Disciplina	Prof. João Choma	
PROJETO IMPLEMENTAÇÃO E TESTE DE SOFTWARE	Valor	+01 ATV
ATIVIDADE : TESTE ESTRUTURAL	**Turma: ESOFT - 6 - N**	
Aluno: Felipe Saueressig Mello	**RA: 23167656-2**	
Aluno: Andrei Luiz Silva	**RA: 23087473-2**	
Aluno: Henrique Henschel Puccetti	**RA: 23094941-2**	

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

Passo 1: Desenhe o grafo de fluxo correspondente

Passo 2: Calcule a complexidade ciclomática: V(G) = E – N + 2

Passo 3: Determine um conjunto base de caminhos independentes.

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

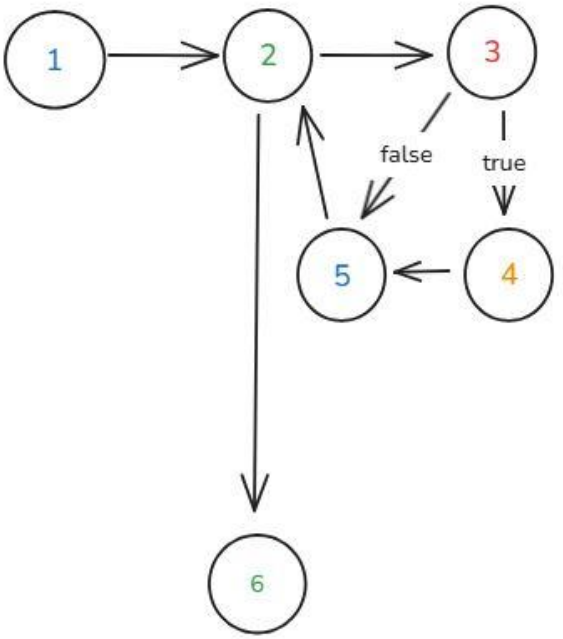
a. divisores.c

a. divisores.c

```c
#include <stdio.h>

int main() {
    int numero, i;

    printf("Digite um numero inteiro: ");
    scanf("%d", &numero);

    printf("Os divisores de %d sao: ", numero);
    for (i = 1; i <= numero; i++) {
        if (numero % i == 0) {
            printf("%d ", i);
        }
    }
    printf("\n");

    return 0;
}
```

V(G) = E - N +2 V(G) = 7 - 6 + 2
V(G) = 3

caminhos:
C1: 1-2-3-4-5-2-6
C2: 1-2-3-4-5-2-3-5-2-6
C3: 1-2-6



PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação	Abordagem:
1	a.C1:	1-2-3-4-5-2-6	
2	a.C2:	1-2-3-4-5-2-3-5-2-6	
3	a.C3:	1-2-6 (não entra no for)	

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01							
02							
03							
04							

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC com no minimo processador i5 10500 e 8 GB de RAM.

Software	Linguagem C, compilador Dev-C++
Ferramental	Tester

N°	Caso de Teste	Identificação do Caso de Teste		Procedimento	Identificação do Procedimento de Teste
1	Execução completa	a.C1		a.P1	a.P1
2	Falha em Nó 3	a.C2		a.P2	a.P2
3	Falha no Nó 2	a.C3		a.P3	a.P3

CASO DE TESTE

Identificação	a.C3	
Itens a Testar	Caminho: 1-2-6	
Entradas	**Campo**	**Valor**
	numero	0
	i	1
Saídas Esperadas	**Campo**	**Valor**
	Os divisores de numero são:	indeterminado
Ambiente		
Procedimento		
Dependência		

