Engenharia de Software II

Aula 9

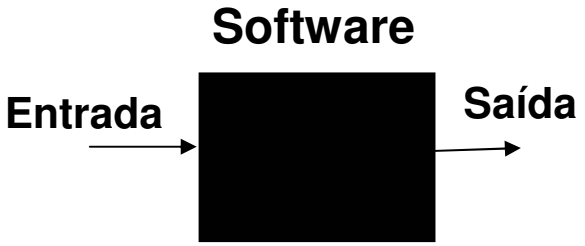
http://www.ic.uff.br/~bianca/engsoft2/

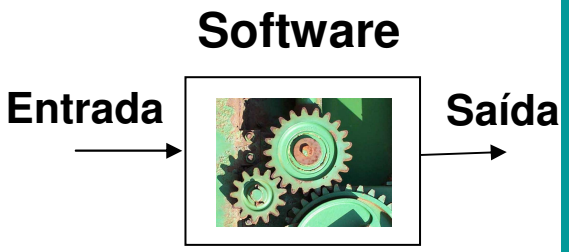
Ementa

- Processos de desenvolvimento de software
- **Estratégias e técnicas de teste de software** (Caps. 13 e 14 do Pressman)
- Métricas para software
- Gestão de projetos de software: conceitos, métricas, estimativas, cronogramação, gestão de risco, gestão de qualidade e gestão de modificações
- Reengenharia e engenharia reversa

Testes Caixa-Preta e Caixa-Branca

- Testes Caixa-Preta
  - São conduzidos na interface do software, sem preocupação com a estrutura lógica interna do software.
- Testes Caixa-Branca
  - São baseados em um exame rigoroso do detalhe procedimental.
  - Caminhos lógicos e colaborações entre componentes são testadas.





Técnicas de Teste Caixa-Branca

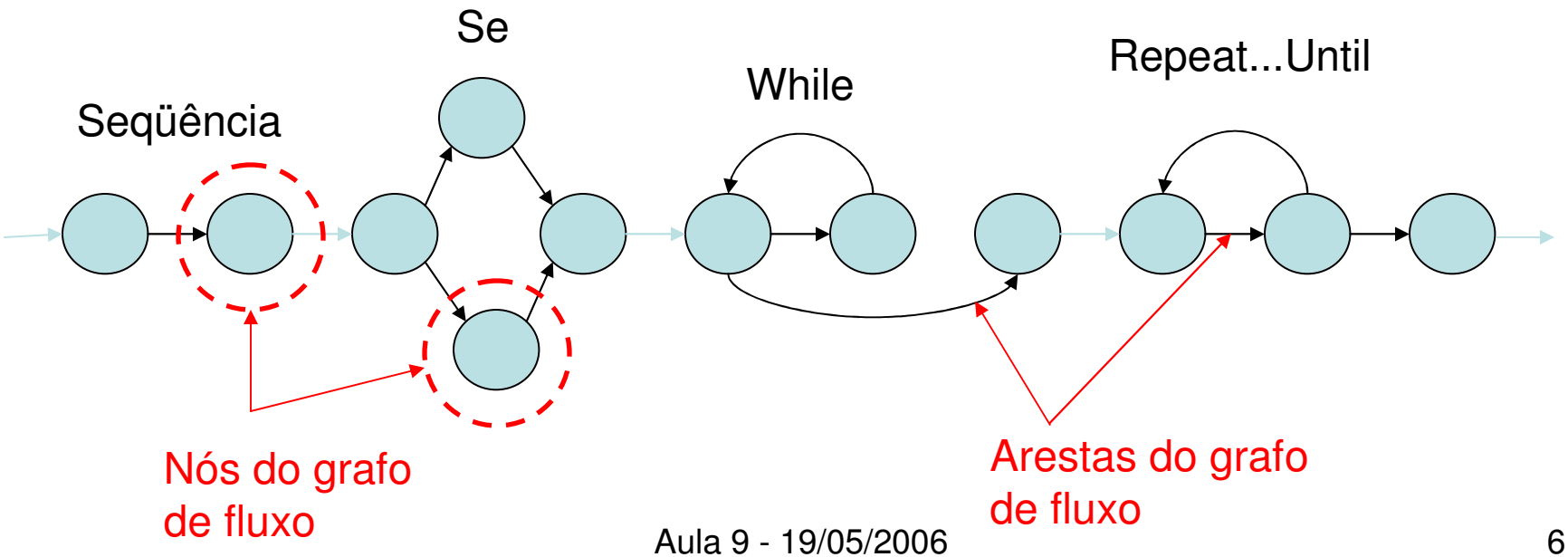
- Usando métodos de teste caixa-branca, podemos derivar casos de teste que:
  1. Garantam que todos os caminhos independentes de um módulo sejam executados pelo menos uma vez.
  2. Exercitem todas as decisões lógicas de seu lado verdadeiro e falso.
  3. Executem todos os ciclos (loops) nos seus limites e dentro de seus intervalos operacionais.
  4. Exercitem as estruturas de dados internas.
- Tipos de teste caixa-branca:
  - Teste de Caminho Básico
  - Teste de Estrutura de Controle

