Simulação e Teste de Software (CC8550)

Aula 07 – Teste de Caixa Branca

Baseado no material desenvolvido pelo prof. Calebe de Paula Bianchini.

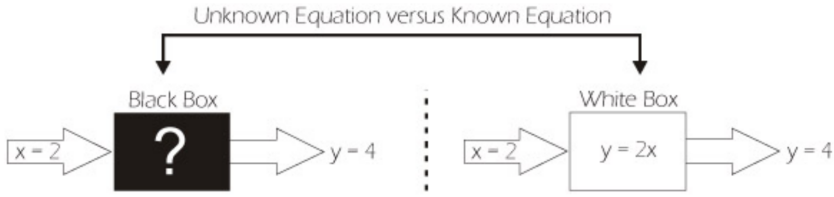
Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

1º Semestre de 2025

Teste de Caixa-Branca

- Verifica não apenas o resultado, mas **como o resultado foi alcançado.**
- Pode ser utilizado também no processo de **depuração de erros.**
- Quando somado ao teste de caixa-preta, pode representar um **aumento de até 60% na qualidade do software.**



Teste Funcional

Teste Estrutural

Características dos Casos de Teste de Caixa-Branca

- Exercitam **todos os caminhos independentes** possíveis no código.
- Verificam **todas as decisões lógicas** com saídas verdadeiras e falsas.
- Executam **ciclos (laços)** dentro dos limites e em seus extremos.
- Avaliam **estruturas de dados internas**, garantindo integridade e validade.

centro universitário FEI

Roteiro para Teste Caixa Branca

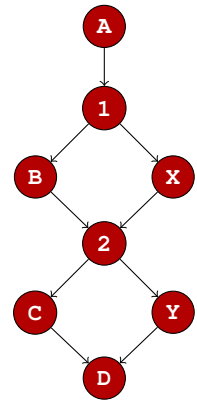
Caminho Independente:

- Caminho no programa que introduz **pelo menos uma nova aresta** (comando ou condição).
- Deve incluir uma aresta que ainda não tenha sido atravessada por caminhos anteriores.
- Cada novo caminho revela uma parte única da lógica do programa.

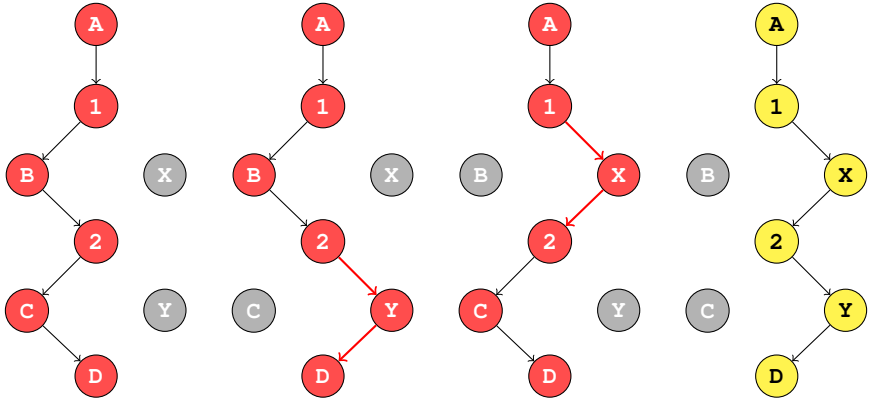
centro universitário FEI

Fluxo de Controle – Exemplo

```
print("A")
if (condition1)
        print("X")
else
        print("B")
if (condition2)
        print("Y")
else
        print("C")
print("D")
```



Caminhos de Execução no Fluxo de Controle



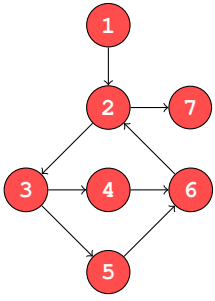
Complexidade Ciclomática

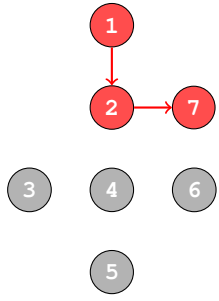
- Métrica que indica o **número de caminhos linearmente independentes** no código.
- Calculada a partir do **grafo de fluxo de controle**:

