Disciplina	Prof. João Choma
PROJETO IMPLEMENTAÇÃO E TESTE DE SOFTWARE	Valor
ATIVIDADE : TESTE ESTRUTURAL	Aluno: Heloísa Scarante
ESOFT - 6 - A	Aluno: Carlos Favarão

Atividade prática de teste Estrutural Passos:

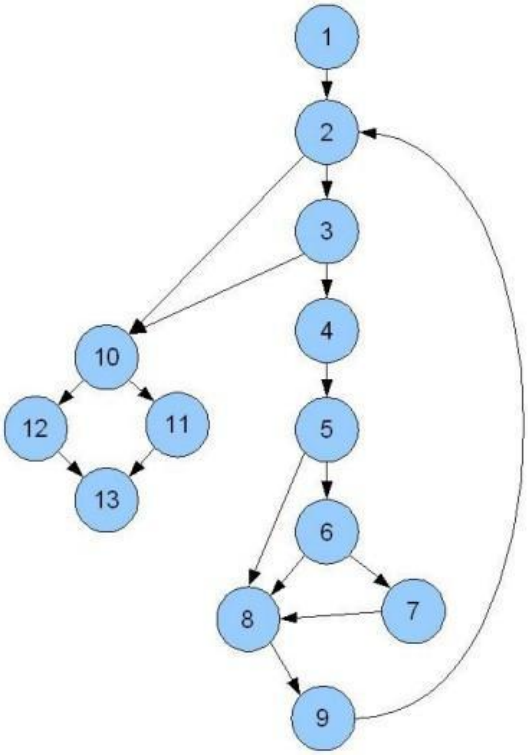
1. Projetar **casos de teste Estruturais** para avaliar os quatro algoritmos dos itens listados abaixo. Conforme o exemplo abaixo, e o excerto do Livro Didático.
2. Preencher os ARTEFATOS de teste abaixo para os testes projetados.
3. Construa, em sua linguagem de preferência os seguintes algoritmos:
    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
1  totalEntradas = 0
   totalValidas = 0
   soma = 0
   2                3
   enquanto valor[i]<>-999 e entradas<100 faça
   4  entradas=entradas+1
      5                6
      se valor[i]>=min e valor[i]<=max então
         validas=validas+1
   7     soma=soma+valor[i]
      senão pule
      fimse
   8  i=i+1
   9 fimenquanto
   se validas>0 então  10
   11    media=soma/validas
   12 senão
         media=-999
   13 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...

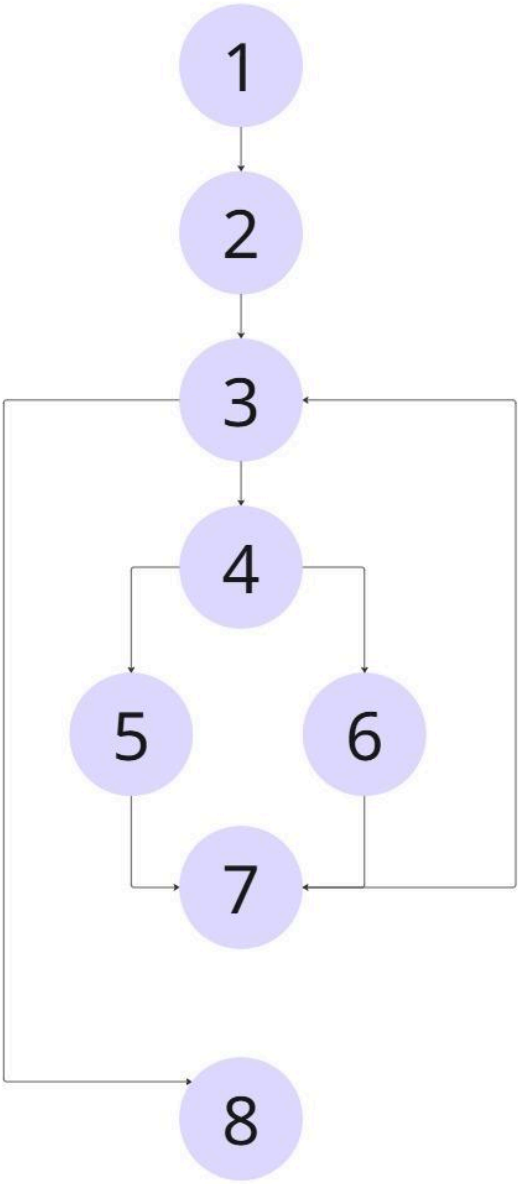
Passo 4: Prepare os casos de teste que vão forçar a execução de cada caminho: O caminho 1 só pode ser testado como parte dos caminhos 4, 5 e 6

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.

