

UNIVALI  
CIÊNCIA DA COMPUTAÇÃO  
ENGENHARIA DE SOFTWARE 2

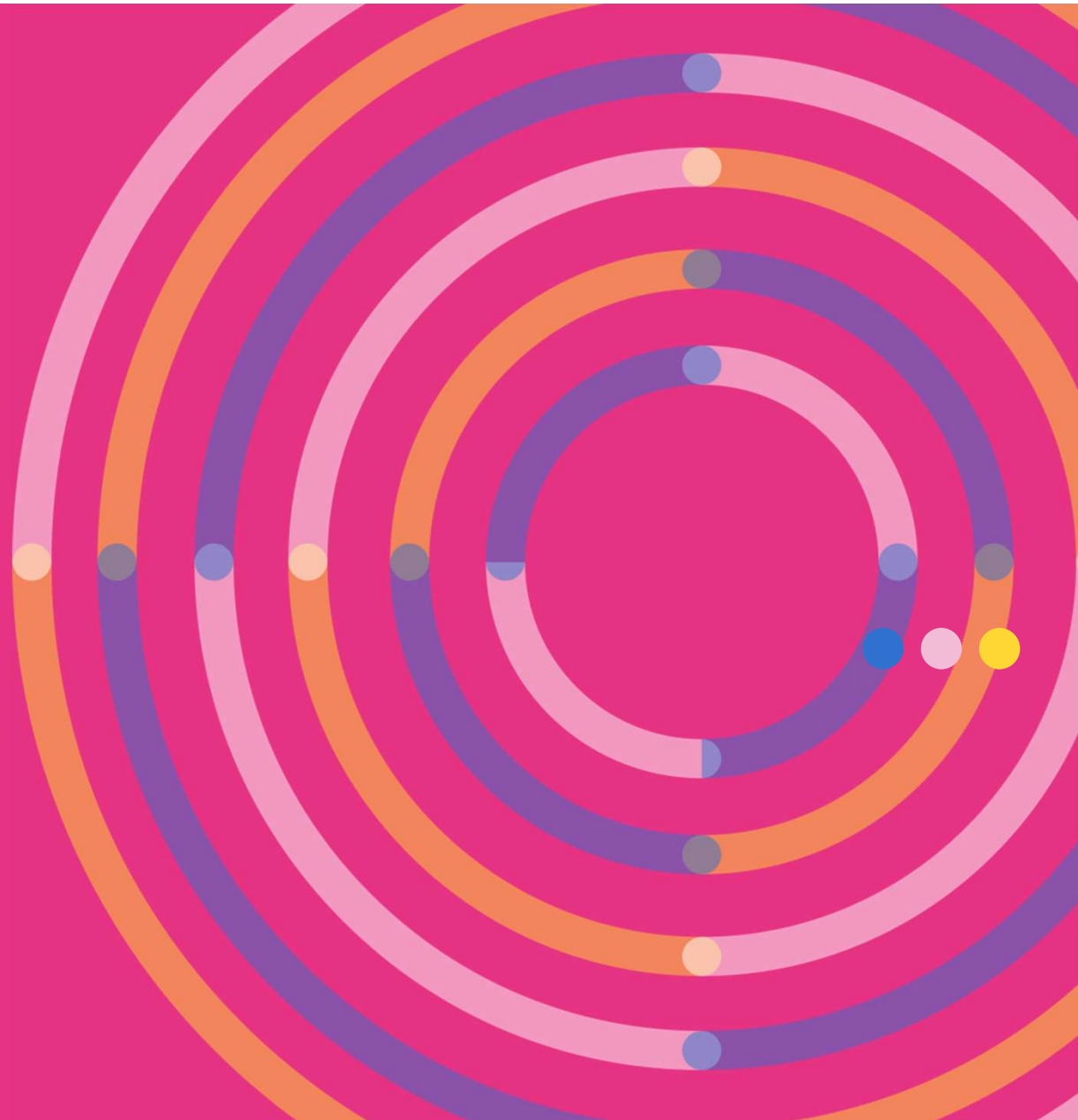
# Técnicas de teste de software

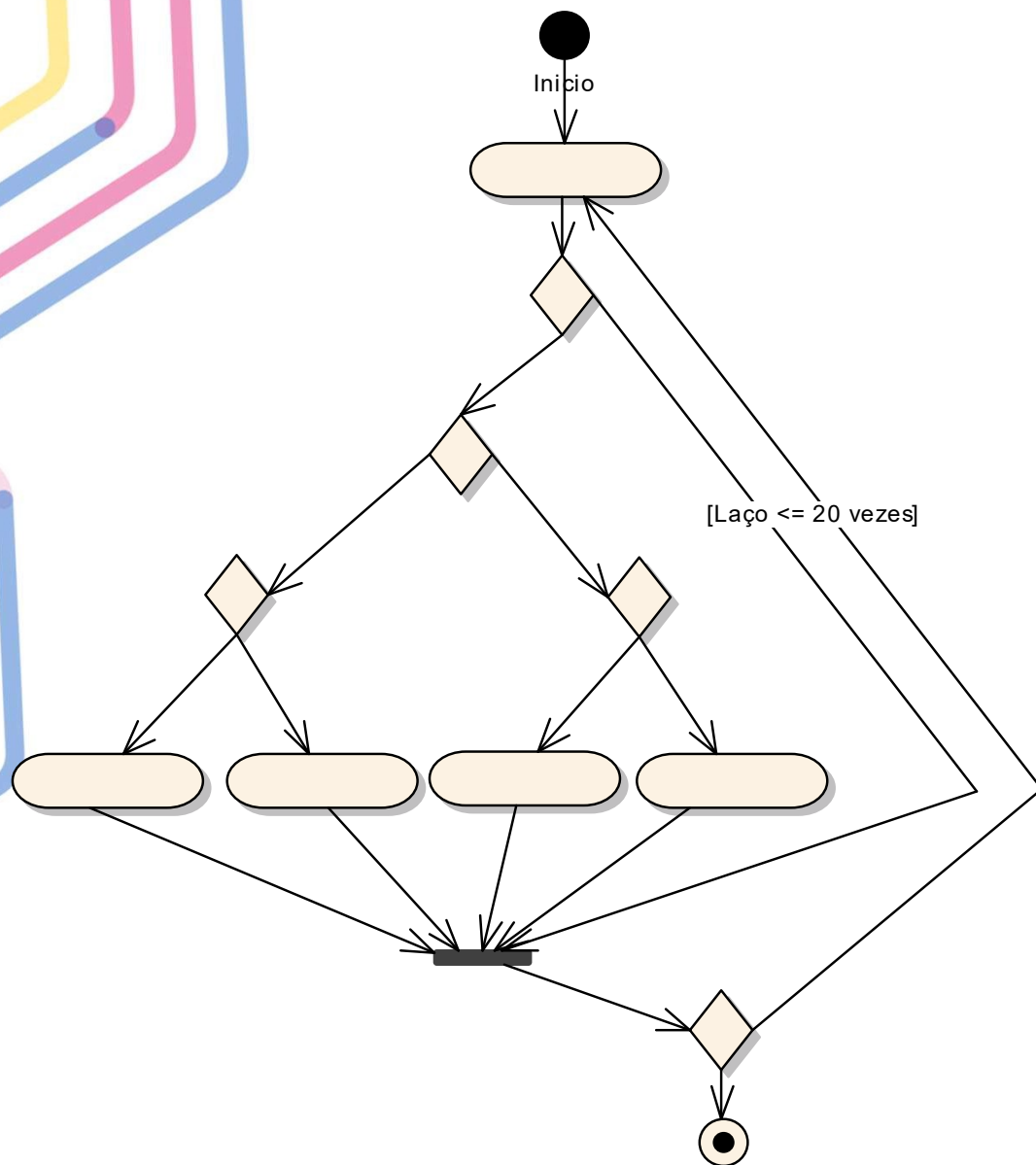
Professora Dra. Adriana Gomes Alves



# Técnicas de Teste – Como Testar?

- Técnica de Teste:
  - É um método sistemático usado para selecionar e/ou gerar testes para serem incluídos em um projeto de teste.
- Exemplos:
  - Teste Estrutural → também chamado de Teste Caixa Branca;
  - Teste Funcional → também chamado de Teste Caixa Preta;





Testes exaustivos são impossíveis, pois o número de caminhos a serem percorridos é muito grande.



