Aula 5 – Teste Funcional

Prof. Paulo R. Nietto

Universidade
Anhembi Morumbi
LAUREATE INTERNATIONAL UNIVERSITIES

Objetivos

- Apresentar Teste Funcional ou Caixa Preta

- Exercícios

Técnicas de Teste

- Testes são indicadores de qualidade do produto mais do que meios de detecção e correção de erros
- Quanto maior o número de defeitos detectados maior o número de defeitos não detectados
- A ocorrência de um número anormal de defeitos em uma bateria de testes indica uma provável necessidade de redesenho dos itens testados

Técnicas de Teste

- Classificam-se em:
  - Testes da Caixa-Preta
  - Testes da Caixa-Branca

Testes da Caixa Preta ou Teste Funcional

- Baseado na **especificação do software**
- Não há preocupação com a lógica interna

Teste da Caixa Preta – Black Box

- Verifica as funcionalidades do software
- O programa é visto como uma caixa preta
- Nenhum conhecimento da sua estrutura interna ou comportamento é assumida
- Casos de testes são gerados tomando como referência requisitos e especificações do software

Teste da Caixa Preta – Black Box

- Para cada valor de entrada verifica-se à saída correspondente esperada
- A geração dos dados de teste pode ser efetuada paralelamente as fases de design e implementação
- Pode ser iniciado assim que o código seja completado

Teste da Caixa Preta – Black Box

- Técnicas de caixa preta sempre devem ser utilizadas durante os testes:
  - Objetivo dos testes da caixa preta é verificar as funcionalidades do componente em teste

Exemplos de erros encontrados no Teste da Caixa Preta

- Funções omitidas ou incorretas
- Erros de interface
- Erros de estruturas de dados
- Erros de acesso a Banco de Dados
- Erros de comportamento ou desempenho
- Etc.

Teste da Caixa Preta – Black Box

- Situação Ideal:
  - Testar o software com todas as combinações possíveis de entradas
  - Demanda muito tempo e dinheiro
- Solução:
  - Fazer uma seleção das possibilidades de casos de teste
  - Existem métodos que ajudam fazer esta seleção

Técnicas de Teste da Caixa Preta

- Partição de Equivalência
- Análise do Valor Limite
- Tabela de Decisão
- Teste de Transição de Estados
- Teste de Caso de Uso
- Etc.

Partição de Equivalência

- Entradas são classificadas em grupos (partições/classes) que tenham um comportamento comum
- Assume-se que o software terá o mesmo comportamento para todo o grupo informado
- Aplicável a um grupo de valores válido ou inválidos (que devem ser rejeitados pelo software)

Partição de Equivalência

- Uma classe de equivalência representa um conjunto de estados válidos e inválidos para condições de entrada

- Uma condição de entrada pode ser:
  - Um valor numérico
  - Um intervalo de valores
  - Um conjunto de valores relacionados
  - Uma condição booleana

Gerando classes de equivalência

1. Identificar o domínio de todas as possíveis entradas
2. Dividir este domínio de dados em classes de equivalência
3. Gerar classes de equivalências de entradas corretas
4. Gerar classes de equivalência de entradas incorretas

Diretrizes para definição das classes de equivalência

- **Intervalo**
  - são definidas uma classe válida e duas inválidas
- **Valor específico**
  - são definidas uma classe válida e duas inválidas
- **Membro de um conjunto**
  - são definidas uma classe válida e uma inválida
- **Entrada for booleana**
  - são definidas uma classe válida e uma inválida

Exemplo

1. Definir casos de teste para o salário que deve ser um valor entre R$ 1501 e R$ 37000

   1 classe válida (1600)
   2 classes inválidas (37250, 1400)

2. Definir casos de teste para um programa que aceita como entrada números inteiros maiores 10.000 e menores que 99.999

   1 classe válida (11.000)
   2 classes inválidas (9.000, 110.000)

Análise do valor limite

- O comportamento do software nos limites de uma partição de equivalência é onde ocorre a maior probabilidade de se encontrar problemas

- Mal entendidos nos requisitos podem gerar erro na definição de uma partição
  - EX: idade deve ser maior que 10 anos
    - X>= 10 ou X > 10????

