



**Curso Superior de Tecnologia em Análise  
e Desenvolvimento de Sistemas**  
**Disciplina: Qualidade e Teste de Software**

Profª Drª Marcia Cassitas Hino  
2º Semestre/2023

## **Técnicas de Teste**

Teste funcional (caixa preta)  
Teste estrutural (caixa branca)

## **Tipos de Teste**

Teste de funcionalidade  
Teste de usabilidade  
Teste de carga  
Teste de volume  
Teste de configuração  
Teste de compatibilidade  
Teste de segurança  
Teste de desempenho  
Teste de instalação  
Teste de confiabilidade  
Teste de recuperação  
Teste de contingência

## **Níveis de Teste**

Teste de unidade  
Teste de integração  
Teste de sistema  
Teste de aceitação

# TESTE ESTRUTURAL (CAIXA BRANCA)

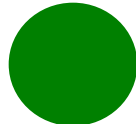

- O teste estrutural baseia-se no conhecimento da estrutura interna do programa, sendo os aspectos de implementação fundamentais para a geração/seleção dos casos de teste a serem executados.
- Em geral, a maioria dos critérios da técnica estrutural utiliza uma representação de programa conhecida como “Grafo de Fluxo de Controle” (GFC) ou “Grafo de Programa”.

- Caminhos lógicos da execução do software são testados.
- Não é viável testar todos os caminhos lógicos de um programa (teste exaustivo).
  - Considere um programa com 100 linhas e dois ciclos aninhados que executam de 1-20 vezes cada.
  - Dentro do ciclo interior existem 4 construções *if-then-else*.
  - Existem  $10^{14}$  caminhos possíveis de execução.
  - Se um mega processador executar 1 caso de teste a cada milissegundo, serão necessários **3170** anos para completar os testes.





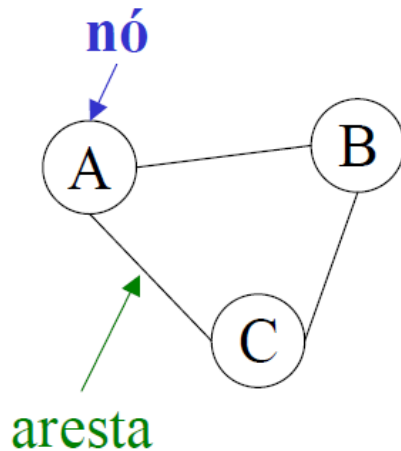
Os casos de teste no teste estrutural devem:

- Garantir que todos os caminhos independentes de um módulo tenham sido exercitados pelo menos uma vez;
  - Exercitar todas as decisões lógicas em seus lados verdadeiro e falso;
  - Executar todos os ciclos nos seus limites e dentro de seus intervalos operacionais;
  - Avaliar as estruturas de dados internas.
- 
- 

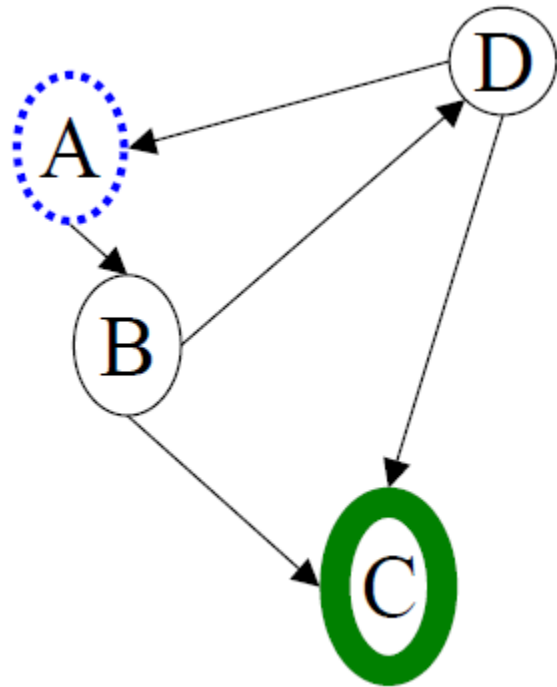
# Grafo de Fluxo de Controle

- O grafo de fluxo mostra o fluxo de controle
- Nós representam um ou mais processos
- Arestas representam o fluxo de controle
- Regiões do grafo são áreas limitadas pelas arestas e nós (incluindo a área fora do grafo)

