UFU– Universidade Federal de Uberlândia
FACOM– Faculdade de Computação

V, V & T

Teste Funcional

Baseado nos slides do Prof. Dr. Tiago Silva da Silva (Unifesp)

Objetivo

- Introduzir Teste Funcional (Caixa Preta)
- Introduzir conceitos de Particionamento de Equivalência
- Introduzir conceitos de Análise do Valor Limite

Partes de um Caso de Teste

- Um Caso de Teste bem projetado é dividido em três partes:
  - Entradas (*inputs*)
  - Saídas (*outputs*)
  - Ordem de execução (*order of execution*)

Partes de um Caso de Teste

- Entradas:
  - Geralmente identificadas como dados fornecidos via teclado para o programa executar.
  - Entretanto, os dados de entrada podem ser fornecidos por outros meios:

Partes de um Caso de Teste

- Entradas:
  - Geralmente identificadas como dados fornecidos via teclado para o programa executar.
  - Entretanto, os dados de entrada podem ser fornecidos por outros meios:
    - Dados oriundos de outro sistema que servem de entrada para o programa em teste
    - Dados fornecidos por outro dispositivo
    - Dados lidos de arquivos ou banco de dados
    - O ambiente no qual o programa está sendo executado

Partes de um Caso de Teste

- Saídas esperadas:
  - A forma mais comum de saída é aquela apresentada na tela do computador
  - Além dessa, as saídas podem ser enviadas para:

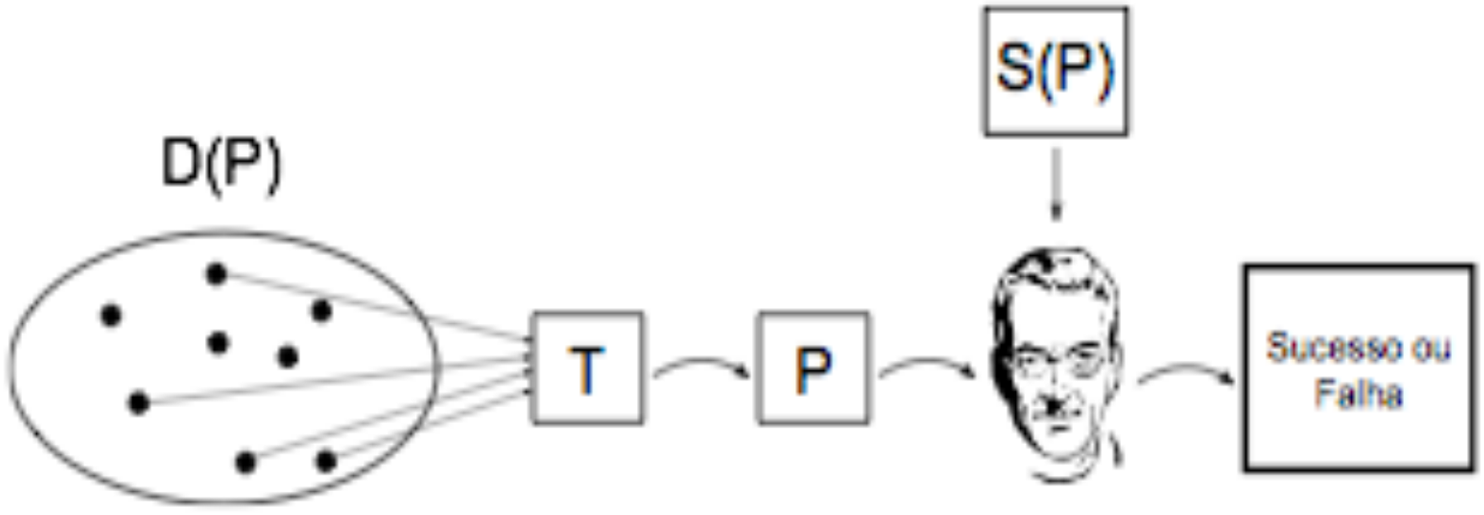
Partes de um Caso de Teste

- Saídas esperadas:
  - A forma mais comum de saída é aquela apresentada na tela do computador
  - Além dessa, as saídas podem ser enviadas para:
    - Outro sistema interagindo com o programa em teste
    - Dados enviados para um dispositivo externo
    - Dados escritos em arquivos ou BDs
    - O estado do sistema ou o ambiente de execução podem ser alterados durante a execução do programa

