Disciplina	Prof. Dacio Machado	
PROJETO IMPLEMENTAÇÃO E TESTE DE SOFTWARE	Valor	+01 ATV
ATIVIDADE: TESTE ESTRUTURAL	Aluno: Andrey Eduardo Indras RA: 23050861-2	
ESOFT - 6 - N - C		

Atividade prática de teste Estrutural Passos:

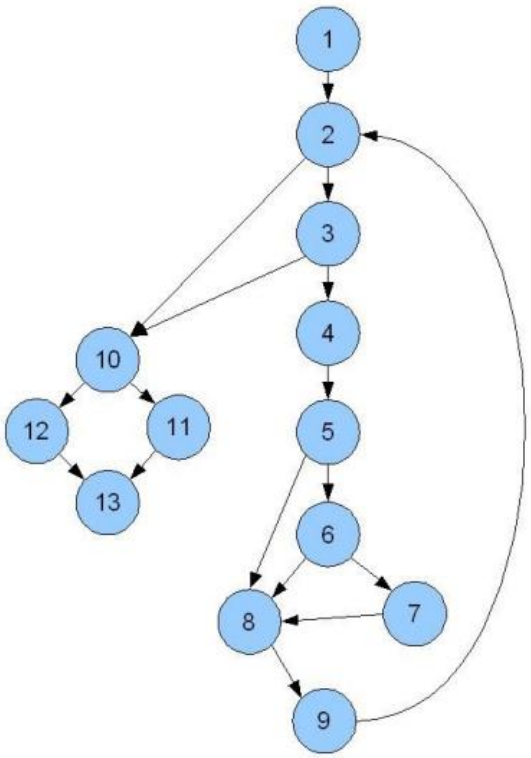
1. Projetar **casos de teste estruturais** para avaliar os quatro algoritmos dos itens listados abaixo. Conforme o exemplo abaixo, e o excerto do Livro Didático.
2. Preencher os ARTEFATOS de teste abaixo para os testes projetados.
3. Construa, em sua linguagem de preferência os seguintes algoritmos:
    a. Um algoritmo que lê um número e imprime a lista dos seus divisores
    b. Um algoritmo que lê dois números e calcula o máximo divisor comum pelo método de Euclides.
    c. Um algoritmo que lê as 4 notas de um aluno e diga se ele passou por média, está em final ou reprovou
    d. Um algoritmo em que dado dois números *n* e *k* (n< k), calcule e apresente a combinatória de n elementos tomados k a k

Exemplo de Desenvolvimento: Derivar os casos de teste para um programa que calcula a média das entradas válidas, usando o método do caminho básico.

```
Procedimento media
INTERFACE ACEITA valor, min, max
INTERFACE RETORNA media, entradas, validas

var
   valor[1..100] vetor de real
   media, entradas, validas, min, max, soma: real
   i : inteiro
inicio
   i = 1
   totalEntradas = 0
   totalValidas = 0
   soma = 0
   enquanto valor[i]<>-999 e entradas<100 faça
      entradas=entradas+1
      se valor[i]>=min e valor[i]<=max então
         validas=validas+1
         soma=soma+valor[i]
      senão pule
      fimse
      i=i+1
   fimenquanto
   se validas>0 então
      media=soma/validas
   senão
      media=-999
   fimse
fim
```

Passo 1: Desenhe o grafo de fluxo correspondente



Passo 2: Calcule a complexidade ciclomática. V(G) = 6 regiões V(G) = 17 arestas – 13 nós + 2 = 6 V(G) = 5 nós predicados + 1 = 6

Passo 3: Determine um conjunto base de caminhos independentes.

Caminho 1: 1-2-10-11-13
Caminho 2: 1-2-10-12-13
Caminho 3: 1-2-3-10-11-13
Caminho 4: 1-2-3-4-5-8-9-2...
Caminho 5: 1-2-3-4-5-6-8-9-2...
Caminho 6: 1-2-3-4-5-6-7-8-9-2...

