Universidade Estadual de Santa Cruz – UESC

Discente: Lucas Vieira de Almeida	Matrícula: 202320101	Curso: Ciência da Computação
Docente: Marcelo Ossamu Honda	Disciplina: Linguagem de Programação I	Data de entrega: 13/09/2023

ALGORITMOS

1. Salário Bruto x Salário Líquido

- **Pseudocódigo**

```
Algoritmo "salário_líquido"

VAR
sbruto, sliquido, inss, irpf, vale, taxas, desconto, sminimo = real

Inicio

 Escreva(“Valor do salário bruto:”)
 Leia(sbruto)
 Escreva(“Valor do vale alimentação(insira 0 caso não receba):”)
 Leia(vale)

 inss <- sbruto * (9.0/100)
 irpf <- sbruto * (27.5/100)
 taxas <- inss + irpf
 desconto <- taxas + vale
 sliquido <- sbruto – desconto⁹

 Se (sliquido < 1320), então
```

sminimo <- 1320 + vale

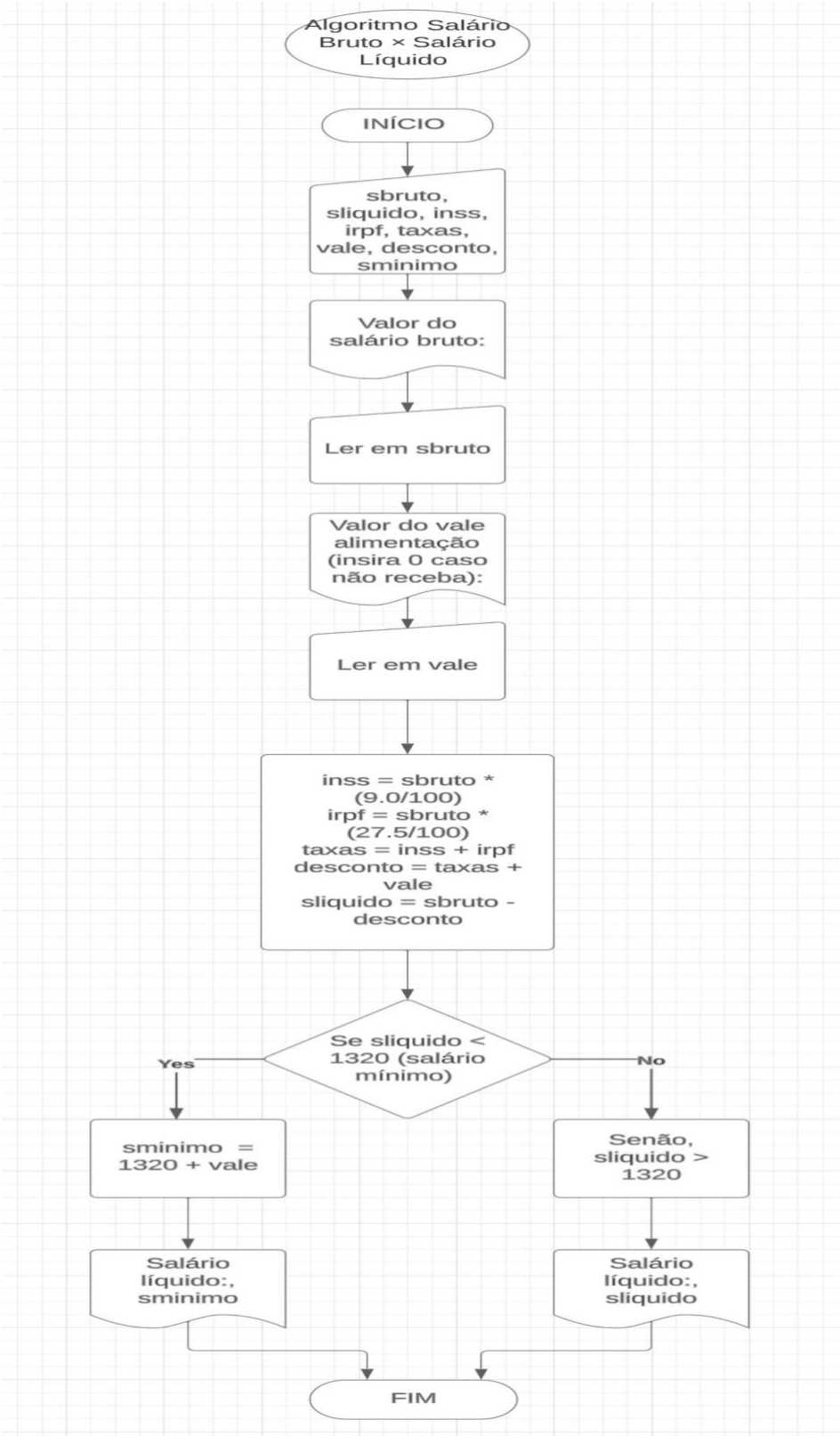
Escreva("Salário líquido: R$, sminimo")

