lista 3

exercício 1)

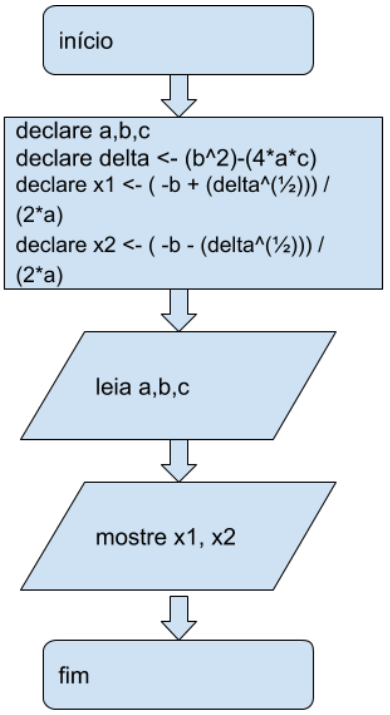
linguagem estruturada

1- início
2- declare a,b,c
3- leia a,b,c
4- declare delta <- (b^2) - (4*a*c)
5- declare x1 <- (-b + (delta^(½))) / (2*a)
6- declare x2 <- (-b - (delta^(½))) / (2*a)
7- mostre x1, x2
8- fim

linguagem natural

1- início
2- declarar 3 variáveis
3- pedir para ler as três variáveis
4- declarar delta valendo (b^2) - (4*a*b)
5- declarar x1 para receber o primeiro resultado valendo (-b + (delta^(½))) / (2*a)
6- declarar x2 para receber o primeiro resultado valendo (-b - (delta^(½))) / (2*a)
7- mostrar as duas variáveis
8- fim

fluxograma



teste de mesa

a <- 4
b<- 8

delta <- 8^2 - 4*4*8 = 64
x1 <- -8 + 64 / 8 = 7
x2 <- -8 - 64 / 8 = 9

x1 <- 7   x2 <- 9

exercício 2)

linguagem estruturada

```
ccaracter =1
cespaco = 1
folha = 1

altura = int(input("Digite a altura da árvore: "))
caracter = input("Digite o caracter: ")

if (altura < 6):
  altura = 6
if (altura > 24):
  altura = 24
folha = 1
contador = 1
while ( contador <= (altura -3)):
  cespaco = 1
  while ( cespaco < ( 40 - contador)):
    print(" ", end="")
    cespaco = cespaco + 1
  ccaracter = 1
  while (ccaracter <= folha):
    print(caracter, end="")
    ccaracter = ccaracter + 1
  print("")
  folha = folha + 2
  contador = contador + 1

contador = 1
while ( contador <= 2):
  cespaco = 1
  while ( cespaco < 39):
    print(" ", end="")
    cespaco = cespaco + 1
  print(caracter)
  contador = contador + 1
```

```
cespaco = 1
while ( cespaco <37):
  print(" ", end="")
  cespaco = cespaco + 1

ccaracter = 1
while ( ccaracter <= 5):
  print(caracter, end="")
  ccaracter = ccaracter + 1
print(" ")
```

linguagem natural

fluxograma

teste de mesa

exercício 3)

linguagem estruturada

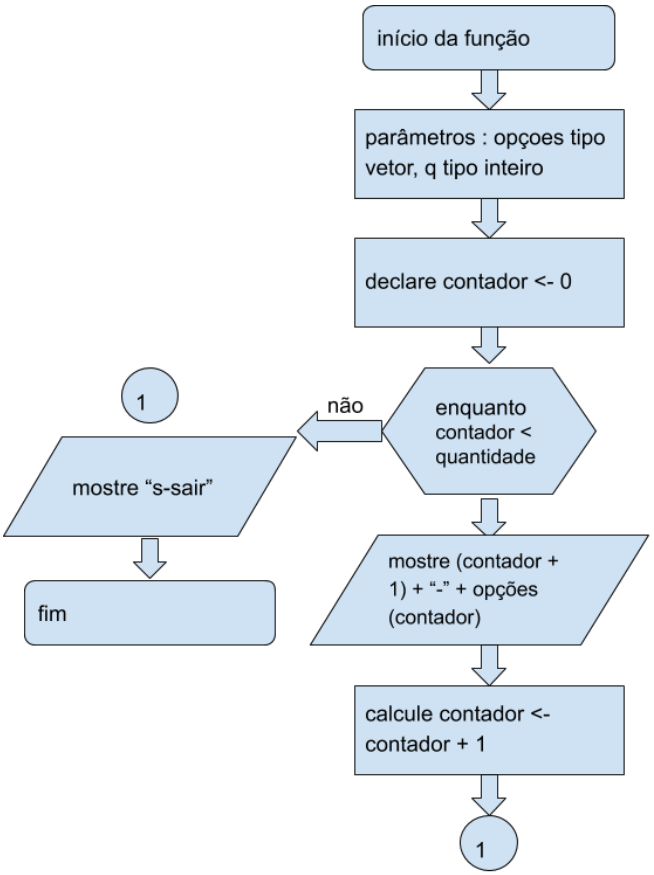
1- função menu
2- parâmetros: opções tipo vetor, q tipo inteiro
3- declare contador <- 0
4- enquanto contador < quantidade faça
        4.1- mostre (contador + 1) + “-” + opções (contador)
        4.2- calcule contador <- contador + 1
5- fim faça
6- mostre “s - sair”
7- fim

linguagem natural

1- declarar função
2- colocar um parâmetro opções tipo vetor e outro q tipo inteiro
3- declarar um contador valendo 0