## Impossibilidade do teste completo em função de:



- ❖ Não é possível testar todas as entradas de um programa;
- ❖ Não é possível testar todas as combinações de dados para um programa;
- ❖ Não é possível testar todos os caminhos do programa;
- ❖ Não é possível testar todas as potenciais falhas, tais como as geradas por erro de design de interface ou análise incompleta dos requisitos.



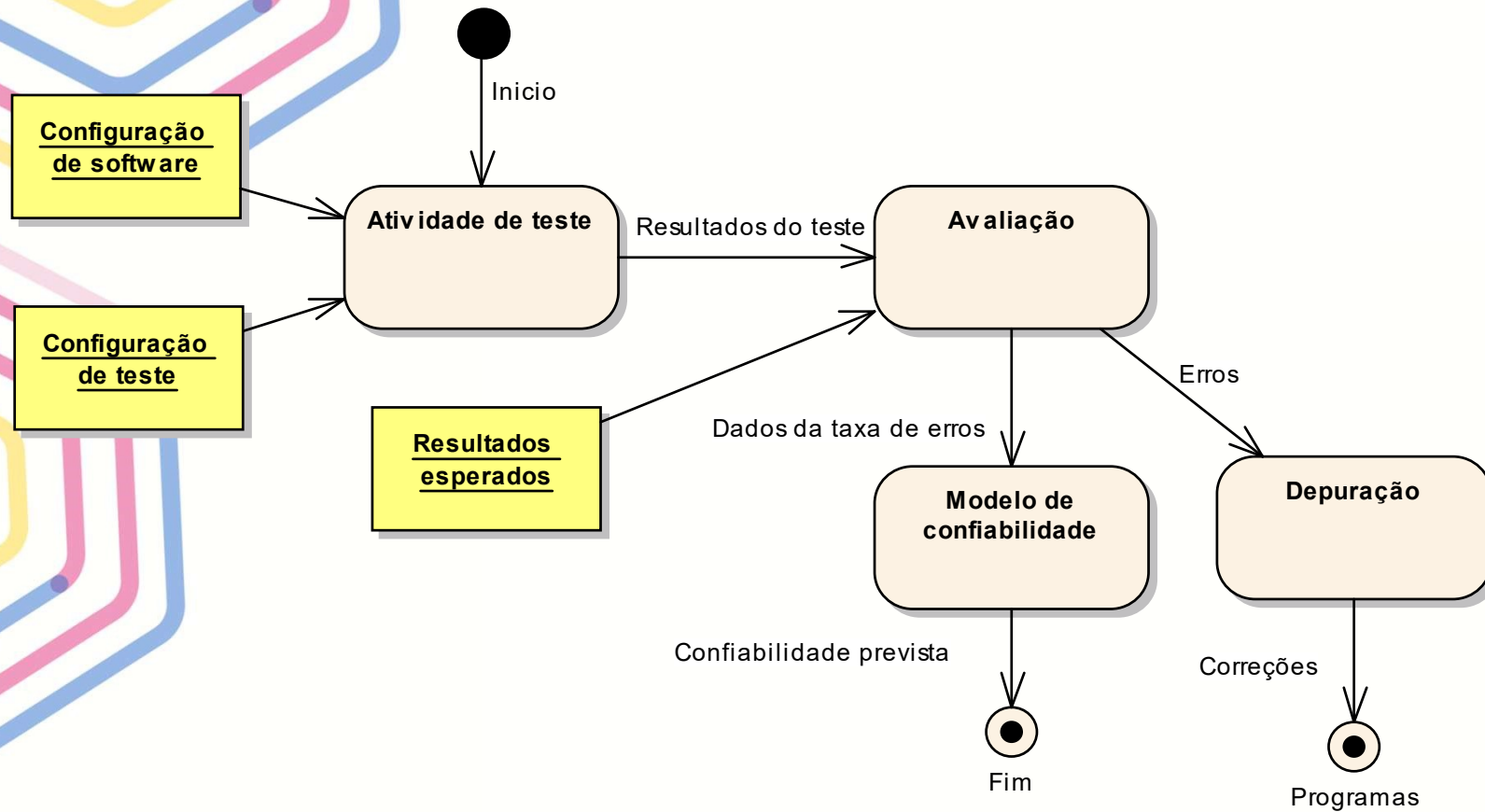
# O que é Caso de Teste?

**“É um conjunto de entradas de testes, condições de execução e resultados esperados desenvolvidos com o objetivo de exercitar um caminho particular de um programa ou verificar a conformidade com um requisito específico.”  
(IEEE 610, 1990)**

**São as entradas específicas que serão usadas e os procedimentos que serão seguidos quando o software for testado.” (Patton, 2005).**



# Fluxo de informação de teste



Fonte: adaptado de (PRESMAN, 1995)

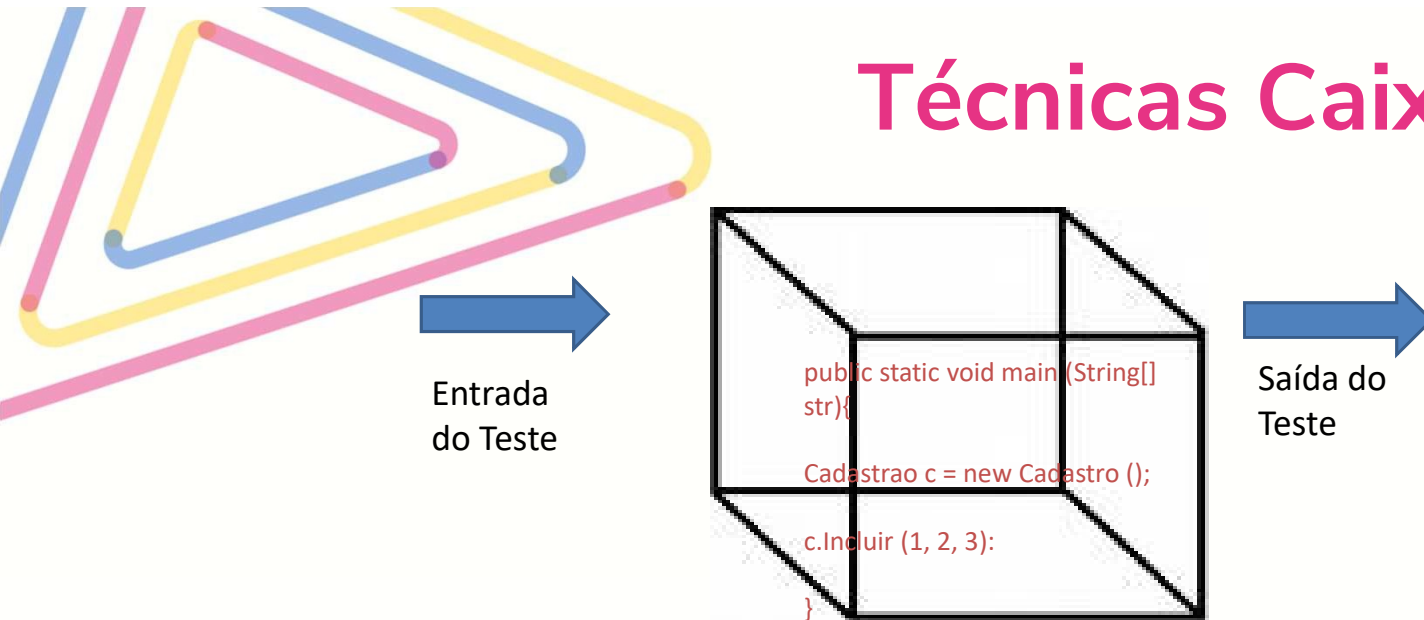


## Resultados dos testes

- Se os erros forem muito graves
  - Duvidar da qualidade do software
- Se forem fáceis de corrigir
  - A qualidade e confiabilidade são aceitáveis OU
  - Os testes são inadequados
- Se não encontrar nenhum erro
  - Não há dúvidas de que os testes são inadequados e os erros estão escondidos no software.