Senão, se (sliquido > 1320), então

Escreva(Salário líquido: R$, sliquido)

FimAlgoritmo

- ***Fluxograma***

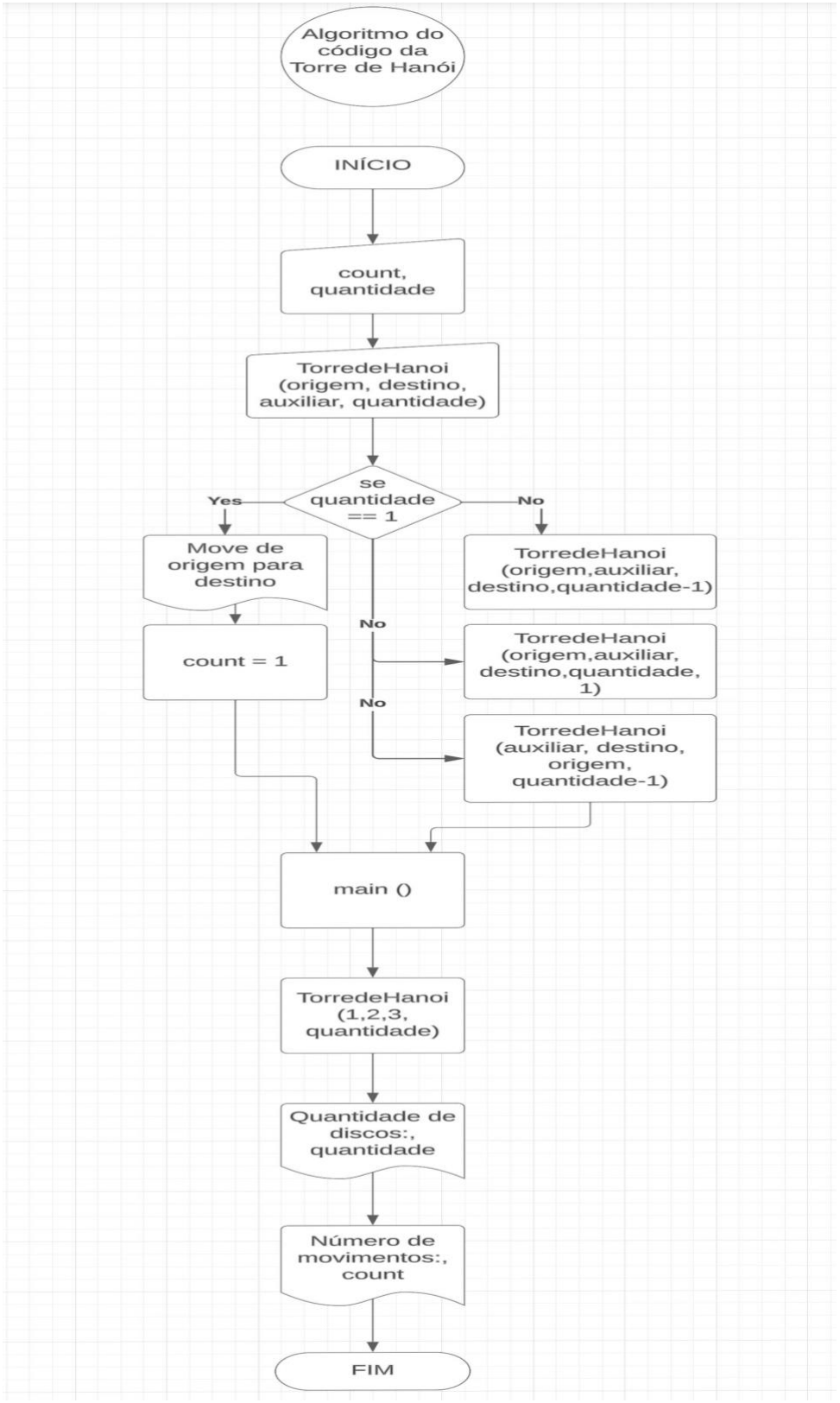

Algoritmo Salário Bruto × Salário Líquido
INÍCIO
sbruto, sliquido, inss, irpf, taxas, vale, desconto, sminimo
Valor do salário bruto:
Ler em sbruto
Valor do vale alimentação (insira 0 caso não receba):
Ler em vale
inss = sbruto * (9.0/100)
irpf = sbruto * (27.5/100)
taxas = inss + irpf
desconto = taxas + vale
sliquido = sbruto - desconto
Se sliquido < 1320 (salário mínimo)
Yes
No
sminimo = 1320 + vale
Senão, sliquido > 1320
Salário líquido:, sminimo
Salário líquido:, sliquido
FIM

2. **Torre de Hanói**

- **Pseudocódigo Algoritmo da Torre de Hanói**

```
Algoritmo "torre_de_hanoi"

VAR
  count = 0, quantidade = 3

Início

TorredeHanoi(origem,destino,auxiliar, quantidade )

Se quantidade == 1, então
Escreva("Move de origem para destino")
count <- 1
Senão
TorredeHanoi(origem,destino, auxiliar e quantidade -1)
TorredeHanoi(origem, destino, auxiliar, 1)
TorredeHanoi(auxiliar,destino,origem, quantidade-1)

main ()

TorredeHanoi(1,2,3,quantidade)
Escreva("Quantidade de discos:", quantidade)
Escreva("Número de movimentos:", count)

FimAlgoritmo
```

- ***Fluxograma***


Algoritmo do código da Torre de Hanói
INÍCIO
count, quantidade
TorredeHanoi (origem, destino, auxiliar, quantidade)
se quantidade == 1
Yes
No
Move de origem para destino
count = 1
TorredeHanoi (origem,auxiliar, destino,quantidade-1)
No
TorredeHanoi (origem,auxiliar, destino,quantidade, 1)
No
TorredeHanoi (auxiliar, destino, origem, quantidade-1)
main ()
TorredeHanoi (1,2,3, quantidade)
Quantidade de discos:, quantidade
Número de movimentos:, count
FIM

3. ***Array 3D***

- ***Pseudocódigo***

Algoritmo Array 3D

// Definindo as dimensões do array tridimensional

inteiro array[10][20][5]