4- enquanto o contador for menor que a quantidade
5- mostre contador + 1 um espaço e opções recebendo contador
6- calcule contador somando + 1
7- mostre “s- sair”
8- fim

fluxograma



teste de mesa

opçoes <- adicionar, remover, limpar
q <- 3
1 - adicionar
contador <- 0 + 1
2 - remover
contador <- 1 + 1
3 - limpar
contador <- 2 + 1
s - sair

exercício 4)

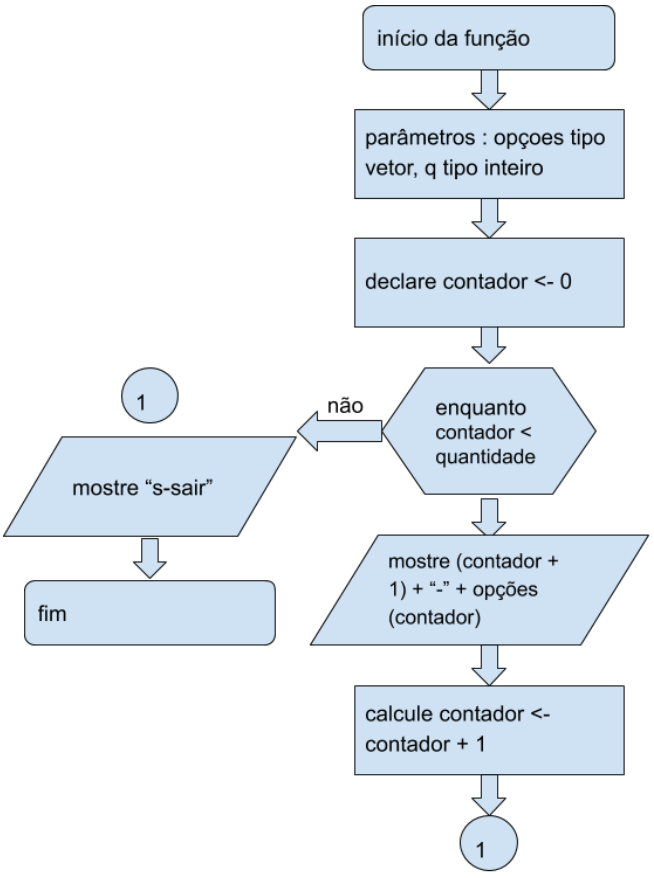
linguagem estruturada

1- início
2- declare km, tanque,media
3- enquanto km >= 0 faça
    3.1- leia km
    3.2- se km >= 0
        3.2.1- leia tanque
        3.2.2- calcule média <- km/tanque
        3.2.3- mostre média
    3.3- fim se
4- fim faça
5- fim

linguagem natural

1- início
2- declarar 3 variáveis, km,tanque,media
3- enquanto km for maior ou igual a zero
4- leia km
5- se km for maior ou igual a zero
6- leia tanque
7- calcule media recebendo km dividido por tanque
8- mostre media
9- fim

fluxograma



teste de mesa

km <- 100
tanque <- 50
media <- 100/50 = 2

exercício 5)

