Teste Estrutural para Engenharia de Software

Teste Fluxo do Grafo de Dados (FGD)

v3

Considerações Def-Uso

- A partir das definições e usos de variáveis de um determinado código é possível criar um **Grafo Def-Uso** para ele.

- Nele são adicionadas informações a respeito do fluxo de dados, caracterizando associações entre pontos do programa nos quais é atribuído um valor a uma variável e pontos nos quais esse valor é utilizado.

Conceito

- Teste estrutural (caixa-branca) extensão do Grafo Fluxo de Controle;
- Utiliza a análise do fluxo de dados em Grafo como fonte de informação para derivar os requisitos de teste;
- Baseia-se nas **definições e uso de variáveis**, ou seja, nas associações entre a definição de uma variável e seus possíveis usos.
- *Um caminho não executado ou somente com definição ou sem uso de variável, indica possível falha em caso de teste de software.*

Critérios para definições (Def)
quando o valor é atribuído à variável pela 1ª. Vez e em posteriores com alteração de valor:

- i) no lado esquerdo de um comando de atribuição;
- ii) em um comando de entrada; ou
- iii)em chamadas de procedimentos e retorno de parâmetro de saída.

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

Critérios para uso (Uso) quando o valor é processado:

- A ocorrência de uma variável é um uso quando a referência a essa variável não a estiver definindo. Dois tipos de usos são distinguidos: uso-c e uso-p;

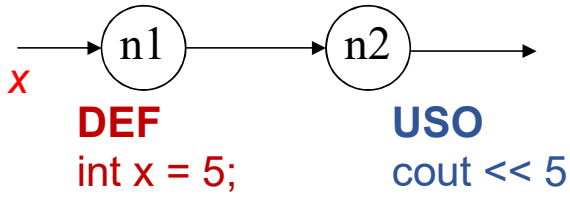
- Uso-c afeta diretamente uma computação sendo realizada ou permite que o resultado de uma definição seja observada (operação aritmética, saídas de impressão e afins);

- Uso-p afeta diretamente o fluxo de controle do programa nas sentenças predicados.

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

Par Def – Uso (Uso-c; Uso-p)

Um par def-uso (DU) para a variável *x* é um par de nós (n1, n2), de modo que *x* está em **DEF** (n1). A definição de *x* em n1 atinge n2, *x* está **USO** em (n2)



Em outras palavras, o valor atribuído a *x* em n1 é usado em n2 uma vez que a definição atinge n2, o valor não é morto ao longo de algum caminho n1 ... n2.

Ocorrências de variáveis

- **Definição (def):** ocorre quando uma variável recebe um valor declarativo na primeira ocorrência e em ocorrências posteriores quando altera o valor, por exemplo:

```
int var = 0;  ...    var = var + 1;
```

- **Uso Computacional (uso-c):** ocorre quando a variável é utilizada em uma computação, por exemplo a operação aritmética, saída para impressão, envio de parâmetros:

```
var = var + 1 ;
```

- **Uso Predicativo (uso-p):** ocorre quando a variável é utilizada em uma condição lógica, por exemplo:

```
if (var != '\n' ) {...}
```

