```

1  algoritmo "Lista 06.01 - Estrutura_Decisao_Simples"
2  var
3      numero: inteiro
4  inicio
5      escreva("Digite um número: ")
6      leia(numero)
7      // Verifica se o resto da divisão por 2 é igual a 0 (número par)
8      se (numero % 2 = 0) entao
9          escreva("O número digitado é par")
10         fimse
11 fimalgoritmo

```

```

Digite um número:
10
O número digitado é par
>>> Fim da execução do programa !

```

- 6.2 Modifique o algoritmo do exercício anterior, alterando seu nome para `Estrutura_Decisao_Composta` e acrescente a possibilidade de mostrar o texto "O número digitado é ímpar", quando for essa a situação.

```

1  algoritmo "Lista 06.02 - Estrutura_Decisao_Composta"
2  var
3      numero: inteiro
4  inicio
5      escreva("Digite um número: ")
6      leia(numero)
7      // Estrutura composta para verificar se é par ou ímpar
8      se (numero mod 2 = 0) entao
9          escreva("O número digitado é par")
10         senao
11             escreva("O número digitado é ímpar")
12         fimse
13 fimalgoritmo

```

```

Digite um número: 10
O número digitado é par
>>> Fim da execução do programa !

```

```

Digite um número: 7
O número digitado é ímpar
>>> Fim da execução do programa !

```

6.3 Faça um algoritmo que receba como entrada, via teclado, o valor do salário e a graduação de um militar e calcula e mostra o valor do salário reajustado com um índice que varia, de acordo com a graduação, da seguinte forma: soldado: 50%, cabo: 40%, sargento: 30% e outra graduação: 20%.

```
1  algoritmo "Lista 06.03 - Reajuste_Salarial_Militar"
2  var
3      salario, salario_reajustado: real
4      graduacao: caractere
5  inicio
6      escreva("Informe o salário atual: R$ ")
7      leia(salario)
8      escreval("Informe a graduação (soldado, cabo, sargento ou outra): ")
9      leia(graduacao)
10
11  // Estrutura de escolha para aplicar o reajuste correspondente
12  escolha graduacao
13      caso "soldado", "Soldado", "SOLDADO"
14          salario_reajustado <- salario * 1.50 // Reajuste de 50%
15      caso "cabo", "Cabo", "CABO"
16          salario_reajustado <- salario * 1.40 // Reajuste de 40%
17      caso "sargento", "Sargento", "SARGENTO"
18          salario_reajustado <- salario * 1.30 // Reajuste de 30%
19      outrocaso
20          salario_reajustado <- salario * 1.20 // Reajuste de 20% para outras graduações
21  fimescolha
22
23  escreval("O valor do salário reajustado é: R$ ", salario_reajustado:8:2)
24  finalgoritmo
```

```
Informe o salário atual: R$ 2000
Informe a graduação:
Soldado, Cabo, Sargento ou outra: SOLDADO
O valor do salário reajustado é: R$ 3000.00

>>> Fim da execução do programa !
```

6.4 Escreva e implemente um algoritmo que receba como entrada, via teclado, a idade de uma pessoa e exiba a informação se ela é menor de idade, maior de idade ou idoso, conforme ela possua menos que 18 anos, entre 18 e 65 anos ou acima de 65 anos, respectivamente. Exemplo de saída:

Informe sua idade: 70
Você é considerada uma pessoa

```
1  algoritmo "Lista 06.04 - Classificacao_Por_Idade"
2  var
3      idade: inteiro
4  inicio
5      escreva("Informe sua idade: ")
6      leia(idade)
7
8      // Avalia as faixas etárias
9      se (idade < 18) entao
10         escreval("Você é considerada uma pessoa menor de idade.")
11     senao
12         se (idade <= 65) entao
13             escreval("Você é considerada uma pessoa maior de idade.")
14         senao
15             escreval("Você é considerada uma pessoa idosa.")
16         fimse
17     fimse
18 fimalgoritmo
```

Informe sua idade: 70
Você é considerada uma pessoa idosa.

>>> Fim da execução do programa !

Informe sua idade: 15
Você é considerada uma pessoa menor de idade.

>>> Fim da execução do programa !

6.5 Escreva e implemente um algoritmo que calcule o IMC (Índice de Massa Corporal) de uma pessoa, de acordo com seu peso e altura. O programa deve informar se a pessoa está abaixo do peso (IMC menor que 20), normal (IMC entre 20 e 25), excesso de peso (entre 26 e 30), obesa (IMC entre 31 e 35) ou com obesidade mórbida (acima de 35).

