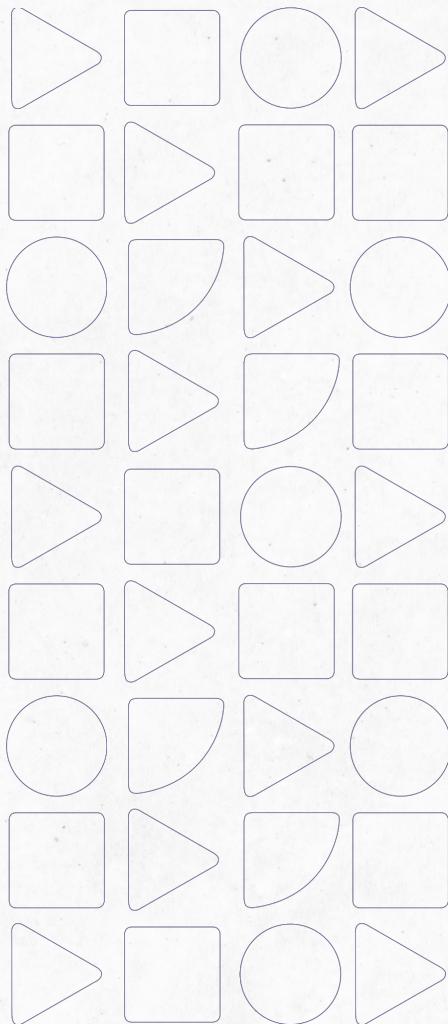


## Testes estruturais

**Disciplina:** Qualidade de *Software*



## Conteúdos:

Testes estruturais.

## Habilidade(s):

Compreender como funcionam os testes estruturais em problemas aplicados.

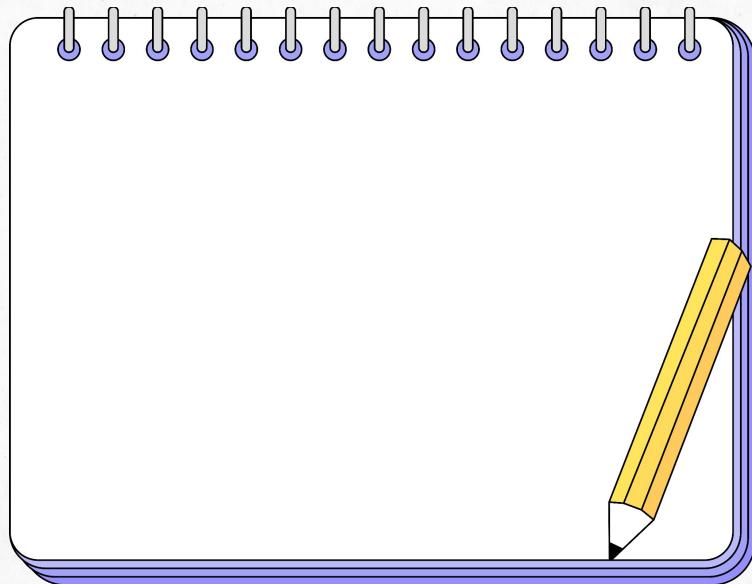
# Bloco 1

---

Entendendo a importância dos testes estruturais.

# Vamos preencher juntos?

Qual é a importância da profissão responsável pela qualidade de *software*?



A importância dos testes para o desenvolvimento de *software* é amplamente reconhecida e abrange uma série de benefícios e razões essenciais.

Na aula de hoje, exploraremos os **testes estruturais**.



# Testes estruturais

Eles se concentram em avaliar a estrutura interna do código-fonte, procurando atingir determinados objetivos de cobertura.

Três objetivos comuns de cobertura são mencionados:

1

**instrução e cobertura:** garantir que todas as instruções do programa sejam executadas pelo menos uma vez durante os testes;

2

**cobertura de filiais:** certificar-se de que todas as ramificações no código sejam percorridas durante os testes, inclusive as vazias;

3

**cobertura de condição:** certificar-se de que todas as combinações possíveis de valores lógicos verdadeiro e falso sejam testadas.