Exemplo de métrica básica

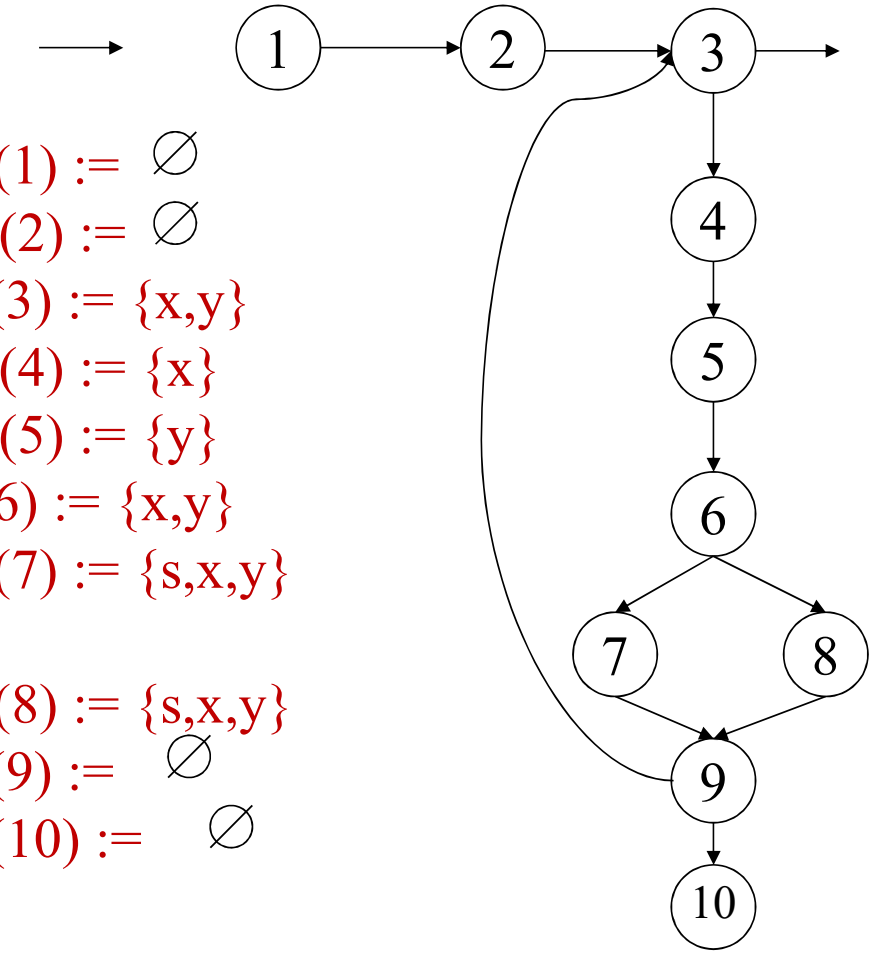
Def e *Use* (Caminhos Genéricos)

Exemplo

```
void proces(int y){

1 int s = 0;
2 int x = 0;
3  while (x<y) {
4     x = x+3;
5     y = y+2;
6     if (x+y<10)
7        s = s+x+y;
      else
8        s = s+x-y;

9  }

10}
```

Def(1) := {s}, Use(1) := ∅
Def(2) := {x}, Use(2) := ∅
Def(3) := ∅, Use(3) := {x,y}
Def(4) := {x}, Use(4) := {x}
Def(5) := {y}, Use(5) := {y}
Def(6) := ∅, Use(6) := {x,y}
Def(7) := {s}, Use(7) := {s,x,y}

Def(8) := {s}, Use(8) := {s,x,y}
Def(9) := ∅, Use(9) := ∅
Def(10) := ∅, Use(10) := ∅



Sobre *def* e *uso* efetivos (para variável x)

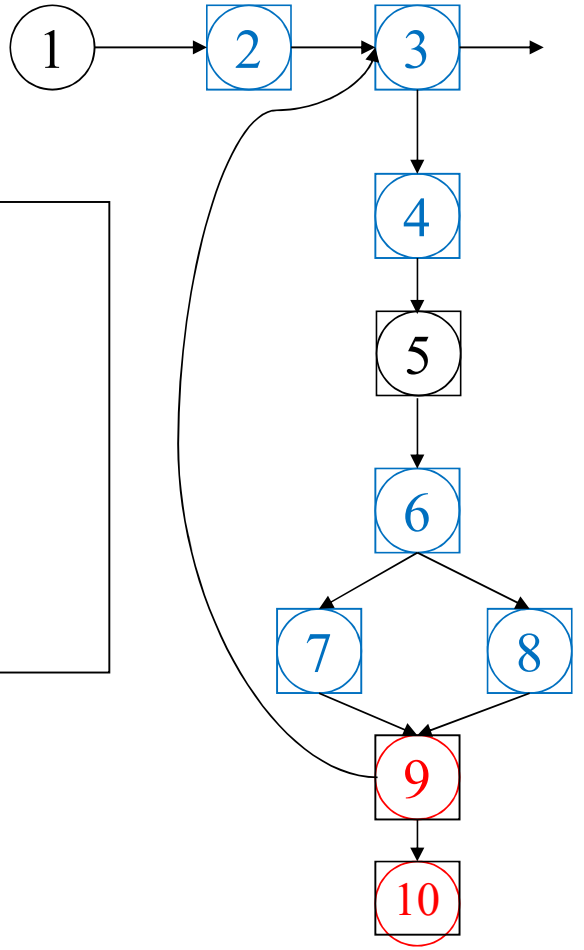
Uma definição de variável x no nodo n1 alcança nodo n2 se e somente se houver um caminho entre n1 e n2 que não contenha uma definição de x.

