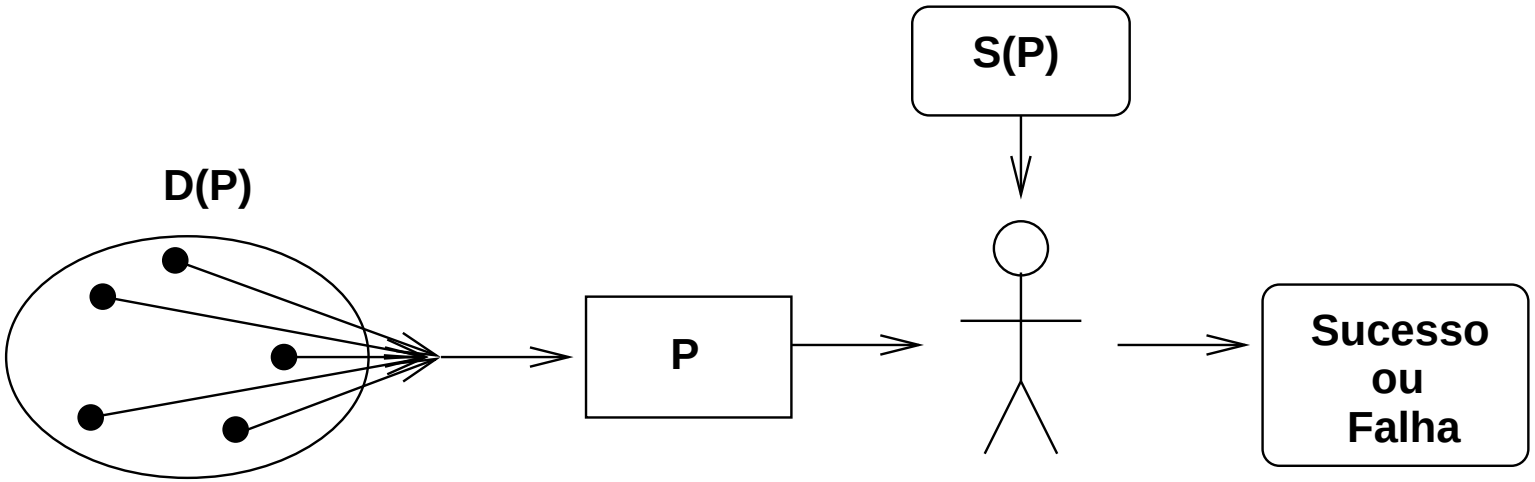
SSC5877 – Validação Verificação e Teste de Software