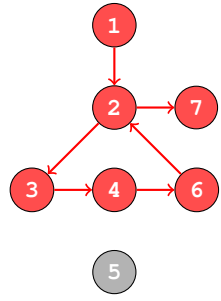
$$V(G) = E - N + 2P$$

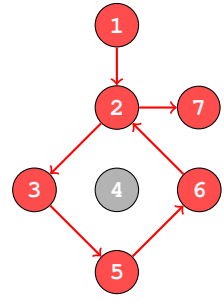
  - $V(G)$: complexidade ciclomática do grafo G;
  - E: número de arestas (edges);
  - N: número de nós (nodes);
  - P: número de componentes conexas (geralmente $P = 1$ para programas com um único ponto de entrada e saída).

Exemplo de Caminhos Independentes









- **Caminho 1:** $1 \rightarrow 2 \rightarrow 7$
- **Caminho 2:** $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \rightarrow 2 \rightarrow 7$
- **Caminho 3:** $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 2 \rightarrow 7$

centro universitário FEI

Teste de Caminho Base

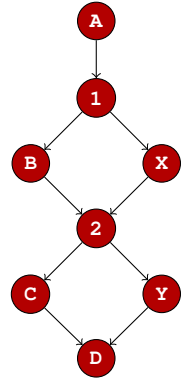
- Utiliza os **caminhos independentes** como base para elaborar casos de teste.
- Garante que cada comando e decisão do programa seja executado pelo menos uma vez.
- **Não assegura cobertura completa,** pois combinações internas podem ainda não ser testadas.

centro universitário FEI

Exemplo: Conjunto de Testes por Caminhos Independentes

Código de exemplo:

```
print("A")
if (condition1)
        print("X")
else
        print("B")
if (condition2)
        print("Y")
else
        print("C")
print("D")
```



Complexidade ciclomática: $V(G) = E - N + 2P = 9 - 8 + 2 = 3$

Caminhos de teste:

Caminho	Condição 1	Condição 2	Fluxo
1	True	True	A → X → Y → D
2	False	True	A → B → Y → D
3	False	False	A → B → C → D

Teste de Comando

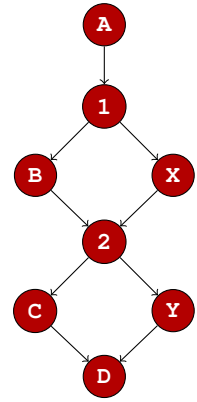
- Também conhecido como **teste de cobertura de instruções**.
- Garante que **cada comando (ou instrução)** do programa seja executado pelo menos uma vez durante os testes.
- Importante para detectar:
  - Código morto (instruções nunca executadas);
  - Erros de lógica em blocos condicionais e laços.
- Aplica-se principalmente a:
  - Condições: `if`, `else`, `switch/case`
  - Repetições: `for`, `while`, `do/while`

Objetivo: executar todos os blocos de código ao menos uma vez.

centro universitário FEI

Exemplo: Cobertura de Comando

```
print("A")
if condition1:
        print("X")
else:
        print("B")
if condition2:
        print("Y")
else:
        print("C")
print("D")
```



Casos de teste:

- **CT01:** condition1 = True e condition2 = True
  Fluxo: A → X → Y → D.
- **CT02:** condition1 = False e condition2 = False
  Fluxo: A → B → C → D.

Cobertura: Os casos de teste CT01 e CT02 garantem que todas as instruções sejam executadas ao menos uma vez.