# Notação

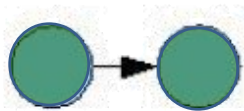


- **Grafo** representa relações entre entidades
- **Nós (ou vértices):** representam as entidades
- **Arestas:** representam as relações entre as entidades

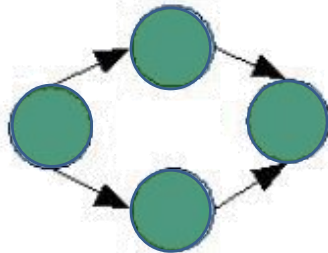


- **Nó de entrada (inicial):** representa um ponto de entrada do programa ou sistema. Pode ter 1 ou mais nós de entrada.
- **Nó de saída (terminal):** um nó que não tem arcos saindo dele. Representa um ponto de saída de um programa ou sistema.

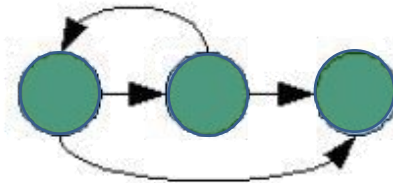




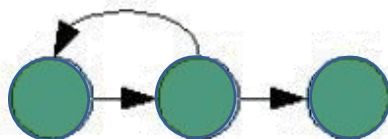
Seqüência



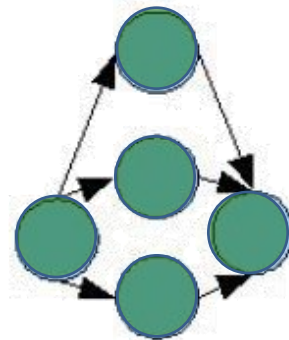
Se-então-senão



Enquanto

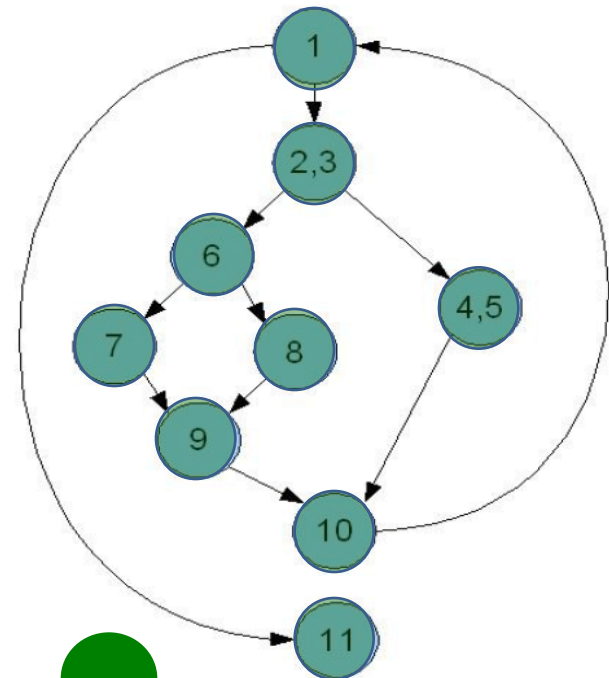
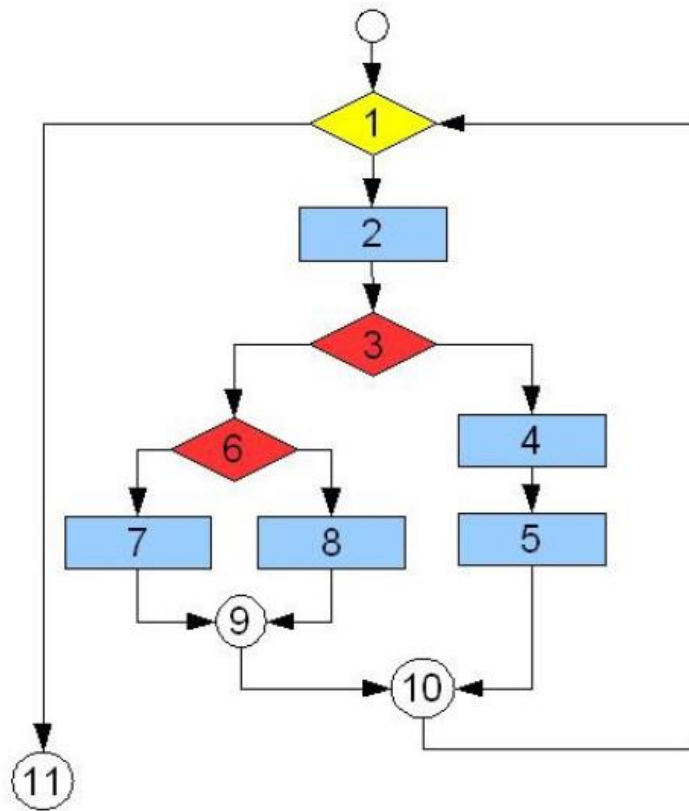