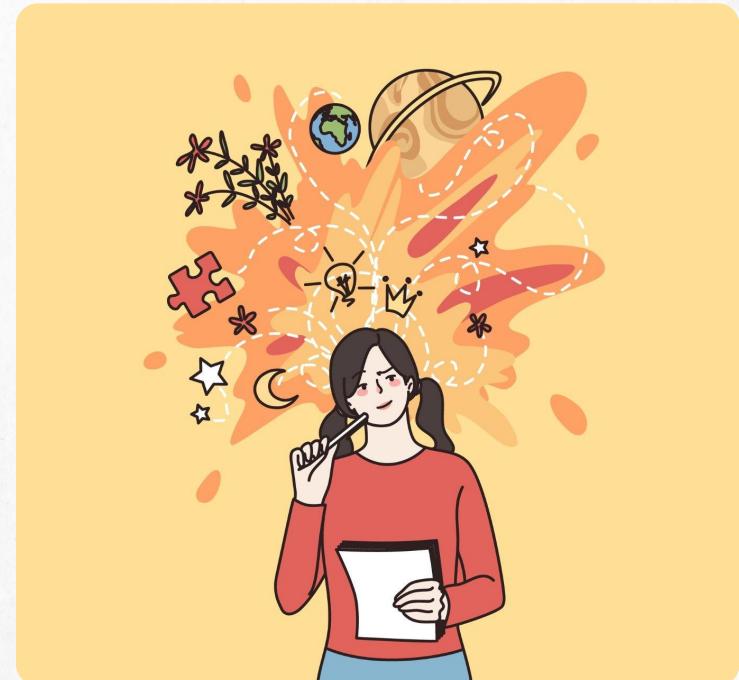
# Métodos e aplicação dos testes estruturais

São geralmente aplicados a testes unitários, ou seja, testes que avaliam unidades individuais de código (como funções ou métodos). Não há uma exigência de teste estrutural específica para testes de integração ou testes de sistema.



## Objetivos do projeto do caso de teste

O objetivo fundamental é definir um conjunto de dados que assegure que cada instrução ou ramo do código seja executado pelo menos uma vez durante os testes.



## Estratégia de teste evolutivo

É aplicado para atingir os objetivos de teste estrutural.

Isso envolve a geração automática de dados de teste, alcançando a melhor cobertura possível dos critérios de teste.

Os testes evolutivos buscam dividir o teste em **objetivos parciais**, cada um representando uma estrutura de programa específica que deve ser executada para alcançar a cobertura desejada.



# Funções objetivo

Para cada objetivo parcial, uma função objetivo é formulada.

Essa função é dividida em duas partes:

## Nível de aproximação:

avalia o quanto próximo um teste está de alcançar o objetivo parcial.

## Cálculo da distância local:

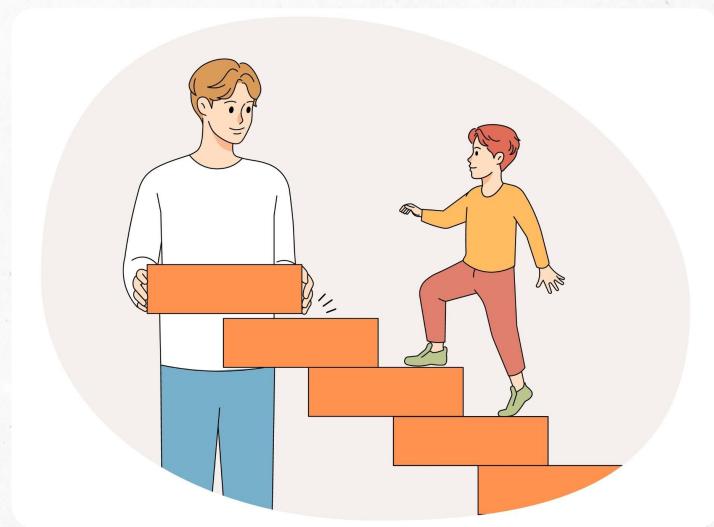
calcula a distância entre um teste específico e a estrutura de programa.