Passo 4: Prepare os casos de teste que vão forçar a execução de cada caminho:

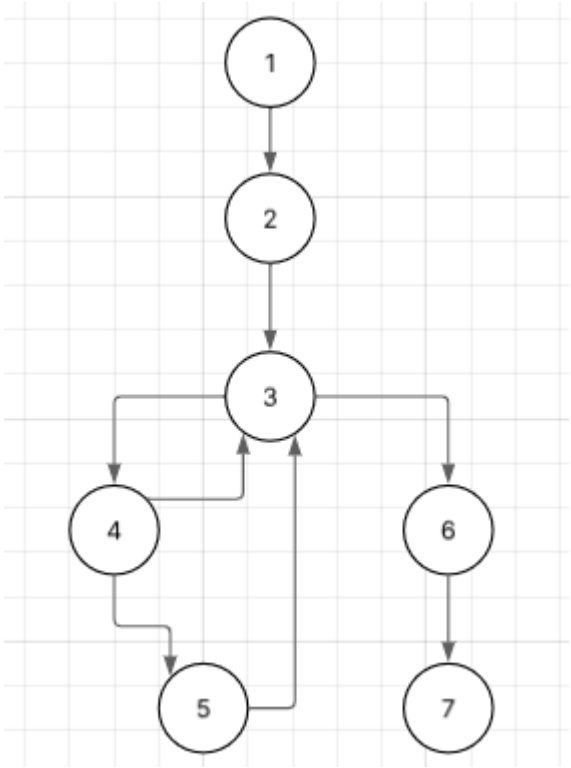
O caminho 1 só pode ser testado como parte dos caminhos 4, 5 e 6

Caminho 2: valor (i) = -999; resultados esperados: média = -999 e os outros valores com os valores iniciais.

Caminho 6: valor (i) = entrada válida; resultados esperados: média correta baseada em n valores e totais apropriados.

Algoritmo A

Passo 1: Grafo de fluxo



Passo 2: Complexidade ciclomática

V(G) = E - N + 2
V(G) = 10 - 9 + 2
V(G) = 3

Passo 3: Caminhos Independentes

Caminho 1: 1-2-3-6-7
Caminho 2: 1-2-3-4-3-6-3-6-7
Caminho 3: 1-2-3-4-5-3-6-7

Passo 4: Casos de Teste

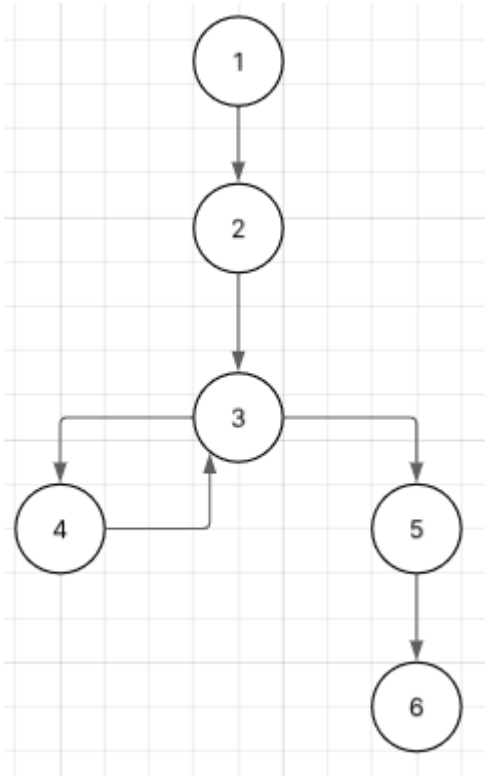
Caminho 1:
Entrada: num = 0
Resultado: lista vazia []

Caminho 2:
Entrada: num = 7
Resultado [1, 7] (números primos)

Caminho 3:
Entrada: num = 6
Resultado: [1, 2, 3, 6] (número com vários divisores)