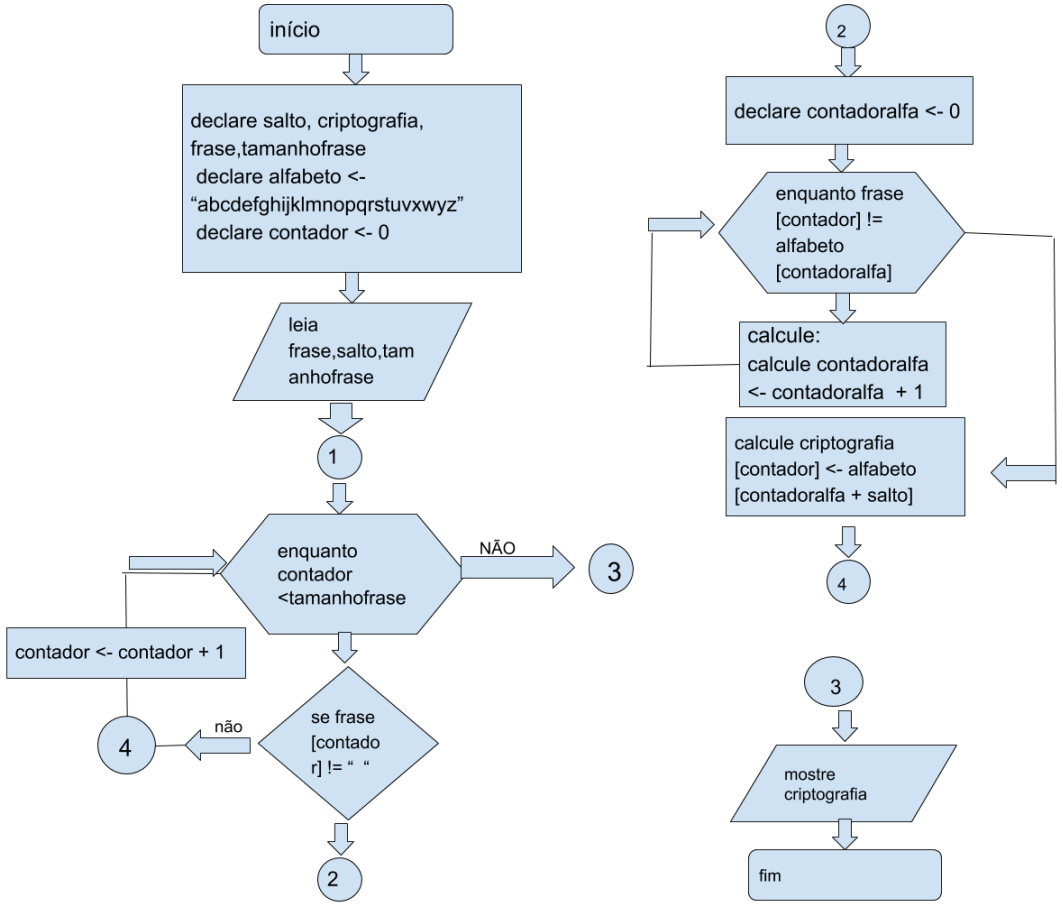
linguagem estruturada

1- início
2- declare salto, criptografia, frase,tamanhofrase
3- declare alfabeto <- “abcdefghijklmnopqrstuvxwyz”
4- declare contador <- 0
5- declare contadoralfa <- 0
6- leia frase,salto,tamanhofrase
7- enquanto contador < tamanhofrase faça
    7.1- se frase (contador) != “ “
        7.1.1- enquanto frase (contador) != alfabeto (contadoralfa)
            7.1.1.1- calcule contadoralfa <- contadoralfa + 1
        7.1.2- calcule criptografia (contador) <- alfabeto (contadoralfa + salto)
    7.2- fim se
    7.3- calcule <- contador + 1
8- fim faça
9- mostre criptografia
10- fim

linguagem natural

1- início
2- declare 4 variáveis, salto, criptografia,frase e tamanhofrase
3- declare alfabeto recebendo todas as letras do alfabeto
4- declare um contador valendo a zero
5- declare outro contador pro alfabeto valendo zero
6- ler frase,salto e tamanhofrase
7- enquanto o contador for menor que o tamanhofrase
8- se frase usando contador for diferente de “ “
9- e enquanto frase usando contador for diferente de alfabeto usando o contador de alfabeto
10- calcule contador do alfabeto recebendo + 1
11- calcule criptografia usando o contador recebendo alfabeto usando contador do alfabeto mais o salto
12- calcule contador recebendo mai 1
13- mostre criptografia
14- fim

fluxograma



teste de mesa

alfabeto <- “abcdefghijklmnopqrstuvxwyz
frase <- oi
salto <- 2
tamanhofrase <- 2
Contador <- 0
contadoralfa <- 0
Contdoralfa <- 0 + 1
Contdoralfa <- 1 + 1
….(Isso se repete 11 vezes)
Contdoralfa <- 13 + 1
Criptografia{0} <- alfabeto {14 + 2}
Contador <- 0 + 1
Contadoralfa <- 0
Contdoralfa <- 0 + 1
Contdoralfa <- 1 + 1
….(Isso se repete 6 vezes)
Contdoralfa <- 8 + 1
Criptografia{1} <- alfabeto {9 + 2}
Criptografia <- qk

exercício 6)