Partes de um Caso de Teste

- Oráculo:
  - Todas as formas de entrada e saída são relevantes
  - Durante o projeto de um caso de teste, determinar correção da saída esperada é função do oráculo (*oracle*)
  - Oráculo corresponde a um mecanismo (programa, processo ou dados) que indica ao projetista de casos de testes se a saída obtida para um caso de teste é aceitável ou não

Cenário Típico



Partes de um Caso de Teste

- Ordem de execução:
  - Casos de teste em cascata
  - Casos de teste independentes

Partes de um Caso de Teste

- Ordem de execução:
  - Casos de teste **em cascata**
    - quando os casos de teste devem ser executados um após o outro, em uma ordem específica. O estado do sistema deixado pelo primeiro caso de teste é reaproveitado pelo segundo e assim sucessivamente.

Partes de um Caso de Teste

- Ordem de execução:
  - Casos de teste **em cascata**
  - Vantagem:
  - Desvantagem:

Partes de um Caso de Teste

- Ordem de execução:
  - Casos de teste **em cascata**
  - Vantagem: casos de testes tendem a ser pequenos e simples. Fáceis de serem projetados, criados e mantidos.
  - Desvantagem:

Partes de um Caso de Teste

- Ordem de execução:
  - Casos de teste **em cascata**
  - Vantagem: casos de testes tendem a ser pequenos e simples. Fáceis de serem projetados, criados e mantidos.
  - Desvantagem: se um caso de teste falhar os casos de teste subsequentes também podem falhar.

Partes de um Caso de Teste

- Ordem de Execução:
  - Casos de teste **independentes**
    - Cada caso de teste é inteiramente autocontido.

Partes de um Caso de Teste

- Ordem de Execução:
  - Casos de teste **independentes**
  - Vantagem:
  - Desvantagem:

Partes de um Caso de Teste

- Ordem de Execução:
  - Casos de teste **independentes**
    - Vantagem: casos de teste podem ser executados em qualquer ordem.
    - Desvantagem:

Partes de um Caso de Teste

- Ordem de Execução:
  - Casos de teste **independentes**
    - Vantagem: casos de teste podem ser executados em qualquer ordem.
    - Desvantagem: casos de teste tendem a ser grandes e complexos, mais difíceis de serem projetados, criados e mantidos.

Tipos de Teste

- Diferentes tipos de testes podem ser utilizados para verificar se um programa se comporta como o especificado.
- Basicamente, os testes podem ser classificados em teste caixa-preta (*black-box testing*) - **funcional -**, teste caixa-branca (*white-box testing*) - **estrutural -** ou teste **baseado em defeito** (*fault-based testing*).
- Esses tipos de teste correspondem às chamadas **técnicas de teste.**

Tipos de Teste

- A técnica de teste é definida pelo tipo de informação utilizada para realizar o teste.
  - Técnica **caixa-preta** - os testes são baseados exclusivamente na especificação de requisitos do programa. Nenhum conhecimento de como o programa está implementado é requerido.
  - Técnica **caixa-branca** - os testes são baseados na estrutura interna do programa, ou seja, na implementação do mesmo.
  - Técnica **baseada em defeito** - os testes são baseados em informações históricas sobre defeitos cometidos frequentemente durante o processo de desenvolvimento de software.

Teste Funcional

Teste Funcional

- Particionamento de Equivalência (*Equivalence Partition*).
- Analise do Valor Limite (*Boundary Value Analysis*).

Teste Funcional

- Também chamado Caixa Preta por considerar o produto em teste como uma **caixa** da qual só se conhece a entrada e a saída, ou seja, nenhum conhecimento de como o produto internamente é utilizado.
- Critérios dessa técnica baseiam-se somente na especificação de requisitos para derivar os requisitos de testes.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.
  - Os casos de testes são construídos.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.
  - Os casos de testes são construídos.
  - O conjunto de testes é executado.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.
  - Os casos de testes são construídos.
  - O conjunto de testes é executado.
  - As saídas obtidas são comparadas com as saídas esperadas.