# Técnicas Caixa Branca



- Seu objetivo é determinar defeitos na estrutura interna ou no código do software.
- São realizados pelos programadores nos seus códigos.
- Testa-se os IF, WHILE, FOR e todas as estruturas de controle, acesso a dados e interface.
- Exemplos: Teste do caminho e teste de fluxo de dados



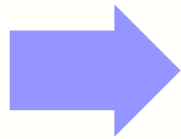


# Técnicas Caixa Branca



## Vantagens

- Possibilita o teste da lógica do software.
- Possibilita o teste de atributos estruturais, tais como a eficiência do código.



## Desvantagens

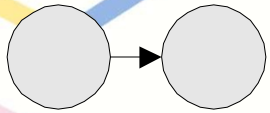
- Não garante que os requisitos dos usuários foram atendidos.
- Não pode simular situações reais

# Teste do caminho básico

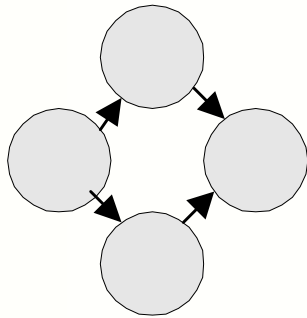
O método de caminho básico possibilita que o projetista do caso de teste derive uma medida da complexidade lógica de um projeto procedimental e use essa medida como guia para definir um conjunto básico de caminhos de execução.