Dado: fórmula para cálculo do IMC: $\text{peso} / (\text{altura})^2$, sendo o peso (em kg) e a altura (em metros).

```
1  algoritmo "Lista 06.05 - Calculo_IMC"
2  var
3    peso, altura, imc: real
4  inicio
5    escreva("Informe o seu peso (em kg): ")
6    leia(peso)
7    escreva("Informe a sua altura (em metros): ")
8    leia(altura)
9
10   // Cálculo do IMC: peso dividido pela altura ao quadrado
11   imc <- peso / (altura * altura) // O VisuAlg também aceita: peso / (altura ^ 2)
12
13   // Avaliação das faixas de IMC
14   se (imc < 20) entao
15     escreval("Abaixo do peso (IMC: ", imc:5:2, ")")
16   senao
17     se (imc <= 25) entao
18       escreval("Normal (IMC: ", imc:5:2, ")")
19     senao
20       se (imc <= 30) entao
21         escreval("Excesso de peso (IMC: ", imc:5:2, ")")
22       senao
23         se (imc <= 35) entao
24           escreval("Obesa (IMC: ", imc:5:2, ")")
25         senao
26           escreval("Com obesidade mórbida (IMC: ", imc:5:2, ")")
27       fimse
28     fimse
29   fimse
30   fimse
31  fimalgoritmo
```

```
Informe o seu peso (em kg):
100
Informe a sua altura (em metros): 1.82
Obesa (IMC: 30.19)

>>> Fim da execução do programa !
```

6.6 Elabore um algoritmo que receba como entrada, via teclado, os valores das coordenadas (x,y) de um ponto no plano cartesiano e forneça como saída, no monitor, a localização do ponto (na origem, sobre o eixo x, sobre o eixo y, no 1º quadrante, no 2º quadrante, no 3º quadrante ou no 4º quadrante). Exemplo de saída:

O ponto de coordenada (x,y) encontra-se no 1º quadrante.

```

1  algoritmo "Lista 06.06 - Localizacao_Plano_Cartesiano"
2  var
3      x, y: real
4  inicio
5      escreva("Informe a coordenada X: ")
6      leia(x)
7      escreva("Informe a coordenada Y: ")
8      leia(y)
9      // Verifica a origem e os eixos primeiro
10     se (x = 0) e (y = 0) entao
11         escreval("O ponto de coordenada (", x, ",", y, ") encontra-se na origem.")
12     senao
13         se (x = 0) entao
14             escreval("O ponto de coordenada (", x, ",", y, ") encontra-se sobre o eixo y.")
15         senao
16             se (y = 0) entao
17                 escreval("O ponto de coordenada (", x, ",", y, ") encontra-se sobre o eixo x.")
18             senao
19                 // Se não está na origem nem nos eixos, verifica os quadrantes
20                 se (x > 0) e (y > 0) entao
21                     escreval("O ponto de coordenada (", x, ",", y, ") encontra-se no 1º quadrante.")
22                 senao
23                     se (x < 0) e (y > 0) entao
24                         escreval("O ponto de coordenada (", x, ",", y, ") encontra-se no 2º quadrante.")
25                 senao
26                     se (x < 0) e (y < 0) entao
27                         escreval("O ponto de coordenada (", x, ",", y, ") encontra-se no 3º quad.")
28                 senao
29                     // Se não é 1º, 2º ou 3º, só pode ser o 4º quadrante
30                     escreval("O ponto de coordenada (", x, ",", y, ") encontra-se no 4º quad.")
31             fimse
32         fimse
33     fimse
34 fimse
35 fimse
36 fimse
37 fimalgoritmo

```

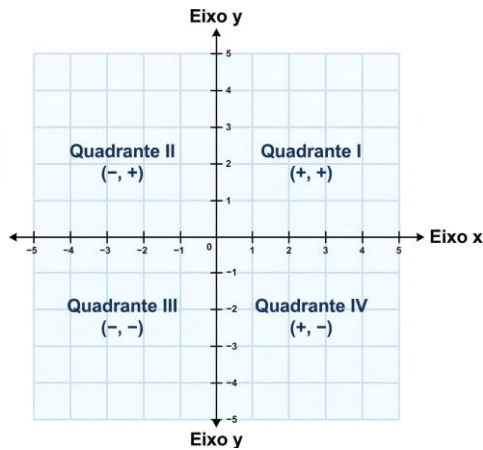
```

Informe a coordenada X: -5
Informe a coordenada Y: 8
O ponto de coordenada ( -5, 8) encontra-se no 2º quadrante.

>>> Fim da execução do programa !