Repita



Caso

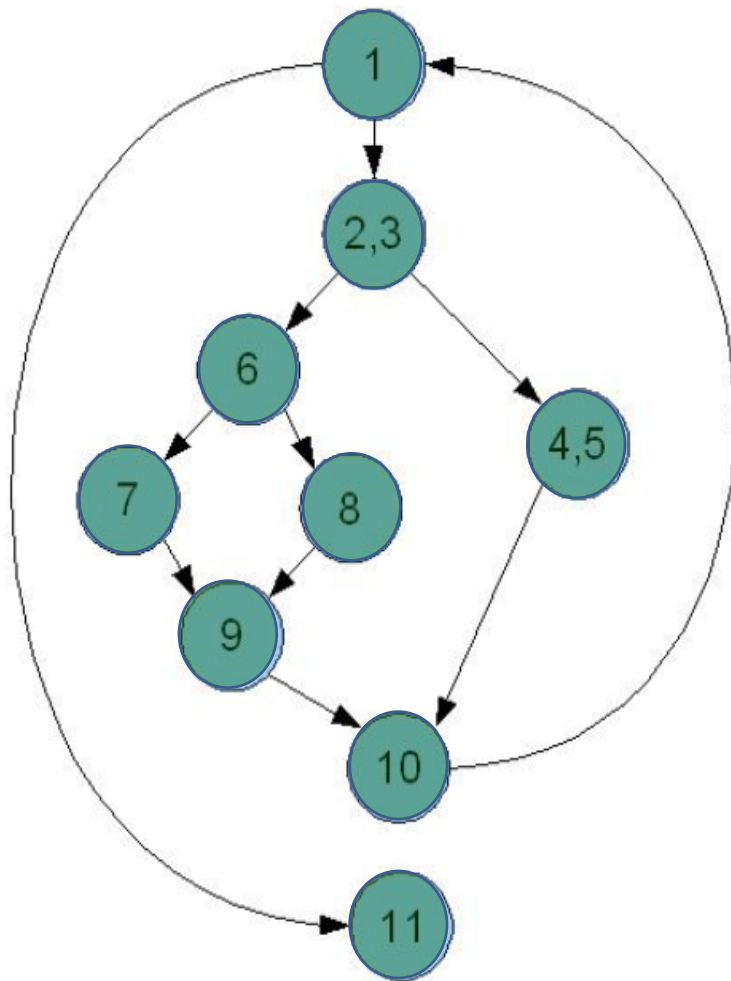
- Derivando o grafo de fluxo a partir de um fluxograma



- Derivando o grafo de fluxo a partir de um pseudocódigo



```
1:  enquanto existir registro faça
2:      leia registro
3:      se registro.campo1 = 0 então
4:          processar registro e armazenar em buffer
5:          incrementar contador
6:      senão se registro.campo2 = 0 então
7:          resetar contador
      senão
8:          processar registro e armazenar em arquivo
9:      fimse
10:     fimse
11:  fimenquanto
```



```
1: enquanto existir registro faça
2:   leia registro
3:   se registro.campo1 = 0 então
4:     processar registro e armazenar em buffer
5:     incrementar contador
6:   senão se registro.campo2 = 0 então
7:     resetar contador
8:   senão
9:     processar registro e armazenar em arquivo
10:  fimse
11: fimse
12: fimenquanto
```