Algoritmo B

Passo 1: Desenho do grafo de fluxo



Passo 2: Complexidade ciclomática

V(G) = E - N + 2
V(G) = 6 - 6 + 2
V(G) = 2

Passo 3: Caminhos Independentes

Caminho 1: 1-2-3-5-6
Caminho 2: 1-2-3-4-3-5-6

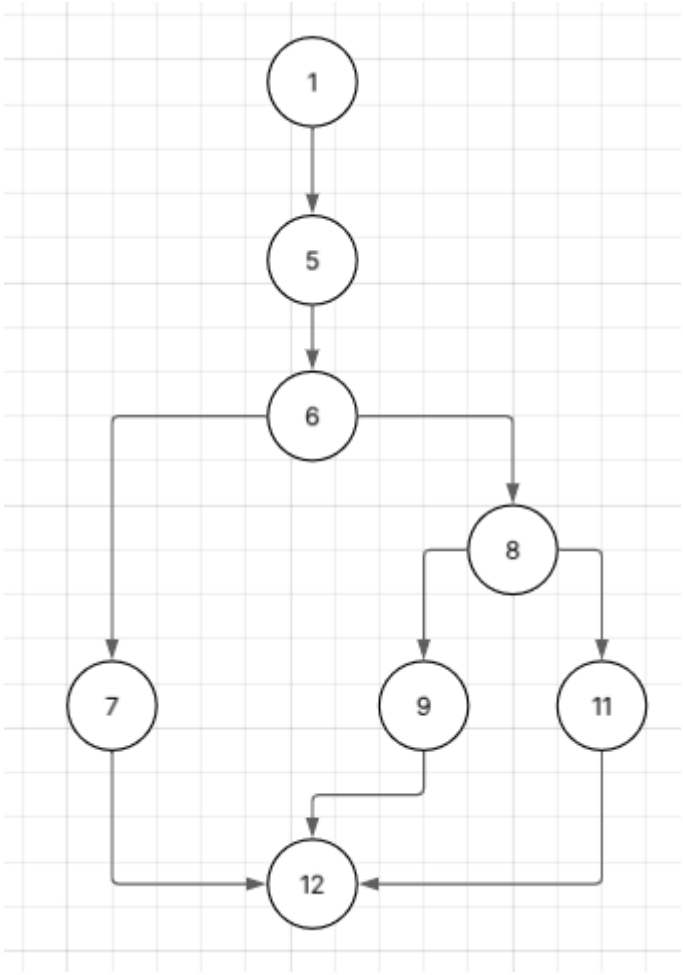
Passo 4: Casos de Teste

Caminho 1:
Entrada: a = 5, b = 0
Resultado: "divisor comum é: 5"

Caminho 2:
Entrada: a = 48, b = 18
Resultado: "divisor comum é: 6"

Algoritmo C

Passo 1: Desenho do grafo de fluxo



Passo 2: Complexidade ciclomática

V(G) = E - N + 2
V(G) = 10 - 9 + 2
V(G) = 3

Passo 3: Caminhos Independentes

Caminho 1: 1-2-3-4-5-6-7-12
Caminho 2: 1-2-3-4-5-6-8-9-12
Caminho 3: 1-2-3-4-5-6-8-10-11-12

Passo 4: Casos de Teste

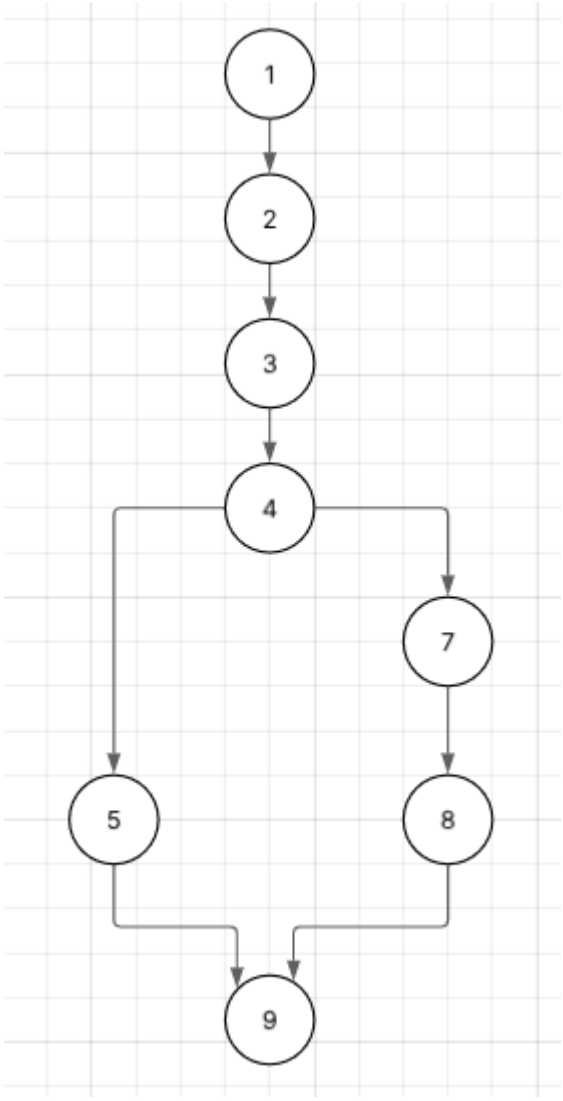
Caminho 1:
Entrada: a=8.0, b=7.5, c=7.0, d=8.5
Resultado: "Passou por média" (média = 7.75)

Caminho 2:
Entrada: a=6.0, b=5.5, c=6.5, d=5.0;
Resultado: "Em final" (média = 5.75)