```

Teste de mesa do algoritmo 06.06:



Atenção! A solução do algoritmo proposto está na página anterior. Esta explicação tem apenas função didática, não é exigida como resposta quando solicitado que se faça um algoritmo qualquer.

Para simular a entrada em todas as 7 opções de caminhos possíveis nesse algoritmo, precisamos definir 7 conjuntos de coordenadas (x, y). Aqui estão os valores de teste que utilizaremos:

1. **Origem:** $x = 0$, $y = 0$
2. **Eixo Y:** $x = 0$, $y = 5$
3. **Eixo X:** $x = 3$, $y = 0$
4. **1º Quadrante:** $x = 2$, $y = 4$
5. **2º Quadrante:** $x = -2$, $y = 4$
6. **3º Quadrante:** $x = -3$, $y = -1$
7. **4º Quadrante:** $x = 4$, $y = -2$

Abaixo está o teste de mesa estruturado. Ele demonstra a ordem cronológica em que o algoritmo avalia cada condição até encontrar a correta.

Cenário	Condição Avaliada como VERDADEIRA	Saída no monitor
1. Origem	$(x = 0)$ e $(y = 0)$	O ponto de coordenada (0,0) encontra-se na origem.
2. Eixo Y	1. $(x=0)$ e $(y=0)$ ->FALSO 2. $(x = 0)$ -> VERDADEIRO	O ponto de coordenada (0,5) encontra-se sobre o eixo y.
3. Eixo X	1. $(x=0)$ e $(y=0)$ ->FALSO 2. $(x=0)$ ->FALSO 3. $(y = 0)$ -> VERDADEIRO	O ponto de coordenada (3,0) encontra-se sobre o eixo x.
4. 1º Quad.	1. $(x=0)$ e $(y=0)$ ->F 2. $(x=0)$ ->F 3. $(y=0)$ ->F 4. $(x > 0)$ e $(y > 0)$ -> VERDADEIRO	O ponto de coordenada (2,4) encontra-se no 1º quadrante.
5. 2º Quad.	1 a 4. Avaliações anteriores ->F 5. $(x < 0)$ e $(y > 0)$ -> VERDADEIRO	O ponto de coordenada (-2,4) encontra-se no 2º quadrante.
6. 3º Quad.	1 a 5. Avaliações anteriores ->F 6. $(x < 0)$ e $(y < 0)$ -> VERDADEIRO	O ponto de coordenada (-3,-1) encontra-se no 3º quadrante.
7. 4º Quad.	1 a 6. Avaliações anteriores ->F 7. Cai no último senao (todas as restrições anteriores falharam).	O ponto de coordenada (4,-2) encontra-se no 4º quadrante.

- 6.7 Crie um algoritmo que receba como entrada o percurso a ser realizado em quilômetros e o tipo (capacidade) de viatura a ser empregada e informe o consumo estimado de combustível, sabendo-se que o consumo das viaturas do tipo 1/4 ton, 2,5 ton e 5 ton são, respectivamente, 8, 6 e 4 quilômetros por litro de combustível.

```
1  algoritmo "Lista 06.07 - Consumo_Viatura"
2  var
3      percurso, consumo, rendimento: real
4      cod_viatura: inteiro
5  inicio
6      escreva("Informe o percurso a ser realizado (em km): ")
7      leia(percurso)
8      escreval("Informe o código da viatura: [1] 1/4 ton; [2] 2.5 ton; [3] 5 ton: ")
9      leia(cod_viatura)
10
11     // Define o rendimento (km/l) com base no tipo da viatura
12     escolha cod_viatura
13         caso 1
14             rendimento <- 8
15         caso 2
16             rendimento <- 6
17         caso 3
18             rendimento <- 4
19         outrocaso
20             escreval("Tipo de viatura inválido.")
21             rendimento <- 0
22     fimsecolha
23
24     // Calcula e exibe o consumo apenas se uma viatura válida foi informada
25     se (rendimento > 0) entao
26         consumo <- percurso / rendimento
27         escreval("O consumo estimado é de: ", consumo:8:2, " litros de combustível.")
28     fimse
29 fimalgoritmo
```

```
Informe o percurso a ser realizado (em km): 1000
Informe o código da viatura: [1] 1/4 ton; [2] 2.5 ton; [3] 5 ton:
1
O consumo estimado é de: 125.00 litros de combustível.

>>> Fim da execução do programa !
```

