UNOESTE – Universidade do Oeste Paulista
FIPP – Faculdade de Informática de Presidente Prudente

Teste de Software – Parte III

Trabalhos de qualidade não podem serem feitos sem concentração, auto-sacrifício, esforço, padronização e as vezes uma certa dúvida do novo.

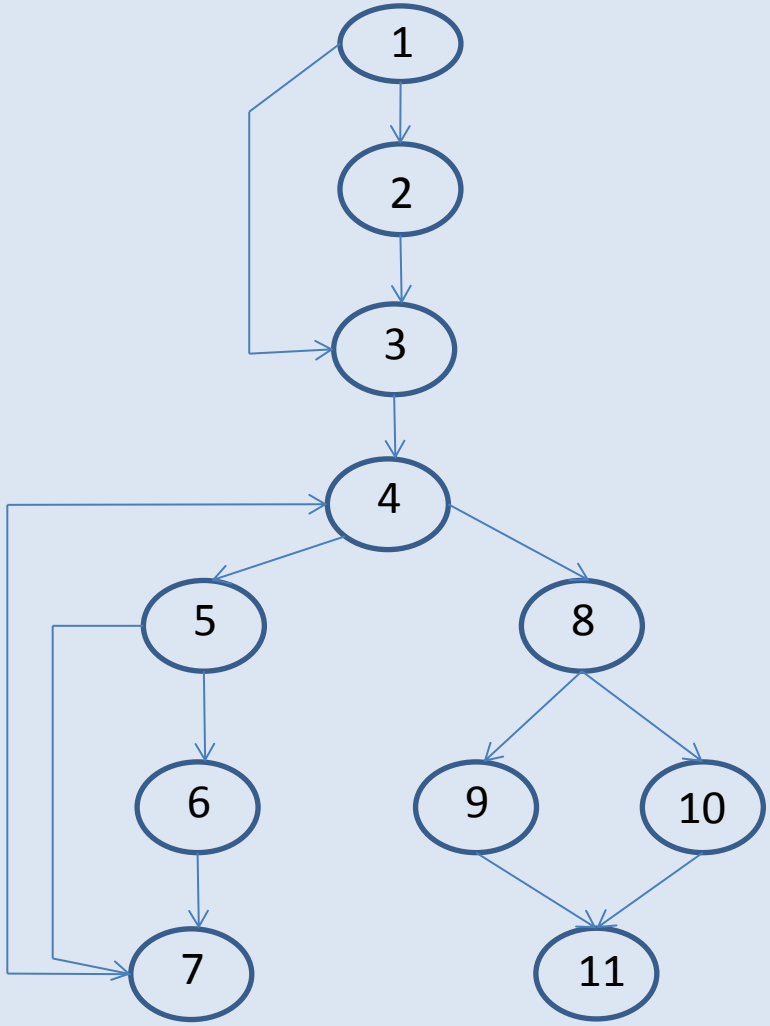
Prof. Msc. Francisco Virginio Maracci – Julho 2011

Teste Estrutural

- Aspectos de implementação são fundamentais para a escolha dos casos de teste
- Utiliza-se de uma representação de programa conhecida como
  - Grafo de fluxo de controle ou grafo de programa
- Pelo grafo de programa pode ser escolhido os componentes que devem ser executados
  - Teste estrutural