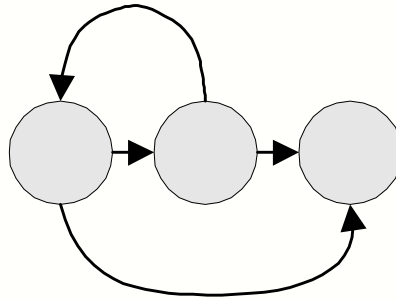
# Notação do grafo de fluxo



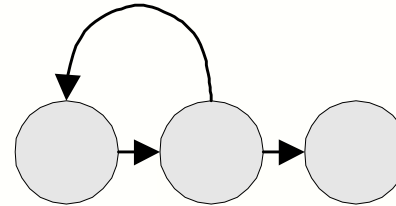
Seqüência



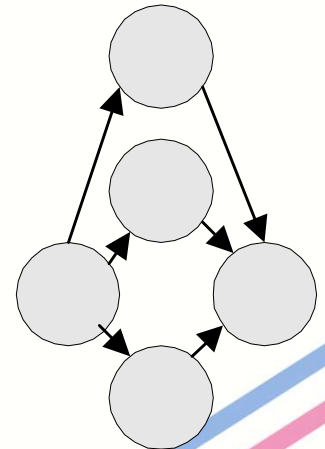
If



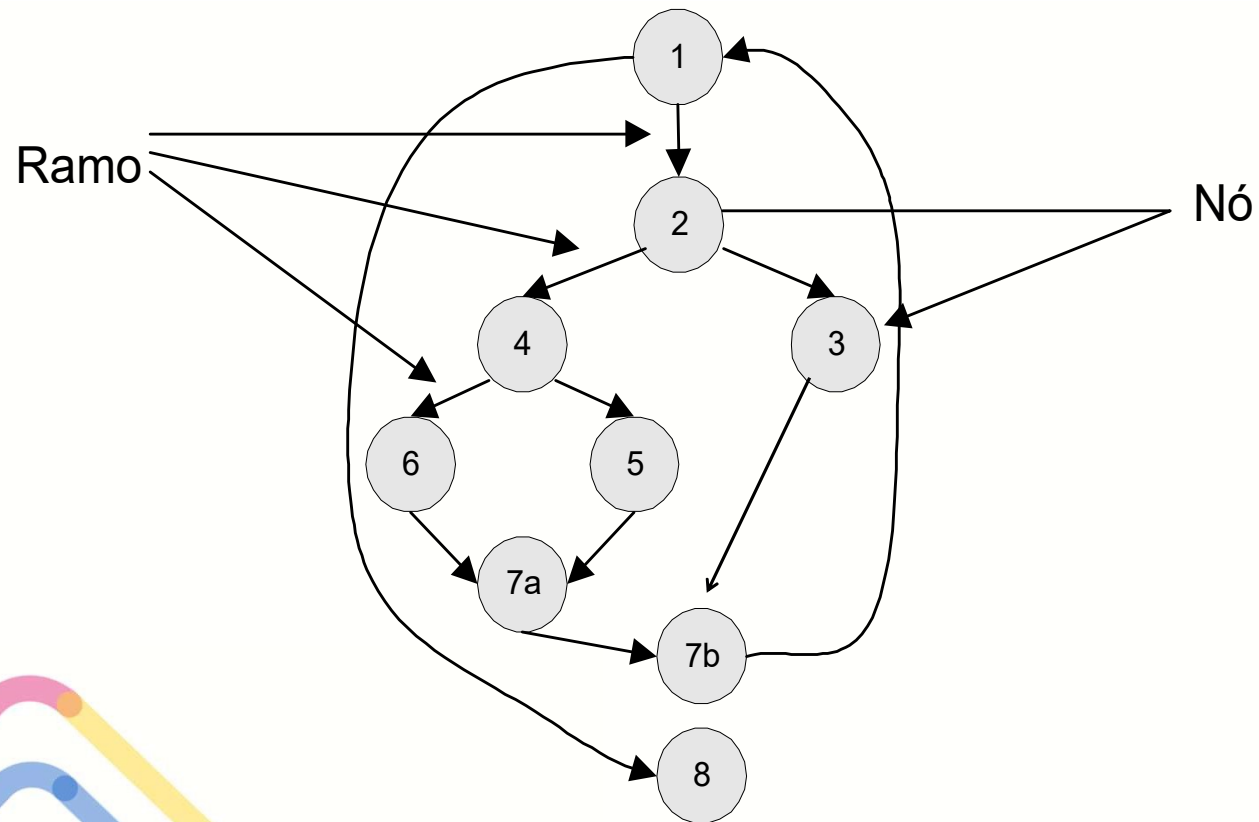
While



Repeat

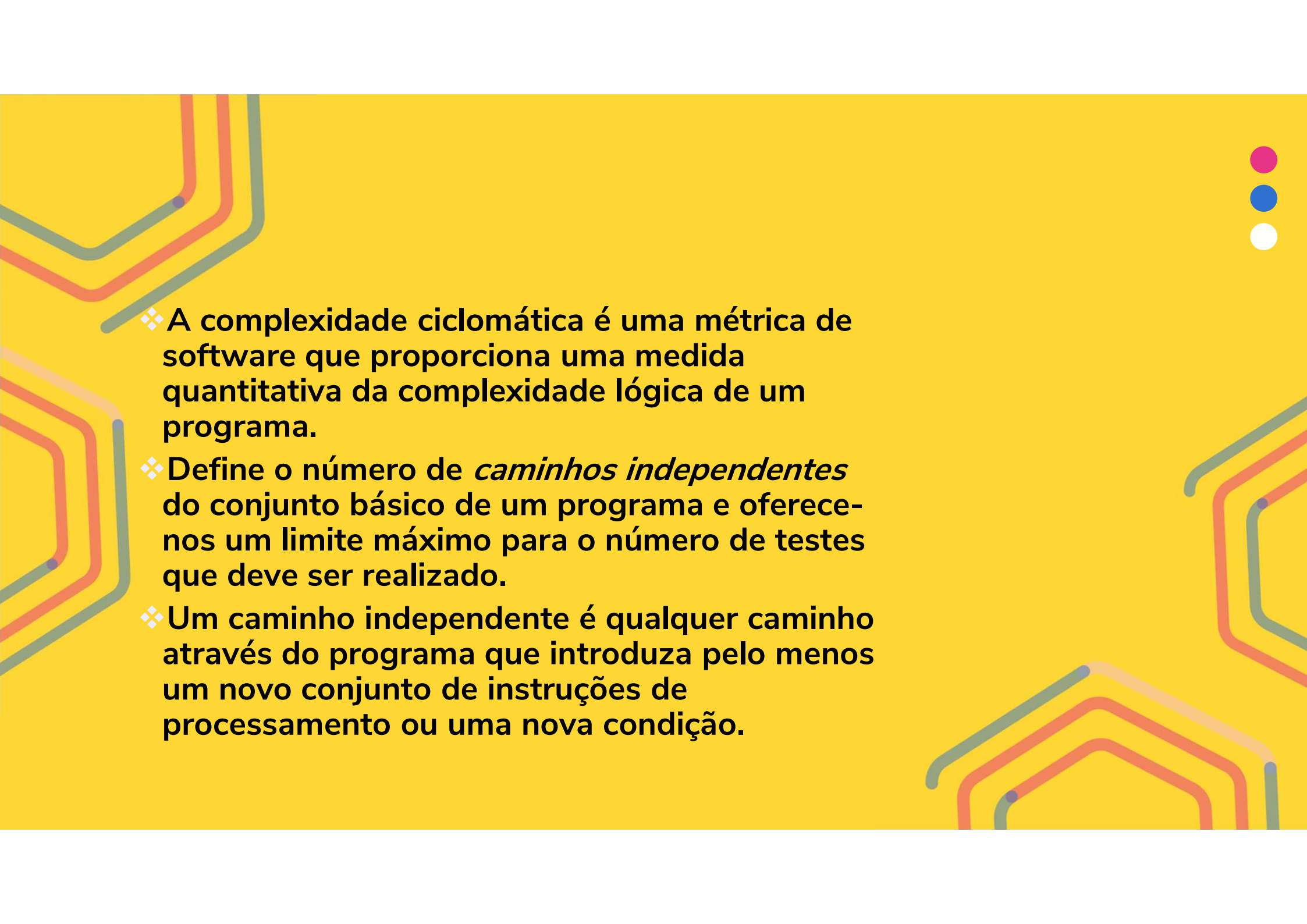


Case

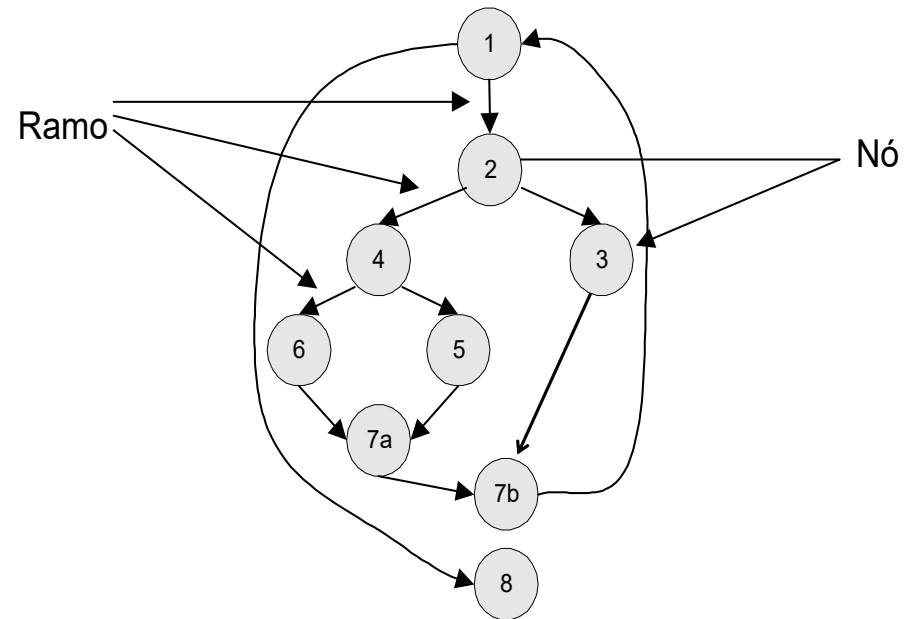


1:	do while records remain
	read record;
2:	if record field 1 = 0
3:	then process record;
	store in buffer;
	increment counter;
4:	elseif record field 2 = 0
5:	then reset counter;
6:	else process record;
	store in file;
7a:	endif
	endif
7b:	enddo
8:	end



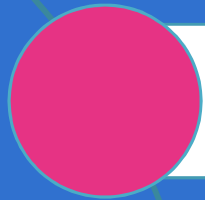
- 
- The background is a solid yellow color. In the top right corner, there are three vertically aligned circles: a pink one at the top, a blue one in the middle, and a white one at the bottom. On the left and right sides, there are decorative elements consisting of multiple parallel lines in shades of pink, red, and grey, which curve and bend in various directions, creating a stylized, abstract pattern.
- ❖ A complexidade ciclomática é uma métrica de software que proporciona uma medida quantitativa da complexidade lógica de um programa.
  - ❖ Define o número de *caminhos independentes* do conjunto básico de um programa e oferece-nos um limite máximo para o número de testes que deve ser realizado.
  - ❖ Um caminho independente é qualquer caminho através do programa que introduza pelo menos um novo conjunto de instruções de processamento ou uma nova condição.

- Caminhos possíveis para o exemplo anterior.
- Caminho 1: 1-8
- Caminho 2: 1-2-3-7b-1-8
- Caminho 3: 1-2-4-5-7a-7b-1-8
- Caminho 4: 1-2-4-6-7a-7b-1-8

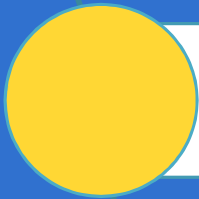




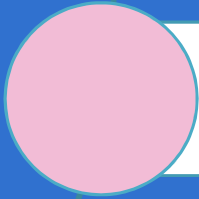
# Teste de Caixa branca



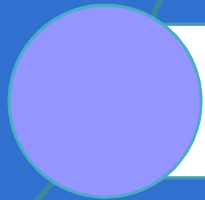
Elaborar os casos de teste.



Para cada caminho definido, atribuir valores as entradas e definir os resultados esperados.