Análise do valor limite

- Casos de teste que exploram as fronteiras tem uma maior probabilidade de detectar os erros

- É usada em conjunto com partição por equivalência

Análise do valor limite

- Um valor limite é um valor de entrada ou de saída que está na fronteira de uma partição de equivalência ou na menor distância de cada lado dela

Diretrizes definir casos de teste usando análise de fronteira para um intervalo

- Verifica as fronteiras/limites:
  - Os valores limites exatos (2 valores)
  - Valores adjacentes (2 fora do limite)
  - Usar o menor incremento possível

- Sugestão:
  - Dois casos válidos para ambos extremos
  - Dois casos inválidos para além dos limites do intervalo

Exemplo – Considerando o menor incremento possível

1. Definir casos de teste para o salário que deve ser um valor entre R$ 1501 e R$ 37000

2 classes válida (1501 , 37.000)
2 classes inválidas (1500, 37001)

Caso de Teste

Id Teste	Entrada	Resultado esperado
1	Informar o valor 1501	Valor válido
2	Informar o valor 37.000	Valor válido
3	Informar o valor 1500	Salário inválido
4	Informar o valor 37001	Salário inválido

Exemplo – Considerando o menor incremento possível

2. Definir casos de teste para um programa que aceita como entrada números inteiros maiores ou iguais 10.000 e menores que 99.999

Caso de Teste

Id Teste	Entrada	Resultado esperado
1	Informar o valor 10000	Valor válido
2	Informar o valor 99.999	Valor válido
3	Informar o valor 9.999	Salário inválido
4	Informar o valor 100.000	Salário inválido

Tabela de Decisão

- Os dois métodos anteriores consideram dados de entrada e saída como independentes

- As dependências entre as entradas e os efeitos nas saídas não são considerados ao gerar os casos de teste

Tabela de Decisão

- Existem requisitos de sistema que possuem condições lógicas (regras de negócio complexas)
- Tabela de Decisão permite que as regras de negócio sejam especificadas de forma compacta
- Devem ser usadas em programas que tenham muitas decisões lógicas

Tabela de Decisão

- Contém as condições que disparam ações no software e os resultados para cada combinação de condições