Teste Funcional

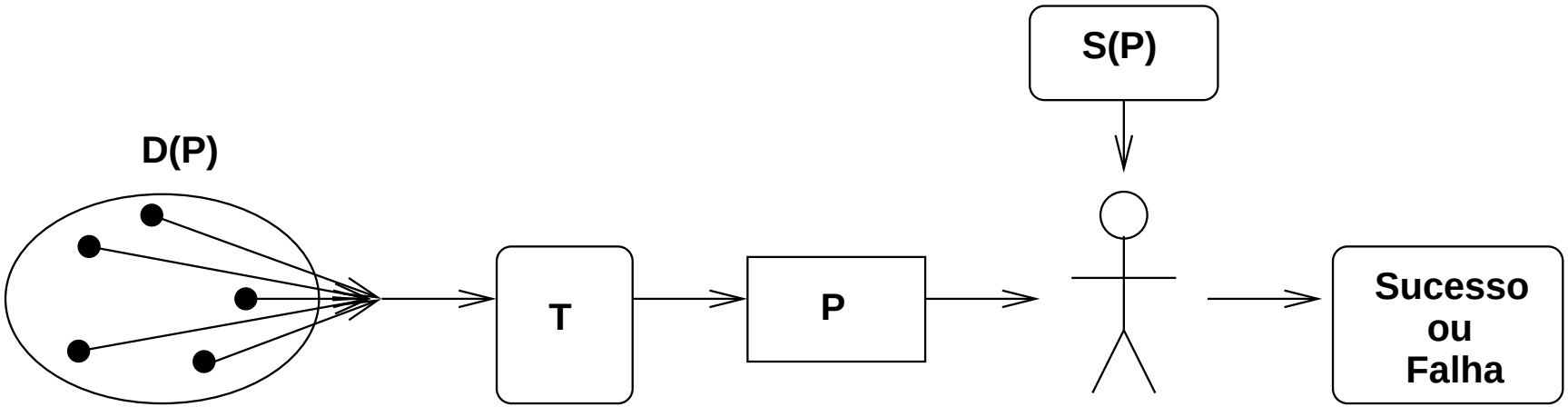
Prof. Marcio E. Delamaro

delamaro@icmc.usp.br

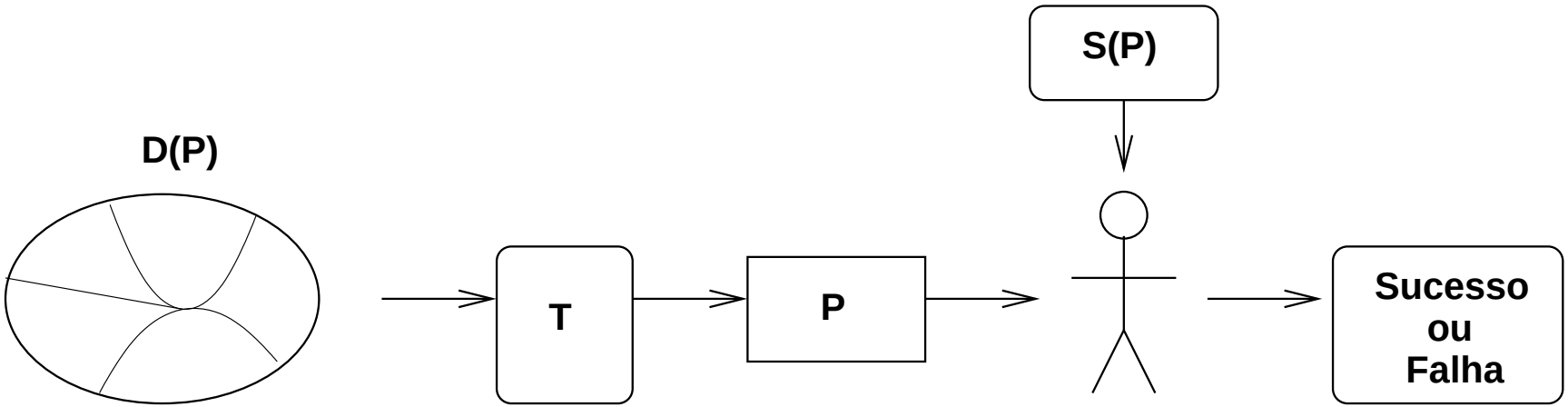
Teste exaustivo



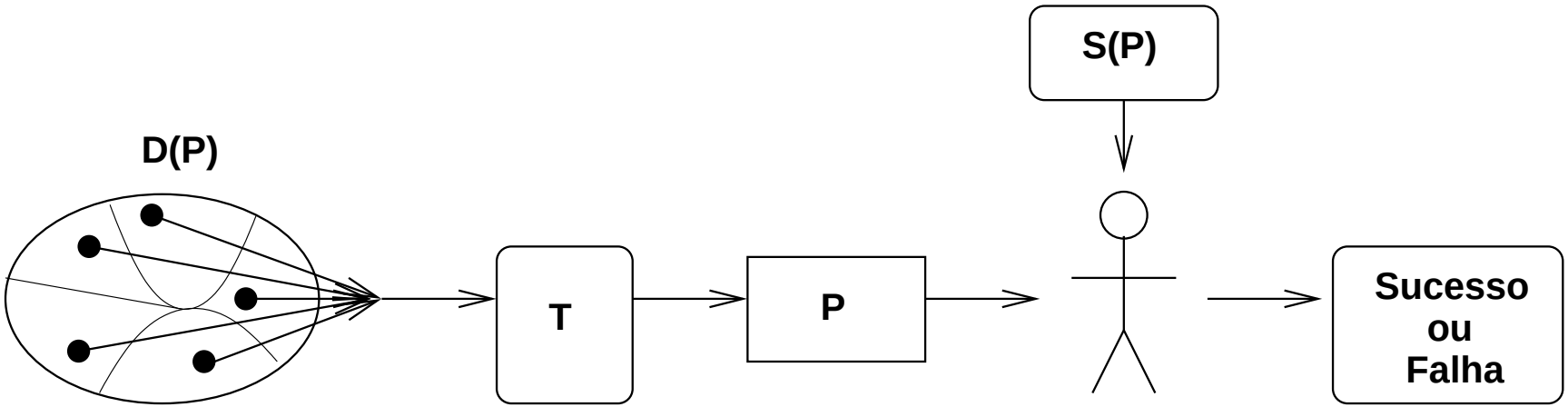
Teste randômico



Teste de subdomínios



Teste de subdomínios



Como computar domínios

- Teste funcional

Como computar domínios

- Teste funcional
- Teste estrutural

Como computar domínios

- Teste funcional
- Teste estrutural
- Teste baseado em defeitos

Teste funcional

- O que diferencia as técnicas de teste é a forma de particionar o domínio
- No caso de teste funcional, apenas a especificação é utilizada
- Quando o testador seleciona casos de teste dos subdomínios, o conjunto de testes resultante é uma boa representação de todo o domínio