Teste Funcional

- Os passos básicos para se aplicar um critério de teste de caixa preta são:
  - A especificação de requisitos é analisada
  - Entradas válidas são escolhidas (com base na especificação) para determinar se o produto em teste se comporta corretamente. Entradas inválidas também são escolhidas para verificar se são detectadas e manipuladas adequadamente.
  - As saídas esperadas para as entradas escolhidas são determinadas.
  - Os casos de testes são construídos.
  - O conjunto de testes é executado.
  - As saídas obtidas são comparadas com as saídas esperadas.
  - Um relatório é gerado para avaliar o resultado dos teses.

Teste Funcional

- **Vantagens**:

Teste Funcional

- **Vantagens**:
  - Pode ser utilizado em todas as fases de teste
  - Independente do paradigma de programação utilizado
  - Eficaz em detectar determinados tipos de erros (funcionalidade ausente, por exemplo)

Teste Funcional

- **Desvantagens:**

Teste Funcional

- **Desvantagens:**
  - Dependente de uma boa especificação de requisitos que, em geral, não é bem feita
  - Não é possível garantir que partes essenciais ou críticas do software sejam executadas
  - Para encontrar todos os defeitos utilizando teste funcional é necessário o teste exaustivo

Teste Funcional - Particionamento de Equivalência

Particionamento de Equivalência

- Critério utilizado para reduzir o número de casos de teste procurando garantir uma boa cobertura do código do produto em teste
- Empregado intuitivamente por programadores mesmo sem conhecer o critério
- Exemplo: sistema de RH - empregar pessoas com base na idade

0 – 16	Não empregar.
16 – 18	Pode ser empregado tempo parcial.
18 – 55	Pode ser empregado tempo integral.
55 – 99	Não empregar.

- Como deveriam ser derivados casos de teste para o exemplo?

Particionamento de Equivalência

- O módulo deveria ser testado considerando as idades: 0, 1, 2, 3, 4, 5, 6, 7, 8, ..., 90, 91, 92, 93, 94, 95, 96, 97, 98, 99?
- Considere que o módulo que resolve o problema anterior tenha sido implementado assim:

```
1  if (idade == 0) empregar = "NAO";
2  if (idade == 1) empregar = "NAO";
3  ...
4  if (idade == 15) empregar = "NAO";
5  if (idade == 16) empregar = "PAR";
6  if (idade == 17) empregar = "PAR";
7  if (idade == 18) empregar = "INT";
8  if (idade == 19) empregar = "INT";
9  ...
10 if (idade == 53) empregar = "INT";
11 if (idade == 54) empregar = "INT";
12 if (idade == 55) empregar = "NAO";
13 if (idade == 56) empregar = "NAO";
14 ...
15 if (idade == 98) empregar = "NAO";
16 if (idade == 99) empregar = "NAO";
```

Particionamento de Equivalência

- Caso o programa tenha sido implementado dessa forma, a única forma de testá-lo adequadamente seria executar o módulo com valores de 0...99
- Caso haja tempo suficiente, esse é o melhor teste a ser realizado
- O problema é que da forma como o código foi implementado, a execução de um dado caso de teste não diz nada a respeito da execução do próximo

Particionamento de Equivalência

- Agora considere esta implementação para o mesmo problema:

```
1  if (idade >= 0 && idade <= 16)
2          empregar = "NAO";
3  if (idade >= 16 && idade <= 18)
4          empregar = "PAR";
5  if (idade >= 18 && idade <= 55)
6          empregar = "INT";
7  if (idade >= 55 && idade <= 99)
8          empregar = "NAO";
```

- Dada essa implementação, fica claro que não é necessário testar para todos os valores 0, 1, 2, ... 14, 15 e 16, por exemplo.
- Apenas um valor precisa ser testado.
- Qual seria este valor?

Particionamento de Equivalência

- Qualquer valor dentro do intervalo tem a mesma importância, ou seja, qualquer valor escolhido é adequado
- O mesmo se aplica para os demais intervalos de dados
- Tais intervalos determinam o que é chamado de **classe de equivalência**
- Qualquer valor no intervalo de uma classe é considerado equivalente em termos de teste. Assim sendo:
  - Se um caso de teste de uma classe de equivalência revela um erro, qualquer caso de teste da mesma classe também revelaria.