1. Um algoritmo que lê um número e imprime a lista dos seus divisores

```
void listar_divisores(int n) {
    printf("Divisores de %d: ", n);
    for (int i = 1; i <= n; i++) {
        if (n % i == 0) {
            printf("%d ", i);
        }
    }
    printf("\n");
}
```



V(G) = E - N + 2

E = 9

N = 8

V(G) = 9 - 8 + 2

V(G) = 3

Caminho 1 - 1,2,3,4,5,7,3,8

Caminho 2 - 1,2,3,4,6,7,3,8

Caminho 3 - 1,2,3,8

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação		ABORDAGEM:
1	Estrutura de repetição	Testar se o for percorre corretamente até n		**Teste estrutural (caixa branca), baseado em caminhos independentes.**
2	Condição de divisor	Testar se if (n % i == 0) retorna divisores		
3	Saída formatada	Testar se imprime todos os divisores corretamente		

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	Testa número positivo	16/09/2025	17/09/2025	4	sim	Helo	n=8
02	Testar número primo	16/09/2025	17/09/2025	2	sim	Helo	n=7
03	testar n=0	16/09/2025	17/09/2025	6	sim	Helo	saída vazia
04	testar n negativo	16/09/2025	17/09/2025	2	sim	Helo	nenhum divisor

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC Windows/Linux com compilador GCC
Software	Compilador C (GCC)
Ferramental	IDE Code::Blocks / VSCode + Extensões C/C++

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	CT01	entrada positiva		Executar e verificar saída "1 2 3 6"	PT01
2	CT02	entrada numero primo		Executar e verificar saída "1 7"	PT02
3	CT03	entrada vazia		Executar e verificar saída apenas quebra	PT03
4	CT04	entrada negativa		Executar e verificar saída apenas quebra	PT04

CASO DE TESTE

Identificação	Entrada positiva	
Itens a Testar	listagem correta de divisores; formato da saída; execução do for	
Entradas	**Campo**	**Valor**
	n	6
Saídas Esperadas	**Campo**	**Valor**
	terminal	1 2 3 6
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar programa que chama listar_divisores(6) e executar; verificar saída.	
Dependência	código-fonte com listar_divisores, compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT01
Objetivo	Verificar que todos os divisores de 6 são impressos corretamente.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Criar teste.c contendo listar_divisores e main(){ listar_divisores(6); }. 2. gcc teste.c -o teste 3. ./teste 4. Conferir se a saída é Divisores de 6: 1 2 3 6. 5. Registrar resultado (OK / NOK) e observações.

CASO DE TESTE

Identificação	Entrada numero primo	
Itens a Testar	condição `if` verdade para divisores 1 e n; formato da saída.	
Entradas	**Campo**	**Valor**
	n	7
Saídas Esperadas	**Campo**	**Valor**
	terminal	1 e 7
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar programa que chama listar_divisores(7) e executar; verificar saída.	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT02
Objetivo	Confirmar que para número primo apenas `1` e `n` são impressos.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Inserir listar_divisores(7) no main. 2. Compilar e executar. 3. Verificar saída Divisores de 7: 1 7. 4. Registrar resultado.

CASO DE TESTE

Identificação	Entrada vazia	
Itens a Testar	comportamento para n = 0 (laço não executa); formato da mensagem quando não há divisores.	
Entradas	**Campo**	**Valor**
	n	0
Saídas Esperadas	**Campo**	**Valor**
	terminal	nada
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar/executar com listar_divisores(0); verificar que não aparecem números após os dois-pontos.	
Dependência	código-fonte com listar_divisores, compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT03
Objetivo	Validar que para n=0 o laço não imprime divisores (comportamento atual do código).
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. main chama listar_divisores(0). 2. Compilar/Executar. 3. Confirmar saída Divisores de 0: (sem números). 4. Registrar resultado e observações (se for esperado comportamento ou bug).