## Critérios

Os critérios estruturais são, em geral, classificados em:

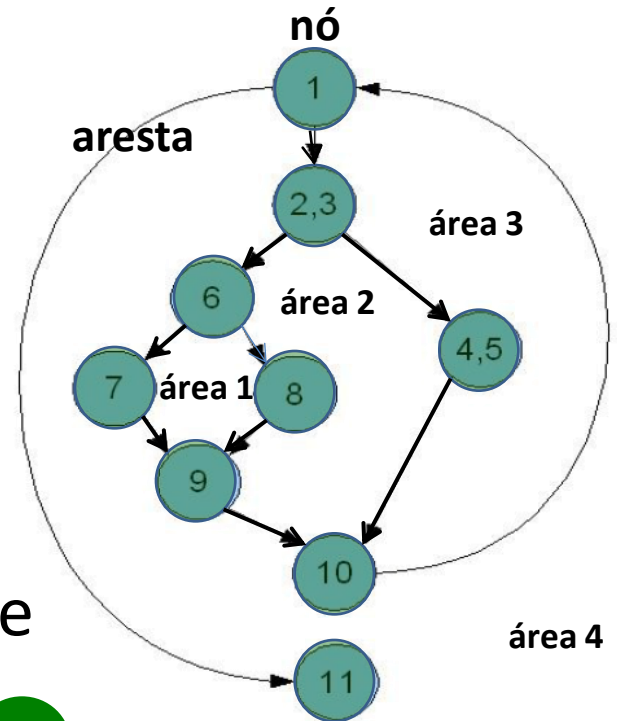
- critérios baseados na complexidade,
- critérios baseados em fluxo de controle e
- critérios baseados em fluxo de dados

## Baseada em Complexidade

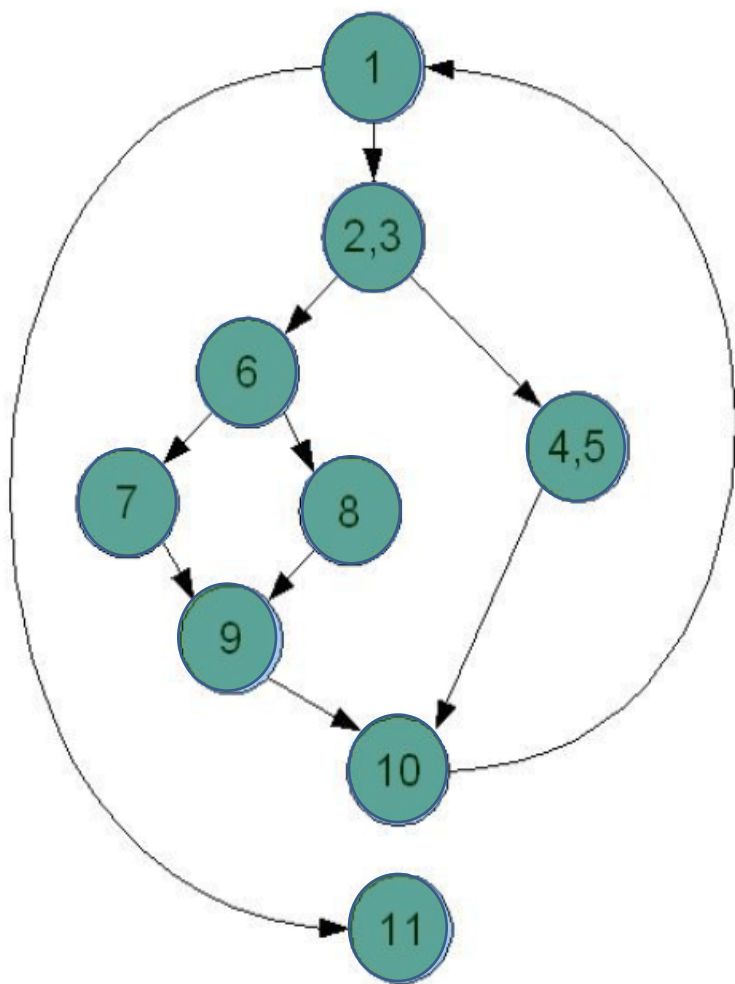
- Complexidade ciclomática é uma métrica que gera uma medida quantitativa da complexidade lógica de um programa.
- Define a quantidade de caminhos independentes de um programa e sugere uma quantidade máxima de testes a serem executados para garantir que todas as instruções foram executadas.