6.8 Formule um algoritmo que receba como entrada quatro números e conte e apresente quantos deles são negativos.

```
1  algoritmo "Lista 06.08 - Conta_Negativos"
2  var
3      n1, n2, n3, n4: real
4      contador_negativos: inteiro
5  inicio
6      // Inicializa o contador com zero
7      contador_negativos <- 0
8      // Leitura e verificação do 1º número
9      escreva("Digite o 1º número: ")
10     leia(n1)
11     se (n1 < 0) entao
12         contador_negativos <- contador_negativos + 1
13     fimse
14     // Leitura e verificação do 2º número
15     escreva("Digite o 2º número: ")
16     leia(n2)
17     se (n2 < 0) entao
18         contador_negativos <- contador_negativos + 1
19     fimse
20     // Leitura e verificação do 3º número
21     escreva("Digite o 3º número: ")
22     leia(n3)
23     se (n3 < 0) entao
24         contador_negativos <- contador_negativos + 1
25     fimse
26     // Leitura e verificação do 4º número
27     escreva("Digite o 4º número: ")
28     leia(n4)
29     se (n4 < 0) entao
30         contador_negativos <- contador_negativos + 1
31     fimse
32     // Apresenta o resultado final
33     escreval("A quantidade de números negativos digitados é: ", contador_negativos)
34  fimalgoritmo
```

```
Digite o 1º número: -1
Digite o 2º número: 3
Digite o 3º número: -9
Digite o 4º número: 0
A quantidade de números negativos digitados é:  2

>>> Fim da execução do programa !
```

6.9 Faça um algoritmo que receba, via teclado, as medidas dos três lados de um triângulo, e informe seu tipo (equilátero, isósceles ou escaleno).

Observação: antes que seja verificado o tipo do triângulo, o algoritmo deve verificar se as medidas dos lados fornecidas pelo usuário, formam triângulo, ou seja, a medida de um lado deve ser menor do que a soma dos outros dois.

```
1  algoritmo "Lista 06.09 - Tipo_Triangulo"
2  var
3      l1, l2, l3: real
4  inicio
5      escreva("Informe a medida do 1º lado: ")
6      leia(l1)
7      escreva("Informe a medida do 2º lado: ")
8      leia(l2)
9      escreva("Informe a medida do 3º lado: ")
10     leia(l3)
11
12     // Verifica a condição de existência: um lado deve ser menor que a soma dos outros dois
13     se (l1 < l2 + l3) e (l2 < l1 + l3) e (l3 < l1 + l2) entao
14
15         // Classificação do triângulo
16         se (l1 = l2) e (l2 = l3) entao
17             escreval("As medidas formam um triângulo EQUILÁTERO.")
18         senao
19             se (l1 <> l2) e (l2 <> l3) e (l1 <> l3) entao
20                 escreval("As medidas formam um triângulo ESCALENO.")
21             senao
22                 // Se não é equilátero nem escaleno, obrigatoriamente tem apenas dois lados iguais
23                 escreval("As medidas formam um triângulo ISÓSCELES.")
24             fimse
25         fimse
26
27     senao
28         escreval("As medidas fornecidas NÃO formam um triângulo.")
29     fimse
30 fimalgoritmo
```

```
Informe a medida do 1º lado: 3
Informe a medida do 2º lado: 5
Informe a medida do 3º lado: 8
As medidas fornecidas NÃO formam um triângulo.

>>> Fim da execução do programa !
```

```
Informe a medida do 1º lado: 6
Informe a medida do 2º lado: 4
Informe a medida do 3º lado: 7
As medidas formam um triângulo ESCALENO.

>>> Fim da execução do programa !
```

6.10 Elabore um algoritmo que receba, via teclado, um número inteiro entre 1 e 12 (representando cada mês do ano) e imprima se este mês tem 28, 30 ou 31 dias. Caso o usuário digite um número que não esteja entre 1 e 12, o algoritmo deve mostrar uma mensagem de entrada inválida e encerrar.