// Preenchendo o array com valores sequenciais
valor = 1

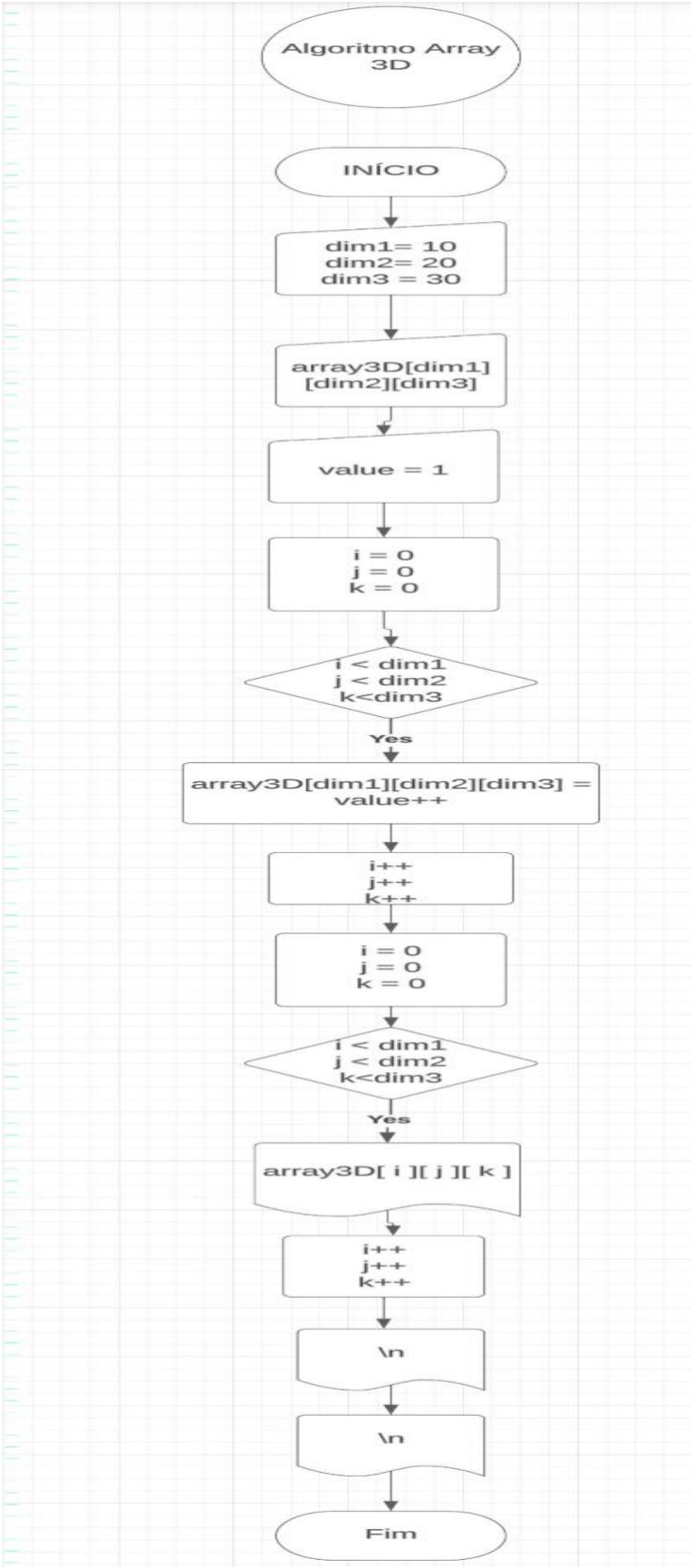
para cada i de 0 até 9
para cada j de 0 até 19
para cada k de 0 até 4
array[i][j][k] = valor
valor = valor + 1
fim para
fim para
fim para

// Apresentando os valores do array

imprimir("Valores do array tridimensional:")
para cada i de 0 até 9
para cada j de 0 até 19
para cada k de 0 até 4
imprimir(array[i][j][k], " ")
fim para
imprimir("\n")

```
fim para

imprimir("\n")

FimAlgoritmo
```

- ***Fluxograma***


Algoritmo Array 3D
INÍCIO
dim1= 10
dim2= 20
dim3 = 30
array3D[dim1] [dim2][dim3]
value = 1
i = 0
j = 0
k = 0
i < dim1
j < dim2
k<dim3
Yes
array3D[dim1][dim2][dim3] = value++
i++
j++
k++
i = 0
j = 0
k = 0
i < dim1
j < dim2
k<dim3
Yes
array3D[i][j][k]
i++
j++
k++
\n
\n
Fim