É calculada por 3 formas diferentes:

- A quantidade de áreas do grafo
- $V(G) = E - N + 2$  onde E é a quantidade de ramos e N a quantidade de nós do grafo G
- $V(G) = P + 1$  onde P é a quantidade de nós predicado (são àqueles que podem desviar o fluxo da execução: if, while, switch, etc.) do grafo G



- Caminho independente é o que introduz ao menos um novo conjunto de instruções.



### Caminhos independentes

- 1, 11
- 1, 2, 3, 4, 5, 10, 1, 11
- 1, 2, 3, 6, 7, 9, 10, 1, 11
- 1, 2, 3, 6, 8, 9, 10, 1, 11

### Caminho não independente

- 1, 2, 3, 4, 5, 10, 1, 2, 3, 6, 8, 9, 10, 1, 11



## Baseado em Fluxo de Controle

Utilizam apenas características de controle da execução do programa, como comandos ou desvios, para determinar quais estruturas são necessárias.

Os critérios mais conhecidos são:

- **Todos-Nós:** exige que a execução do programa passe, ao menos uma vez, em cada vértice do grafo;
- **Todas-Arestas:** requer que cada aresta do grafo, isto é, cada desvio de fluxo de controle do programa, seja exercitada pelo menos uma vez;
- **Todos-Caminhos:** requer que todos os caminhos possíveis do programa sejam executados.

É importante ressaltar que a cobertura do critério **Todos-Nós** é o **mínimo esperado** de uma boa atividade de teste, pois, do contrário, o programa testado é entregue sem a certeza de que todos os comandos presentes foram executados ao menos uma vez.

## Baseado em fluxo de dado

- Utilizam a análise de fluxo de dados como fonte de informação para derivar os requisitos de teste.
- Seleciona caminhos de teste de acordo com as definições e dos usos das variáveis do programa (potenciais usos)

## Caso de Teste

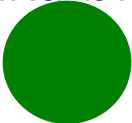

Para usar grafos para projetar casos de teste deve-se seguir os seguintes passos:

- Desenhar o grafo
- Calcular a complexidade ciclomática
- Determinar um conjunto base de caminhos independentes



# Resumindo



- Métodos estruturais se baseiam na estrutura de controle do programa
  - Funções ou métodos mais simples podem ser testados com métodos funcionais (caixa preta) enquanto que funções ou métodos mais complexos podem ser melhor testados com métodos estruturais (caixa branca)
- 
- 