Caminho 3:
Entrada: a=3.0, b=4.0, c=2.5, d=4.5;
Resultado: "Reprovado" (média = 3.5)

Algoritmo D

Passo 1: Desenho do grafo de fluxo



Passo 2: Complexidade ciclomática

V(G) = E - N + 2
V(G) = 7 - 7 + 2
V(G) = 2

Passo 3: Caminhos Independentes

Caminho 1: 1-2-3-4-5-9
Caminho 2: 1-2-3-4-6-7-8-9

Passo 4: Casos de Teste

Caminho 1:
Entrada: n = -1, k = 5;
Resultado: "Vai dar não, moio pra tu"

Caminho 1 (alternativo):
Entrada: n = 6, k = 4
Resultado: "Vai dar não, moio pra tu"

Caminho 2:
Entrada: n = 2, k = 5;
Resultado: "C(5, 2) = 10"

PLANOS DE TESTE A SER DESCRITO:

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação		ABORDAGEM:
1	Algoritmo de Divisores	Lê um número e imprime a lista dos seus divisores		análise de fluxo de controle para números positivos, negativos e zero. análise dos laços de repetição e condições de parada. cobertura de todas as condições de aprovação/reprovação. validação de entrada e cálculo fatorial.
2	Algoritmo MDC (Euclides)	Lê dois números e calcula o máximo divisor comum pelo método de Euclides		
3	Média de 4 notas	Lê 4 notas de um aluno e determina se passou por média, está em final ou reprovou		
4	Algoritmo de Combinatória	Dados dois números n e k (n<k), calcula a combinatória de n elementos tomados k a k		

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	Análise de código e criação de grafos de fluxo	11/09/2025	11/09/2025	**3h**		Andrey	Algoritmos: a, b, c, d
02	Cálculo da complexidade ciclomática	11/09/2025	11/09/2025	**1h**	01	Andrey	Para todos algoritmos
03	Definição de caminhos independentes	11/09/2025	11/09/2025	**2h**	02	Andrey	Casos de teste
04	Execução e validação dos testes	11/09/2025	11/09/2025	**3h**	03	Andrey	Relatório final

AMBIENTE DE TESTE

Ambiente	**Descrição**
Hardware	PC Intel i5, 8GB RAM, SSD 256GB
Software	Windows 10, Python 3.12
Ferramental	VS Code para execução, sem frameworks de teste extras, Lucidchart para grafos de fluxo

IDENTIFICAÇÃO DE CASO DE TESTE / IDENTIFICAÇÃO DE PROCEDIMENTO DE TESTE

N°	**Caso de Teste**	**Identificação do Caso de Teste**		**Procedimento**	**Identificação do Procedimento de Teste**
1	Algoritmo de Divisores	CT01_DIVISORES – CT03_DIVISORES		Testar entradas válidas, limites e zero	PT01_DIVISORES
2	Algoritmo MDC (Euclides)	CT04_MDC – CT05_MDC		Testar entradas múltiplos, zero	PT02_MDC
3	Média de 4 notas	CT06_MEDIA – CT08_MEDIA		Testar cenários de aprovação, final e reprovação	PT03_MEDIA
4	Algoritmo de Combinatória	CT09_COMBINATORIA – CT11_COMBINATORIA		Testar entradas válidas e inválidas	PT04_COMBINATORIA

CASO DE TESTE

Identificação	CT01_DIVISORES	
Itens a Testar	Algoritmo de divisores - entrada zero	
Entradas	**Campo**	**Valor**
	num	0
Saídas Esperadas	**Campo**	**Valor**
		Divisores de 0: []
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT01_DIVISORES	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT02_DIVISORES	
Itens a Testar	Algoritmo de divisores - número primo	
Entradas	**Campo**	**Valor**
	num	7
Saídas Esperadas	**Campo**	**Valor**
		Divisores de 7: [1, 7]
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT01_DIVISORES	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT03_DIVISORES	
Itens a Testar	Algoritmo de divisores - número com vários divisores	
Entradas	**Campo**	**Valor**
	num	6
Saídas Esperadas	**Campo**	**Valor**
		Divisores de 6: [1, 2, 3, 6]
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT01_DIVISORES	
Dependência	Nenhuma	