4. *Array Unimensional*

- ***Pseudocódigo***

```
Função bubbleSort(arr, n):
    Para i de 0 até n – 1:
        Para j de 0 até n – i – 1:
            Se arr[j] > arr[j + 1]:
                Trocar arr[j] com arr[j + 1]

Função calcularMedia(arr, n):
    Soma = 0
    Para i de 0 até n – 1:
        Soma += arr[i]
    Retornar soma / n

Função calcularMediana(arr, n):
    Se n % 2 == 0:
        Retornar (arr[n / 2 – 1] + arr[n / 2]) / 2.0
    Senão:
        Retornar arr[n / 2]

Função calcularDesvioPadrao(arr, n, media):
    somaQuadrados = 0
    Para i de 0 até n – 1:
        somaQuadrados += (arr[i] – media) ^ 2
    Retornar raizQuadrada(somaQuadrados / n)

Função encontrarValoresRepetidos(arr, n):
    valorAtual = arr[0]
```

```
    contagem = 1
    Imprimir “Valores Repetidos:”
    Para i de 1 até n – 1:
        Se arr[i] == valorAtual:
            Incrementar contagem
        Senão:
            Imprimir “Valor:”, valorAtual, “Quantidade:”, contagem
            valorAtual = arr[i]
            contagem = 1
    Imprimir “Valor:”, valorAtual, “Quantidade:”, contagem

Função removerRepetidos(arr, n, novoTamanho):
    novoArray = AlocarMemoria(n * tamanho(int))
    Se falhaNaAlocacao(novoArray):
        Sair com código de falha
    valorAtual = arr[0]
    novoArray[0] = valorAtual
    *novoTamanho = 1
    Para i de 1 até n – 1:
        Se arr[i] != valorAtual:
            valorAtual = arr[i]
            novoArray[*novoTamanho] = valorAtual
            Incrementar *novoTamanho
    Retornar novoArray

Função principal(argc, argv):
    Declarar e Inicializar myArray[TAM]
    Inicializar gerador de números aleatórios com o tempo atual
    Imprimir “Elemento      Valor”
    Para i de 0 até TAM – 1:
```

```
    myArray[i] = GerarNúmeroAleatório(0, 1000)
    Imprimir i + 1, myArray[i]

Chamar bubbleSort(myArray, TAM)

Imprimir “Array Ordenado:”
Imprimir “Elemento      Valor”
Para i de 0 até TAM – 1:
    Imprimir i + 1, myArray[i]

Media = calcularMedia(myArray, TAM)
Mediana = calcularMediana(myArray, TAM)
desvioPadrao = calcularDesvioPadrao(myArray, TAM, media)

Imprimir “Média:”, media
Imprimir “Mediana:”, mediana
Imprimir “Desvio Padrão:”, desvioPadrao

encontrarValoresRepetidos(myArray, TAM)

novoTamanho
novoArray = removerRepetidos(myArray, TAM, &novoTamanho)

Imprimir “Novo Array sem Repetições:”
Imprimir “Elemento      Valor”
Para i de 0 até novoTamanho – 1:
    Imprimir i + 1, novoArray[i]

Media = calcularMedia(novoArray, novoTamanho)
Mediana = calcularMediana(novoArray, novoTamanho)
```

desvioPadrao = calcularDesvioPadrao(novoArray, novoTamanho, media)