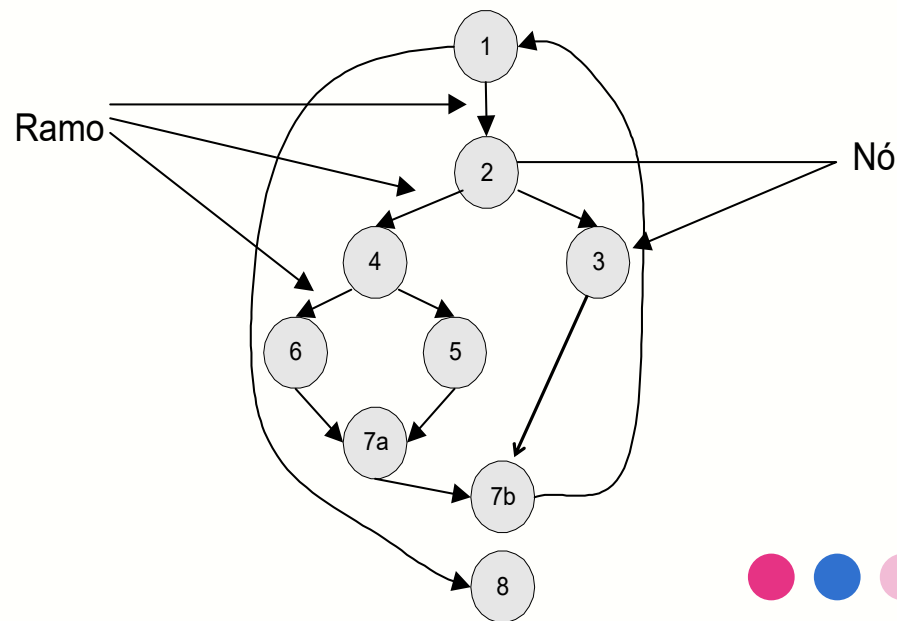
Cada caso de teste é executado e comparado com os resultados esperados.



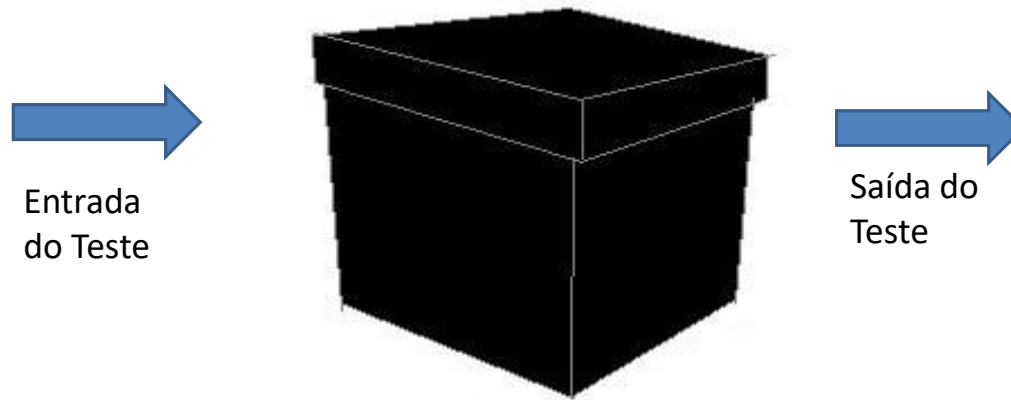
Assim que todos os casos de testes tenham sido executados, o analista pode certificar-se que todas as instruções foram executadas ao menos uma vez.

Caminho	Entradas	Resultado esperado
Caso 1 (Caminho 1: 1-8)	Arquivo vazio	Vai para o fim do programa
Caso 2 (Caminho 2: 1-2-3-7b-1-8)	Field 1=0	Incrementou o Contador
Caso 3 (Caminho 3: 1-2-4-5-7a-7b-1-8)	Field 1=1 / Field 2=0	Zerou o Contador
Caso 4 (Caminho 4: 1-2-4-6-7a-7b-1-8)	Field 1=1 / Field 2=1	Armazenou no arquivo

1:	do while records remain
	read record;
2:	if record field 1 = 0
3:	then process record;
	store in buffer;
	increment counter;
4:	elseif record field 2 = 0
5:	then reset counter;
6:	else process record;
	store in file;
7a:	endif
	endif
7b:	enddo
8:	end



# Técnicas Caixa Preta



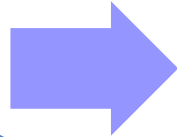
- São baseadas em requisitos, tendo por objetivo determinar se os requisitos foram total ou parcialmente satisfeitos pelo software.
- Verifica apenas os resultados produzidos e não requer conhecimento interno do sistema, apenas conhecimento dos requisitos de sistema / negócio.
- Exemplos: Classe de equivalência e análise do valor limite




# Técnicas Caixa Preta

## Vantagens

- Os casos de testes são independentes do modo como o software é implementado.
- Elaboração dos casos de testes podem ocorrer em paralelo com a implementação.



## Desvantagens

- Possibilidade de casos de testes redundantes.
  - Possibilidade de deixar passar erros de lógica no software.
- 

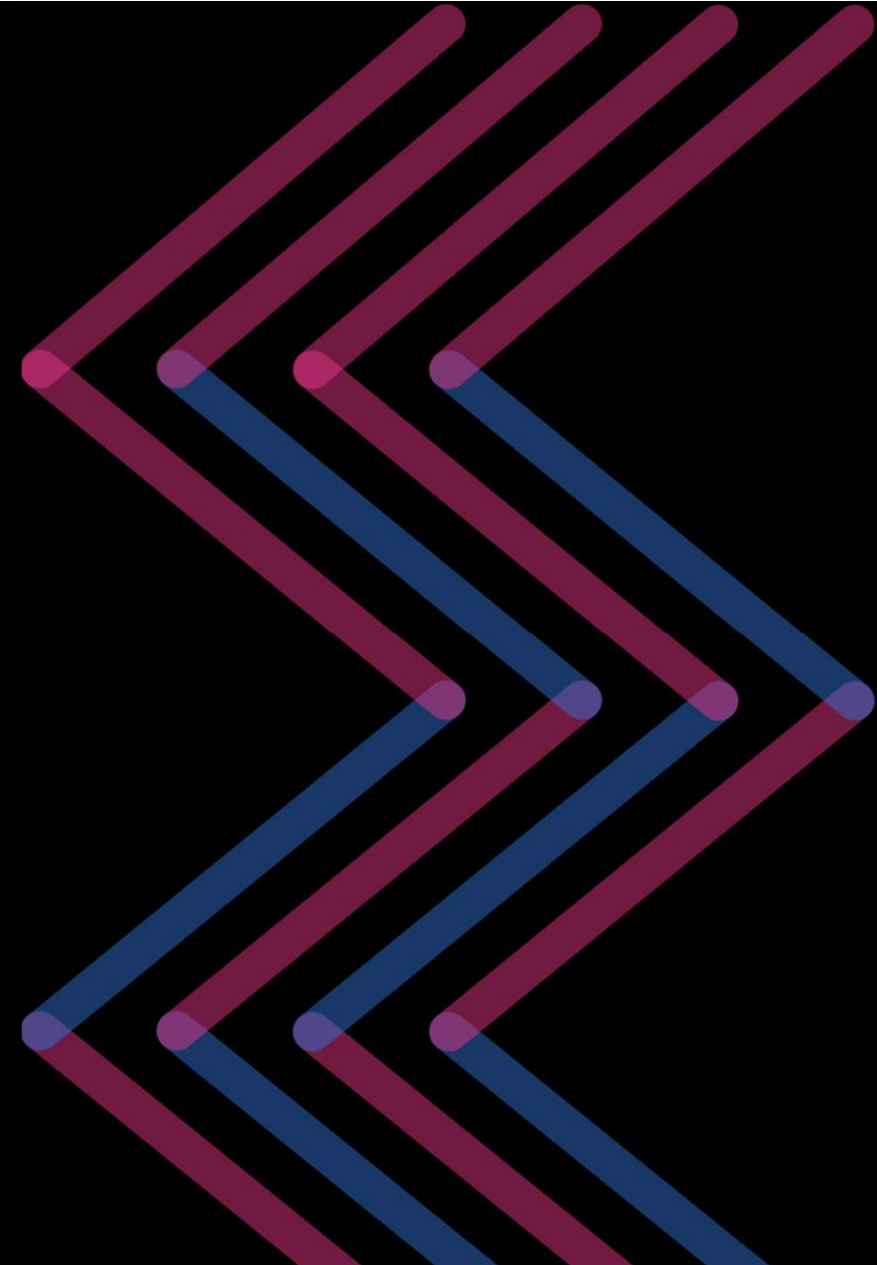
## Categorias de erros

- ❖ Funções incorretas ou ausentes
- ❖ Erros de interface
- ❖ Erros nas estruturas de dados ou no acesso a banco de dados externo
- ❖ Erros de desempenho
- ❖ Erros de inicialização e término.