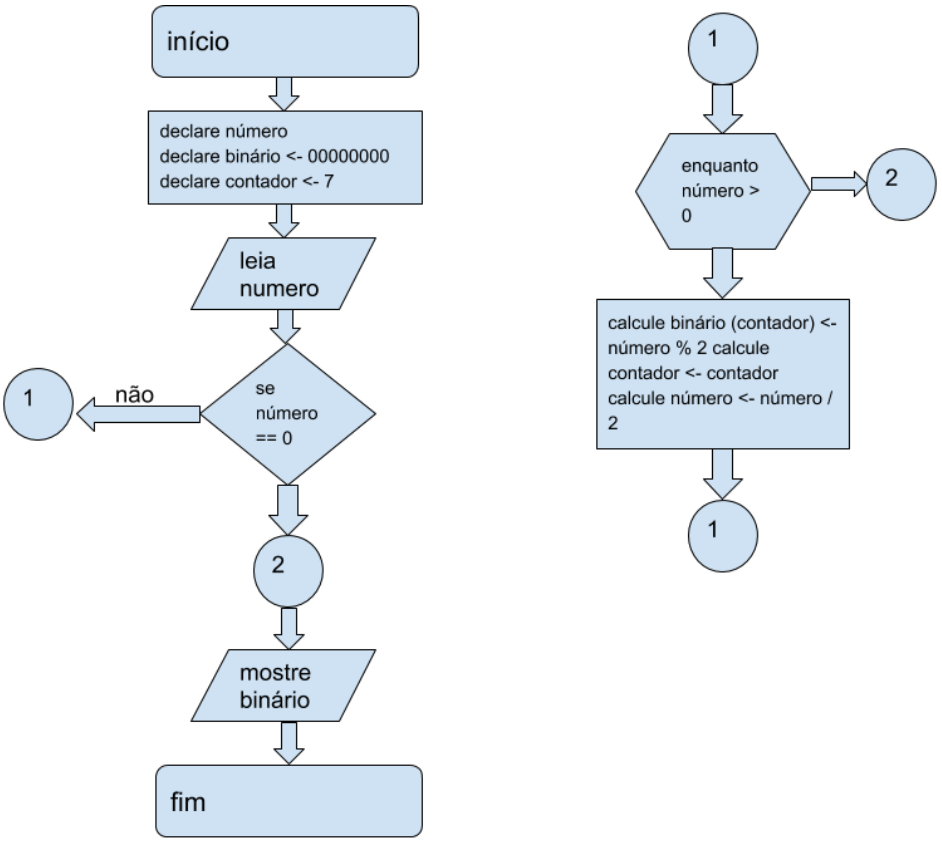
linguagem estruturada

1- início
2- declare número
3- declare binário <- 00000000
4- declare contador <- 7
5- leia número
6- se número == 0
7- se não
    7.1- enquanto número > 0 faça
        7.1.1- calcule binário (contador) <- número % 2
        7.1.2- calcule contador <- contador - 1
        7.1.3- calcule número <- número / 2
    7.2- fim faça
8- fim se
9- mostre binário
10- fim

linguagem natural

1- início
2- declare número
3- declare binario recebendo 8 zeros
4- declare um contador recebendo 7
5- leia número
6- se numero for igual a zero para
7- se não, enquanto número for maior que zero
8- calcule binario usando o contador recebendo o resto da divisão por 2 do número
9- calcule contador recebendo mais 1
10- calcule numero recebendo numero dividido por 2
11- mostre binario
12- fim

fluxograma



teste de mesa

numero <- 40
binario (7) <- 40 % 2 = 0
contador <- 7-1 = 6
numero <- 40/2 = 20
binário (6) <- 20 % 2 = 0
contador <- 6-1 = 5
numero <- 20/2 = 10
binário (5) <- 10 % 2 = 0
contador <- 5-1 = 4
numero <- 10/2 = 5
binário (4) <- 5 % 2 = 1
contador <- 4-1 = 3
numero <- 5/2 = 2
binário (3) <- 2 % 2 = 0
contador <- 3-1 = 2
numero <- 2/2 = 1
binário (2) <- 1 % 2 = 1
contador <- 2-1 = 1
numero <- 1/2 = 0
Binario <- 101000

exercício 7)