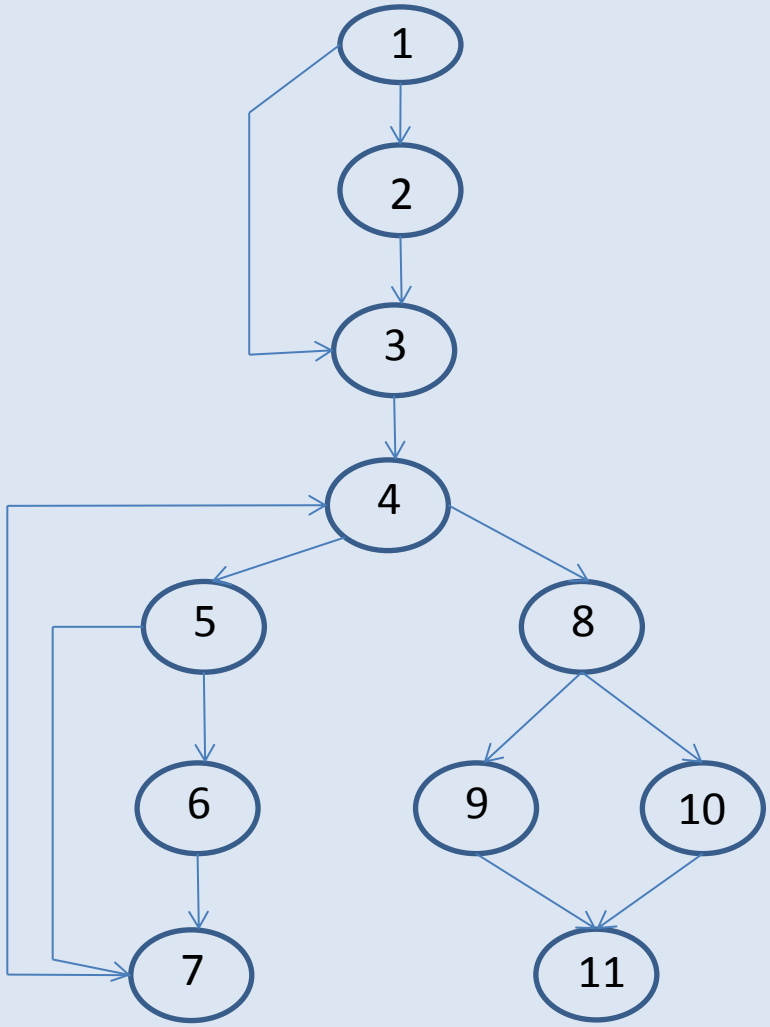
Grafos

- Nó
  - 6
  - 1
  - 2
- Arco/arestas
  - (7,4) (1,3) (5,7) (1,2) (8,9) (9,10)
- Laço
  - (4,5,6,7)
- Caminho
  - (1,2,3,4,8,10,11)
  - (1,2,3,4,8,9,11)
  - (1,2,3,4,5,6,7)



Montagem

```
/* 01 */  {
/* 01 */      char  achar;
/* 01 */      int   length, valid_id;
/* 01 */      length = 0;
/* 01 */      printf ("Digite um possível identificador\n");
/* 01 */      printf ("seguido por <ENTER>: ");
/* 01 */      achar = fgetc (stdin);
/* 01 */      valid_id = valid_starter (achar);
/* 01 */      if (valid_id)
/* 02 */                  length = 1;
/* 03 */      achar = fgetc (stdin);
/* 04 */      while (achar != '\n')
/* 05 */      {
/* 05 */                  if (!(valid_follower (achar)))
/* 06 */                              valid_id = 0;
/* 07 */                  length++;
/* 07 */                  achar = fgetc (stdin);
/* 07 */      }
/* 08 */      if (valid_id && (length >= 1) && (length < 6) )
/* 09 */                  printf ("Valido\n");
/* 10 */      else
/* 10 */                  printf ("Invalido\n");
/* 11 */  }
```



Critérios de teste estrutural

- Baseia-se em diferente tipos de conceitos para determinar os requisitos de teste
  - Nó → todos-nós
  - Arco → todos-arcos
  - Laço → Boundary-Interior
  - Caminho → todos-caminhos
- Em geral os critérios são classificados em:
  - Baseados em fluxo de controle
  - Baseado em fluxo de dados
  - Baseados na complexidade

Baseado em Complexidade

- Requer que a complexidade do programa seja calculada
- Utiliza esta informação para derivar os requisitos de teste
- Utiliza-se de caminhos e complexidade dos mesmos para escolher e executar os testes

Teste Estrutural Baseado em Complexidade

- Possui como um dos critérios o calculo da complexidade ciclomática após a realização do grafo de programa
- A complexidade ciclomática oferece um limite máximo para o número de caminhos linearmente independentes
  - Um caminho linearmente independente é qualquer caminho do programa que introduza pelo menos um novo conjunto de instruções de processamento ou uma nova condição que deverá ser testada
- Estes caminhos estabelecem um conjunto básico para o grafo de fluxo de controle e se os casos de teste forem projetados para executar estes caminhos é garantido que cada instrução do programa será executada pelo menos uma vez
  - conseqüentemente um limite máximo do número de casos de teste que deve ser projetado e executado para garantir a total cobertura do programa

Teste Estrutural Baseado em Complexidade

- Existem três maneiras existem de realizar o calculo, são elas:
  - Estabelecendo o número de regiões em um grafo. Uma região pode ser descrita como uma área incluída no plano do grafo (grafo deve ser planar). Grafo planar é o grafo que pode ser reapresentado no plano sem que as arestas se cruzem. O número de regiões é estabelecido contando todas as áreas delimitadas e a área não delimitada fora do grafo.
  - V(G) = E-N+2, onde E é o total de arcos e N o total de nós do grafo.
  - V(G)=P+1, onde P é o número de nós predicativos contido no grafo de fluxo de controle. Nós predicativos são os nós que possuem a execução de um comando que realiza um desvio no programa.