CASO DE TESTE

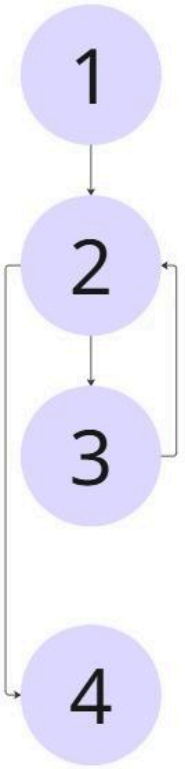
Identificação	Entrada numero negativo	
Itens a Testar	comportamento para n negativo (laço não executa); mensagem exibida.	
Entradas	**Campo**	**Valor**
	n	-5
Saídas Esperadas	**Campo**	**Valor**
	terminal	nada
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar programa que chama listar_divisores(-5) e executar; verificar saída.	
Dependência	código-fonte com listar_divisores, compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT04
Objetivo	Verificar comportamento atual para n negativo (esperado: loop não executa).
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. main chama listar_divisores(-5). 2. Compilar/Executar. 3. Confirmar saída Divisores de -5: sem números. 4. Registrar resultado e comentar se isso é aceitável / precisa ajuste.

2. Um algoritmo que lê dois números e calcula o máximo divisor comum pelo método de Euclides.

```
int mdc(int a, int b) {
    while (b != 0) {
        int temp = b;
        b = a % b;
        a = temp;
    }
    return a;
}
```



V(G) = E - N + 2

E = 4

N = 4

V(G) = 4 - 4 + 2

V(G) = 2

Caminho 1 - 1,2,3,2,4

Caminho 2 - 1,2,4

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação		ABORDAGEM:
1	Caso básico	Verificar mdc entre dois números (ex.: 48 e 18)		**Caixa branca (testes baseado em caminhos) + caixa preta (valores representativos).**
2	falta de argumento	Um dos argumentos 0 (a, 0) ou (0, b)		
3	entrada vazia	Ambos zero (0,0) e números primos entre si		

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	testa mdc composto	16/09/2025	17/09/2025	4	sim	Helo	48,18
02	teste 1 zero	16/09/2025	17/09/2025	2	sim	Helo	a=0,b=5
03	teste vazio	16/09/2025	17/09/2025	6	sim	Helo	0,0

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC Windows/Linux com compilador GCC
Software	Compilador C (GCC)
Ferramental	IDE Code::Blocks / VSCode + Extensões C/C++

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	CT01	teste mdc composto		Executar e verificar saída "6"	PT01
2	CT02	teste 1 zero		Executar e verificar saída "a"	PT02
3	CT03	entrada vazia		Executar e verificar saída apenas quebra	PT03

CASO DE TESTE

Identificação	teste mdc composto	
Itens a Testar	cálculo correto do máximo divisor comum; fluxo do laço.	
Entradas	**Campo**	**Valor**
	n	48
	n	18
Saídas Esperadas	**Campo**	**Valor**
	valor	6
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar e executar rotina que chama mdc(48,18); verificar retorno 6.	
Dependência	implementação da função	

PROCEDIMENTO DE TESTE

Identificação	PT01
Objetivo	Verificar quemdc (48,18) retorna 6
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Implementar main com printf("%d", mdc(48,18)); 2. gcc teste.c -o teste 3. ./teste 4. Verificar saída 6. Registrar OK/NOK.

CASO DE TESTE

Identificação	teste falta de argumento	
Itens a Testar	laço não executado; função retorna a.	
Entradas	**Campo**	**Valor**
	a	5
	b	0
Saídas Esperadas	**Campo**	**Valor**
	valor	5
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar/executar mdc(5,0).	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT02
Objetivo	Validar que quando b==0 inicialmente, retorna a sem entrar no laço.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	Implementar main com printf("%d", mdc(0,5)); gcc teste.c -o teste ./teste Verificar saída 5. Registrar OK/NOK.

CASO DE TESTE

Identificação	Entrada vazia	
Itens a Testar	comportamento para ambos zero (caso limite).	
Entradas	**Campo**	**Valor**
	a	0
	b	0
Saídas Esperadas	**Campo**	**Valor**
	valor	0
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar/executar mdc(0,0).	
Dependência	código-fonte , compilador (gcc).	