- Vantagem:
  - Cria combinações de condições que ainda não foram testadas

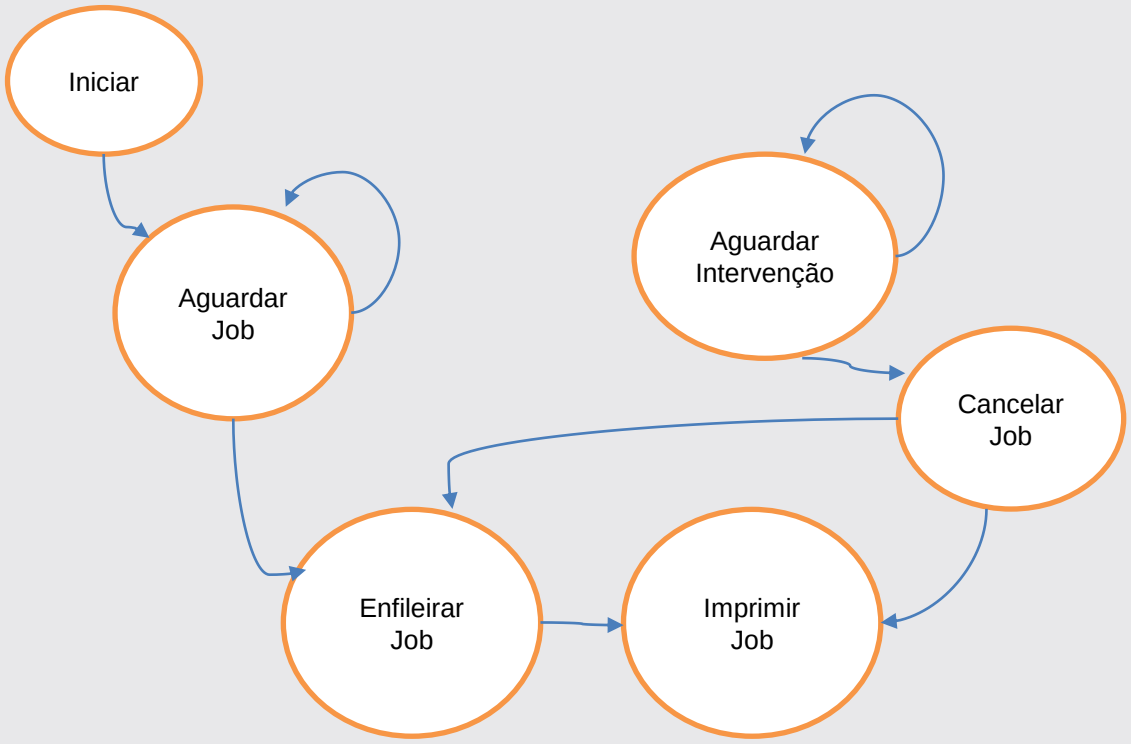
Exemplo - ATM

Condição	1	2	3	4	5
Cartão válido	N	S	S	S	S
Senha válida	-	N	N	S	S
Senha inválida pela terceira vez	-	N	S	N	N
Saldo disponível	-	-	-	N	S
Ação					
Rejeita cartão	S	N	N	N	N
Informar senha novamente	N	S	N	N	N
Apreende cartão	N	N	S	N	N
Escolhe outra opção	N	N	N	S	N
Libera dinheiro	N	N	N	N	S

Teste de Transição de Estados

- As saídas geradas por um sistema dependem da condição atual do sistema ou de seu estado anterior
  - Depende também do histórico de execução ou eventos anteriores
  - Exemplo: Servidor de impressão

Exemplo – Servidor de impressão



Exemplo – Servidor de impressão

- Responde a eventos baseado em seu estado:
  - Aguardando Job
  - Enfileirando Job
  - Imprimindo Job
  - Aguardando intervenção usuário
  - Aguardando intervenção operador

Teste de Transição de Estados

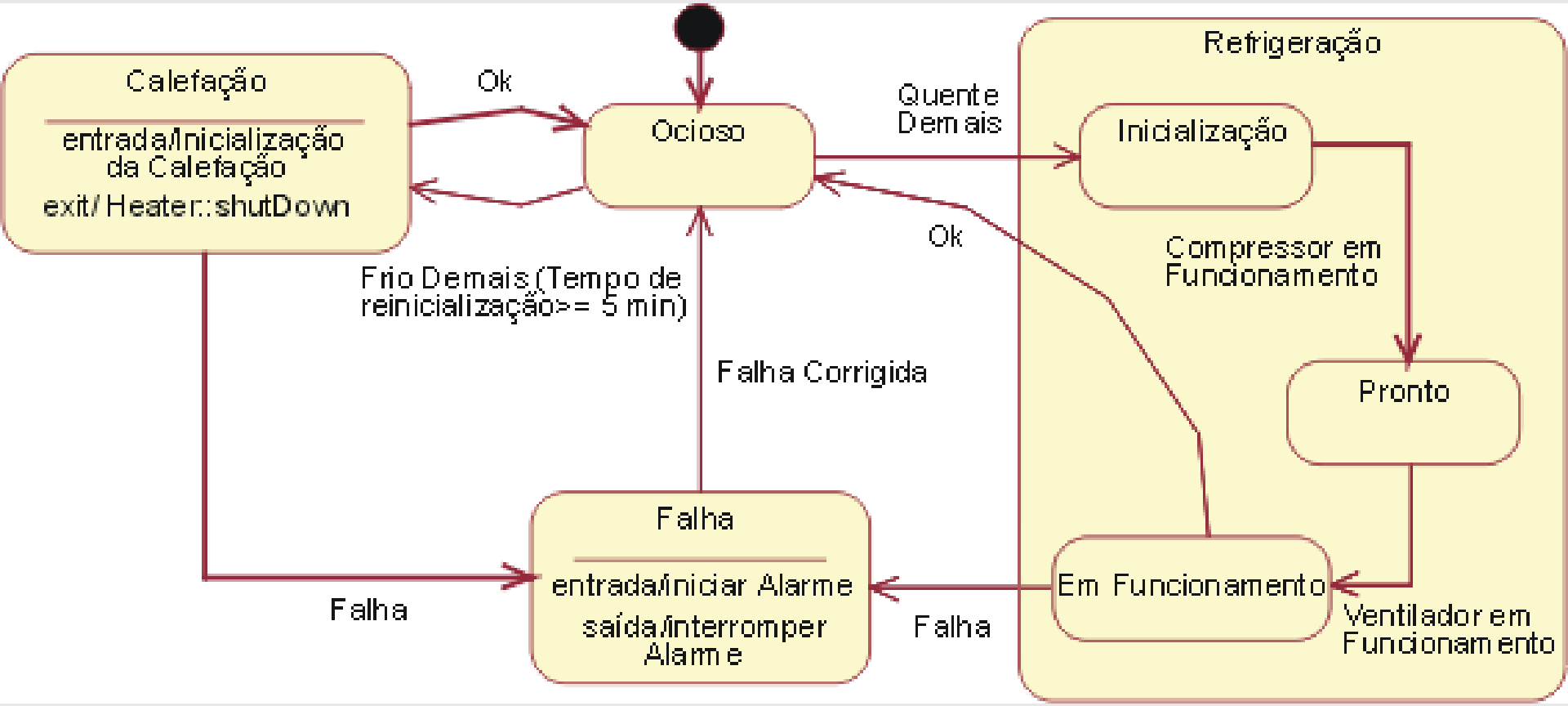
- Não basta definir dados de entrada e saída, deve-se definir transações