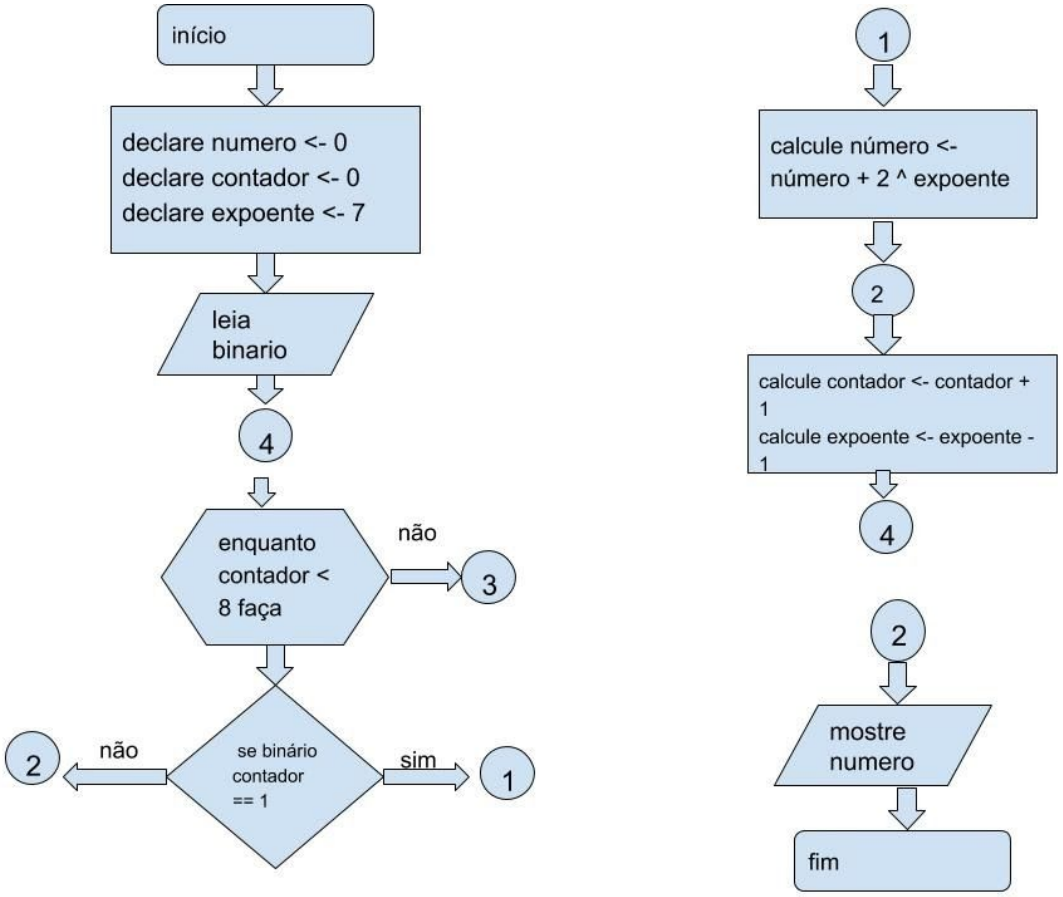
linguagem estruturada

1- início
2- declare binário
3- declare numero <- 0
4- declare contador <- 0
5- declare expoente <- 3
6- leia binário
7- enquanto contador < 3 faça
    7.1- se binário [contador] == 1
        7.1.1- calcule número <- número + 2 ^ expoente
    7.2- fim se
    7.3- calcule contador <- contador + 1
    7.4- calcule expoente <- expoente - 1
8- fim faça
9- mostre número
10- fim

linguagem natural

1- início
2- declare binario
3- declare numero recebendo zero
4- declare contador recebendo zero
5- declare expoente recebendo 7
6- leia binario
7- enquanto contador for menor que 8
8- se binario receber 1
9- calcule numero recebendo numero + 2 ^expoente
10- calcule contador e expoente recebendo + 1
11- mostre numero
12- fim

fluxograma



teste de mesa

binario <- 0101
calcule número <- 1 * 2 ^ 2 = 4
calcule contador <- 0 + 1 = 1
calcule expoente <- 3 - 1 = 2
calcule número <- 1 * 2 ^ 1 = 1
calcule contador <- 1 + 1 = 2
calcule expoente <- 1 - 1 = 0
numero <- 4 + 1 = 5

exercício 8)

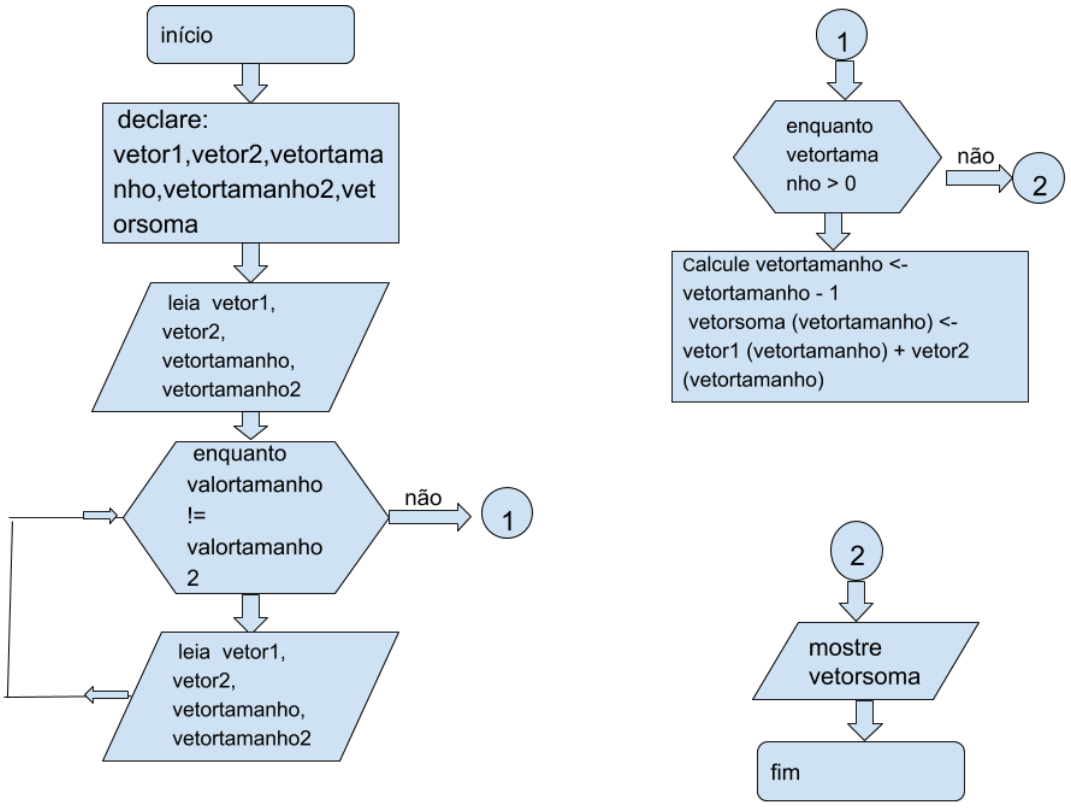
linguagem estruturada

1- início
2- declare vetor1,vetor2,vetortamanho,vetortamanho2,vetorsoma
3- leia vetor1, vetor2, vetortamanho, vetortamanho2
4- enquanto valortamanho != valortamanho2 faça
    4.1- leia vetor1, vetor2, vetortamanho, vetortamanho2
5- fim faça
6- enquanto vetortamanho > 0
    6.1- calcule vetortamanho <- vetortamanho - 1
    6.2- vetorsoma (vetortamanho) <- vetor1 (vetortamanho) + vetor2 (vetortamanho)
7- fim faça
8- mostre vetorsoma
9- fim