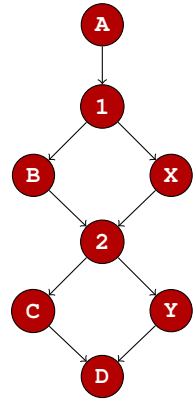
Teste de Ramos

- Garante que **cada decisão lógica** (nó do grafo) seja avaliada em seus dois resultados:
  - Verdadeiro
  - Falso
- Cada bifurcação no grafo deve ser testada em ambas as direções.

centro universitário FEI

Exemplo: Cobertura de Ramos

```
print("A")
if condition1:
        print("X")
else:
        print("B")
if condition2:
        print("Y")
else:
        print("C")
print("D")
```



Caso de Teste	Nó 1 (condition1)	Nó 2 (condition2)
CT01	Verdadeiro	Verdadeiro
CT02	Verdadeiro	Falso
CT03	Falso	Verdadeiro
CT04	Falso	Falso

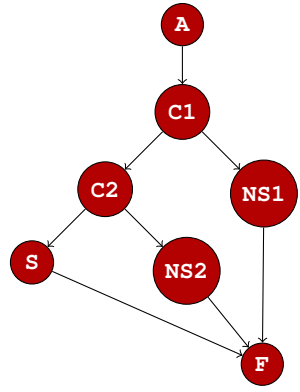
Teste de Condição

- Avalia todas as **condições compostas** de forma isolada.
- Cada subcondição deve ser testada com os valores:
  - Verdadeiro
  - Falso
- Casos de teste devem cobrir combinações internas de expressões lógicas.

centro universitário FEI

Exemplo: Teste de Condição

```
print("Inicio")
if (a > 0 and b < 10):
        print("Condição␣Satisfeita")
else:
        print("Condição␣Não␣Satisfeita")
print("Fim")
```



Caso de Teste	a > 0?	b < 10?	Valores	Resultado
CT1	V	V	a=5, b=5	Satisfeita
CT2	V	F	a=5, b=15	Não Satisfeita
CT3	F	V	a=-3, b=5	Não Satisfeita
CT4	F	F	a=-3, b=15	Não Satisfeita

Teste de Ciclo

- Foca exclusivamente na **validade de laços de repetição**.
- Permite identificar falhas em estruturas como while, for, do-while.
- Define diferentes classes de comportamento de execução dos ciclos.

centro universitário FEI

Ciclo Simples

- Testa laços de repetição com base em limites predefinidos:
  1. Pular o ciclo (0 iterações)
  2. Apenas uma passagem
  3. Duas passagens
  4. m passagens, onde $m < n$
  5. $n-1$, n e $n+1$ passagens
- Para o exemplo apresentado, considera-se $n = 3$.

centro universitário FEI

Ciclo Aninhado

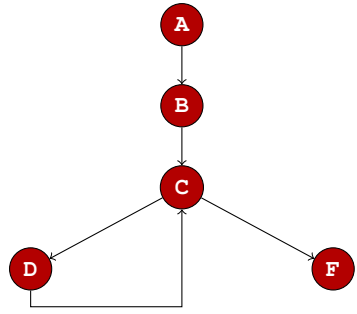
- Quando há laços dentro de laços, a quantidade de combinações cresce **exponencialmente**.
- A técnica:
  - Começa testando os **ciclos internos**, mantendo os externos no mínimo.
  - À medida que ?sai?, testa os externos com mais iterações e mantém os internos em valores padrão.

Ciclo Concatenado

- Laços consecutivos que não são aninhados.
- Se **independentes**, aplica-se teste de ciclo simples em cada um.
- Se **dependentes** (ex.: um ciclo define o limite do outro), tratam-se como **aninhados**.

Exemplo: Teste de Ciclo (Ciclo Simples)

```
print("Inicio")
i = 0
while i < len(numeros):
        raiz = sqrt(numeros[i])
        print("Raiz:", raiz)
        i += 1
print("Fim")
```



Pular o Ciclo

- **Entrada:** `numeros = []` (lista vazia)
- O laço não é executado nenhuma vez.
- Testa a condição inicial de parada.

centro universitário FEI

Uma e Duas Passagens

- **CT1:** `numeros = [9]` 1 iteração (raiz = 3)
- **CT2:** `numeros = [9, 25]` 2 iterações (raiz = 3 e 5)
- Testam a execução mínima e intermediária do ciclo.