- Para cada transação definir:
  - Estado antes da transição
  - Evento que disparou a transição
  - Reação esperada
  - Próximo estado

Utilização dos Teste de Transição de Estados

- Onde a funcionalidade é influenciada pelo estado do componente em teste
  - Softwares industriais embarcados e de automação em geral
  - Testar fluxos de telas (ex: internet)
  - Sistemas Orientados a Objetos

Exemplo – Sistema de Calefação



Lista de idéias de Teste – Sistema de Calefação

- O estado Ocioso recebe o evento Muito Quente
- O estado Ocioso recebe o evento Muito Frio
- O subestado Inicialização do estado Refrigeração recebe o evento Compressor Funcionando
- O subestado Pronto do estado Refrigeração recebe o evento Ventilador Funcionando
- O subestado Funcionando do estado Refrigeração recebe o evento OK
- O subestado Funcionando do estado Refrigeração recebe o evento Falha
- O estado Falha recebe o evento Falha Solucionada
- O estado Calefação recebe o evento OK
- O estado Calefação recebe o evento Falha

Teste de Caso de Uso

- Casos de teste podem ser derivados de casos de uso

- Um caso de uso descreve interações entre os atores e o sistema

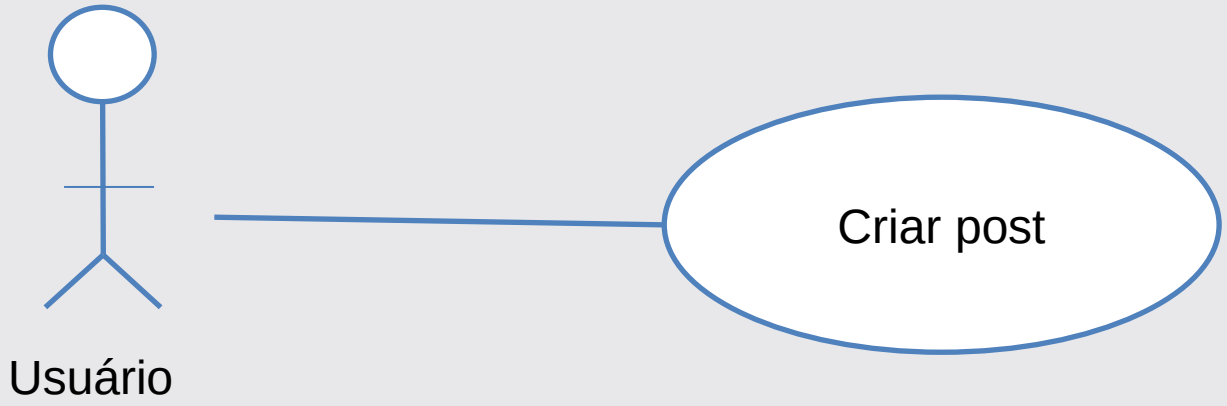
Teste de Caso de Uso

- Para gerar os casos de testes a partir dos casos de uso considera-se:
  - Pré-condições
  - Outras possíveis condições
  - Resultados esperados
  - Pós-condições

Utilização de Teste de Caso de Uso

- Descobrir defeitos no fluxo do processamento durante a utilização do software
- Construir testes de aceite junto aos usuários finais
- Descobrir defeitos de integração de componentes

Teste de Caso de Uso: Exemplo



Teste de Caso de Uso: Exemplo

Caso de Uso: Criar post

Pré-condição: Usuário logado no sistema

Fluxo Principal:

1. Usuário seleciona a opção de novo post
2. Sistema abre página de postagem
3. Usuário digita post
4. Usuário seleciona a opção salvar o post
5. Sistema salva o post
6. Sistema carrega a página com post salvo.

Teste de Caso de Uso: Exemplo

Caso de Teste: Criar post

Pré-condição: Usuário logado no sistema

Cenário:

1. Clicar no botão “novo post”
2. Digitar texto
3. Clicar no botão “salvar como rascunho” o post

Resultado Esperado:

Sistema salvar o post e mostrar a página com post salvo.