Teste de Caminho Básico

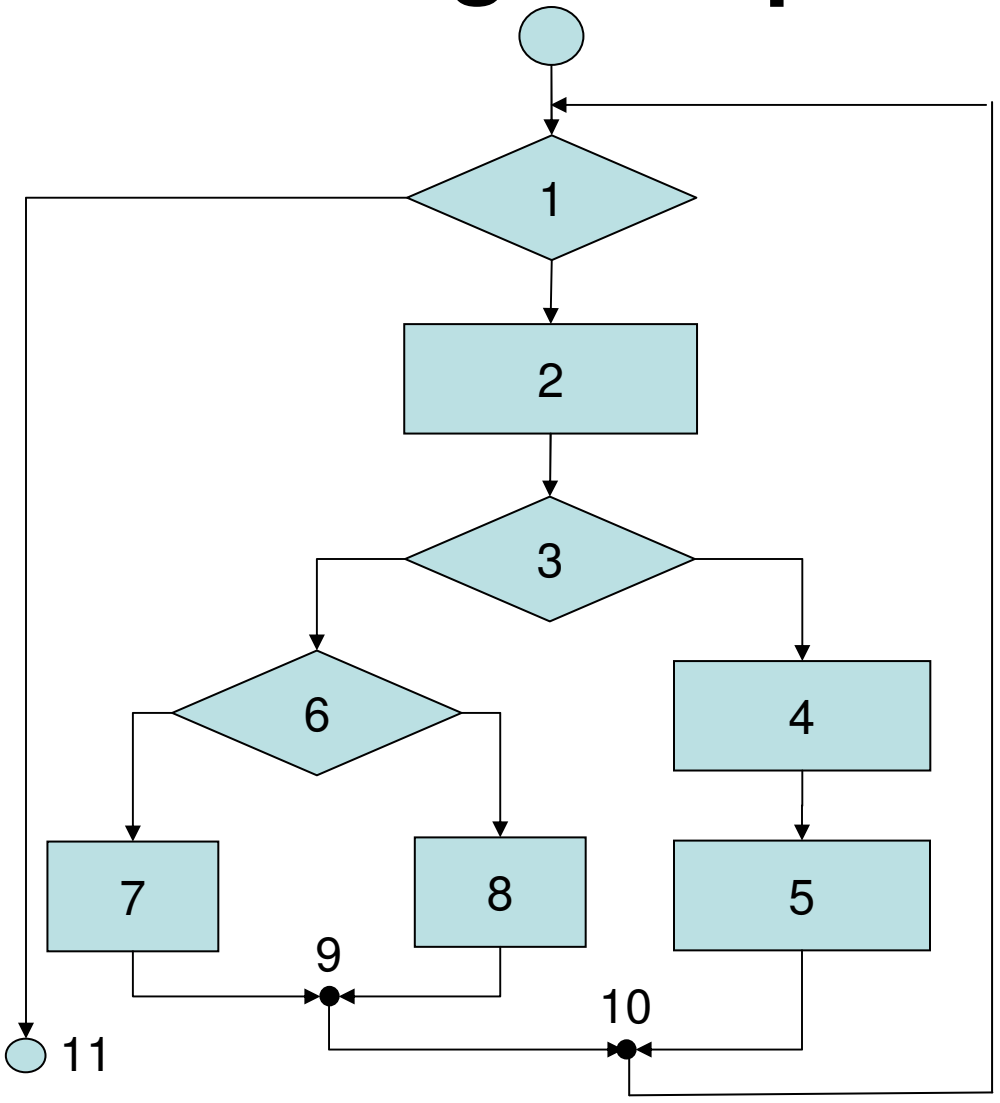
- Permite ao projetista de caso de teste medir a complexidade lógica de um projeto procedimental.
  - Essa medida é usada como guia para definir um conjunto básico de caminhos de execução.
  - Casos de teste derivados para exercitar o conjunto básico executam garantidamente cada comando pelo menos uma vez.