Teste Estrutural Baseado em Complexidade

A questão abaixo refere-se ao seguinte trecho de programa

```
begin
      read (a,b,c)
      tipo = "escaleno"
      if (a=b) or (b=c) or (a=c) then
            tipo = "isosceles";
      if (a=b) and (b=c) then
            tipo = "equilátero";
      if (a>=b+c) or (b>=a+c) or (c>=a+b) then
            tipo = "não é um triangulo";
      if (a<=0) or (b<=0) or (c<=0) then
            tipo = "dados inválidos";
      write (tipo)
end
```

Considere as seguintes afirmativas:

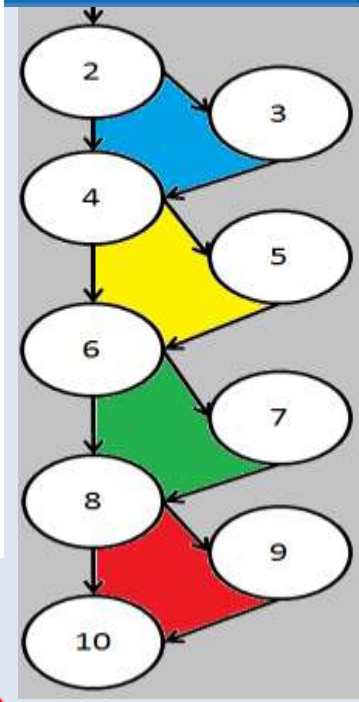
I - É possivel exercitar todos os comandos do programa com 5 casos de teste.
II - Um limite superior do número de caminhos linearmente independentes do grafo de fluxo do programa é 4.
III - Admitindo que os nós do grafo de fluxo possam representar condições compostas, e que, portanto, cada comando do programa acima possa ser representado num único nó, o número de regiões de seu grafo de fluxo é 4.

Assinale a alternativa CORRETA:

A. Apenas a afirmativa I é verdadeira.
B. Apenas a afirmativa II é verdadeira
C. Apenas a afirmativa III é verdadeira
D. Apenas as afirmativas I e II são verdadeiras
E. Todas as afirmativas são verdadeiras.

Teste Estrutural Baseado em Complexidade

- Para o exercício são cinco regiões (azul, amarela, verde, vermelha e cinza) o que torna a alternativa I verdadeira e II e III falsas.
- V(G) = E-N+2, onde E é o total de arcos e N o total de nós do grafo. Neste caso V(G) = 13-10+2 = 5. Portanto, a alternativa I é verdadeira e as II e III são falsas.
- V(G)=P+1, onde P é o número de nós predicativos contido no grafo de fluxo de controle. Nós predicativos são os nós que possuem a execução de um comando que realiza um desvio no programa (2,4,6 e 8). VG=4+1. Portanto, a alternativa I é verdadeira e as II e III são falsas.



Considere as seguintes afirmativas:

I - É possivel exercitar todos os comandos do programa com 5 casos de teste.
II - Um limite superior do número de caminhos linearmente independentes do grafo de fluxo do programa é 4.
III - Admitindo que os nós do grafo de fluxo possam representar condições compostas, e que, portanto, cada comando do programa acima possa ser representado num único nó, o número de regiões de seu grafo de fluxo é 4.

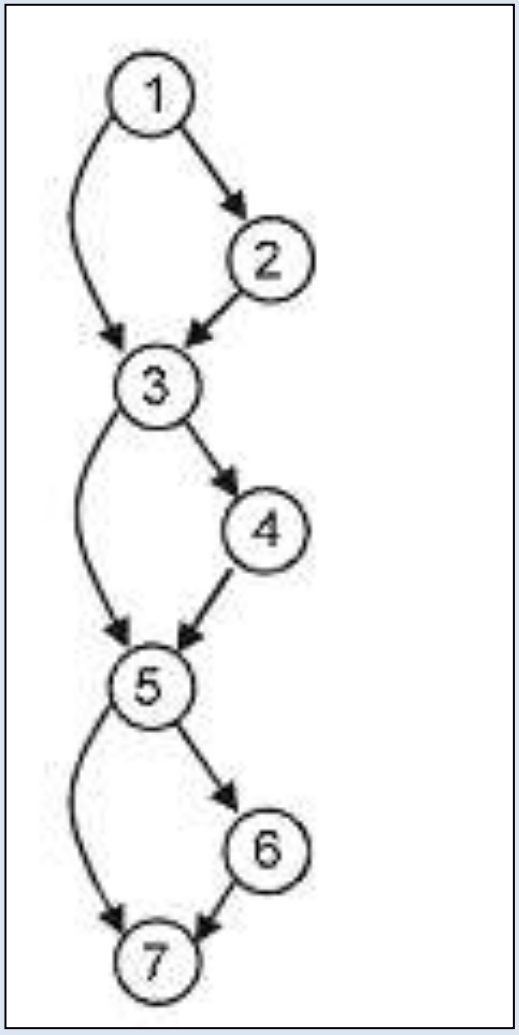
Assinale a alternativa CORRETA:

A. Apenas a afirmativa I é verdadeira.
B. Apenas a afirmativa II é verdadeira
C. Apenas a afirmativa III é verdadeira
D. Apenas as afirmativas I e II são verdadeiras
E. Todas as afirmativas são verdadeiras.

Teste Estrutural Baseado em Complexidade

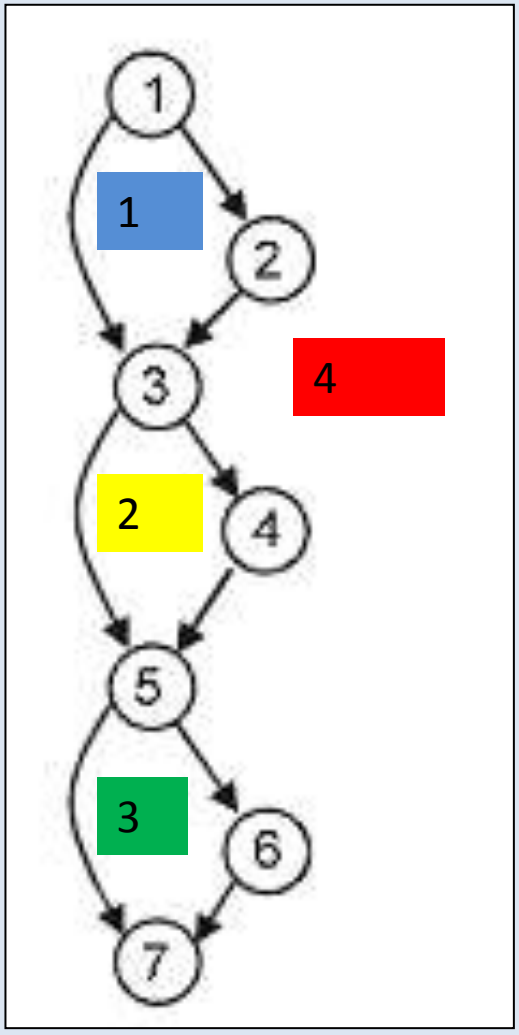
Ao longo de todo o desenvolvimento do software, devem ser aplicadas atividades de garantia de qualidade de software (GQS), entre as quais se encontra a atividade de teste. Um dos critérios de teste utilizados para gerar casos de teste é o denominado critério dos caminhos básicos, cujo número de caminhos pode ser determinado com base na complexidade ciclomática. Considerando-se o grafo de fluxo de controle apresentado na figura ao lado, no qual os nós representam os blocos de comandos e as arestas representam a transferência de controle, qual a quantidade de caminhos básicos que devem ser testados no programa associado a esse grafo de fluxo de controle, sabendo-se que essa quantidade é igual à complexidade ciclomática mais um?

(A) 1.
(B) 3.
(C) 4.
(D) 7.
(E) 8.



Teste Estrutural Baseado em Complexidade

Ao longo de todo o desenvolvimento do software, devem ser aplicadas atividades de garantia de qualidade de software (GQS), entre as quais se encontra a atividade de teste. Um dos critérios de teste utilizados para gerar casos de teste é o denominado critério dos caminhos básicos, cujo número de caminhos pode ser determinado com base na complexidade ciclomática. Considerando-se o grafo de fluxo de controle apresentado na figura ao lado, no qual os nós representam os blocos de comandos e as arestas representam a transferência de controle, qual a quantidade de caminhos básicos que devem ser testados no programa associado a esse grafo de fluxo de controle, sabendo-se que essa quantidade é igual à complexidade ciclomática mais um?



(A) 1.
(B) 3.
(C) 4.
(D) 7.
(E) 8.