DEF(1) := {s}, USE(1) := ∅
DEF(2) := {x}, USE(2) := ∅
DEF(3) := ∅, USE(3) := {x,y}
DEF(4) := {x}, USE(4) := {x}
DEF(5) := {y}, USE(5) := {y}
DEF(6) := ∅, USE(6) := {x,y}
DEF(7) := {s}, USE(7) := {s,x,y}
DEF(8) := {s}, USE(8) := {s,x,y}

Caminhos genéricos:

1,2,3,4,5,6,7

1, 2,3,4,5,6,8



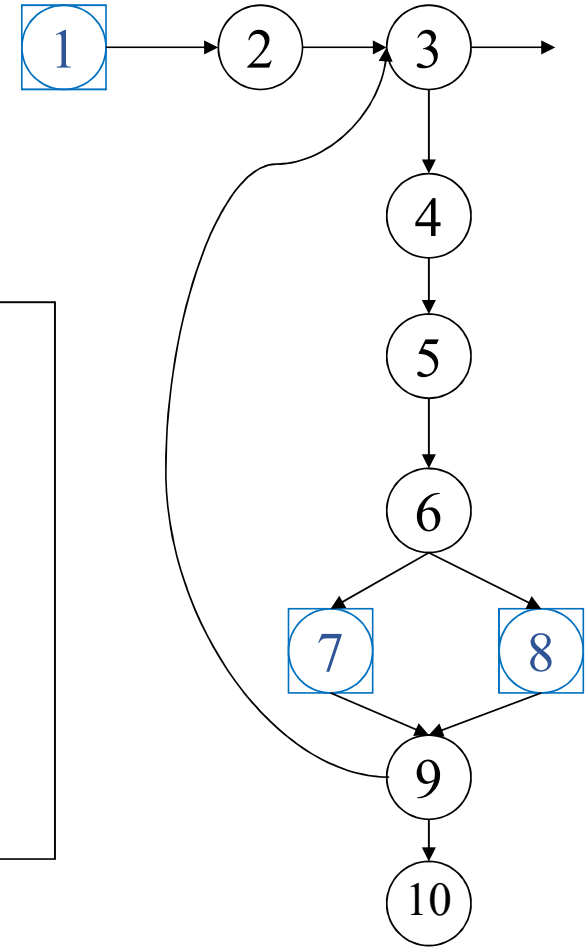

```
void proces(int y){

1 int s = 0;
2 int x = 0;
3  while (x<y) {
4     x = x+3;
5     y = y+2;
6     if (x+y<10)
7        s = s+x+y;
      else
8        s = s+x-y;

9  }

10}
```

Exemplo do par Def-Use (para variável s)

Definição - Uso, dois pares DU:
1-7,

1-8

DEF(1) := {s}, USE(1) :=∅
DEF(2) := {x}, USE(2) := ∅
DEF(3) := ∅ , USE(3) := {x,y}
DEF(4) := {x}, USE(4) := {x}
DEF(5) := {y}, USE(5) := {y}
DEF(6) := ∅ , USE(6) := {x,y}
DEF(7) := {s}, USE(7) := {s,x,y}
DEF(8) := {s}, USE(8) := {s,x,y}



Exemplo de Análise com Def-Uso
Critérios TODAS-DEFINIÇÕES e TODOS-USOS

Exemplo para análise com a variável *length*

Considerações para caso de teste de software

- **Todas-Definições:** cada definição de variável seja exercitada pelo menos uma vez, não importa se por um uso-c ou por um uso-p, *critério genérico*.

- **Todos-Usos:** todas as associações entre uma definição de variável e seus usos (*usos-c e usos-p*) sejam exercitadas pelos casos de teste, através de pelo menos um caminho livre de definição, ou seja, um caminho onde a variável não é redefinida.