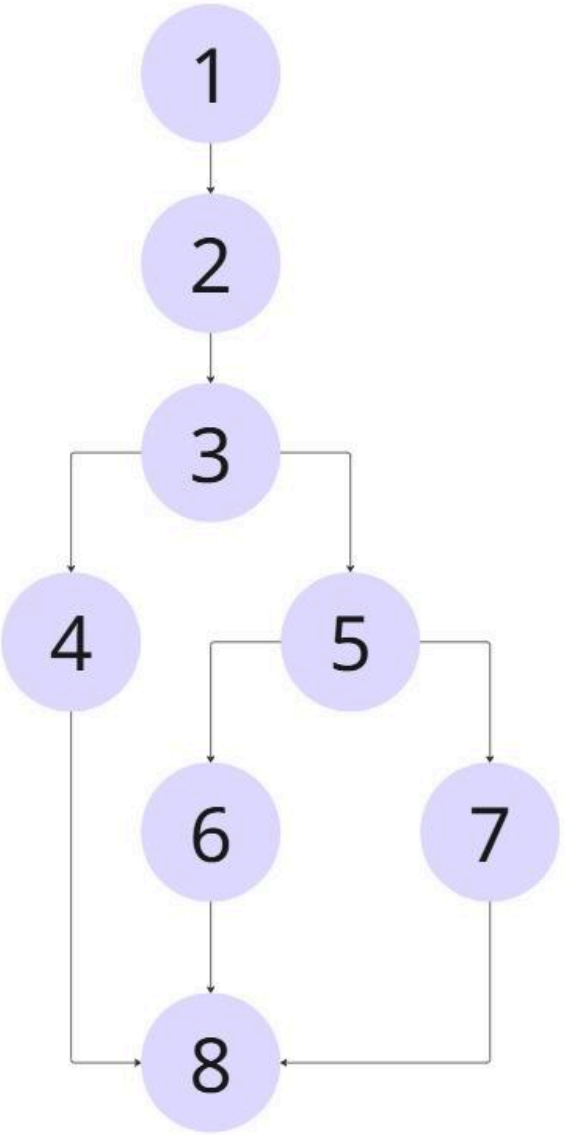
PROCEDIMENTO DE TESTE

Identificação	PT03
Objetivo	Registrar comportamento do sistema para entrada (0,0).
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	Implementar main com printf("%d", mdc(0,5)); gcc teste.c -o teste ./teste Verificar saída 5. Registrar OK/NOK.

3. Um algoritmo que lê as 4 notas de um aluno e diga se ele passou por média, está em final ou reprovou

```
void situacao_aluno(float n1,
float n2, float n3, float n4) {
   float media = (n1 + n2 + n3 +
n4) / 4.0;

   printf("Média = %.2f -> ",
media);
   if (media >= 7.0) {
      printf("Aprovado\n");
   } else if (media >= 4.0) {
      printf("Final\n");
   } else {
      printf("Reprovado\n");
   }
}
```



V(G) = E - N + 2

E = 9

N = 8

V(G) = 9 - 8 + 2

V(G) = 3

Caminho 1 - 1,2,3,4,8

Caminho 2 - 1,2,3,5,6,8

Caminho 3 - 1,2,3,5,7,8

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

Nº	Item	Especificação		ABORDAGEM:
1	Aprovado	Verificar cálculo da média e classificação “Aprovado” (media ≥ 7.0)		**Caixa branca (testes baseado em caminhos) + caixa preta (valores representativos).**
2	entradas negativas	tratar entradas fora do intervalo (ex: negativas, >10)		
3	reprovado	Verificar média < 4.0 → Reprovado		

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	testa aluno aprovado	16/09/2025	17/09/2025	4	sim	Helo	8,9
02	testa entradas negativas	16/09/2025	17/09/2025	2	sim	Helo	-10, 0
03	testa alunos reprovados	16/09/2025	17/09/2025	6	sim	Helo	4,3

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC Windows/Linux com compilador GCC
Software	Compilador C (GCC)
Ferramental	IDE Code::Blocks / VSCode + Extensões C/C++