Particionamento de Equivalência

- Tal critério de teste assume que na especificação de requisitos existe uma indicação precisa das classes de equivalência
- Além disso, também é assumido que o programador não implementou algo estranho como o seguinte:

```
1  if (idade >= 0 && idade <= 16)
2          empregar = "NAO";
3  if (idade >= 16 && idade <= 18)
4          empregar = "PAR";
5  if (idade >= 18 && idade <= 41)
6          empregar = "INT";
7  // início comado estranho
8  if (idade == 42 && nome == "Fulano")
9          empregar = "INT-DIF";
10 if (idade == 42 && nome != "Fulano")
11         empregar = "INT";
12 // fim comando estranho
13 if (idade >= 55 && idade <= 99)
14         empregar = "NAO";
```

Particionamento de Equivalência

- Observe que com esse critério de teste, o número de casos de teste é reduzido de 100 para 4 (um para cada classe de equivalência)

- Casos de teste inválidos devem ser considerados?

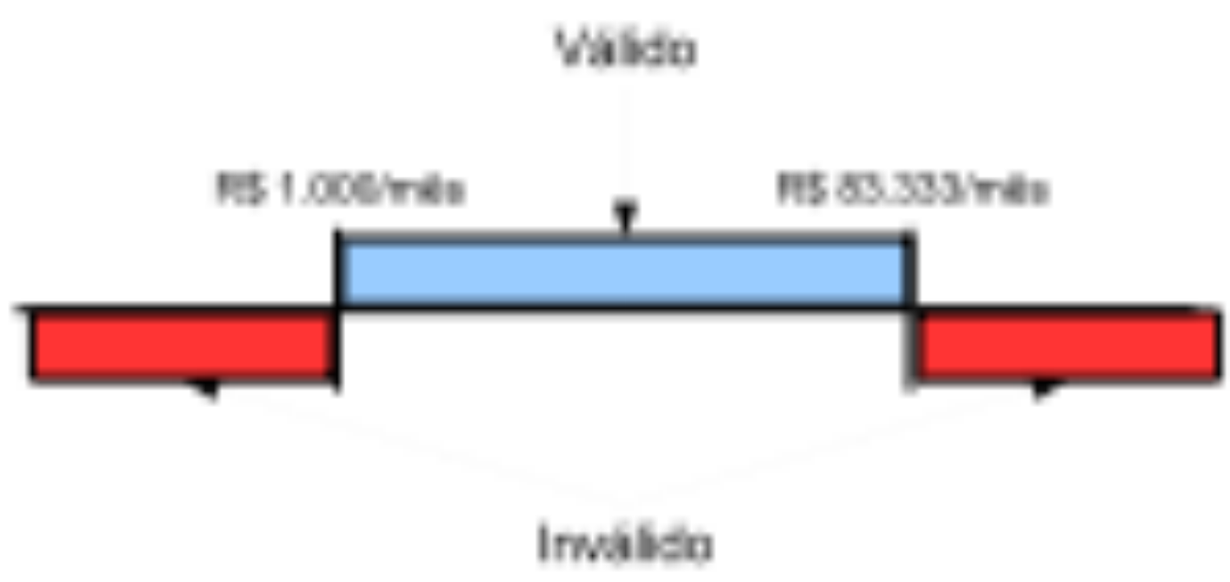
Particionamento de Equivalência

- Passos de Aplicação:

1. Identificar as classes de equivalência (requisitos de teste do critério)
2. Criar casos de teste para as classes de equivalência válidas
3. Criar um caso de teste para cada classe de equivalência (entradas inválidas são grandes fontes de defeitos)
4. Casos de teste adicionais podem ser criados caso haja tempo e recursos suficientes
    - Com base em sua experiência, o testador pode criar casos de teste adicionais