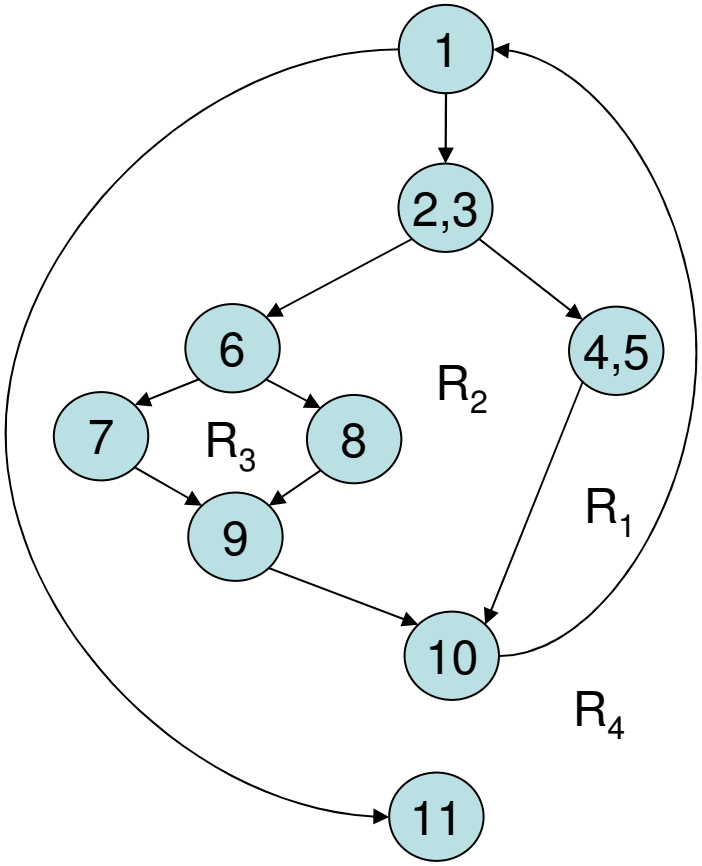
Notação de Grafo de Fluxo

- Serve como notação útil para entender o fluxo de controle e ilustrar a abordagem de caminho básico.