IDENTIFICAÇÃO DE CASO DE TESTE / IDENTIFICAÇÃO DE PROCEDIMENTO DE TESTE

Nº	Caso de Teste	Identificação do Caso de Teste		Procedimento	Identificação do Procedimento de Teste
1	CT01	Aprovado (média > 7.0)		Executar e verificar saída “Média = 7.75 -> Aprovado”	PT01
2	CT02	Entradas invalidas		Executar com notas inválidas; registrar saída e necessidade de validação	PT02
3	CT03	Reprovado (média < 4.0)		Executar e verificar saída “Média = 3.00 -> Reprovado”	PT03

CASO DE TESTE

Identificação	aluno aprovado	
Itens a Testar	Cálculo da média; ramo media >= 7.0 → “Aprovado”.	
Entradas	**Campo**	**Valor**
	n1	8
	n2	7,5
	n3	9
	n4	6,5
Saídas Esperadas	**Campo**	**Valor**
	valor	média = 7,75 : aprovado
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar programa com main chamando situacao_aluno(8.0,7.5,9.0,6.5); executar e verificar saída.	
Dependência	implementação da função	

PROCEDIMENTO DE TESTE

Identificação	PT01
Objetivo	Verificar que média > 7 resulta em “Aprovado”.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Implementar main(){ situacao_aluno(8.0,7.5,9.0,6.5); }. 2. gcc teste.c -o teste 3. ./teste 4. Verificar se a saída é Média = 7.75 -> Aprovado. 5. Registrar resultado (OK / NOK) e observações.

CASO DE TESTE

Identificação	entrada invalida	
Itens a Testar	Comportamento quando notas inválidas são fornecidas (ex.: negativas ou maiores que 10). **Observação:** função atual **não valida** intervalo de notas — documentar necessidade de requisito se desejar validação.	
Entradas	**Campo**	**Valor**
	n1	-1
	n2	5
	n3	6
	n4	8
Saídas Esperadas	**Campo**	**Valor**
	valor	invalido
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar/executar com valores inválidos; verificar saída e registrar necessidade de validação.	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT02
Objetivo	Verificar comportamento atual com notas fora do intervalo e registrar se é aceitável.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. main(){ situacao_aluno(-1.0,5.0,6.0,8.0); } 2. Compilar (gcc teste.c -o teste) 3. Executar (./teste) 4. Verificar saída Média = 4.50 -> Final. 5. Registrar resultado e **recomendação**: se for necessário, abrir requisição para adicionar validação de entrada (ex.: validar 0..10 e retornar erro/avisar).

CASO DE TESTE

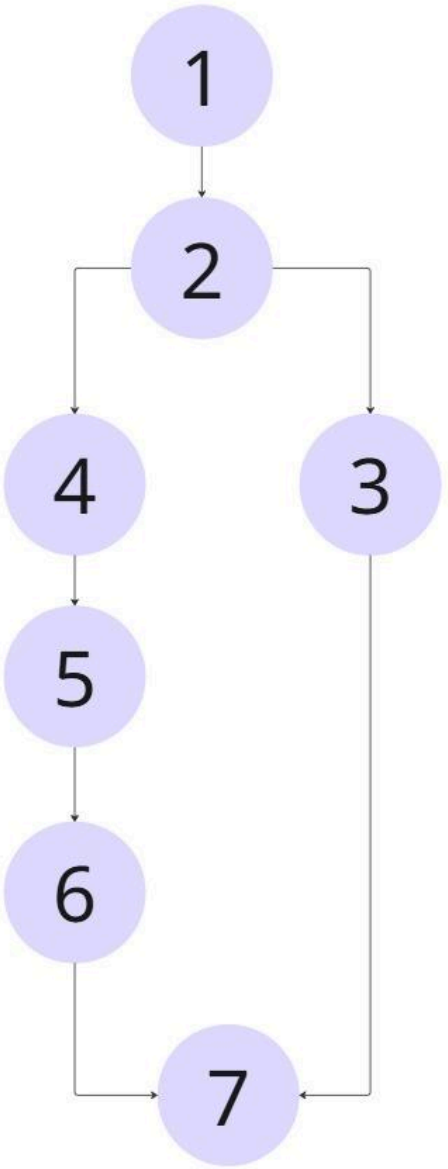
Identificação	Reprovado	
Itens a Testar	else : “reprovado”	
Entradas	**Campo**	**Valor**
	n1	2
	n2	3
	n3	4
	n4	3
Saídas Esperadas	**Campo**	**Valor**
	valor	média 3.00 : reprovado
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	compilar/executar situacao_aluno(2,3,4,3); verificar saída	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT03
Objetivo	Confirmar que média < 4 resulta em “Reprovado”.
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. main(){ situacao_aluno(2,3,4,3); } 2. Compilar e executar. 3. Verificar Média = 3.00 -> Reprovado. 4. Registrar resultado.