# Seleção e combinação de indivíduos

São representados da seguinte forma:

Os testes que estão mais próximos de alcançar o objetivo parcial são selecionados como “pais”.

Os testes selecionados como “pais” são combinados, gerando novos testes “descendentes”.



# Bloco 2

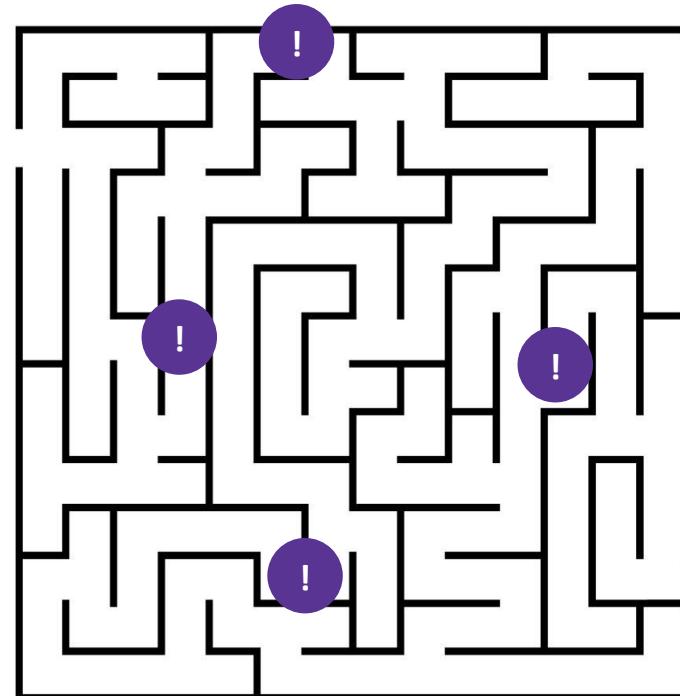
---

Compreendendo o cálculo do nível de aproximação.

# Labirinto do conhecimento

Responda as perguntas do caminho para chegar ao final.

- Qual é o foco dos testes estruturais no desenvolvimento de *software*?
- Por que os testes estruturais são mais comuns em testes unitários no *software*?
- Qual é o objetivo principal ao projetar casos de teste no contexto dos testes estruturais?
- O que significa dividir o teste em objetivos parciais?





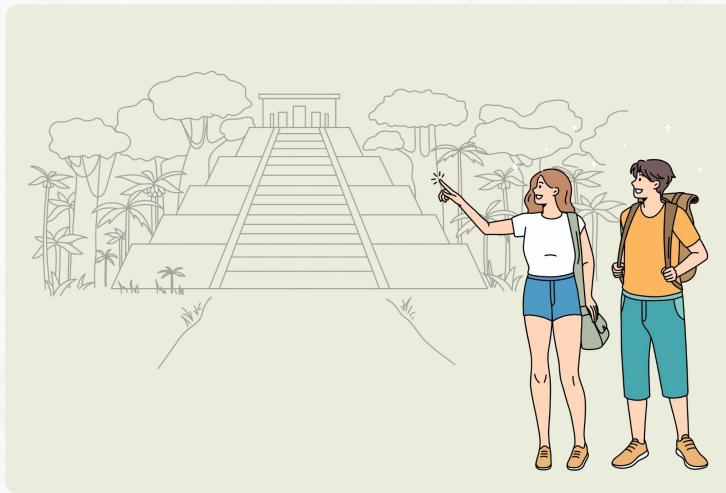
## Parabéns!

Baseando-se nos seus conhecimentos, você conseguiu atingir o objetivo final. Assim como essa dinâmica, o próximo tema se assemelha a um labirinto. Vamos explorá-lo?