De Fluxograma para Grafo de Fluxo

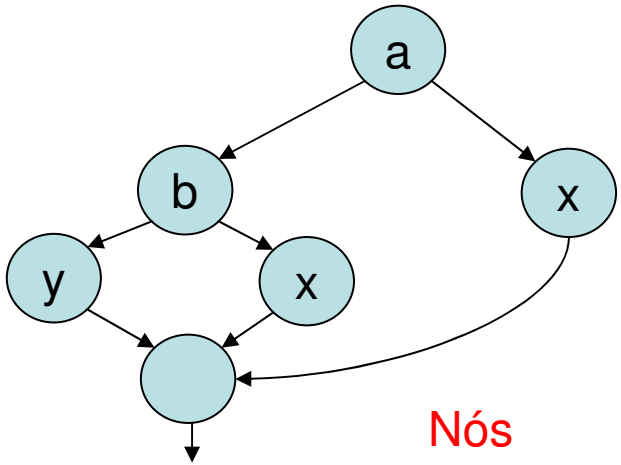




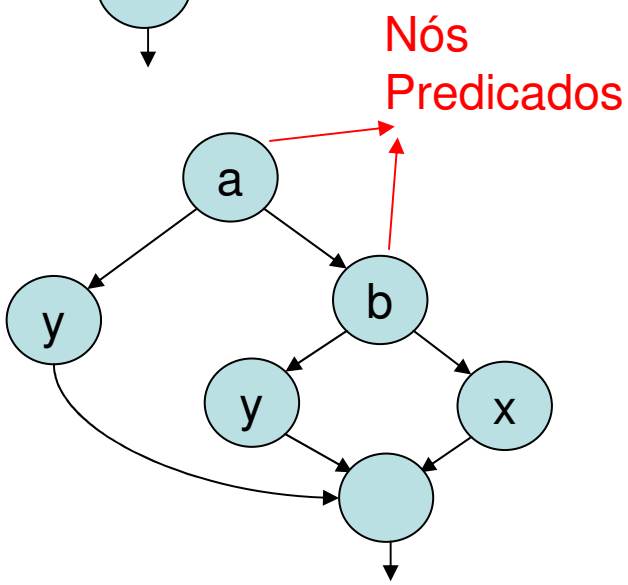
Áreas limitadas por arestas e nós são chamadas de **regiões.**

Lógica Composta

```
SE a OU b
ENTÃO procedimento x
SENÃO procedimento y
FIM-SE
```

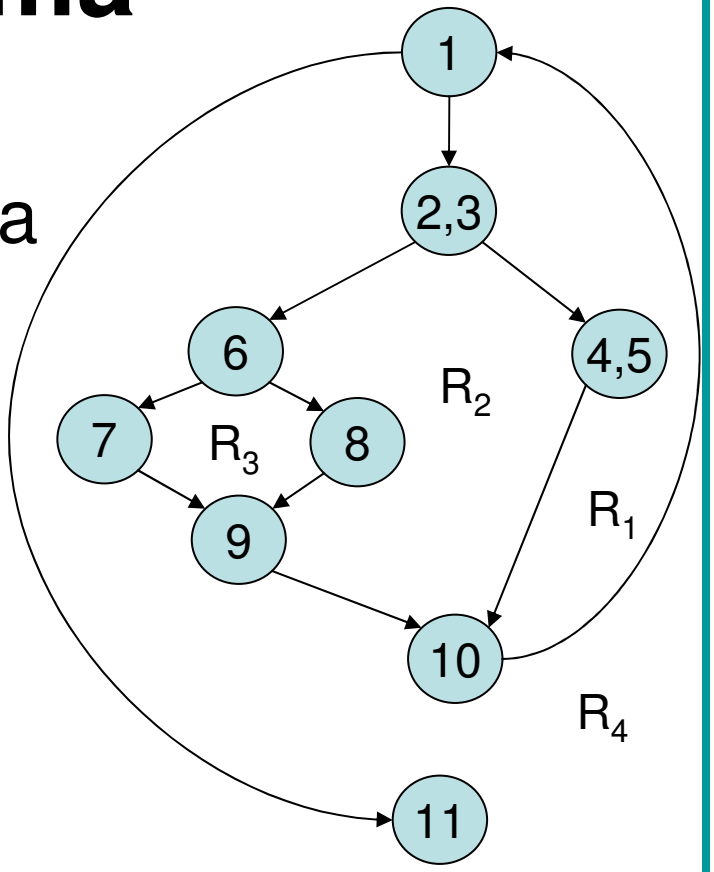



```
SE a E b
ENTÃO procedimento x
SENÃO procedimento y
FIM-SE
```



Caminhos Independentes de Programa

- Um conjunto de caminhos independentes para o grafo da figura é:
  - Caminho 1: 1-11
  - Caminho 2: 1-2-3-4-5-10-1-11
  - Caminho 3: 1-2-3-6-8-9-10-1-11
  - Caminho 4: 1-2-3-6-7-9-10-1-11
- Um caminho independente deve incluir pelo menos uma aresta nova.



Conjunto-Base

- É um conjunto de caminhos independentes em que cada aresta aparece pelo menos uma vez.
- Testes projetados para forçar a execução do conjunto-base garantem que:
  - Todo comando do programa terá sido executado pelo menos uma vez.
  - Cada condição terá sido executada no seu lado verdadeiro e no seu lado falso.
- O conjunto-base não é único.
  - Diversos conjuntos-base diferentes podem ser derivados para um dado projeto procedimental.