Imprimir “Média (Novo Array):”, media

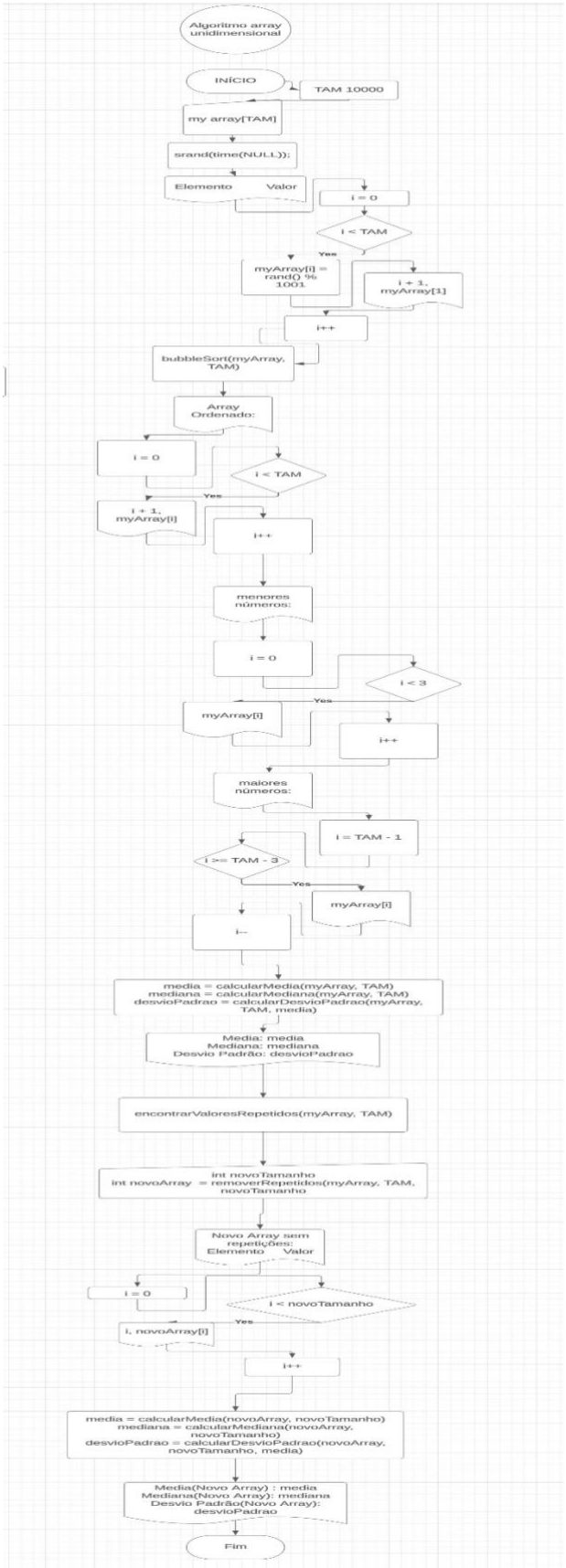
Imprimir “Mediana (Novo Array):”, mediana

Imprimir “Desvio Padrão (Novo Array):”, desvioPadrao

LiberarMemoria(novoArray)

FimAlgoritmo

- ***Fluxograma***


Algoritmo array unidimensional
INÍCIO
TAM 10000
my array[TAM]
srand(time(NULL));
Elemento Valor
i = 0
i < TAM
Yes
myArray[i] = rand() % 1001
i + 1, myArray[1]
i++
bubbleSort(myArray, TAM)
Array Ordenado:
i = 0
i < TAM
Yes
i + 1, myArray[i]
i++
menores números:
i = 0
i < 3
Yes
myArray[i]
i++
maiores números:
i = TAM - 1
i >= TAM - 3
Yes
myArray[i]
i--
media = calcularMedia(myArray, TAM)
mediana = calcularMediana(myArray, TAM)
desvioPadrao = calcularDesvioPadrao(myArray, TAM, media)
Media: media
Mediana: mediana
Desvio Padrão: desvioPadrao
encontrarValoresRepetidos(myArray, TAM)
int novoTamanho
int novoArray = removerRepetidos(myArray, TAM, novoTamanho
Novo Array sem repetições:
Elemento Valor
i = 0
i < novoTamanho
Yes
i, novoArray[i]
i++
media = calcularMedia(novoArray, novoTamanho)
mediana = calcularMediana(novoArray, novoTamanho)
desvioPadrao = calcularDesvioPadrao(novoArray, novoTamanho, media)
Media(Novo Array) : media
Mediana(Novo Array): mediana
Desvio Padrão(Novo Array): desvioPadrao
Fim