## Imagine o seguinte cenário

Você está explorando um labirinto em busca de um tesouro. Nele, o nível de aproximação é a distância entre onde você está e o tesouro.

Assim, quanto mais perto você está do tesouro, **maior é o nível de aproximação**.



## Mais próximo do caminho final

Considere que você está tentando alcançar um ponto final dentro do labirinto.

Ele tem bifurcações, ou nós de ramificação. Nelas, você pode escolher diferentes caminhos. No final do labirinto, há uma porta que você deve alcançar. Ela é a estrutura desejada do programa.



# Caminho percorrido

A seguir, veja os passos que você realizou para chegar ao final.

- Você está no nó de partida, o nó A. Ou seja, é como se você estivesse no início do labirinto;
- À medida que você avança, cada bifurcação (nó de ramificação) encontrada aumenta o nível de aproximação entre você e a porta final;
- Você escolhe o caminho da esquerda, chamado de nó B. Agora, você está a um nível de aproximação um da porta final;
- Você chega ao nó C (nível dois de aproximação) e depois ao nó D (nível três de aproximação);
- Ao chegar ao nó E, você escolhe o caminho errado e se afasta da porta final (nível quatro de aproximação).



## É assim que ocorre o cálculo do nível de aproximação!

No contexto dos testes de *software*, cada caminho que o programa pode seguir é como um caminho no labirinto.

O nível de aproximação mede o quanto perto um teste está de alcançar a parte do programa que precisa ser testada.

Isso ajuda os testadores a entenderem quais partes do código estão sendo cobertas pelos testes e quanto próximos estão de atingir os objetivos pretendidos.

# Bloco 3

---

O que é o cálculo de distância local?

## Duas verdades e uma mentira



Identifique quais das alternativas abaixo são as verdadeiras e qual delas é a falsa.

Sobre o cálculo do nível de aproximação, é correto afirmar que:

- a** o nível de aproximação mede a distância entre um teste e a estrutura do programa.
- b** o cálculo do nível de aproximação não leva em consideração as condições lógicas.
- c** o cálculo do nível de aproximação considera as ramificações do programa.

## Duas verdades e uma mentira



Identifique quais das alternativas abaixo são as verdadeiras e qual delas é a falsa.

Sobre o cálculo do nível de aproximação, é correto afirmar que:

- a o nível de aproximação mede a distância entre um teste e a estrutura do programa.
- b o cálculo do nível de aproximação não leva em consideração as condições lógicas.
- c o cálculo do nível de aproximação considera as ramificações do programa.

## Vamos observar o exemplo?

Imagine que você está brincando com um jogo de escolha. Nesse jogo, você está em uma sala com várias portas e precisa escolher a certa para encontrar um tesouro.



# Condições das portas

A primeira porta requer que você tenha duas chaves idênticas para abri-la. Essas chaves são chamadas “X” e “Y”.

A segunda porta é um pouco diferente. Ela exige que você tenha uma chave chamada “A” ou outra chave chamada “B”.

Se X e Y tiverem exatamente o mesmo valor, você pode abrir a porta e chegar ao tesouro.



Se, pelo menos, uma das chaves for a correta, você conseguirá alcançar o tesouro.

# Distância local: o que isso significa?

Agora, vamos pensar na distância local como uma forma de medir o quanto perto você está de cumprir a condição de cada porta. Quanto mais próximo você estiver, menor será a distância local.

Porta 1

Imagine que você tem duas chaves:  $X = 5$  e  $Y = 5$ . Agora, para saber o quanto perto você está de abrir a porta, você calcula  $|X - Y| = |5 - 5| = 0$ . Isso significa que a distância local é 0, porque suas chaves são idênticas. Assim, você está bem perto de cumprir a condição da porta.