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

```
#include <stdio.h>
int valid_s(char ch);
int valid_f(char ch);
main (){
/* 1 */ char achar;
/* 1 */ int length, valid_id;
/* 1 */ length = 0;
/* 1 */ valid_id = 1;
/* 1 */ printf ("Identificador: ");
/* 1 */ achar = fgetc (stdin);
/* 1 */ valid_id = valid_s(achar);
/* 1 */ if(valid_id){
/* 2 */ length = 1;
/* 2 */ }
/* 3 */ achar = fgetc (stdin);
/* 4 */ while(achar != '\n'){
/* 5 */ if(!(valid_f(achar))){
/* 6 */ valid_id = 0;
/* 6 */ }
/* 7 */ length++;
/* 7 */ achar = fgetc (stdin);
/* 7 */ }
/* 8 */ if(valid_id && (length >= 1) && (length < 6)){
/* 9 */ printf ("Valido\n"); }
/* 10 */ else {
/* 10 */ printf ("Invalid\n");}
/* 11 */ }
```

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

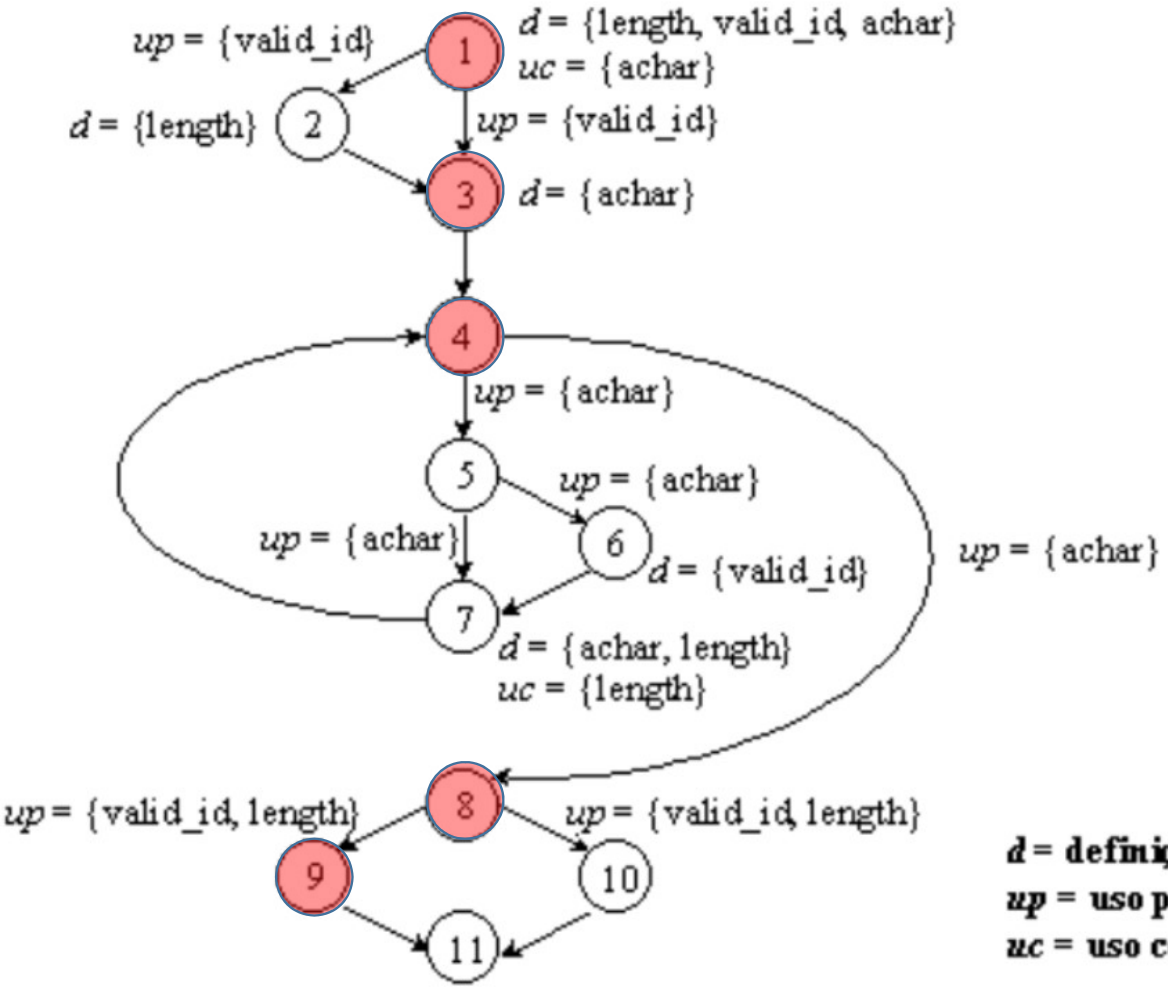