Observação: assumo que fevereiro sempre possui 28 dias.

```
1  algoritmo "Lista 06.10 - Dias_No_Mes"
2  var
3      mes: inteiro
4  inicio
5      escreva("Digite um número correspondente a um mês (1 a 12): ")
6      leia(mes)
7
8      // Estrutura de múltipla escolha para verificar a quantidade de dias
9      escolha mes
10         caso 2
11             escreval("Este mês tem 28 dias.")
12         caso 4, 6, 9, 11
13             escreval("Este mês tem 30 dias.")
14         caso 1, 3, 5, 7, 8, 10, 12
15             escreval("Este mês tem 31 dias.")
16         outrocaso
17             // Trata qualquer entrada fora do intervalo 1-12
18             escreval("Entrada inválida. Programa encerrado.")
19     fimescolha
20 fimalgoritmo
```

```
Digite um número correspondente a um mês (1 a 12): 6
Este mês tem 30 dias.

>>> Fim da execução do programa !
```

```
Digite um número correspondente a um mês (1 a 12): 13
Entrada inválida. Programa encerrado.

>>> Fim da execução do programa !
```

6.11 Implemente um algoritmo que calcula as raízes de uma equação do segundo grau do tipo ax^2+bx+c , utilizando a fórmula de Bhaskara. O algoritmo deverá receber via teclado os valores dos índices a, b e c da equação e mostrar mensagem informando a quantidade de raízes (caso possua), com os seus respectivos valores ou que a equação não possui raízes reais.

Dado: fórmula de Bhaskara: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Exemplos de saída:

```
Informe o valor de a: 1
Informe o valor de b: -5
Informe o valor de c: 6
-----
A equação possui duas raízes: 2 e 3
```

```
1  algoritmo "Lista 06.11 Equacao_Segundo_Grau"
2  var
3      a, b, c, delta, x1, x2: real
4  inicio
5      escreva("Informe o valor de a: ")
6      leia(a)
7      escreva("Informe o valor de b: ")
8      leia(b)
9      escreva("Informe o valor de c: ")
10     leia(c)
11     escreval("-----")
12     // Verifica se realmente é uma equação do 2º grau
13     se (a = 0) entao
14         escreval("O coeficiente 'a' deve ser diferente de zero.")
15     senao
16         delta <- (b^2) - (4 * a * c)
17
18         se (delta < 0) entao
19             escreval("A equação não possui raízes reais.")
20         senao
21             se (delta = 0) entao
22                 x1 <- -b / (2 * a)
23                 escreval("A equação possui uma raiz: ", x1)
24             senao
25                 x1 <- (-b + RaizQ(delta)) / (2 * a)
26                 x2 <- (-b - RaizQ(delta)) / (2 * a)
27                 escreval("A equação possui duas raízes: ", x1, " e ", x2)
28             fimse
29         fimse
30     fimse
31  fimalgoritmo
```

```
Informe o valor de a: 2
Informe o valor de b: -1
Informe o valor de c: 3
-----
A equação não possui raízes reais.
>>> Fim da execução do programa !
```

```
Informe o valor de a: -1
Informe o valor de b: 1
Informe o valor de c: 9
-----
A equação possui duas raízes: -2.54138126514911 e
3.54138126514911
>>> Fim da execução do programa !
```

6.12 Desenvolva um programa que receba a renda anual de um militar e calcula o imposto de renda, cujos índices variam de acordo com as seguintes faixas:

- 1ª faixa: renda menor ou igual a R\$ 21.450,00: 15%.
- 2ª faixa: renda maior que R\$ 21.450,00 e menor ou igual a 51.900,00: 15% sobre o teto da primeira faixa (3217.50), mais 28% do que exceder à primeira faixa.
- 3ª faixa: renda maior que 51.900,00 – alíquotas referentes aos tetos das duas faixas anteriores (11743.50), mais 31% do que exceder à segunda faixa.