Características

- Conjunto de teste é pequeno
- Número de subdomínios é pequeno
- Uma vez definidos os subdomínios, pode-se assumir, com alguma segurança, que qualquer elemento da classe pode ser considerado
- Se um elemento detectar um defeito, qualquer outro também detecta; se não detectar, os outros também não detectam
- Particionar o domínio de saída também é valido

Histórico

- Década de 70
- Métodos para especificar sistemas como Análise e Projeto Estruturados
- Eram mencionados, embora não diretamente, aspectos de validação do sistema com relação à satisfação dos seus requisitos funcionais

Critérios

- Diversos critérios são definidos dentro desta técnica
- Particionamento de Equivalência, Análise do Valor Limite, Grafo Causa-Efeito e Error-Guessing. Além desses, também existem outros, como, por exemplo, Teste Funcional Sistemático, Syntax Testing, State Transition Testing e Graph Matrix

Vantagens

- Pode ser utilizado em todas as fases de teste.
- Independente do paradigma de programação utilizado.
- Eficaz em detectar determinados tipos de defeitos.
- Funcionalidade ausente, por exemplo.

Desvantagens

- Dependente de uma boa especificação o que, em geral, não é bem feito.
- Não é possível garantir que partes essenciais ou críticas do software sejam executadas
- Ruim quando se tem entradas simples mas processamento complexo

Cadeia de caracteres

O programa solicita do usuário um inteiro positivo no intervalo entre 1 e 20 e então lê uma cadeia de caracteres desse comprimento. Após isso, o programa solicita um caractere e retorna a posição na cadeia em que o caractere é encontrado pela primeira vez ou uma mensagem indicando que o caractere não está presente na cadeia. O usuário tem a opção de procurar vários caracteres

Cadeia de caracteres

C:\Users\delamaro\Documents\AAATemp\Nova Pasta\Untitled1.exe

```
Digite o tamanho: 5
Digite o string: abcde
Caractere a procurar: d
Encontrado na posicao 4
Deseja procurar outro? s
Caractere a procurar: x
Nao encontrado
Deseja procurar outro? n
```

Particionamento em classes de equivalência

- Duas etapas
- Identificar classes de equivalência
- Definir os casos de teste para cobrir essas classes

Identificar classes de equivalência

- Identificar condições de entrada relevantes
- Particionar cada uma em dois ou mais grupos
- Para ajudar na identificação das partições, pode-se observar a especificação procurando termos como “intervalo” e “conjunto” ou palavras similares que indiquem que os dados são processados da mesma forma

Classes de equivalência

Condição de entrada	Classes de equivalência válidas	Classes de equivalência inválidas

Identificando as entradas

- Tamanho da cadeia (**T**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres (**CC**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres (**CC**)
- O caractere a ser procurado (**C**)

Identificando as entradas

- Tamanho da cadeia (**T**)
- A cadeia de caracteres (**CC**)
- O caractere a ser procurado (**C**)
- A opção por procurar mais caracteres (**O**)

Identificar as classes

- Algumas guidelines podem ser estabelecidas
- Uma condição de entrada estabelece um intervalo de valores
- *“O número de ítens pode variar de 1 a 999´´*
- Uma classe válida
- $1 \leq$ número de ítens ≤ 999
- Duas classes não válidas
- $1 >$ número de ítens
- $999 <$ número de ítens

Guidelines

- A condição de entrada estabelece uma quantidade de valores
- *“Para um automóvel, de 1 a seis proprietários podem ser relacionados”*
- Uma classe válida
- De um a seis proprietários
- Duas classes inválidas
- Nenhum proprietário
- Mais do que seis proprietários

Guidelines

- A condição de entrada especifica um conjunto de valores que (acredita-se) devem ser tratados de maneiras diversas.
- *"Tipo de veículo deve ser: ônibus, caminhão, taxi, passageiro ou motocicleta"*
- Uma classe válida para cada valor
- Uma classe não válida, por exemplo: trailer