PROCEDIMENTO DE TESTE

Identificação	PT01_DIVISORES
Objetivo	Validar cobertura de caminhos do algoritmo de divisores
Requisitos	Algoritmo implementado, ambiente de teste configurado
Fluxo	1. Executar CT01 com número zero (0) 2. Verificar lista vazia 3. Executar CT02 com número primo (7) 4. Verificar divisores [1,7] 5. Executar CT03 com número composto (6) 6. Verificar divisores [1,2,3,6] 7. Documentar resultados

CASO DE TESTE

Identificação	CT04_MDC	
Itens a Testar	MDC - um número zero	
Entradas	**Campo**	**Valor**
	a	5
	B	0
Saídas Esperadas	**Campo**	**Valor**
		divisor comum é: 5
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT02_MDC	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT05_MDC	
Itens a Testar	MDC - números múltiplos	
Entradas	**Campo**	**Valor**
	A	48
	B	18
Saídas Esperadas	**Campo**	**Valor**
		divisor comum é: 6
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT02_MDC	
Dependência	Nenhuma	

PROCEDIMENTO DE TESTE

Identificação	PT02_MDC
Objetivo	Validar algoritmo de Euclides cobrindo todas as condições estruturais
Requisitos	Algoritmo MDC implementado, ambiente de teste configurado
Fluxo	1. Executar CT04 com um zero (5,0) 2. Verificar retorno 5 3. Executar CT05 com múltiplos (48,18) 4. Verificar cálculo correto através das iterações (divisor 6) 5. Documentar resultados

CASO DE TESTE

Identificação	CT06_MEDIA	
Itens a Testar	Média - aluno aprovado	
Entradas	**Campo**	**Valor**
	a	8.0
	b	7.5
	c	7.0
	d	8.5
Saídas Esperadas	**Campo**	**Valor**
		Passou por média
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT03_MEDIA	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT07_MEDIA	
Itens a Testar	Média - aluno em final	
Entradas	**Campo**	**Valor**
	a	6.0
	b	5.5
	c	6.5
	d	5.0
Saídas Esperadas	**Campo**	**Valor**
		Em final
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT03_MEDIA	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT08_MEDIA	
Itens a Testar	Média - aluno reprovado	
Entradas	**Campo**	**Valor**
	a	3.0
	b	4.0
	c	2.5
	d	4.5
Saídas Esperadas	**Campo**	**Valor**
		Reprovado
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT03_MEDIA	
Dependência	Nenhuma	

PROCEDIMENTO DE TESTE

Identificação	PT03_MEDIA
Objetivo	Cobrir todas as condições de aprovação do sistema de notas
Requisitos	Sistema de média implementado, critérios definidos (>=7 aprovado, >=5 final, <5 reprovado)
Fluxo	1. Executar CT06 com notas para aprovação (média 7.75) 2. Verificar status "Passou por média" 3. Executar CT07 com notas para final (média 5.75) 4. Verificar status "Em final" 5. Executar CT08 com notas para reprovação (média 3.5) 6. Verificar status "Reprovado" 7. Documentar resultados

CASO DE TESTE

Identificação	CT09_COMBINATORIA	
Itens a Testar	Combinatória - entrada inválida (negativo)	
Entradas	**Campo**	**Valor**
	n	-1
	k	5
Saídas Esperadas	**Campo**	**Valor**
		Vai dar não, moio pra tu
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT04_COMBINATORIA	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT10_COMBINATORIA	
Itens a Testar	Combinatória - entrada inválida (n > k)	
Entradas	**Campo**	**Valor**
	n	6
	k	4
Saídas Esperadas	**Campo**	**Valor**
		Vai dar não, moio pra tu
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT04_COMBINATORIA	
Dependência	Nenhuma	

CASO DE TESTE

Identificação	CT11_COMBINATORIA	
Itens a Testar	Combinatória - entrada válida	
Entradas	**Campo**	**Valor**
	n	2
	k	5
Saídas Esperadas	**Campo**	**Valor**
		C(5, 2) = 10
Ambiente	Windows 10, Python 3.12, VS Code	
Procedimento	PT04_COMBINATORIA	
Dependência	Nenhuma	

PROCEDIMENTO DE TESTE

Identificação	PT04_COMBINATORIA
Objetivo	Validar cálculo combinatorial e condição n <= k
Requisitos	Algoritmo de combinatória implementado, validação n <= k, n>=0, k>=0
Fluxo	1. Executar CT09 com negativo (n=-1, k=5) 2. Verificar erro 3. Executar CT10 com n > k (n=6, k=4) 4. Verificar erro 5. Executar CT11 com válida (n=2, k=5) 6. Verificar C(5,2)=10 7. Documentar resultados