linguagem natural

1- início
2- declare 5 variáveis, vetor1,vetor2,vetortamanho,vetortamanho2,vetorsoma
3- leia vetor1, vetor2, vetortamanho, vetortamanho2
4- enquanto valortamanho for diferente de valortamanho2
5- leia vetor1, vetor2, vetortamanho, vetortamanho2
6- enquanto vetortamanho for maior que 0
7- calcule vetortamanho recebendo vetortamanho - 1
8- calcule vetorsoma usando vetortamanho recebendo vetor1 usando vetortamanho + vetor2 usando vetortamanho
9- mostre vetorsoma
10- fim

fluxograma
I’ll



teste de mesa

vetor <- [1,2]
vetor2 <- [1,5]
vetortamanho <- 2
vetortamanho2 <- 2
Vetortamanho <- 2 - 1
vetorsoma (2) <- 2 [1] + 5 [1] = 7
calcule vetortamanho <- 2 - 1 = 1
vetorsoma (1) <- 1 [0] + 1 [0] = 2
vetorsoma <- [2,7]

exercício 9)

linguagem estruturada

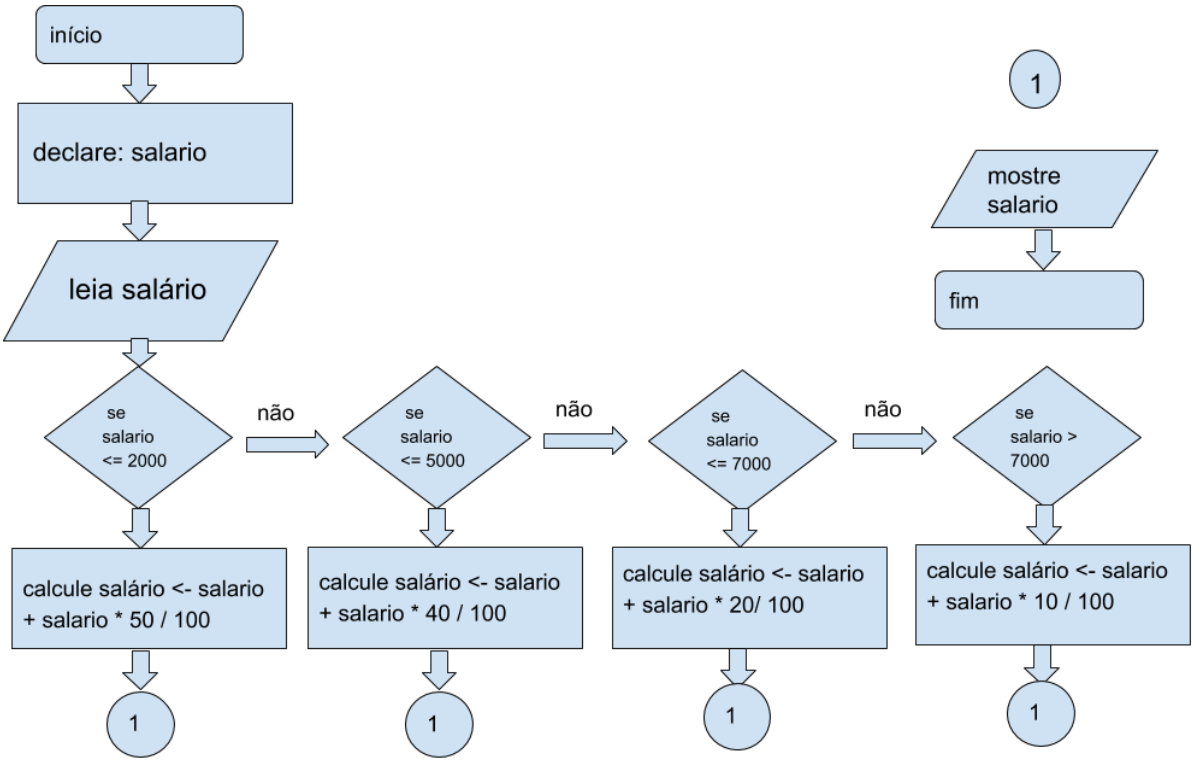
1- início
2- declare salario
3- leia salario
4- se salario <= 2000
    4.1- calcule salário <- salario + salario * 50 / 100
5- se não
    5.1- se salario <= 5000
        5.1.1- calcule salario <- salário + salario * 40 / 100
    5.2- se não
        5.2.1- se salario <= 7000
            5.2.1.1- calcule salario <- salario + salario * 20 / 100
        5.2.2- se não
            5.2.2.1- calcule salario <- salario + salario * 10 / 100
        5.2.3- fim se
6- fim se
7- mostre salario
8- fim

linguagem natural

1- início
2- declare salário
3- leia salário
4- se o salario for menor ou igual a 2000
5- calcule salário recebendo salário + salário vezes 50 /100
6- se o salario for menor ou igual a 5000
7- calcule salário recebendo salário + salário vezes 40 / 100
8- se o salario for menor ou igual a 7000
9- calcule salário recebendo salário + salário vezes 20 / 100
10- se não calcule salário recebendo salário + salário vezes 10 / 100