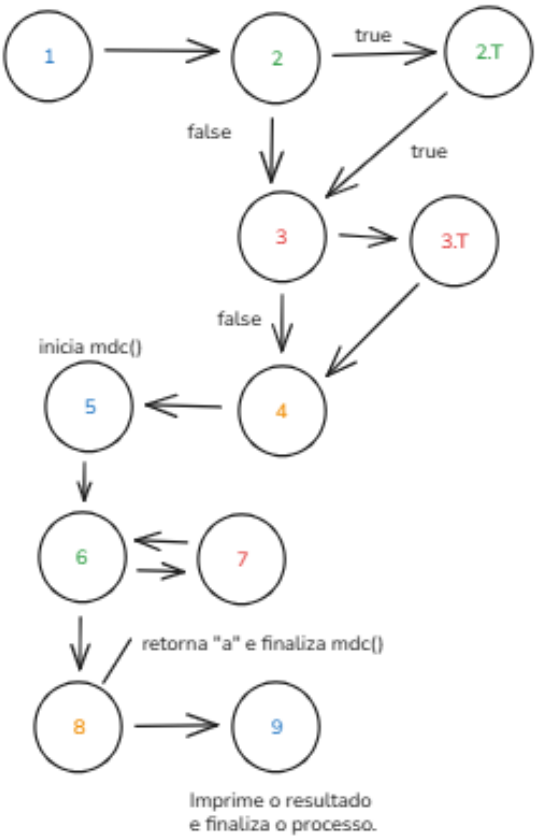
PROCEDIMENTO DE TESTE

Identificação	a.P3
Objetivo	Fazer o programa não excecutar o laço For
Requisitos	Um pc basico
Fluxo	1(numero = 0)→ 2(i = 1, i<=numero)→6(finalizar programa)

b. mdc_Euclides.c

b. mdc_Euclides.c

```c
#include <stdio.h>

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

int main() {
    int num1, num2;

    printf("Digite o primeiro numero: ");
    scanf("%d", &num1);
    printf("Digite o segundo numero: ");
    scanf("%d", &num2);

    // Certifica-se de que os números são positivos para o cálculo
    if (num1 < 0) num1 = -num1;
    if (num2 < 0) num2 = -num2;

    printf("O MDC de %d e %d eh: %d\n", num1, num2, mdc(num1, num2));

    return 0;
}
```



V(G) = E - N +2 V(G) = 13 - 11 +2 V(G) = 4

Caminhos independentes:
C1: 1-2-2.T-3-3.T-4-5-6-7-6-8-9
C2: 1-2-3-4-5-6-7-6-8-9
C3: 1-2-2.T-3-4-5-6-7-6-8-9
C4: 1-2-2.T-3-4-5-6-8-9

obs: O nó6 depende do valor no nó3 onde se o num2 for menor que 0 então nó3 é true e não há como nó6 ser false, se o num2 for maior ou igual a 0 então nó3 é false e nó6 pode ser ou não true. qualquer resultado é independete do nó2, mas há mais possibilidades de caminhos, mas não possuem testes satisfatórios.

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação	Abordagem:
1	b.C1	**1-2-2.T-3-3.T-4-5-6-7-6-8-9**	
2	b.C2	**1-2-3-4-5-6-7-6-8-9**	
3	b.C3	**1-2-2.T-3-4-5-6-7-6-8-9**	
4	b.C4	**1-2-2.T-3-4-5-6-8-9**	

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01							
02							

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC com no minimo processador i5 10500 e 8 GB de RAM.
Software	Linguagem C, compilador Dev-C++
Ferramental	Tester

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	b.C1			b.P1	b.P1
2	b.C2			b.P2	
3	b.C3			b.P3	
4	b.C4			b.P4	

CASO DE TESTE

Identificação		
Itens a Testar		
Entradas	**Campo**	**Valor**
Saídas Esperadas	**Campo**	**Valor**
Ambiente		
Procedimento		
Dependência		