# Derivação dos Casos de Teste

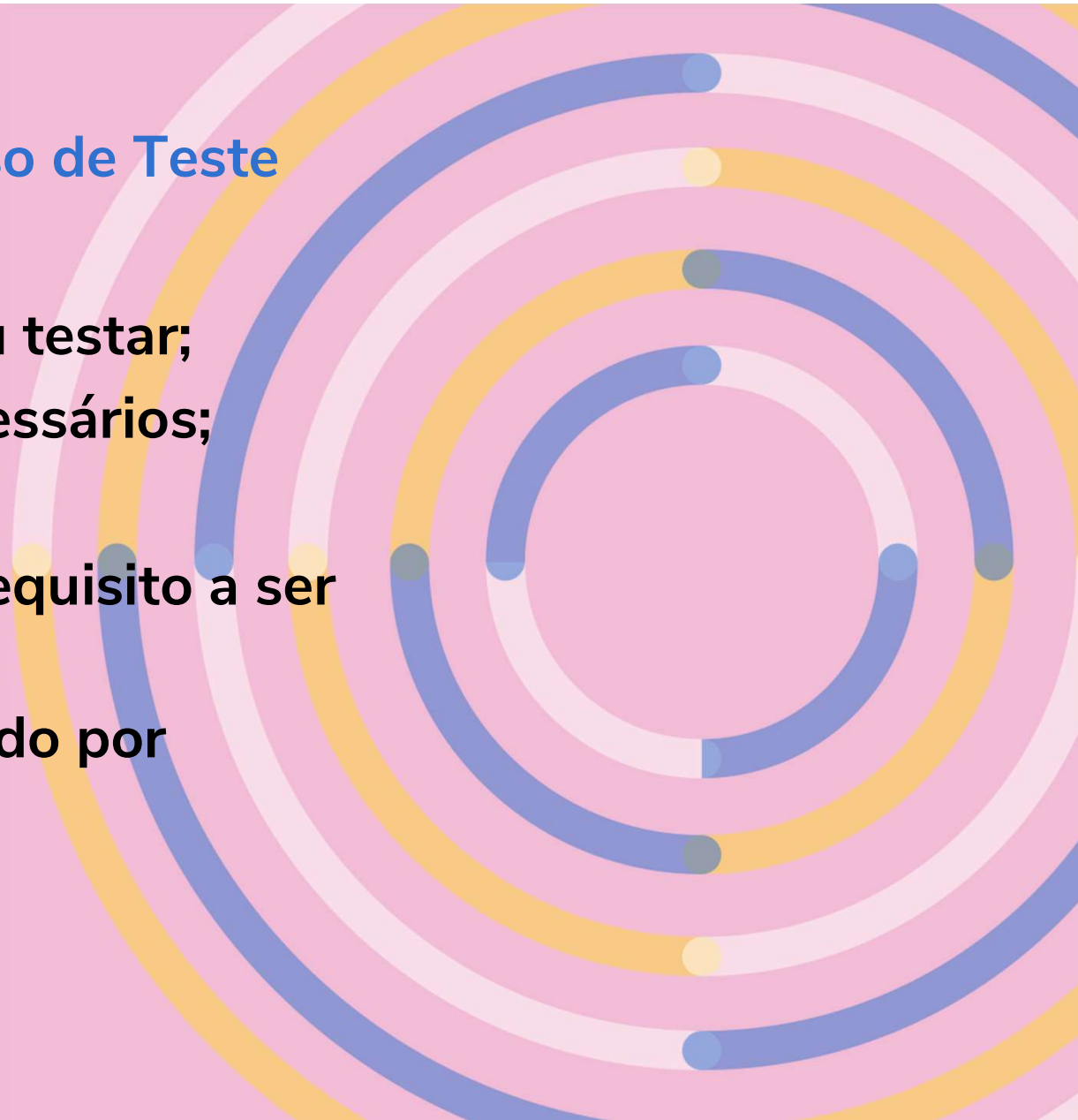
- Os casos de testes são derivados de um documento de especificação que define os requisitos.
- Documentos utilizados para derivação:
  - Especificação de Requisitos
  - Casos de Uso
  - Cenários de Negócios
  - Especificação Funcional

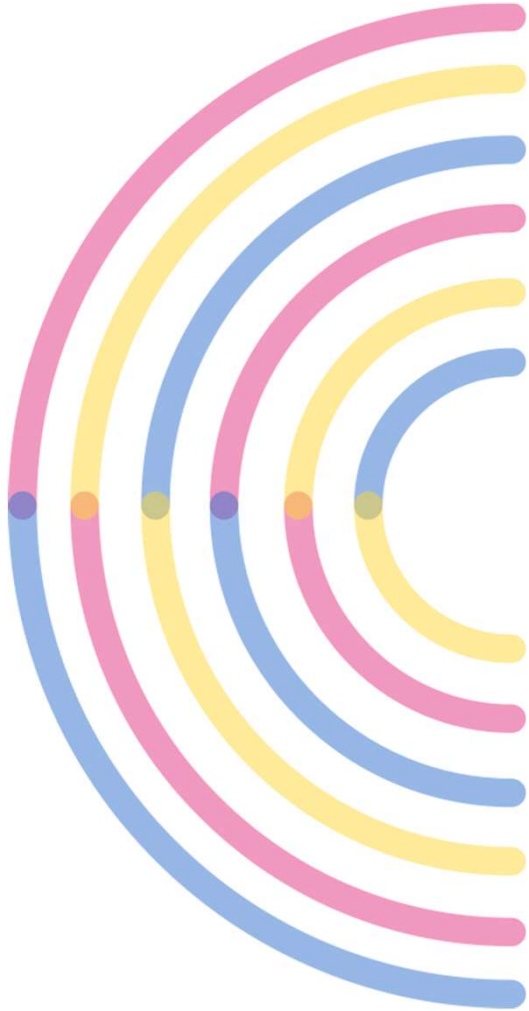




## Características de um Bom Caso de Teste

- **Efetivo:** Testar o que se planejou testar;
- **Econômico:** Sem passos desnecessários;
- **Reutilizável:** Possa ser repetido;
- **Rastreável:** Possa identificar o requisito a ser testado;
- **Autoexplicativo:** Possa ser testado por qualquer pessoa





# Exemplo de Casos de Teste



*Tela Pesquisar obra*

Título:



Autor:

Voltar

Pesquisar

No mínimo quantos casos de testes teríamos nesta situação?