Complexidade Ciclomática

- É uma métrica de software que fornece uma medida quantitativa da complexidade lógica de um programa.
  - Fornece um limite superior para o número de caminhos independentes num conjunto-base.
  - Fornece um limite superior para a quantidade de testes que deve ser conduzida para garantir que todos os comandos sejam executados pelo menos uma vez.

Cálculo da Complexidade Ciclomática

- Pode ser calculada de três maneiras equivalentes:
  1. V(G) = R

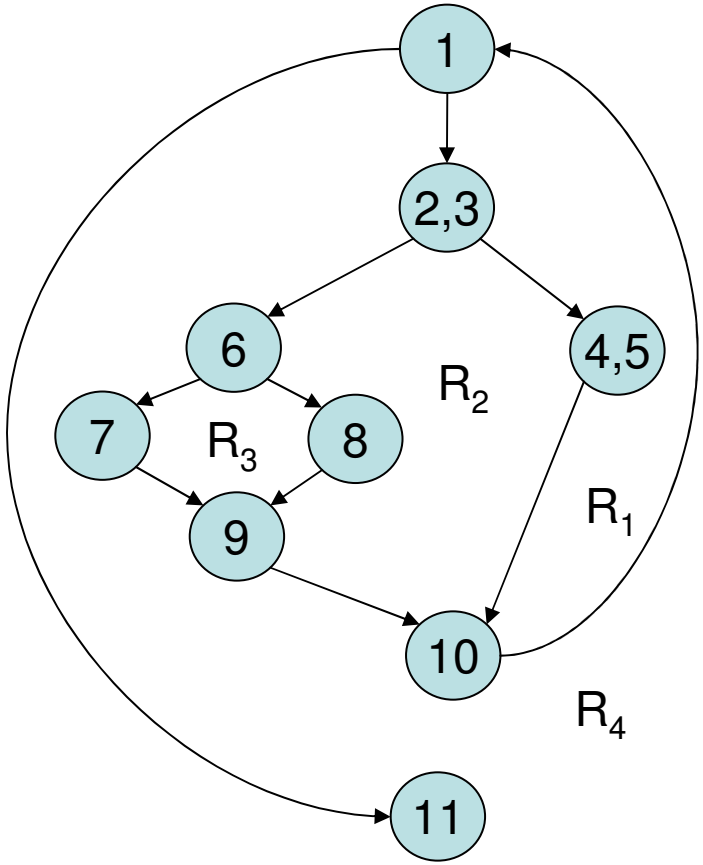
     onde R é o número de regiões do grafo de fluxo.
  2. V(G) = E – N +2

     onde E é o número de arestas e N é o número de nós do grafo G.
  3. V(G) = P + 1

     onde P é o número de nós-predicados contidos no grafo G (só funciona se os nós-predicado tiverem no máximo duas arestas saindo)