Particionamento de Equivalência

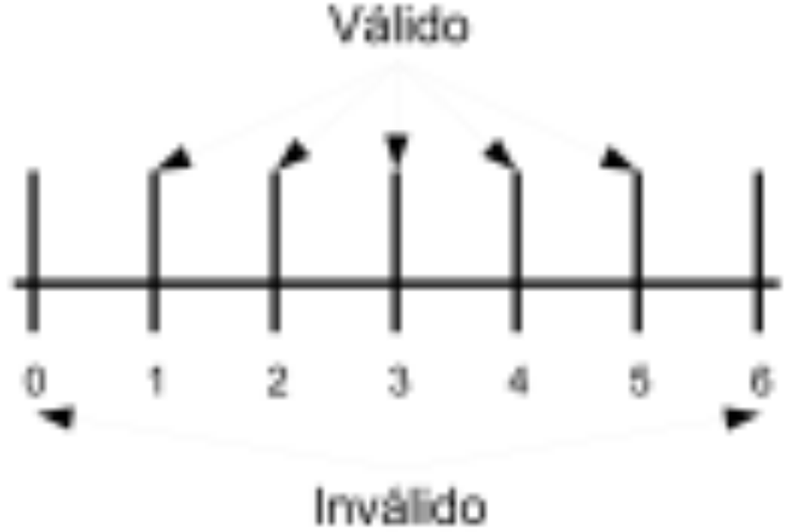
- Definição das Classes:
  - Diferentes tipos de dados exigem diferentes tipos de classe de equivalência
  - Intervalo de dados contínuos (renda para hipoteca de R$1000 a 83000/mês)



  - Em geral, são definidas duas classes inválidas e uma válida
  - Para a classe válida, poderia ser escolhido R$1342/mês
  - Para classes inválidas, poderia ser: R$123/mês e R$90000/mês

Particionamento de Equivalência

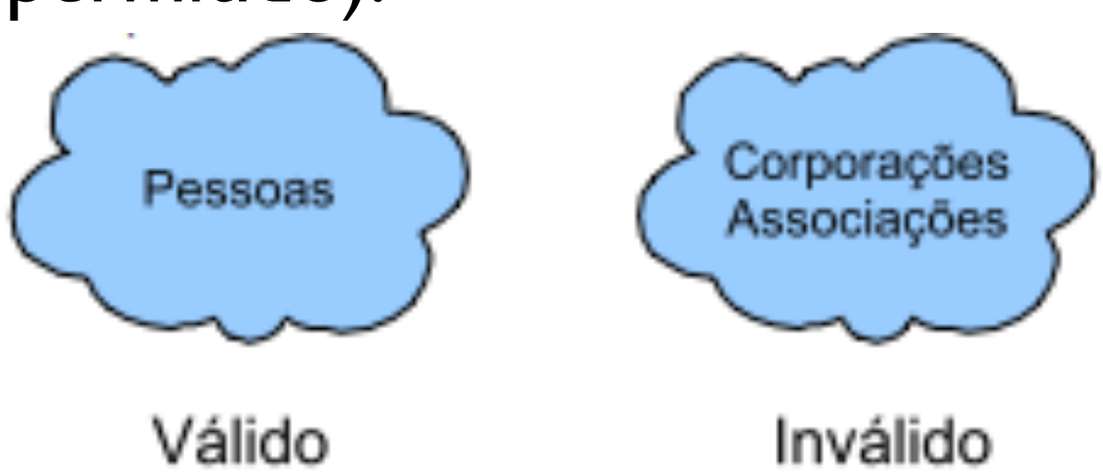
- Definição das Classes:
  - Intervalos de dados discretos (hipotecas de 1 a 5 casas):



- Em geral, são definidas duas classes inválidas e uma válida
- Válida: 2
- Inválidas: -2 e 8

Particionamento de Equivalência

- Definição das Classes:
  - Intervalo de dados simples (somente hipoteca para pessoas é permitido):



- Em geral, são definidas uma classe inválida e uma válida
- Válida: uma pessoa qualquer
- Inválida: uma companhia ou associação

Particionamento de Equivalência

- Definição das Classes:
  - Intervalo de dados de múltipla escolha (três tipos de hipoteca são válidas: condomínio, sobrado e casa térrea):

  - Para o intervalo válido pode-se escolher: condomínio, sobrado ou casa térrea. Escolher somente um ou os três? Depende a criticalidade do programa em teste. Se forem poucos itens, vale a pena selecionar um de cada.
  - O mesmo para classe inválida.