Análise com a variável <*length*>

O subcaminho **(1,3,4,8,9)** é não executável

length não recebe outro valor nenhum

```
int valid_s(char ch) {
/* 1 */ if(((ch >= 'A') && (ch <= 'Z')) ||
           ((ch >= 'a') && (ch <= 'z'))) {
/* 2 */ return (1);
/* 2 */ }
/* 3 */ else
/* 3 */ {
/* 3 */ return (0);
/* 3 */ }
/* 4 */ }

int valid_f(char ch) {
/* 1 */ if(((ch >= 'A') && (ch <= 'Z')) ||
           ((ch >= 'a') && (ch <= 'z')) ||
           ((ch >= '0') && (ch <= '9'))) {
/* 2 */ return (1);
/* 2 */ }
/* 3 */ else
/* 3 */ {
/* 3 */ return (0);
/* 3 */ }
/* 4 */ }
```

1 - Exemplo de caminhos ***Todas-definições***



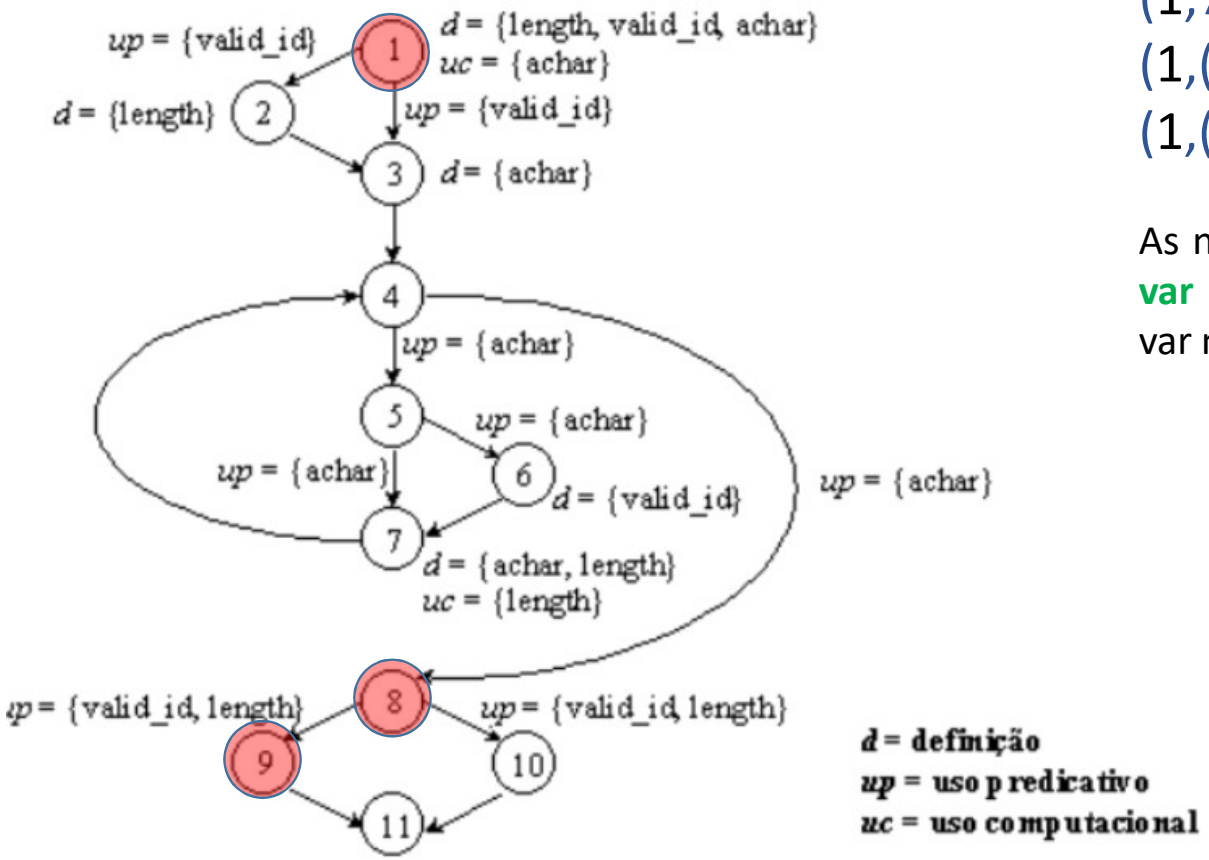
d = definição
up = uso predicativo
uc = uso computacional

Caminhos critério TODAS-DEFINIÇÕES:
(1,3,4,5,7);
(1,3,4,8,9);
(1,3,4,8,10);
(1,3,4,5,6,7); e
(1,2,3...)