PROCEDIMENTO DE TESTE

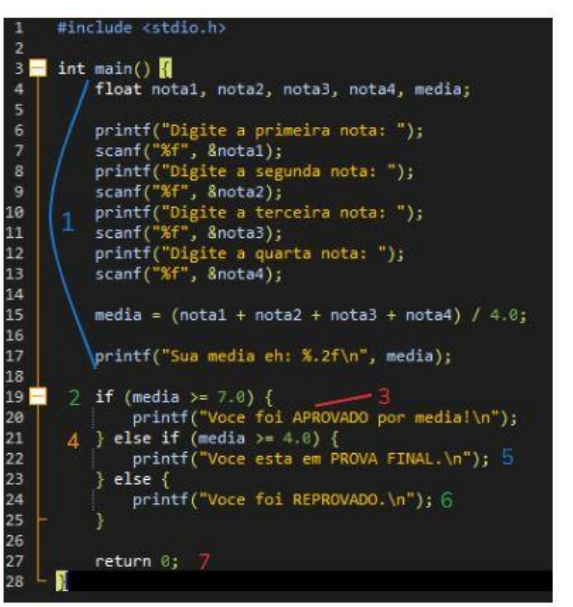
Identificação	
Objetivo	
Requisitos	

Fluxo	

c. situacao_aluno.c

c. situacao_aluno.c

```c
#include <stdio.h>

int main() {
    float nota1, nota2, nota3, nota4, media;

    printf("Digite a primeira nota: ");
    scanf("%f", &nota1);
    printf("Digite a segunda nota: ");
    scanf("%f", &nota2);
    printf("Digite a terceira nota: ");
    scanf("%f", &nota3);
    printf("Digite a quarta nota: ");
    scanf("%f", &nota4);

    media = (nota1 + nota2 + nota3 + nota4) / 4.0;

    printf("Sua media eh: %.2f\n", media);

    if (media >= 7.0) {
        printf("Voce foi APROVADO por media!\n");
    } else if (media >= 4.0) {
        printf("Voce esta em PROVA FINAL.\n");
    } else {
        printf("Voce foi REPROVADO.\n");
    }

    return 0;
}
```



V(G) = E - N +2 V(G) = 8 - 7 + 2 V(G) = 3

Caminhos independetes:
C1: 1-2-3-7
C2: 1-2-4-5-7
C3: 1-2-4-6-7

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação	Abordagem:
1	c.C1:	**1-2-3-7**	
2	c.C2:	**1-2-4-5-7**	
3	c.C3:	**1-2-4-6-7**	

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01							
02							

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC com no minimo processador i5 10500 e 8 GB de RAM.
Software	Linguagem C, compilador Dev-C++
Ferramental	Tester

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					
2					
3					
4					
5					

CASO DE TESTE

Identificação		
Itens a Testar		
Entradas	**Campo**	**Valor**
Saídas Esperadas	**Campo**	**Valor**
Ambiente		
Procedimento		
Dependência		

PROCEDIMENTO DE TESTE

Identificação	
Objetivo	
Requisitos	
Fluxo	

d. combinatoria.c

d. combinatoria.c

```c
#include <stdio.h>

// Função para calcular o fatorial de um número
double fatorial(int num) {
    if (num <= 0) {
        return 0; // Fatorial de números negativos não é definido
    }
    double resultado = 1.0; // Use 1.0 para garantir a operação com ponto flutuante
    int i;
    for (i = 1; i <= num; i++) {
        resultado *= i;
    }
    return resultado;
}

int main() {
    int n, k;
    double resultado;

    printf("Calculo da Combinatoria C(n, k)\n");
    printf("Digite o valor de n (total de elementos): ");
    scanf("%d", &n);
    printf("Digite o valor de k (elementos a serem escolhidos): ");
    scanf("%d", &k);
    if (k > n || k < 0) {
        printf("Erro: k deve ser menor ou igual a n, e k deve ser não-negativo.\n");
        return 1;
    }

    // C(n, k) = n! / (k! * (n-k)!)
    resultado = fatorial(n) / (fatorial(k) * fatorial(n - k));

    printf("A combinatoria de %d elementos tomados %d a %d eh: %.0f\n", n, k, k, resultado);

    return 0;
}
```

V(G) = E - N +2 V(G) = 36 - 26 +2 V(G) = 12

Caminhos independetes:

C1: 1-2-3

C2: 1-2-4-5-6-25-26

C3: 1-2-4-5-7-8-9-8-10-11-12-13-25-26

C4: 1-2-4-5-7-8-9-8-10-11-12-13-25-26

C5: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-20-25-26

C6: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-20-25-26

C7: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-24-25-26

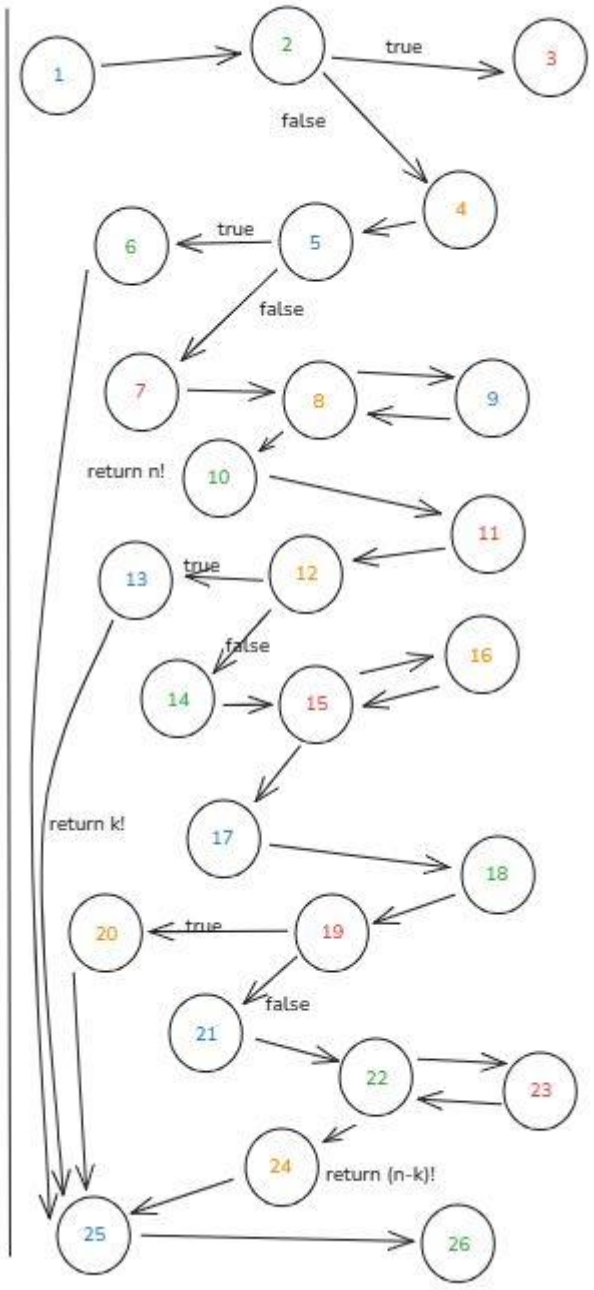
C8: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26

C9: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26

C10: 1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26

C11: 1-2-4-5-6-25-26

C12: 1-2-4-5-7-8-9-8-10-11-12-13-14-15-16-15-17-18-19-20-21-25-26



PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	Item	Especificação	Abordagem:
1	d.C1:	**1-2-3**	
2	d.C2:	**1-2-4-5-6-25-26 (n = 0)**	
3	d.C3:	**1-2-4-5-7-8-9-8-10-11-12-13-25-26 (k = 0)**	
4	d.C4	**1-2-4-5-7-8-9-8-10-11-12-13-25-26 (k<0)**	
5	d.C5	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-20-25-26 ((n-k) = 0)**	
6	d.C6	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-20-25-26 ((n-k) <0)**	
7	d.C7	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26**	
8	d.C8	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26**	
9	d.C9	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26**	
10	d.C10	**1-2-4-5-7-8-9-8-10-11-12-14-15-16-15-17-18-19-21-22-23-22-24-25-26**	
11	d.C11	**1-2-4-5-6-25-26 (n<0)**	
12	d.C12	**1-2-4-5-7-8-9-8-10-11-12-13-14-15-16-15-17-18-19-21-22-24-25-26**	

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01							
02							

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	PC com no minimo processador i5 10500 e 8 GB de RAM.
Software	Linguagem C, compilador Dev-C++
Ferramental	Tester

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					
2					
3					
4					
5					

CASO DE TESTE

Identificação		
Itens a Testar		
Entradas	**Campo**	**Valor**
Saídas Esperadas	**Campo**	**Valor**
Ambiente		
Procedimento		
Dependência		

PROCEDIMENTO DE TESTE

Identificação	
Objetivo	
Requisitos	
Fluxo	