Particionamento de Equivalência

- Definição das Classes:
  - Em geral, não há tempo para a criação de um caso de teste para cada classe válida.
  - Solução: criar o menor número possível de casos de teste que cubram todas as classes válidas
  - Criar um caso de teste para cada classe inválida

Renda	# Moradores	Aplicante	Tipo	Resultado
$5.000	2	Pessoas	Condomínio	Válido
$100	1	Pessoas	Uma família	Inválido
$90.000	1	Pessoas	Uma família	Inválido
$1.342	**0**	Pessoas	Condomínio	Inválido
$1.342	**6**	Pessoas	Condomínio	Inválido
$1.342	1	**Corporação**	Sobrado	Inválido
$1.342	1	Pessoas	**Duplex**	Inválido

Particionamento de Equivalência

- Definição das Classes:
  - Uma abordagem adicional ao critério Particionamento de Equivalência é considerar as saídas.
  - O domínio de saída também é particionado em classes válidas e inválidas
  - Casos de teste que causem tais saídas são então desenvolvidos

Particionamento de Equivalência

- Aplicabilidade e Limitações:
  - Reduz significativamente o número de casos de teste em relação ao teste exaustivo
  - Mais adequado para o teste de produtos com domínios de entrada divididos em intervalos ou conjuntos
  - Assume que os valores dentro da mesma classe são equivalentes
  - Aplicável em todas as fases de teste: unidade, integração, sistema.

Teste Funcional - Análise do Valor Limite

Análise do Valor Limite

- Um dos critérios de teste mais básicos que existe
- Auxilia na seleção de um pequeno subconjunto de casos de teste que mantém uma boa cobertura do código

Análise do Valor Limite

- Considerando o exemplo:

0 – 16	Não empregar.
16 – 18	Pode ser empregado tempo parcial.
18 – 55	Pode ser empregado tempo integral.
55 – 99	Não empregar.

- Observe que os limites, tal como o 16, aparece em duas classes de equivalência. O mesmo ocorre com o 18 e o 55.

Análise do Valor Limite

- As condições anteriores deveriam, na verdade, ser escritas como:

$0 \leq idade < 16$	Não empregar.
$16 \leq idade < 18$	Pode ser empregado tempo parcial.
$18 \leq idade < 55$	Pode ser empregado tempo integral.
$55 \leq idade < 99$	Não empregar.

- ou

$0 \leq idade \leq 15$	Não empregar.
$16 \leq idade \leq 17$	Pode ser empregado tempo parcial.
$18 \leq idade \leq 54$	Pode ser empregado tempo integral.
$55 \leq idade \leq 99$	Não empregar.

Análise do Valor Limite

- O código abaixo implementa as regras anteriores

```
1  if (idade >= 0 && idade <= 15)
2          empregar = "NAO";
3  if (idade >= 16 && idade <= 17)
4          empregar = "PAR";
5  if (idade >= 18 && idade <= 54)
6          empregar = "INT";
7  if (idade >= 55 && idade <= 99)
8          empregar = "NAO";
```

- Valores limites a serem considerados:
  - {-1, 0, 1}, {14, 15, 16}
  - {15, 16, 17}, {16, 17, 18}
  - {17, 18, 19}, {53, 54, 55}
  - {54, 55, 56}, {98, 99, 100}

Análise do Valor Limite

- Passos de Aplicação:

1. Identificar as classes de equivalência (requisitos de teste do critério)
2. Identificar os limites de cada classe
3. Criar casos de teste para os limites escolhendo:
   - Um ponto abaixo do limite
   - O limite
   - Um ponto acima do limite
4. Observe que “acima” e “abaixo” são termos relativos e dependente do valor dos dados
   - Números inteiros: limite = 16; abaixo = 15
   - Números reais: limite $5,00; abaixo = $4,99
5. Casos de teste adicionais podem ser criados dependendo dos recursos disponíveis