Complexidade Ciclomática

1. O grafo de fluxo tem quatro regiões.
2. V(G) = 11 arestas – 9 nós + 2 = 4.
3. V(G) = 3 nós-predicado + 1 = 4

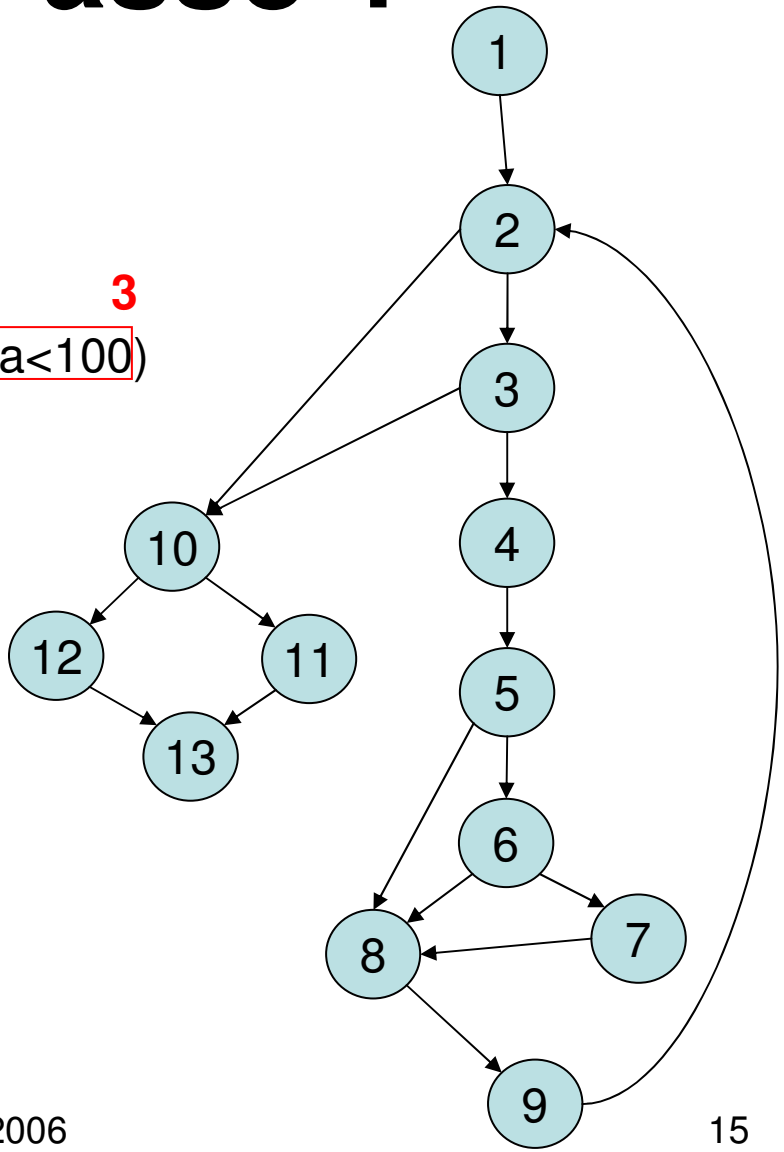


Derivação de Casos de Teste

- O método de teste de caminho básico pode ser aplicado a um projeto procedimental ou ao código-fonte.
- Os seguintes passos devem ser seguidos:
  1. Desenhar o grafo de fluxo correspondente ao projeto ou código-fonte.
  2. Determinar a complexidade ciclomática do grafo de fluxo resultante.
  3. Determinar um conjunto-base de caminhos independentes.
  4. Preparar casos de teste que vão forçar a execução de cada caminho do conjunto-base.

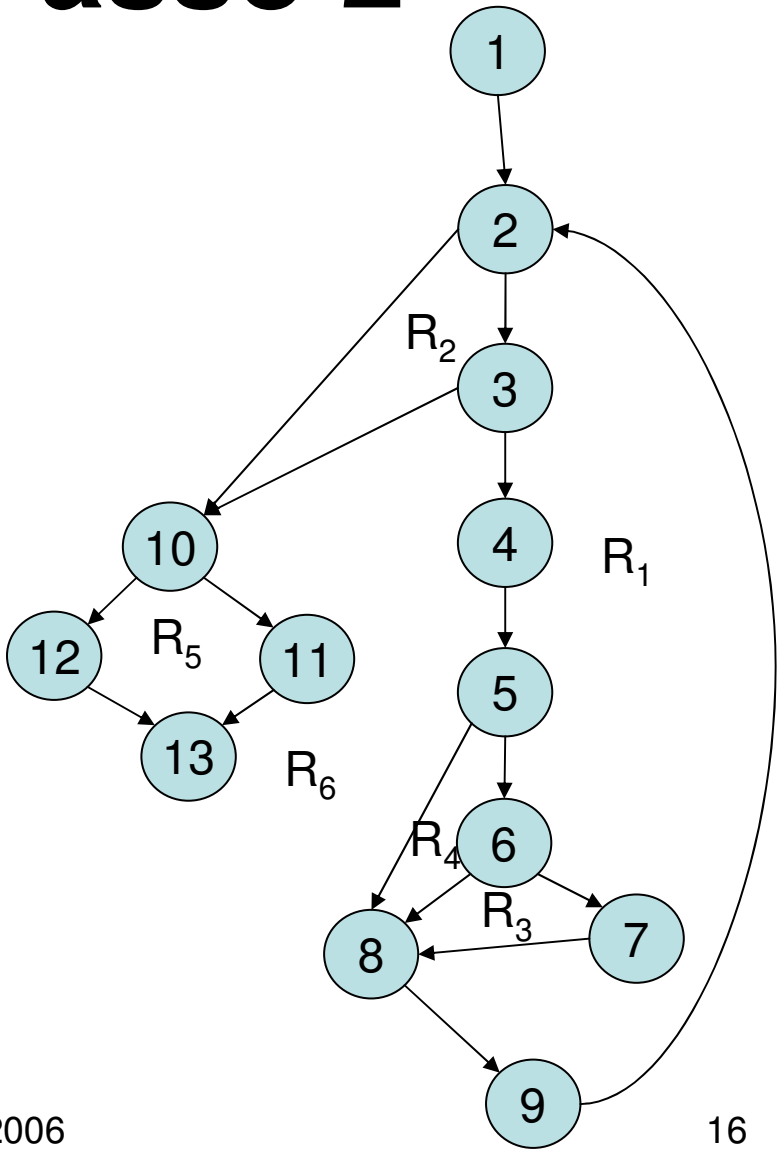
Exemplo: Passo 1

```
Procedimento média(valor[])
i=1;                                                          1
soma=0;
total.entrada=0;
total.válidas=0
                                          2                              3
Faça-Enquanto (valor[i]≠-999 E total.entrada<100)
    incremente total.entrada de 1;                                       4
    Se (valor[i]≥min E valor[i]≤max)                                     5 6
    Então
7           incremente total.válidas de 1;
            soma = soma + valor[i]
    Fim-Se
 8  incremente i de 1;
Fim-Enquanto                                                             9
Se total.válidas>0                                                       10
Então média = soma/total.válidas;                                        11
Senão média = -999;                                                      12
Fim-Se                                                                   13
Fim média
```