Porta 2

Agora, considere que você tem as chaves:  $A = 3$  e  $B = 7$ . Para cada chave, você calcula a distância em relação à condição: a distância para A é  $|3 - 2| = 1$  e a distância para B é  $|7 - 2| = 5$ . Como você escolheu a chave que está mais perto da condição, a distância local é 1.

## Conclusão

A distância local é como medir o quanto perto você está de cumprir as condições de uma porta no jogo.

Quanto menor a distância, mais perto você está de acertar a resposta correta. Isso ajuda a entender quais chaves são as melhores para escolher e quais estão mais perto de abrir a porta, te permitindo encontrar o tesouro.

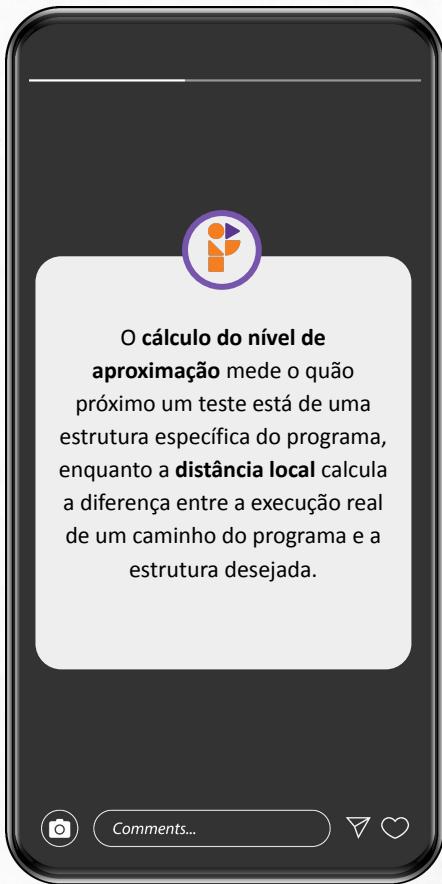


# Bloco 4

---

Conhecendo o teste da caixa branca.





## Teste da caixa branca

É uma abordagem eficaz para encontrar e resolver problemas em *softwares*, identificando erros antes que eles causem problemas reais.

Nesse tipo de teste, também conhecido como **análise de caixa branca**, os testadores inserem dados no sistema e observam como ele processa esses dados para gerar a saída desejada.



# Vantagens do teste da caixa branca

Esse método pode ser usado para aplicativos de serviços da web.



Todos os caminhos independentes dentro de um módulo são testados.



Todas as decisões lógicas são exploradas.



*Loops* são testados em seus limites.



As estruturas de dados internas são examinadas para garantir a sua validade.



Erros em partes de código ocultas são descobertos.



É uma abordagem sistemática, focando na execução equivalente.

## Desvantagens do teste da caixa branca

Esse tipo de teste raramente é prático para a depuração em grandes sistemas e redes.

X

Pode perder casos não abordados explicitamente no código.

X

Requer testadores qualificados e com conhecimento do código.

X

É difícil examinar todos os pedaços de código, podendo haver erros ocultos não detectados.

# Bloco 5

---

É hora de praticar!

## Vamos praticar?

Dividam-se em quatro grupos e analisem a situação de caso que o professor disponibilizará nas folhas de ofício que serão entregues.

A partir dessa situação, expliquem como podem ser realizados os **testes estruturais**, incluindo o **cálculo do nível de aproximação e da distância local**.

- Cada grupo discutirá os principais pontos do material de leitura e preparará uma breve apresentação, destacando os conceitos mais importantes e as suas interações;
- Cada equipe terá cinco minutos para apresentar as conclusões obtidas.



# Bloco 6

---

Vamos apresentar!

## Discussões e reflexões





## Hora dos *feedbacks*!

O professor avaliará a compreensão dos alunos considerando as apresentações em grupo, a participação nas discussões e as respostas dadas durante perguntas feitas na atividade.

# Fechamento

Que bom!

Que pena...

Que tal?

# Referências Bibliográficas

PROZ EDUCAÇÃO. *Apostila de Qualidade de software*. 2023.