# Resposta



Título	Autor	Resultado
V	V	Pesquisa OK
V	-	Pesquisa OK
-	V	Pesquisa OK
-	-	Msg erro
V	I	Msg erro
I	V	Msg erro
I	-	Msg erro
-	I	Msg erro
I	I	Msg erro

**Fazendo este exercício, implicitamente você está utilizando uma técnica de testes.**

# Classe de Equivalência

- Uma mesma classe de equivalência identifica variações de entrada e condições iniciais as quais se esperam produzir o mesmo resultado.
- **Motivações:**
  - Desejo de se realizar um teste completo;
  - Desejo de se evitar redundância.



# Classe de Equivalência

- **Um grupo de casos de testes pertence a mesma classe de equivalência se:**
  - Todos eles testam a mesma coisa;
  - Se um deles encontrar um defeito, provavelmente os outros também encontrarão;
  - Se um deles não encontra erro, provavelmente os outros também não encontrarão.



# Classe de Equivalência


- **Objetivo de agrupar casos de teste em uma mesma classe de equivalência:**
  - **É a redução da quantidade de casos de testes a serem testados evitando redundâncias sem afetar a completude dos testes**







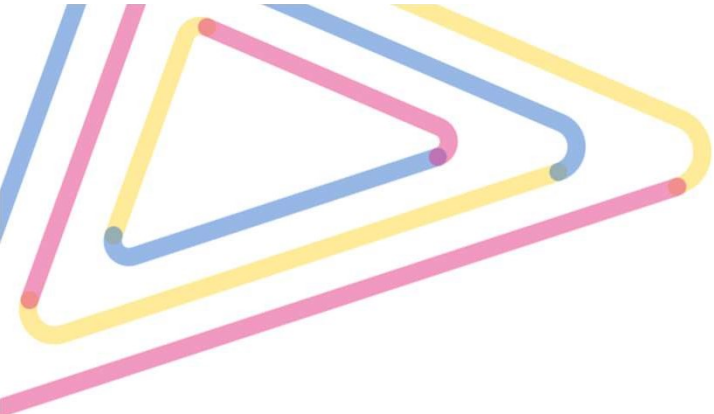
# Classe de Equivalência

- **A elaboração de casos de testes utilizando a classe de equivalência consiste de duas etapas:**
    1. **Identificar as classes de equivalência**
    2. **Selecionar os casos de teste.**
- 


## Identificação da Classe de Equivalência

- Deve ser considerado cada condição de entrada (usualmente uma sentença ou frase da especificação) que deverá ser dividida em dois grupos.
- Tipos de classe de equivalência:
  - Válidas – representam entradas válidas para o programa
  - Inválidas – representam entradas inválidas para o programa
- Recomenda-se uma maior atenção as classes de equivalência inválidas, por serem ótima fonte de erro, já que os desenvolvedores não as consideram em seus testes






## Identificação da Classe de Equivalência

- **Recomendações para identificar boas classes de equivalência:**
    - Não esqueça das classes de equivalência para valores inválidos;
    - Organize a sua classificação em uma tabela;
    - Procure por intervalos de números;
    - Procure por variáveis que devem ser iguais;
    - Procure por grupo de variáveis que devem calcular um determinado valor ou intervalo;
    - Procure por eventos de saídas equivalentes;
    - Procure por ambientes operacionais equivalentes
- 




## Identificação da Classe de Equivalência

- **Exemplos de classes de equivalência identificadas a partir de condições de entrada:**
    - Se uma condição de entrada especifica um intervalo de valores (por exemplo, “O campo código pode receber valores de 1 a 999”), existiria então:
      - Uma classe de equivalência válida ( $1 \leq \text{código} \leq 999$ );
      - Duas classes de equivalência inválidas:
        - Código < 1;
        - Código > 999
    - Se uma condição de entrada especifica um sexo (por exemplo, “O exame de próstata só pode ser realizado por pessoa do sexo masculino”), existiria então:
      - Uma classe de equivalência válida (sexo = masculino)
      - Uma classe de equivalência inválida ( sexo = feminino)
- 



## Seleção dos Casos de Teste

- **Processo para seleção dos casos de testes a partir das classes de equivalências identificadas:**
    - Atribuir um número único para cada classe de equivalência.
    - Até que todas as classes de equivalências válidas tenham sido cobertas por casos de testes, escreva um novo caso de teste para cobrir tanto quanto possível as classes de equivalências válidas descobertas.
    - Até que os casos de testes tenham coberto todas as classes de equivalências inválidas, escreva um caso de teste único para cobrir cada uma das classes de equivalência inválida.
- 

## Seleção dos Casos de Teste

- A razão para que casos de testes individuais cubram classes de equivalências inválidas é que certas entradas errôneas podem mascarar ou substituir outras verificações de entradas errôneas.







## Exemplo de Classe de Equivalência



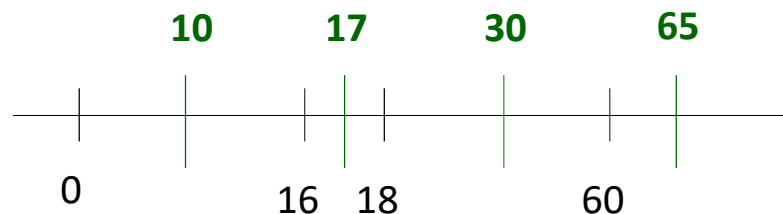
### Regras:

Idade menor que 16 anos → candidato não será contratado

Idade entre 16 e 18 anos → contratado em tempo parcial

Idade maior que 18 até 60 anos → contratado em tempo integral


Idade maior que 60 anos → candidato não será contratado





## Exemplo de Classe de Equivalência

Classe de Equivalência	Caso de Teste	Idade	Resultado Esperado
Idade < 16	CT01	10	Não contratado
16<= Idade <= 18	CT02	17	Contratado tempo parcial
19<= Idade <= 60	CT03	30	Contratado tempo integral
Idade > 60	CT04	65	Não contratado




## Análise do Valor Limite

- **Considerado o melhor método de seleção de casos de teste da técnica caixa preta.**
- **Existe uma maior probabilidade de falhas nos limites de uma classe de equivalência.**
- **Condições limites são situações diretamente acima e abaixo das fronteiras das classes de equivalência de entrada e de saída.**





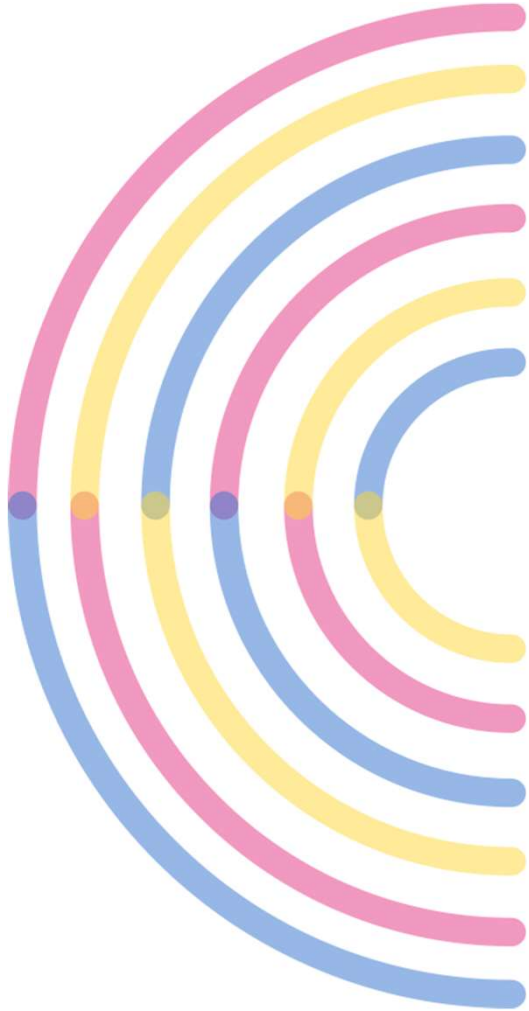
## Análise do Valor Limite

- **Recomendações para Identificar valores limites:**
    - Se uma condição de entrada especifica um intervalo de valores, escreva casos de testes para as extremidades dos intervalos e casos de testes de entradas inválidas para valores logo acima das extremidades. Por exemplo, se um intervalo válido de uma entrada varia de -1 à + 1, os casos de testes para esta situação são: {-1, +1, -1,001, 1,001};
    - se a entrada ou saída de um programa for um conjunto ordenado (um arquivo seqüencial, por exemplo, linear ou uma lista ou uma tabela), foque no primeiro e no último elemento do conjunto;
- 