V(G) = 9-7+2=4, onde E é o total de arcos e N o total de nós do grafo.
V(G)=3+1, onde P é o número de nós predicativos contido no grafo de fluxo de controle. Nós predicativos são os nós que possuem a execução de um comando que realiza um desvio no programa.

Teste Estrutural Baseado em Complexidade

Faça o cálculo para o programa abaixo

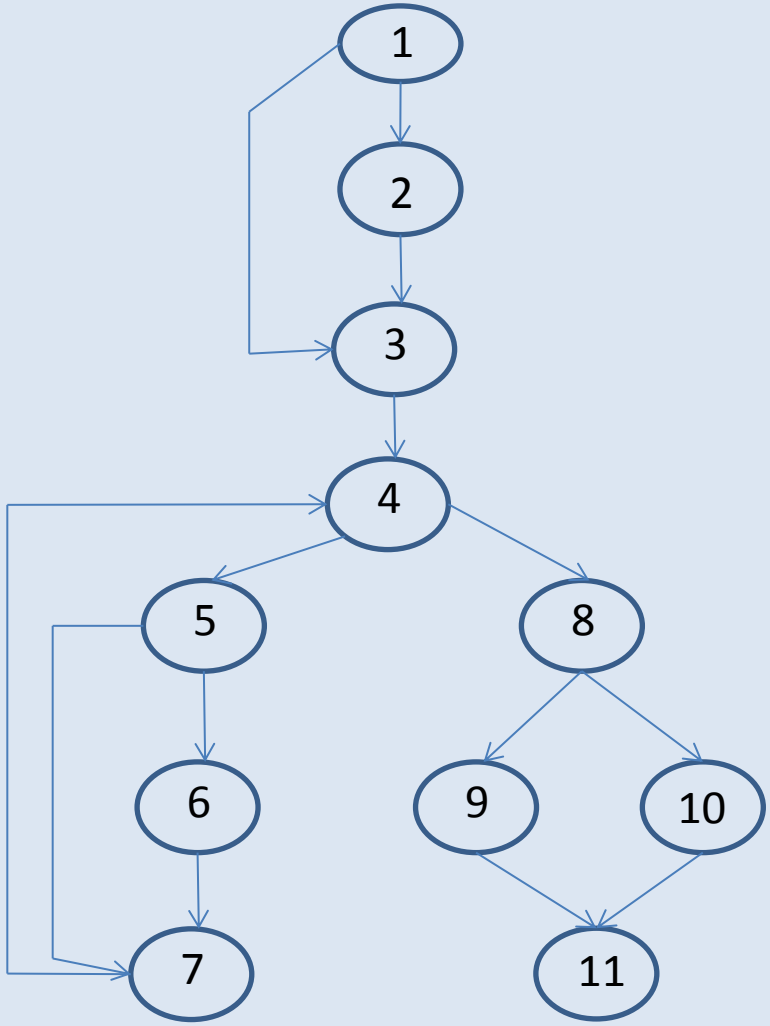
```
public static void search ( int key, int [] elemArray, Result r )
{
  int bottom = 0 ;
  int top = elemArray.length - 1 ;
  int mid ;
  r.found = false ; r.index = -1 ;
  while ( bottom <= top )
   {
     mid = (top + bottom) / 2 ;
     if (elemArray [mid] == key)
     {
      r.index = mid ;
      r.found = true ;
      return ;
     }
    else
   {
     if (elemArray [mid] < key)
        bottom = mid + 1 ;
     else
         top = mid - 1 ;
    }
  }
} // search
```

Baseado em Fluxo de Controle

- Utiliza-se de controle da execução do programa
  - Desvios
  - Comandos
- Critérios:
  - Todos-nós
    - Exige que passe ao menos uma vez por cada nó
      - Cada comando uma vez
  - Todos-arcos
    - Cada desvio do fluxo do programa seja executado uma vez
      - aresta
  - Todos-caminhos
    - Todos os caminhos possíveis sejam executados
  - Boundary-interior
    - Baseia-se nos laços do programa

Baseado em Fluxo de Controle

- Todos-nós
  - 1,2,3,4,5,6,7,8,9,10,11
- Todos-arcos
  - (1,2)(2,3)
  - (1,3)
  - (3,4)
  - (4,5)(5,6)(6,7)
  - (5,7)
  - (7,4)
  - (4,8)(8,9)(9,11)
  - (8,10)(10,11)
- Todos-caminhos
  - (1,2,3,4,5,6,7)
  - (1,2,3,4,8,9,11)
  - (1,2,3,4,8,10,11)
  - (1,3,4,.....)