Guidelines

- Condição de entrada determina uma condição do tipo “tem que ser”
- “Primeira caractere do identificador tem que ser uma letra”
- Uma classe válida
- Primeiro caractere é uma letra
- Classe não válida
- Caractere é um dígito

Guidelines

- Se existe alguma razão para acreditar que o programa não trata elementos de uma classe de maneira uniforme, a classe deve ser dividida em classes menores

Cadeia: T

- Tamanho da cadeia de caracteres

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo
- Uma classe válida: de 1 a 20

Cadeia: T

- Tamanho da cadeia de caracteres
- Se enquadra na primeira guideline: valores num intervalo
- Uma classe válida: de 1 a 20
- Duas não válidas: menor que 1, maior que 20

Cadeia: CC

- Valor da cadeia de caracteres

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde
- 8y4e*

Cadeia: CC

- Valor da cadeia de caracteres
- O valor em si não determina comportamentos diferentes do programa, apenas o seu tamanho
- abcde
- 8y4e*
- Assim, não é necessário usar essa variável como condição de entrada

Cadeia: O

- Opção de continuar ou não

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis
- Duas classes válidas: **s** ou **n**

Cadeia: O

- Opção de continuar ou não
- Se enquadra na terceira guideline: conjunto de valores possíveis
- Duas classes válidas: **s** ou **n**
- Uma classe não válida: **w**, por exemplo

Cadeia: C

- Caractere a ser procurado

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline
- Duas classes válidas: C pertence à string e C não pertence à string

Cadeia: C

- Caractere a ser procurado
- Se enquadra também na terceira guideline
- Duas classes válidas: C pertence à string e C não pertence à string
- Nenhuma classe não válida

Cadeia: classes de equivalência

Variável de entrada	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 \leq T \leq 20$ (1)	$T < 1$ (2) e $T > 20$ (3)
O	S (4) N (5)	Outro (6)
C	Pertence (7) Não pertence (8)	

Computar casos de teste

- Próximo passo é usar as classes para computar os casos de teste necessários
- Regras a seguir:
- Atribuir um número para cada classe
- Definir casos de teste para cobrir o maior número de classes válidas possíveis, até que todas as classes válidas sejam cobertas
- Para cada classe não válida, desenhar um caso de teste específico

Comentário

- Entradas inválidas podem mascarar ou exptrapolar outras entradas inválidas
- Por exemplo: **T** = 34 e **O** = w provavelmente não irá exercitar a classe inválida para a condição **O**

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 3; Não encontrado

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- **T** = -3

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- **T** = -3
- Saída: Tamanho inválido

Cadeia: casos de teste

- **T** = 3 **CC** = abc **C** = c **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 3; Não encontrado
- Classes cobertas: 1, 4, 5, 7, 8 (todas as válidas)
- **T** = -3
- Saída: Tamanho inválido
- Classe coberta: 2

Cadeia: casos de teste

- **T** = 34

Cadeia: casos de teste

- **T** = 34
- Saída: Tamanho inválido

Cadeia: casos de teste

- **T** = 34
- Saída: Tamanho inválido
- Classe coberta: 3

Cadeia: casos de teste

- **T** = 34
- Saída: Tamanho inválido
- Classe coberta: 3
- **T** = 3 **CC** = abc **C** = c **O** = w

Cadeia: casos de teste

- **T** = 34
- Saída: Tamanho inválido
- Classe coberta: 3
- **T** = 3 **CC** = abc **C** = c **O** = w
- Saída: Responda com s ou n

Cadeia: casos de teste

- **T** = 34
- Saída: Tamanho inválido
- Classe coberta: 3
- **T** = 3 **CC** = abc **C** = c **O** = w
- Saída: Responda com s ou n
- Classe coberta: 6

Exercício

- Poderíamos pensar numa condição de entrada como: “Número de vezes que um caractere é procurado no string”
- Nesse caso, como ficaria a tabela de classes? Modifique-a.
- Defina os casos de teste para cobrir essas novas classes.