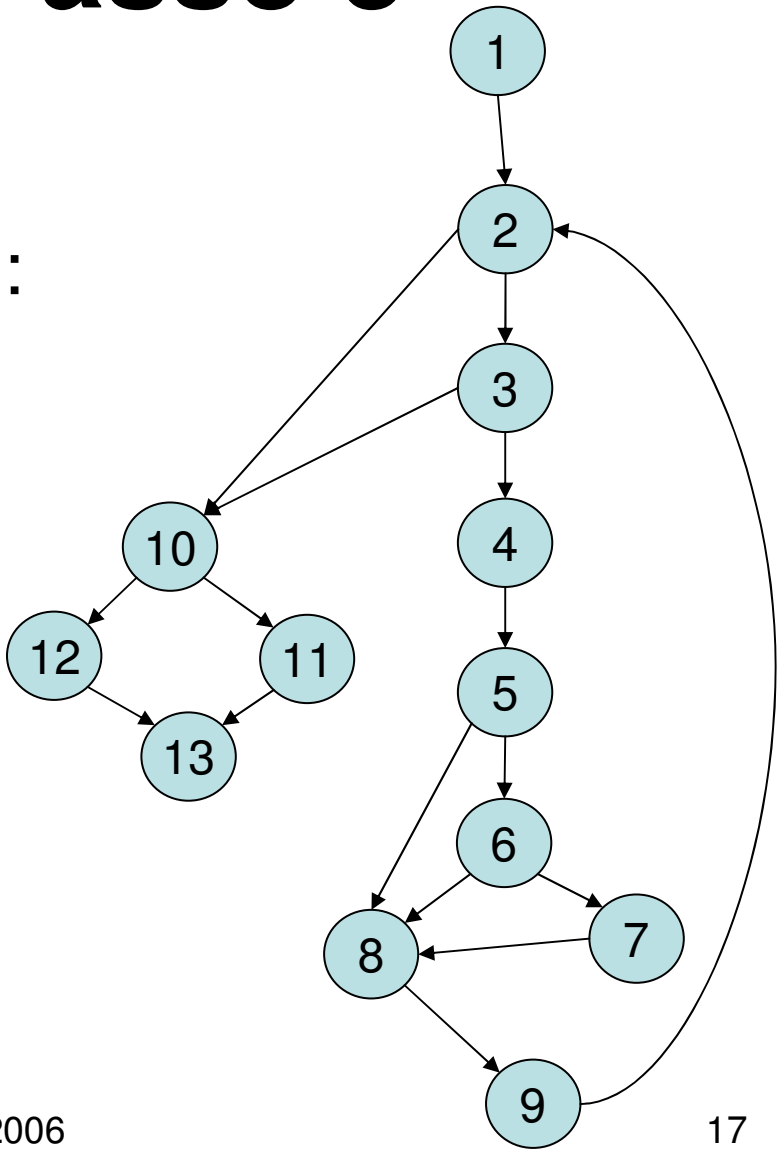
Exemplo: Passo 2

- V(G) = 6 regiões
- V(G) = 17 arestas – 13 nós + 2 = 6
- V(G) = 5 nós-predicados + 1 = 6