Análise do Valor Limite

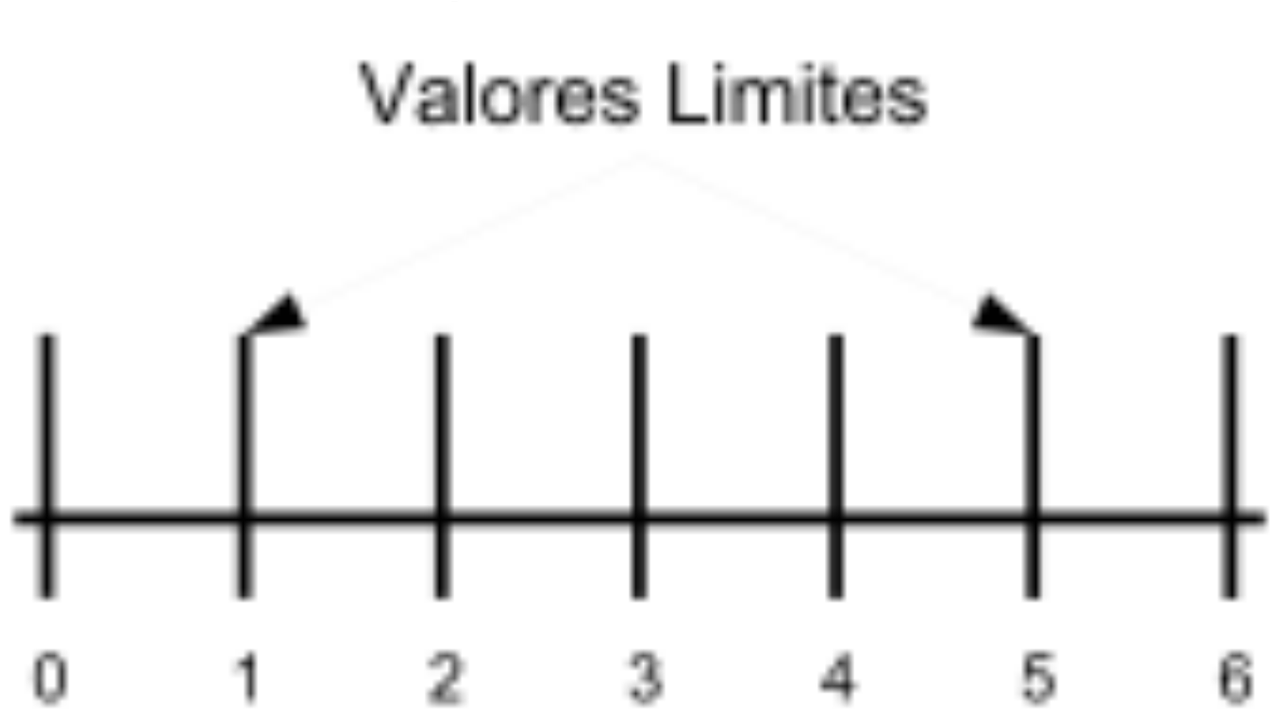
- Definição das Classes
  - Mais adequado para entradas que apresentam valores contínuos

(extraído de Copeland (2004))

- Dados de teste para o limite inferior: {$999, $1000, $1001}
- Dados de teste para o limite superior: {$83332, 83333, 83334}

Análise do Valor Limite

- Definição das Classes
  - Intervalo de dados discretos (hipotecas de 1 a 5 casas):



(extraído de Copeland (2004))

Análise do Valor Limite

Renda	# Moradores	Resultado	Descrição
$1,000	1	Válido	Mín. renda, mín. moradores
$83,333	1	Válido	Max. renda, mín. moradores
$1,000	5	Válido	Mín. renda, max.. moradores
$83,333	5	Válido	Max. renda, max. moradores
$1,000	0	Inválido	Mín. renda, abaixo mín. moradores
$1,000	6	Inválido	Mín. renda, acima max. moradores
$83,333	0	Inválido	Max. renda, abaixo mín. moradores
$83,333	6	Inválido	Max. renda, acima max. moradores
$999	1	Inválido	Abaixo mín. renda, mín. moradores
$83,334	1	Inválido	Acima max. renda, mín. moradores
$999	5	Inválido	Abaixo mín. renda, max. moradores
$83,334	5	Inválido	Acima max. renda, max. moradores

Análise do Valor Limite

- Aplicabilidade e Limitações:
  - Reduz significativamente o número de casos de teste em relação ao teste exaustivo
  - Observa o domínio de saída
  - Mais adequado para o teste de produtos com domínios de entrada divididos em intervalos ou conjuntos
  - Aplicável em todas as fases de teste: unidade, integração, sistema.