# Análise do Valor Limite

- **Recomendações para Identificar valores limites:**

- se uma condição de entrada especifica um número de valores, escreva casos de testes para o número máximo e mínimo desses valores e para os números abaixo e acima desses valores. Por exemplo, se um arquivo de entrada pode conter de 1 a 255 registros, os casos de testes para esta situação são: { 0, 1, 255, 256};



## Exemplo de Análise de Valor Limite

### Regras:

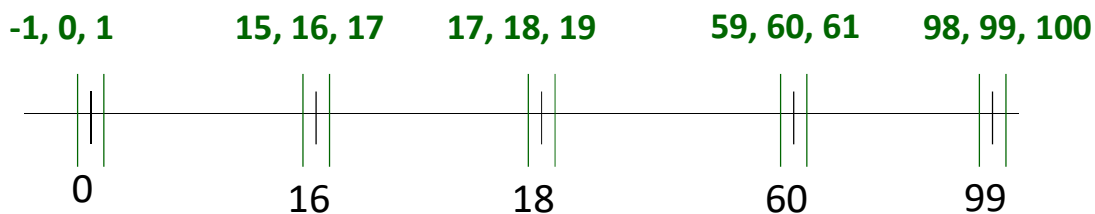
Idade menor que 16 anos → candidato não será contratado

Idade entre 16 e 18 anos → contratado em tempo parcial

Idade maior que 18 até 60 anos → contratado em tempo integral

Idade maior que 60 anos → candidato não será contratado

Campo Idade aceita valores de 0 a 99



# Exemplo de Análise de Valor Limite

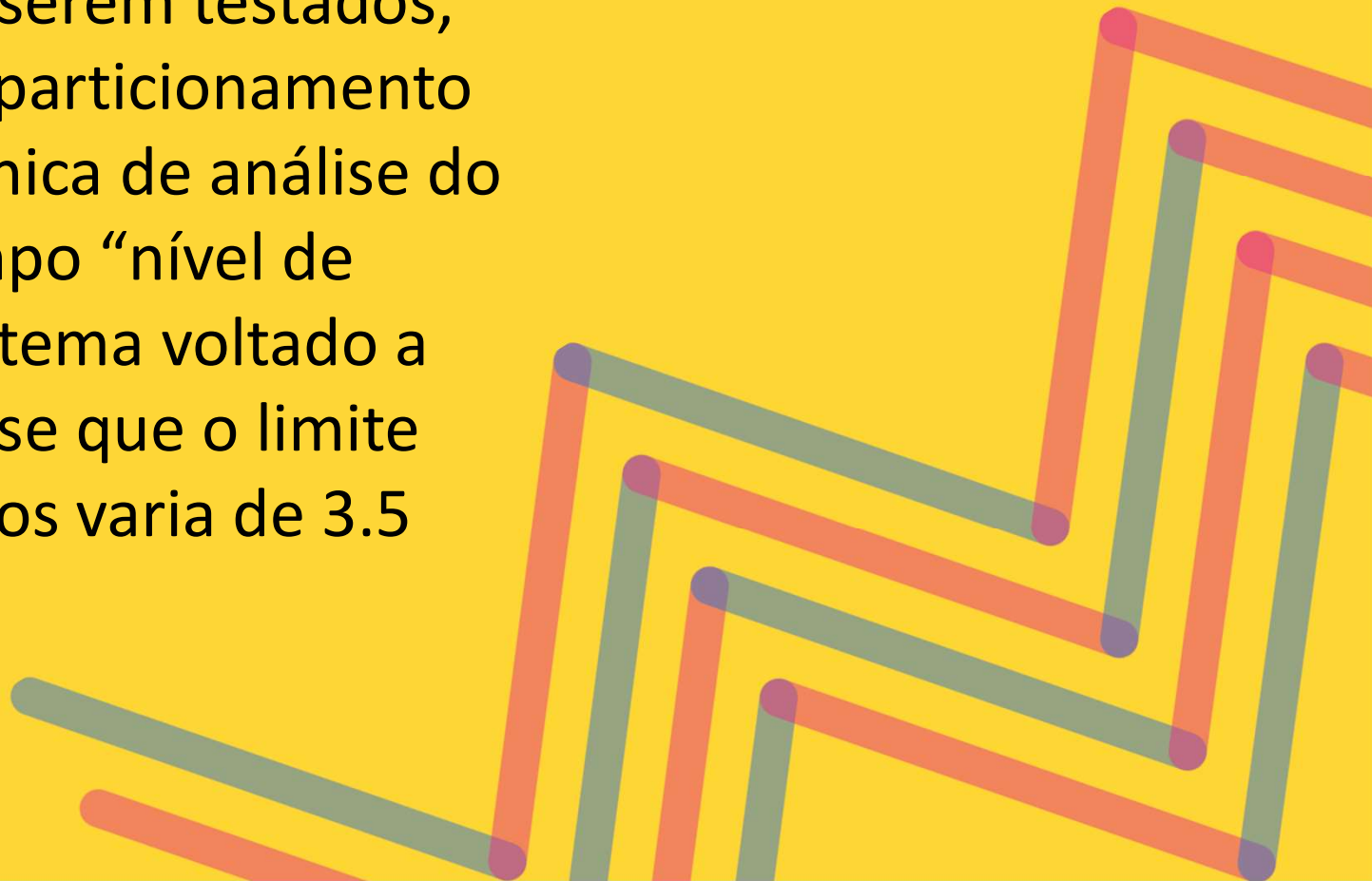
Valor Limite (Idade)	Caso de Teste	Resultado Esperado
-1	CT01	Idade Inválida
0	CT02	Não contratado
1	CT03	Não contratado
15	CT04	Não contratado
16	CT05	Contratado tempo parcial
17	CT06	Contratado tempo parcial
18	CT07	Contratado tempo parcial
19	CT08	Contratado tempo integral
59	CT09	Contratado tempo integral
60	CT10	Contratado tempo integral
61	CT11	Não contratado
98	CT12	Não contratado
99	CT13	Não contratado
100	CT14	Idade Inválida



## Exercício 1



Apresente os valores a serem testados, utilizando a técnica de particionamento de equivalência e a técnica de análise do valor limite, para o campo “nível de combustível” de um sistema voltado a área automotiva, sabe-se que o limite aceitável para os veículos varia de 3.5 litros até 51.1 litros



## Exercício 2



- Utilize a técnica do caminho básico para definir quantos caminhos devem ser testados para o trecho de código abaixo. Apresente o grafo e os caminhos a serem testados.

```
1 início
  leia nro
2  enquanto nro ≠ 0
3    se nro > 0
3.1    nro = pega_inteiro(nro / 2)
        escreva nro
4    senão
        escreva mensagem de erro
5  fim-se
6  fim-enquanto
7 fim
```