Teste Funcional: Considerações Finais

- Como os critérios funcionais se baseiam apenas na especificação, não podem assegurar que partes críticas e essenciais do código tenham sido cobertas.
- Além disso, os próprios critérios apresentam limitações.
- Assim, é fundamental que outras técnicas sejam aplicadas em conjunto, para que o software seja explorado por diferentes pontos de vista.

Exercício 1

Um programa recebe campos numéricos como entrada. Os valores menores que 50 devem ser rejeitados. Valores entre 50 e 101 são considerados como válidos. Valores maiores ou iguais a 102 devem ser rejeitados.

a) Usando partição de equivalência, gere valores de entrada que validem todas estas condições
b) Quais os valores de entrada abrangem a maioria dos limites deste programa?

Exercício 1

a) Devemos gerar valores para validar um intervalo entre 50 e 101. As regrinhas de partição por equivalência dizem para neste caso definir uma classe válida e duas inválidas.

   - um valor válido: 53

   - dois inválidos: 45 e 110

Exercício 1

b) Quando falamos em limites estamos considerando a técnica de análise dos valores limites. A sugestão é usar Dois casos válidos para ambos extremos e dois casos inválidos para além dos limites do intervalo, como o menor incremento possível.

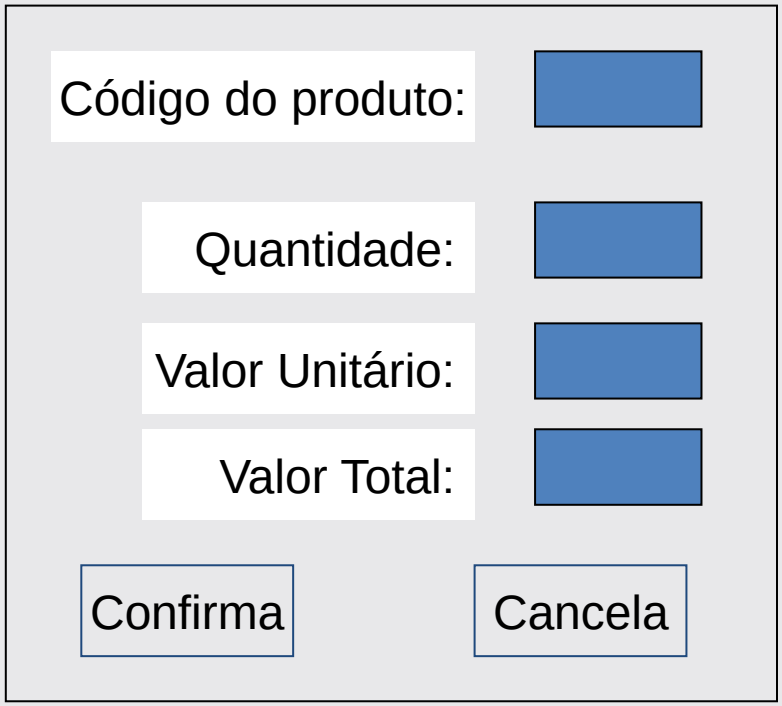
- dois casos válidos para os extremos: 50 e 101

- dois casos inválidos para além dos limites com o menor incremento: 49 e 102.

- para usar partição de equivalência com análise do valor limite teria que escolher um valor válido no meio: 77

Exercício 2

Use partição de equivalência e análise do valor limite para gerar testes da tela abaixo:



Especificações do sistema:
1. O código do produto deve ser um número entre 1 e 99999
2. A quantidade deve ser um valor entre 1 e 100
3. O valor total máximo de um pedido é de R$ 9999,99

Exercício 2

a) O código do produto dever ser um número entre 1 e 99999

Trata-se de um intervalo entre 1 e 99999
- dois casos válidos para os extremos: 1 e 99999
- dois inválidos além dos limites: 0 e 100000
- um valor intermediário: 45000

b) A quantidade deve ser um valor entre 1 e 100

Trata-se de um intervalo entre 1 e 100
- dois casos válidos para os extremos: 1 e 100
- dois inválidos além dos limites: 0 e 101
- um valor intermediário: 50

Exercício 2

c) O valor total máximo de um pedido é de R$ 9999,99

Trata-se de um valor específico. A regra de partição de equivalência para valor específico diz que devem são definidas uma classe válida e duas inválidas, como estamos usando valor limite, os valores já devem testar os limites.