Análise de valor limite

- É mais proveitoso explorar condições limites
- Aquelas que estão sobre, acima e abaixo dos limites das classes de equivalência
- Assim, em vez de selecionar um caso de teste qualquer, deve-se tomar um ou mais casos de teste de modo que cada limitante da classe seja testado
- Além disso, deve-se considerar também o domínio de saída para derivar casos de teste (classes de equivalência de saída)

Guidelines

- Dizem que é difícil definir um “livro de receitas”

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01
- Se especifica uma quantidade de valores, escolher casos de teste nos limites, uma unidade acima e uma abaixo

Guidelines

- Dizem que é difícil definir um “livro de receitas”
- Ainda assim, algumas dicas podem ser observadas
- Se uma condição de entrada define um intervalo de valores: casos de teste nos limites do intervalo e logo além dos limites (casos não válidos)
- Por exemplo se o intervalo for -1.00 a 1.00, devemos tomar -1.00, 1.00, -1.01 e 1.01
- Se especifica uma quantidade de valores, escolher casos de teste nos limites, uma unidade acima e uma abaixo
- Por exemplo se um arquivo de entrada deve conter de 1 a 255 registros, casos de teste devem contemplar 0, 1, 255 e 256 registros

Guidelines

- Aplicar a 1a. regra para o domínio de saída

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, $0,00 e no máximo $1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, $0,00 e no máximo $1.165,25
- Aplicar entrada que cause $0,00 de dedução e outra que cause $1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, $0,00 e no máximo $1.165,25
- Aplicar entrada que cause $0,00 de dedução e outra que cause $1.165,25
- Tente achar casos de teste que cause dedução negativa ou maior que $1.165,25

Guidelines

- Aplicar a 1a. regra para o domínio de saída
- Por exemplo, um programa deve calcular a dedução que uma empresa tem sobre um determinado imposto
- Diz a especificação que essa dedução é de, no mínimo, $0,00 e no máximo $1.165,25
- Aplicar entrada que cause $0,00 de dedução e outra que cause $1.165,25
- Tente achar casos de teste que cause dedução negativa ou maior que $1.165,25
- Nem sempre é possível achar casos de teste além dos limites

Guidelines

- Aplicar 2a. regra para saídas

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta
- Achar casos de teste que façam com que 0, 1 e 4 abstracts sejam mostrados

Guidelines

- Aplicar 2a. regra para saídas
- Por exemplo, um sistema de recuperação de informação mostra no máximo 4 abstracts para cada consulta
- Achar casos de teste que façam com que 0, 1 e 4 abstracts sejam mostrados
- Tentar caso de teste que erroneamente mostre 5 registros

Guidelines

- Se a entrada ou a saída for um conjunto ordenado (um arquivo sequencial ou uma lista ligada), dar atenção ao primeiro e último elemento

Guidelines

- Se a entrada ou a saída for um conjunto ordenado (um arquivo sequencial ou uma lista ligada), dar atenção ao primeiro e último elemento
- Use a sua criatividade para definir outras condições limites

Cadeia: classes

Variável de entrada	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 \leq T \leq 20$ (1)	$T < 1$ (2) e $T > 20$ (3)
O	S (4) N (5)	Outro (6)
C	Pertence (7) Não pertence (8)	

Cadeia: classes

Variável de E/S	Classes de equivalência válidas	Classes de equivalência inválidas
T	$1 \leq T \leq 20$ (1)	$T < 1$ (2) e $T > 20$ (3)
O	S (4) N (5)	Outro (6)
C	Pertence (7) Não pertence (8)	
Posição	$1 \leq pos \leq 20$ (9)	$pos < 1$ (10) $pos > 20$ (11)

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 1; Não encontrado

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 1; Não encontrado
- **T** = 20 **CC** = abcdefghijklmnopqrst **C** = t **O** = n

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 1; Não encontrado
- **T** = 20 **CC** = abcdefghijklmnopqrst **C** = t **O** = n
- Saída: Encontrado na posição 20

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 1; Não encontrado
- **T** = 20 **CC** = abcdefghijklmnopqrst **C** = t **O** = n
- Saída: Encontrado na posição 20
- **T** = 0

Cadeia: casos de teste

- **T** = 1 **CC** = a **C** = a **O** = s **C** = k **O** = n
- Saída: Encontrado na posição 1; Não encontrado
- **T** = 20 **CC** = abcdefghijklmnopqrst **C** = t **O** = n
- Saída: Encontrado na posição 20
- **T** = 0
- Saída: Tamanho inválido

Cadeia: casos de teste

- **T** = 21

Cadeia: casos de teste

- **T** = 21
- Saída: Tamanho inválido

Cadeia: casos de teste

- **T** = 21
- Saída: Tamanho inválido
- **T** = 3 **CC** = abc **C** = c **O** = w

Cadeia: casos de teste

- **T** = 21
- Saída: Tamanho inválido
- **T** = 3 **CC** = abc **C** = c **O** = w
- Saída: Responda com s ou n