11- mostre salário
12- fim

fluxograma



teste de mesa

salario <- 1600
calcule <- 1600 + 1600 * 50 / 100
salário <- 2400

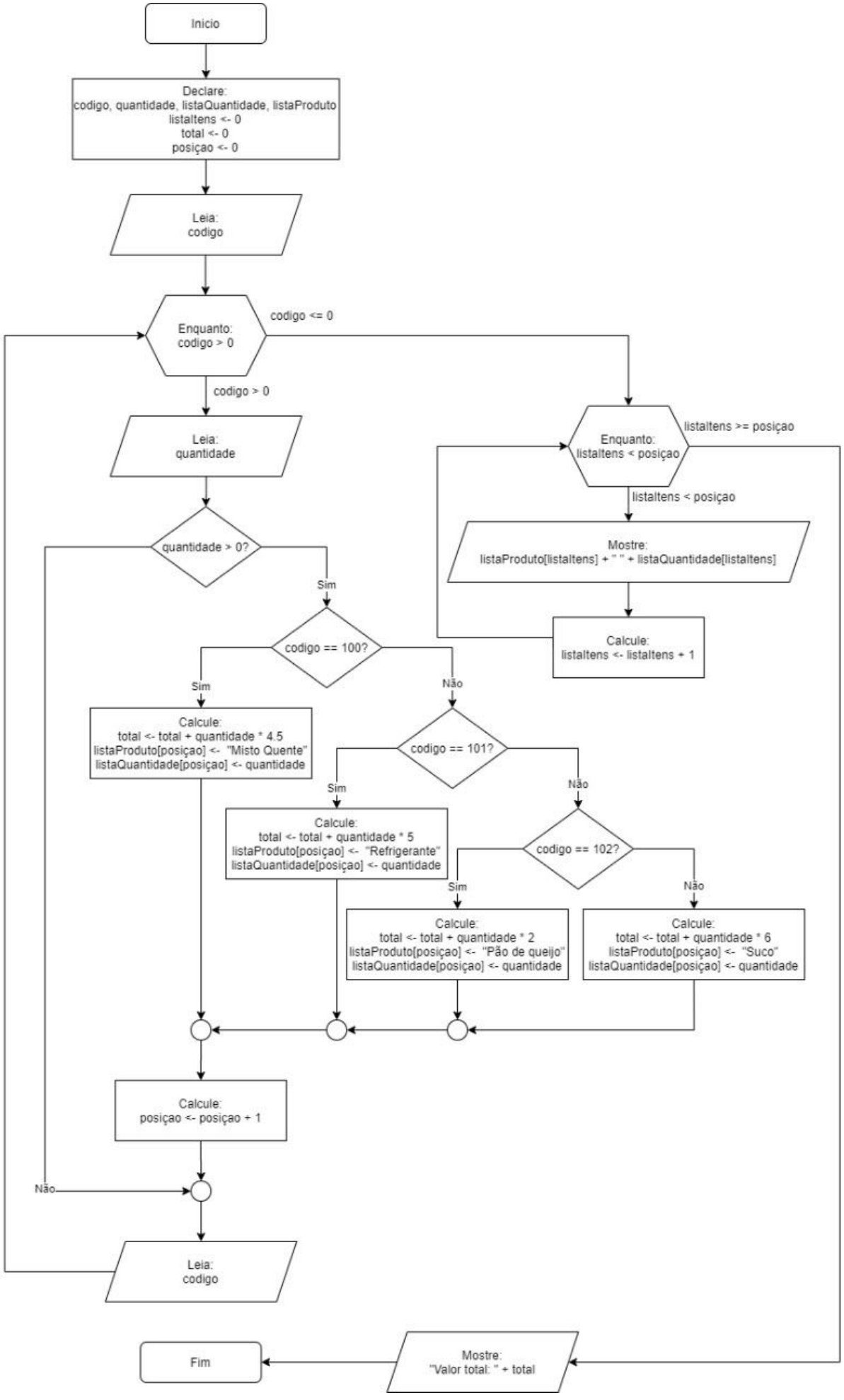
exercício 10)

1- início
2- declare codigo, quantidade, listaquantidade,valor,listaproduto
3- declare posição <- 0
4- declare lista <- 0
5- leia codigo, quantidade
6- enquanto codigo > 0 faça
7- leia- codigo, quantidade
7.1- se codigo == 100 então
7.1.1- calcule- valor <- valor+ quantidade*4.5
7.1.2- calcule listaproduto[posiçao] <- “misto quente”
7.1.3- calcule listaquantidade[posiçao] <- quantidade
7.2- fim se
7.3- se não
7.3.1- se codigo == 101 então
7.3.1.1calcule- valor <- valor + quantidade*5.0
7.3.1.2calcule listaproduto[posiçao] <- “refrigerante”
7.3.1.3calcule listaquantidade[posiçao] <- quantidade
7.4- fim se
7.5- se não
7.5.1se codigo == 102 então
7.5.1.1- calcule- valor <- valor + quantidade*2.0
7.5.1.2- calcule listaproduto[posiçao] <- “pão de queijo”
7.5.1.3- calcule listaquantidade[posiçao] <- quantidade
7.6- fim se
7.7- se não
7.7.1- se codigo == 103 então
7.7.1.1- calcule- valor <- valor + quantidade*6.0
7.7.1.2- calcule listaproduto[posiçao] <- “suco”
7.7.1.3- calcule listaquantidade[posiçao] <- quantidade
7.8- fim se
7.9- calcule posição <- posição + 1
7.10- leia codigo
8- fim faça
9- enquanto lista <= posiçao
9.1- MOSTRE listaProduto[lista] + " " + listaQuantidade[lista]
9.2- CALCULE lista <- lista + 1
10- FIM FAÇA
11- MOSTRE "Valor total: " + valor
12- FIM