4. Um algoritmo em que dado dois números n e k (n>k), calcule e apresente a combinatória de n elementos tomados k a k

```
long long combinatoria(int n,
int k) {
    if (k > n || k < 0) return 0;
    long long numerador =
fatorial(n);
    long long denominador =
fatorial(k) * fatorial(n - k);
    return numerador /
denominador;
}
```



V(G) = E - N + 2

E = 7

N = 7

V(G) = 9 - 8 + 2

V(G) = 2

Caminho 1 - 1,2,3,7

Caminho 2 - 1,2,4,5,6,7

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação		ABORDAGEM:
1	entradas corretas	combinatoria(5,2) deve retornar 10		**Caixa branca (testes baseado em caminhos) + caixa preta (valores representativos).**
2	entradas envalidas	k > n e k < 0 → retorna 0		
3	limites	k = 0 → retorna 1; k = n → 1		

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	Teste básico C(5,2)	16/09/2025	17/09/2025	4	sim	Helo	espera 10
02	Teste limites (k=0,k=n)	16/09/2025	17/09/2025	2	sim	Helo	C(5,0)=1, C(5,5)=1
03	teste maior (C(10,3))	16/09/2025	17/09/2025	6	sim	Helo	espera 120

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC Windows/Linux com compilador GCC
Software	Compilador C (GCC)
Ferramental	IDE Code::Blocks / VSCode + Extensões C/C++

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	CT01	C(5,2) — caso básico		executar e verificar retorno 10	PT01
2	CT02	Limites: k=0 / k=n		Executar C(5,0) e C(5,5) e verificar 1	PT02
3	CT03	entradas invalidas		Executar C(5,6) e C(5,-1); verificar 0	PT03

CASO DE TESTE

Identificação	Combinação com k=0	
Itens a Testar	Cálculo da combinatória com k=0	
Entradas	**Campo**	**Valor**
	n	5
	k	0
Saídas Esperadas	**Campo**	**Valor**
	valor	1
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	Executar combinatoria(5,0) e verificar retorno 1	
Dependência	implementação da função	

PROCEDIMENTO DE TESTE

Identificação	PT01
Objetivo	Verificar se combinatoria(5,0) retorna 1
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Implementar main com printf("%lld", combinatoria(5,0)); 2. Compilar com gcc teste.c -o teste 3. Executar ./teste e verificar saída 1

CASO DE TESTE

Identificação	Combinação com k>n	
Itens a Testar	Condição de retorno 0 quando k	
Entradas	**Campo**	**Valor**
	n	3
	k	5
Saídas Esperadas	**Campo**	**Valor**
	valor	0
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	Executar combinatoria(3,5) e verificar retorno 0	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT02
Objetivo	Verificar se combinatoria(3,5) retorna 0
Requisitos	arquivo teste.c com função e main, gcc, terminal.
Fluxo	1. Implementar main com printf("%lld", combinatoria(3,5)); 2. Compilar com gcc teste.c -o teste 3. Executar ./teste e verificar saída 0

CASO DE TESTE

Identificação	Combinação com valores normais	
Itens a Testar	Cálculo correto da combinatória	
Entradas	**Campo**	**Valor**
	n	5
	k	2
Saídas Esperadas	**Campo**	**Valor**
	valor	10
Ambiente	PC com compilador C (GCC), terminal	
Procedimento	Executar combinatoria(5,2) e verificar retorno 10	
Dependência	código-fonte , compilador (gcc).	

PROCEDIMENTO DE TESTE

Identificação	PT03
Objetivo	Confirmar que média < 4 resulta em “Reprovado”.
Requisitos	Arquivo teste.c com função combinatoria e fatorial implementadas
Fluxo	1. Implementar main com printf("%lld", combinatoria(5,2)); 2. Compilar com gcc teste.c -o teste 3. Executar ./teste e verificar saída 10