- uma válida: 9999,99
- duas inválidas: 10000,00 e 0 (não existe pedido com valor zerado ou negativo!)

Exercício 3

Considerando o trecho abaixo da tabela de decisão do ATM. Caso se queira cobrir todas as possibilidades de combinação de partições possíveis qual o número de testes que precisam ser realizados?

Condição	1	2	3	4
Cartão válido	N	S	S	S
Senha válida	-	N	N	S
Senha inválida pela terceira vez	-	N	S	N
Ação				
Rejeita cartão	S	N	N	N
Informar senha novamente	N	S	N	N
Apreende cartão	N	N	S	N

Exercício 3

A tabela de decisão gera as saídas para as possíveis condições quando satisfeitas ou não. Assim, caso seja necessário gerar casos de testes para validar todas as condições deve-se multiplicar a quantidade de condições pelo número de situações. Neste exercício seriam necessários 6X4 = 24 casos de teste para abranger todas as situações.

Exercício 4

Quais as características esperadas de uma pessoa que vai realizar o teste da caixa-preta? Que tipo de conhecimento ela deve ter? Ela deve ser um profundo conhecedor de ferramentas de programação?

Exercício 4

Para executar teste da caixa preta, não há necessidade de conhecimento de técnicas ou ferramentas específicas. O tester não precisa conhecer de ferramenta de programação. Serão verificadas as funcionalidades do sistema sem preocupação de como o sistema foi implementado.

Exemplo: Programa Cadeia de Caracteres

- Especificação:
  - Fluxo Principal: O usuário informa uma cadeia de caracteres contendo de 1 a 20 caracteres e um caractere a ser procurado. O sistema informa a posição na cadeia em que o caractere foi encontrado pela primeira vez.
  - Fluxo Alternativo: O caractere informado não está presente na cadeia previamente informada.
    - Uma mensagem é exibida informando que o caractere não pertence à cadeia informada.
  - Fluxo de Exceção: A cadeia informada está vazia ou seu tamanho é maior do que 20.
    - Uma mensagem de erro é exibida indicando que a cadeia de caracteres deve ter tamanho de 1 a 20.
  - Fluxo de Exceção: Não é informado um caractere a ser procurado ou é informado mais de um caractere.
    - Uma mensagem de erro é exibida indicando que um caractere a ser procurado deve ser informado.

Programa Cadeia de Caracteres

- Entradas:
  - Cadeia de caracteres
  - Caractere a ser procurado
- Saídas possíveis:
  - A posição da primeira ocorrência do caractere a ser procurado na cadeia.
  - Mensagem informando que o caractere não pertence à cadeia informada.
  - Mensagem de erro informando que a cadeia é inválida.
  - Mensagem de erro informando que o caractere a ser procurado é inválido.

Programa Cadeia de Caracteres

- Classes de Equivalência:

Entrada	**Classes de Equivalência Válidas**	**Classes de Equivalência Inválidas**
Cadeia de caracteres	Qualquer cadeia de tamanho T, tal que $1 \leq T \leq 20$.	Qualquer cadeia de tamanho T, tal que $T < 1$ e $T > 20$.
Caractere a ser procurado	Caractere que pertence à cadeia.	Caractere não informado.
	Caractere que não pertence à cadeia.	Mais de um caractere é informado.

Programa Cadeia de Caracteres

- Casos de Teste:

Cadeia de Caracteres	**Caractere a ser procurado**	**Saída Esperada**
abc	b	2
abc	d	O caractere não pertence à cadeia informada.
abcdefghijklm nopqrstuvxyz	<<qualquer>>	Erro: Cadeia inválida.
abc	xyz	Erro: Mais de um caractere informado.
abc	<<caractere não informado>>	Erro: Nenhum caractere informado.

Referências

- SOMMERVILLE, Ian. Engenharia de Software. São Paulo: Addisson-Wesley, 2004
- PRESSMAN, Roger S. Engenharia de Software. São Paulo: Makron Books, 1995.
- CRAIG, R.D. Systematic Software Testing. New York: Artech House, 2002.
- Jeff Tian. Software Quality Engineering - Testing, Quality Assurance, and Quantifiable Improvement. IEEE Computer Society. John Wiley & Sons, Inc. 2005.

Obrigado

paulo.nietto@animaeducacao.com.br

Universidade
Anhembi Morumbi
LAUREATE INTERNATIONAL UNIVERSITIES