linguagem natural

1- início
2- declare codigo, quantidade, listaquantidade,valor,listaproduto
3- declare posição recebendo 0
4- declare lista recebendo 0
5- leia codigo e quantidade
6- enquanto codigo for maior que 0 leia codigo e quantidade
7- se codigo for igual a 100 multiplique a quantidade por 4.5, coloque misto quente na listaproduto usando posição, calcule listaquantidade usando posição recebendo quantidade
8- se codigo for igual a 101 multiplique a quantidade por 5, coloque refrigerante na listaproduto usando posição, calcule listaquantidade usando posição recebendo quantidade
9- se codigo for igual a 102 multiplique a quantidade por 2, coloque pão de queijo na listaproduto usando posição, calcule listaquantidade usando posição recebendo quantidade
10- se codigo for igual a 103 multiplique a quantidade por 6, coloque suco na listaproduto usando posição, calcule listaquantidade usando posição recebendo quantidade
11- calcule posiçao recebendo posiçao + 1
12- lea codigo
13- enquanto lista for menor ou igual a posiçao MOSTRE listaProduto[lista] + " " + listaQuantidade[lista]
14- calcule lista recebendo lista + 1
15- mostre valor total
16- fim

fluxograma


Inicio
Declare:
codigo, quantidade, listaQuantidade, listaProduto
listaItens <- 0
total <- 0
posiçao <- 0
Leia:
codigo
Enquanto:
codigo > 0
codigo <= 0
codigo > 0
Leia:
quantidade
quantidade > 0?
Sim
codigo == 100?
Sim
Calcule:
total <- total + quantidade * 4.5
listaProduto[posiçao] <- "Misto Quente"
listaQuantidade[posiçao] <- quantidade
Não
codigo == 101?
Sim
Calcule:
total <- total + quantidade * 5
listaProduto[posiçao] <- "Refrigerante"
listaQuantidade[posiçao] <- quantidade
Não
codigo == 102?
Sim
Calcule:
total <- total + quantidade * 2
listaProduto[posiçao] <- "Pão de queijo"
listaQuantidade[posiçao] <- quantidade
Não
Calcule:
total <- total + quantidade * 6
listaProduto[posiçao] <- "Suco"
listaQuantidade[posiçao] <- quantidade
Calcule:
posiçao <- posiçao + 1
Não
Leia:
codigo
Enquanto:
listaItens < posiçao
listaItens >= posiçao
listaItens < posiçao
Mostre:
listaProduto[listaItens] + " " + listaQuantidade[listaItens]
Calcule:
listaItens <- listaItens + 1
Mostre:
"Valor total: " + total
Fim

fluxograma

Teste de Mesa

Passo	Comando	Variáveis							Mostre
		codigo	quantidade	listaQuantidade	listaProduto	listaItens	total	posiçao	
3	listaItens <- 0					0			
4	total <- 0					0	0		
5	posiçao <- 0					0	0	0	
6	leia: codigo	101				0	0	0	
7.1	leia: quantidade	101	4			0	0	0	
7.2.2.1.1	total <- total + quantidade * 5	101	4			0	20	0	
7.2.2.1.2	listaProduto[posiçao] <- "Refrigerante"	101	4		[“Refrigerante”]	0	20	0	
7.2.2.1.3	listaQuantidade[posiçao] <- quantidade	101	4	[4]	[“Refrigerante”]	0	20	0	
7.3	posiçao <- posiçao + 1	101	4	[4]	[“Refrigerante”]	0	20	1	
7.5	leia: codigo	102	4	[4]	[“Refrigerante”]	0	20	1	
7.1	leia: quantidade	102	10	[4]	[“Refrigerante”]	0	20	1	
7.2.2.1.1	total <- total + quantidade * 2	102	10	[4]	[“Refrigerante”]	0	40	1	

7.2.2.1.2	listaProduto[posiçao] <- "Pão de queijo"	102	10	[4]	[“Refrigerante”, "Pão de queijo"]	0	40	1	
7.2.2.1.3	listaQuantidade[posiçao] <- quantidade	102	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	0	40	1	

7.5	leia: codigo	-1	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	0	40	1	
9.1	mostre: listaProduto[listaItens] + " " + listaQuantidade[listaItens]	-1	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	0	40	1	Refrigerante - 4
9.2	listaItens <- listaItens + 1	-1	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	1	40	1	
9.1	mostre: listaProduto[listaItens] + " " + listaQuantidade[listaItens]	-1	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	1	40	1	Pão de queijo - 10
11	mostre: "Valor total: " + total + “R$”	-1	10	[4, 10]	[“Refrigerante”, "Pão de queijo"]	1	40	1	Valor total: 40R$