# Vamos praticar?

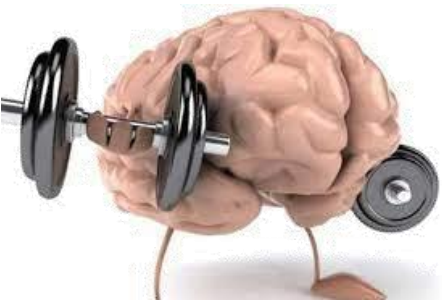


# EXERCÍCIO

## Descrição:

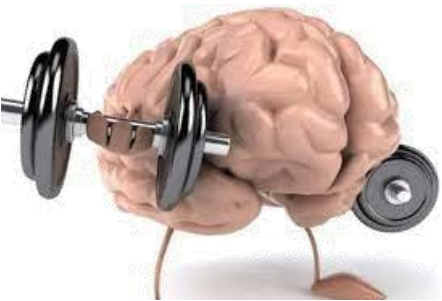
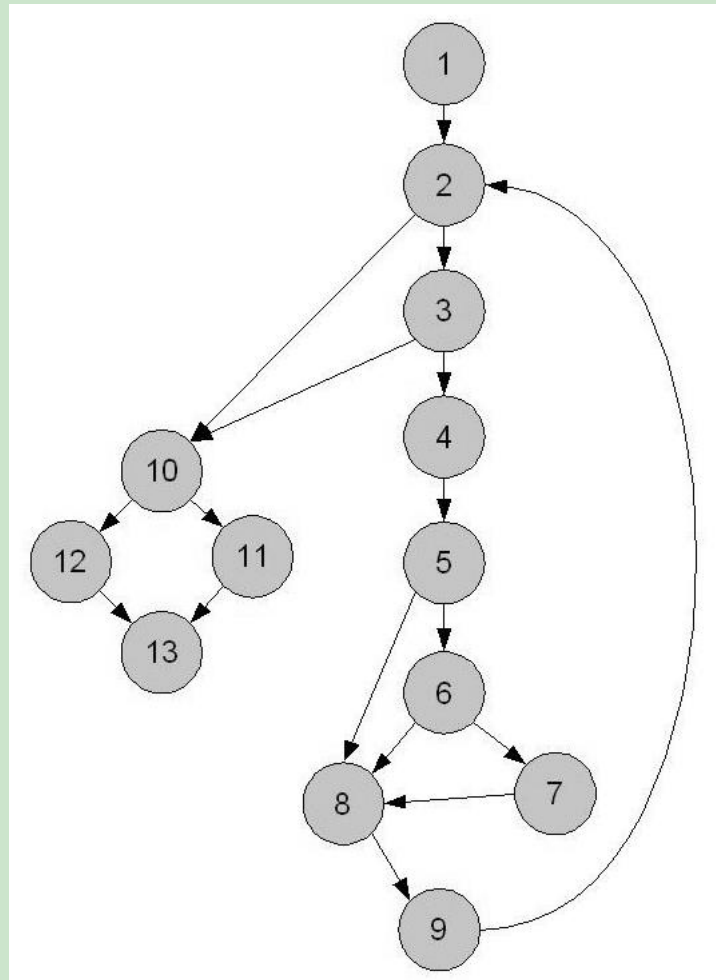
Identificar os casos de teste para um programa ao lado, usando o método do caminho básico.

```
i = 1
totalEntradas = 0
totalValidas = 0
soma = 0
enquanto valor [i] <> -999 e entradas < 100
    entradas = entradas+1
    se valor [i] >= min e valor [i] <= max
        validar = validas + 1
        soma = soma + valor [i]
    senão
        pule
    fim-se
    i = i + 1
fim-enquanto
se validas > 0
    media = soma/validas
senão
    media = -999
fim-se
fim
```



# RESPOSTA EXERCÍCIO

Passo 1:  
Desenho do grafo





# RESPOSTA EXERCÍCIO

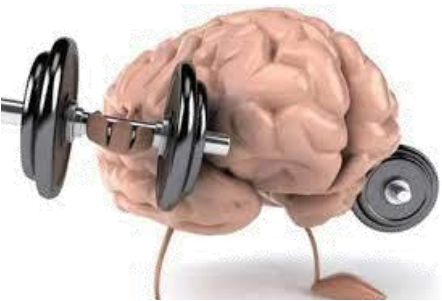
Passo 2:

Calcular a complexidade ciclomática

$$V(G) = 6 \text{ regiões}$$

$$V(G) = 17 \text{ arestas} - 13 \text{ nós} + 2 = 6$$

$$V(G) = 5 \text{ nós predicados} + 1 = 6$$



# RESPOSTA EXERCÍCIO

Passo 3:

Determinar um conjunto base de caminhos independentes

Caminho 1: 1, 2, 10, 11, 13

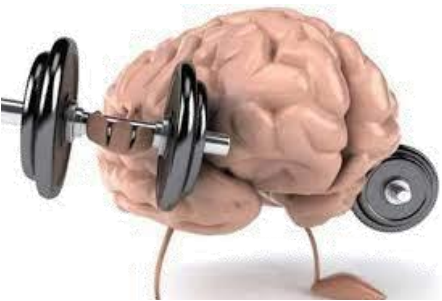
Caminho 2: 1, 2, 10, 12, 13

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...





*Obrigada!*