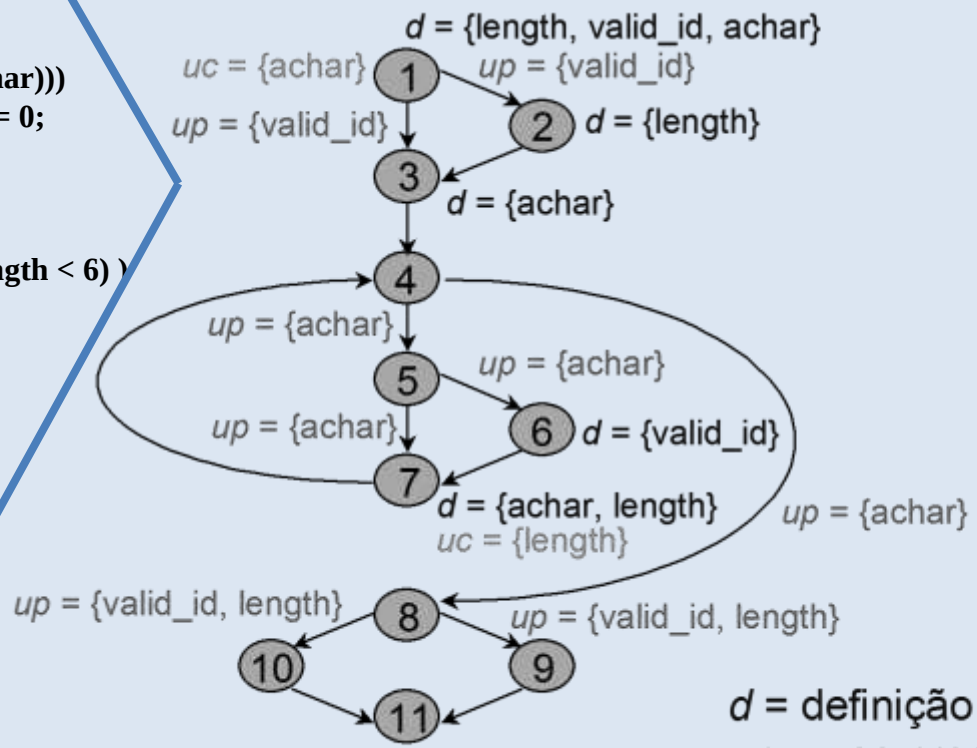


Baseado em Fluxo de Dados

- Utiliza-se de informações do fluxo de dados do programa para determinar os requisitos de teste
- São estabelecidos por intermédio de definições de variáveis e referências a elas
- São adicionadas informações a respeito do fluxo de dados do programa no grafo de programa
  - Atribuição de valores a uma variável
- Tipos
  - Todas-definições
    - Requer que cada definição de variável seja exercitada pelo menos uma vez
  - Todos-usos
    - Requer que todas as associações entre uma definição de variável e seus usos sejam exercitadas pelos casos de teste
    - Tipos de uso
      - Uso computacional → afeta diretamente a computação
      - Uso predicativo → afeta diretamente o fluxo de controle do programa

Baseado em Fluxo de Dados

```
/* 01 */  {
/* 01 */        char  achar;
/* 01 */        int   length, valid_id;
/* 01 */        length = 0;
/* 01 */        printf ("Digite um possível identificador\n");
/* 01 */        printf ("seguido por <ENTER>: ");
/* 01 */        achar = fgetc (stdin);
/* 01 */        valid_id = valid_starter (achar);
/* 01 */        if (valid_id)
/* 02 */                    length = 1;
/* 03 */        achar = fgetc (stdin);
/* 04 */        while (achar != '\n')
/* 05 */        {
/* 05 */                    if (!(valid_follower (achar)))
/* 06 */                                valid_id = 0;
/* 07 */                    length++;
/* 07 */                    achar = fgetc (stdin);
/* 07 */        }
/* 08 */        if (valid_id && (length >= 1) && (length < 6) )
/* 09 */                    printf ("Valido\n");
/* 10 */        else
/* 10 */                    printf ("Invalido\n");
/* 11 */  }
```



Grafo Def-Uso do *identifier*

d = definição
up = uso predicativo
uc = uso computacional

Baseado em Fluxo de Dados

- Todas-definições?
- Todos-usos?
  - Exemplo
    - Todas-definições→ (1,2, length,valid_id,achar) (3,4, achar)