Exemplo: Passo 3

- Temos que especificar 6 caminhos independentes:
  1. 1-2-10-12-13
  2. 1-2-10-11-13
  3. 1-2-3-10-11-13
  4. 1-2-3-4-5-8-9-2-...
  5. 1-2-3-4-5-6-8-9-2-...
  6. 1-2-3-4-5-6-7-8-9-2-...



Exemplo: Passo 4

- Caso de Teste do Caminho 1
  - valor[1] = -999
  - Resultado esperado: média=-999 e outros totais com valores iniciais
- Caso de Teste do Caminho 2
  - valor[1] = 1, valor[2]=3,valor[3]=2, valor[4]=-999, min=0, max=10
  - Resultado esperado: média=2, total.válidas=3, total.entradas=3.
- Caso de Teste do Caminho 3
  - valor[1]=valor[2]=...=valor[100]=valor[101]=1, min=0, max=10
  - Resultado esperado: média=1, total.válidas=100, total.entradas=100

Teste de Estrutura de Controle

- O teste de caminho básico é simples e altamente eficaz, mas não é suficiente por si só.
- Exitem outros tipos de testes que focam nas estruturas de controle:
  - Teste de Condição
  - Teste de Fluxo de Dados
  - Teste de Ciclo

Teste de Condição

- Exercita todas as condições lógicas contidas em um módulo de programa para garantir que elas não contém erros.
  - Condição simples: é uma variável booleana ou uma expressão relacional, possivelmente precedida por um NÃO.
  - Condição composta: duas ou mais condições simples ligadas por um operador OU, E ou NÃO.

Teste de Fluxo de Dados

- Seleciona caminhos de teste de acordo com a localização das definições e dos usos das variáveis no programa.
- Dado um comando S, definimos:
  - DEF(S) = { X | comando S contém uma definição de X}
  - USO(S) = { X | comando S contém um uso de X}
- Uma variável X definida em S é considerada viva em S', se existir um caminho de S para S' que não contenha nenhuma outra definição de X.
- Uma cadeia definição-uso (DU) da variável X é da forma [X,S,S'], X pertence a DEF(S) e USO(S') e a definição de X no comando S está viva no comando S'.
- Estratégia: exigir que cada cadeia DU seja coberta pelo menos uma vez.

Teste de Ciclo

- Focaliza exclusivamente na validade de construções de ciclos (*loops*).
  - Ciclos simples
  - Ciclos concatenados
  - Ciclos aninhados
  - Ciclos desestruturados

Ciclos Simples

- O seguinte conjunto de testes pode ser aplicado a ciclos simples em que n é o número máximo de passagens permitidas:
  1. Pule o ciclo completamente.
  2. Apenas uma passagem pelo ciclos.
  3. Duas passagens pelo ciclo.
  4. *m* passagens pelo ciclo em $m < n$.
  5. $n - 1$, n, $n + 1$ passagens pelo ciclo.

Ciclos Aninhados

- Para evitar que o número de testes cresça geometricamente, deve seguir a seguinte abordagem:
  1. Comece testando o ciclo mais interno, ajustando os outros ciclos para valores mínimos.
  2. Trabalhe em direção ao exterior, conduzindo testes para o ciclo seguinte, mas mantendo todos os ciclos externos nos valores mínimos e os internos nos valores “típicos”.
  3. Continue até que todos os ciclos sejam testados.

Ciclos Concatenados

- Dois possíveis casos:
  - Contadores independentes
    - Usar a abordagem de ciclos simples para cada um deles.
  - Contador de ciclo para o ciclo 1 é usado como valor inicial para o ciclo 2.
    - Usar a abordagem de ciclos aninhados.

Ciclos Desestruturados

- Devem ser reprojetados.