$m < n$ e $n-1$, n, $n+1$ Passagens

- Para $n = 3$:
- **CT3:** numeros = [9, 25] $m = 2 < n$
- **CT4:** numeros = [9, 25, 4, 36] Testa 3 e 4 passagens
- Permite verificar a estabilidade do laço com diferentes volumes de dados.

centro universitário FEI

Teste de Caixa Branca

Exercícios para Entrega

Exercício 1 – Caminhos Independentes

Código:

```python
def verificar(n):
    if n > 0:
        if n % 2 == 0:
            return "Par positivo"
        else:
            return "Ímpar positivo"
    elif n < 0:
        return "Negativo"
    else:
        return "Zero"
```

Tarefa:

- Desenhe o grafo de fluxo de controle.
- Calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Defina os casos de teste.

centro universitário FEI

Exercício 2 – Teste de Comando e Ramos

Código:

```
1  def classificar(x):
2      if x > 100:
3          return "Alto"
4      if x > 50:
5          return "Médio"
6      return "Baixo"
```

Tarefa:

- Desenhe o grafo de fluxo de controle.
- Calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Elabore um conjunto mínimo de testes para cobrir:
  - Todos os comandos
  - Todas as decisões (ramos)

centro universitário FEI

Exercício 3 – Teste de Condição

Código:

```
1 def acesso(idade, membro):
2     if idade >= 18 and membro:
3         return "Permitido"
4     return "Negado"
```

Tarefa:

- Desenhe o grafo de fluxo de controle.
- Calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Crie casos de teste que explorem todas as **combinações** possíveis da condição composta.
- Explique como os testes cobrem os resultados possíveis (Verdadeiro/Falso).

centro universitário FEI

Exercício 4 – Teste de Ciclo

Código:

```
1  def somar_ate(n):
2      soma = 0
3      for i in range(n):
4          soma += i
5      return soma
```

Tarefa:

- Desenhe o grafo de fluxo de controle.
- Calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Realize testes para as seguintes condições:
  - O laço é **ignorado** (0 iterações)
  - O laço executa **uma única vez**
  - O laço executa **várias vezes**
- Determine o valor de saída esperado para cada caso.

centro universitário FEI

Exercício 5 – Teste de Ciclo (Aninhado)

Código:

```
1  def multiplicar_matrizes(m, n):
2      for i in range(m):
3          for j in range(n):
4              print(f"Posição ({i}, {j})")
```

Tarefa:

- Desenhe o grafo de fluxo de controle.
- Calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Realize testes para os seguintes cenários:
  - Ambos os laços são **ignorados**
  - Apenas o laço j é **ignorado**
  - Um laço executa **uma vez** e outro **várias vezes**
  - Ambos os laços executam **várias vezes**
- Determine quantas vezes a linha print é executada em cada caso.

centro universitário FEI

Exercício 6 – Teste Completo

Código:

```
1  def analisar(numeros):
2      total = 0
3      for n in numeros:
4          if n > 0 and n % 2 == 0:
5              total += n
6          elif n < 0:
7              total -= 1
8          else:
9              continue
10     if total > 10:
11         return "Acima"
12     return "Abaixo"
```

Tarefa:

- Desenhe o grafo de fluxo de controle e calcule a complexidade ciclomática.
- Identifique os caminhos independentes.
- Elabore casos de teste para:
  - Cobertura de **comando** e **ramo**
  - Cobertura de **condição** (composta: `n > 0 and n % 2 == 0`)
  - Comportamentos do **laço** (0, 1, várias iterações)
- Para cada caso, indique o valor de retorno esperado.

centro universitário FEI

Simulação e Teste de Software (CC8550)

Aula 07 – Teste de Caixa Branca

Baseado no material desenvolvido pelo prof. Calebe de Paula Bianchini.

Prof. Luciano Rossi

Ciência da Computação
Centro Universitário FEI

1º Semestre de 2025