Informe a renda: **57000.00**

2ª faixa de imposto

Imposto devido: R\$ 13171,50

Alíquota real (%): 23,10

```
1  algoritmo "Imposto_Renda_Militar"
2  var
3      renda, imposto, aliquota_real: real
4      faixa: inteiro
5  inicio
6      escreva("Informe a renda: ")
7      leia(renda)
8      escreval("-----")
9
10     // Avaliação das faixas de contribuição
11     se (renda <= 21450.00) entao
12         faixa <- 1
13         imposto <- renda * 0.15
14     senao
15         se (renda <= 51900.00) entao
16             faixa <- 2
17             // 3217.50 é o teto da 1ª faixa (21450 * 0.15)
18             imposto <- 3217.50 + ((renda - 21450.00) * 0.28)
19         senao
20             faixa <- 3
21             // 11743.50 é a soma dos tetos das faixas anteriores
22             imposto <- 11743.50 + ((renda - 51900.00) * 0.31)
23         fimse
24     fimse
25
26     // Cálculo da alíquota real (porcentagem do imposto sobre a renda total)
27     aliquota_real <- (imposto / renda) * 100
28
29     // Impressão dos resultados formatados
30     escreval(faixa, "ª faixa de imposto")
31     escreval("Imposto devido: R$ ", imposto:8:2)
32     escreval("Alíquota real (%): ", aliquota_real:8:2)
33 fimalgoritmo
```

Informe a renda: 10000

1ª faixa de imposto

Imposto devido: R\$ 1500.00

Alíquota real (%): 15.00

>>> Fim da execução do programa !

6.13 Construa um algoritmo que leia quatro números digitados pelo usuário, some o primeiro com o segundo número, some o terceiro com o quarto número. Informe qual soma é a maior ou se são iguais. Caso o usuário digitar um valor igual a zero ou negativo, em qualquer um dos quatro números, o algoritmo deve mostrar uma mensagem de valor inválido e finalizar.

```
1  algoritmo "Soma_e_Comparacao"
2  var
3      n1, n2, n3, n4, soma1, soma2: real
4  inicio
5      escreva("Digite o 1º número: ")
6      leia(n1)
7      escreva("Digite o 2º número: ")
8      leia(n2)
9      escreva("Digite o 3º número: ")
10     leia(n3)
11     escreva("Digite o 4º número: ")
12     leia(n4)
13
14     // Validação da entrada: nenhum número pode ser zero ou negativo
15     se (n1 <= 0) ou (n2 <= 0) ou (n3 <= 0) ou (n4 <= 0) entao
16         escreval("Valor inválido inserido. O programa será finalizado.")
17     senao
18         soma1 <- n1 + n2
19         soma2 <- n3 + n4
20
21     se (soma1 > soma2) entao
22         escreval("Soma do 1º com o 2º núm. (", soma1, ") é MAIOR que a soma do 3º + 4º (", soma2, ").")
23     senao
24         se (soma2 > soma1) entao
25             escreval("A soma do 3º com o 4º núm. (", soma2, ") é MAIOR que a soma do 1º + 2º (", soma1, ").")
26         senao
27             escreval("As duas somas são IGUAIS (valor: ", soma1, ").")
28         fimse
29     fimse
30     fimse
31 fimalgoritmo
```

```
Digite o 1º número: 4
Digite o 2º número: 7
Digite o 3º número: 3
Digite o 4º número: 8
As duas somas são IGUAIS (valor: 11).

>>> Fim da execução do programa !
```

```
Digite o 1º número: 1
Digite o 2º número: 8
Digite o 3º número: 2
Digite o 4º número: 3
Soma do 1º com o 2º núm. ( 9 ) é MAIOR que a soma do 3º + 4º ( 5 ) .

>>> Fim da execução do programa !
```

6.14 Construa um programa em que o usuário informe, via teclado, as coordenadas (x,y) do centro de um círculo (em um plano cartesiano), bem como o seu raio; e forneça, também, as coordenadas (x,y) de um ponto de teste. De posse dos dados inseridos pelo usuário, o algoritmo deverá informar se o ponto de teste está situado no interior do círculo (mas não no centro), no centro do círculo, na circunferência (fronteira) ou fora do círculo.

Dados: Equação da circunferência:

$$r = \sqrt{(x_p - x_c)^2 + (y_p - y_c)^2} \text{ , onde:}$$

x_p = coordenada x do ponto de teste
 x_c = coordenada x do centro do círculo
 y_p = coordenada y do ponto de teste
 y_c = coordenada y do centro do círculo

Exemplo de execução do algoritmo:

Coordenada xc: 0
 Coordenada yc: 0
 Raio: 5
 Coordenada xp: 6
 Coordenada yp: 0

O ponto está fora do círculo

```

1  algoritmo "Posicao_Ponto_Circulo"
2  var
3    xc, yc, raio, xp, yp, distancia: real
4  inicio
5    // Entrada de dados do círculo
6    escreva("Coordenada xc: ")
7    leia(xc)
8    escreva("Coordenada yc: ")
9    leia(yc)
10   escreva("Raio: ")
11   leia(raio)
12   // Entrada de dados do ponto de teste
13   escreva("Coordenada xp: ")
14   leia(xp)
15   escreva("Coordenada yp: ")
16   leia(yp)
17   // Cálculo da distância entre o centro e o ponto de teste
18   distancia <- RaizQ((xp - xc)^2 + (yp - yc)^2)
19   // Estruturas de decisão para classificar a posição
20   se (xp = xc) e (yp = yc) entao
21     escreval("O ponto está situado no centro do círculo")
22   senao
23     se (distancia < raio) entao
24       escreval("O ponto está situado no interior do círculo")
25     senao
26       se (distancia = raio) entao
27         escreval("O ponto está situado na circunferência (fronteira)")
28       senao
29         escreval("O ponto está fora do círculo")
30     fimse
31   fimse
32   fimse
33 fimalgoritmo
  
```

```

Coordenada xc: 3
Coordenada yc: 5
Raio: 6
Coordenada xp: 8
Coordenada yp: 6
O ponto está situado no interior do círculo

>>> Fim da execução do programa !
  
```

6.15 Faça um programa que receba o código da carga, o peso da carga em toneladas, calcule e apresente: o peso da carga em quilos, o valor do frete, o valor do seguro (percentual sobre o valor da carga) e o valor total a ser pago (frete e seguro).

Dados:

Código da carga	Frete por quilo	Seguro da carga
10 a 20	R\$ 5.00	3%
21 a 30	R\$ 8.00	5%
31 a 40	R\$ 10.00	8%

Exemplo de execução do algoritmo:

Código da carga: **20**

Peso da carga (t): **10.5**

Valor da carga: **R\$ 600000.00**

Peso da carga: 10500 Kg

Frete: R\$ 52500.00

Seguro: R\$ 18000.00

Valor total: R\$ 70500.00

```

1  algoritmo "Calculo_Frete_Seguro"
2  var
3      codigo_carga: inteiro
4      peso_toneladas, peso_quilos, valor_carga: real
5      frete_por_quilo, percentual_seguro: real
6      valor_frete, valor_seguro, valor_total: real
7  inicio
8      escreva("Código da carga: ")
9      leia(codigo_carga)
10     escreva("Peso da carga (t): ")
11     leia(peso_toneladas)
12     escreva("Valor da carga: R$ ")
13     leia(valor_carga)
14     escreval("-----")
15
16     // Definição das taxas de acordo com o código da carga
17     se (codigo_carga >= 10) e (codigo_carga <= 20) entao
18         frete_por_quilo <- 5.00
19         percentual_seguro <- 0.03 // 3%
20     senao
21         se (codigo_carga >= 21) e (codigo_carga <= 30) entao
22             frete_por_quilo <- 8.00
23             percentual_seguro <- 0.05 // 5%
24         senao
25             se (codigo_carga >= 31) e (codigo_carga <= 40) entao
26                 frete_por_quilo <- 10.00
27                 percentual_seguro <- 0.08 // 8%
28             senao
29                 frete_por_quilo <- 0
30                 percentual_seguro <- 0
31                 escreval("Código da carga inválido.")
32         fimse
33     fimse
34 fimse
35
36 // Só executa os cálculos e exibe resultados se o código for válido
37 se (frete_por_quilo > 0) entao
38     // Conversão de toneladas para quilos (1 tonelada = 1000 kg)
39     peso_quilos <- peso_toneladas * 1000
40     // Cálculos financeiros

```

```
41 valor_frete <- peso_quilos * frete_por_quilo
42 valor_seguro <- valor_carga * percentual_seguro
43 valor_total <- valor_frete + valor_seguro
44 // Saída formatada
45 escreval("Peso da carga: ", peso_quilos, " Kg")
46 escreval("Frete: R$ ", valor_frete:8:2)
47 escreval("Seguro: R$ ", valor_seguro:8:2)
48 escreval("Valor total: R$ ", valor_total:8:2)
49 fimse
50 fimalgoritmo
```

```
Código da carga: 10
Peso da carga (t): 25000
Valor da carga: R$ 33000
-----
Peso da carga: 25000000 Kg
Frete: R$ 125000000.00
Seguro: R$ 990.00
Valor total: R$ 125000990.00

>>> Fim da execução do programa !
```