O subcaminho **(1,3,4,8,9)**
é não executável,
e qualquer caminho completo que
o inclua também é não executável.

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

2 - Exemplo critério ***Todos-Usos***



Em relação ao critério Todos-Usos, seriam requeridas as associações:

(1,7, length);

(1,(8,9),length) e

(1,(8,10), length)

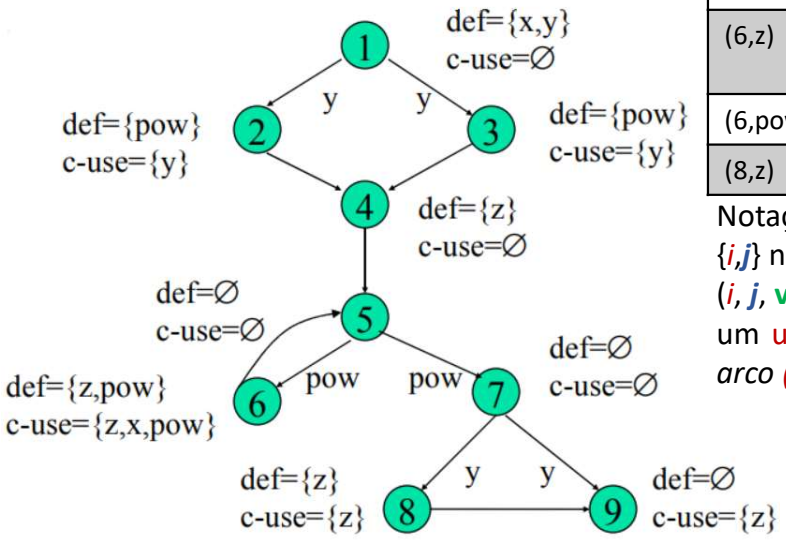
As notações (i, ***j***, **var**) e (i,(***j***, ***k***),**var**) indicam que a variável **var** é definida no nó *i* e existe um uso computacional de var no nó *j* ou um uso predicativo de **var** no *arco (j, k).*

A associação **(1,(8,9), length)** é não executável pois o único caminho que livre de definição possível de exercitá-la seria um caminho que incluísse o subcaminho (1,3,4,8,9).

Já para **(1,7,length)** qualquer caminho completo executável incluindo um dos subcaminhos (1,3,4,5,6,7), (1,3,4,5,7) seria suficiente para exercitá-la.

Análises pelos Critérios: Genérico; Todas-Definições e Todos-Usos
Caminhos para Casos de Teste

```
#include <iostream>
using namespace std;
int main(){
/*1*/int x,y; float z,pow;
/*1*/cin >> x >> y;
/*1*/if (y<0){
/*2*/    pow=0-y;}
     else {
/*3*/    pow=y;
     }
/*4*/z=1.0;
/*5*/while (pow !=0){
/*6*/     z=z*x;
/*6*/     pow=pow-1;
    }
/*7*/if (y<0)
/*8*/     z=1.0/z;
/*9*/cout << "Saida: " << z;
}
```



Resultado da métrica GFD para todas variáveis.
Caminhos que devem ser Casos de Testes

(Nodo definição,var)	Definição – Uso-c		Definição – Uso-p		Caminhos Genéricos Para Casos de Testes
	Def -> Use-c	(i, j, var)	Def -> Use-p	(i,(j, k),var)	
(1,x)	{6}	(1,6,x)	∅	∅	1,2,4,5,6 1,3,4,5,6
(1,y)	{2,3}	(1,3,y)	{(1,2),(1,3),(7,8),(7,9)}	(1(7,8),y) (1(7,9),y)	1,2,4,5,7,8,9; 1,3,4,... 1,3,4,5,7,9; 1,2,4,...
(2,pow)	{6}	(2,6,pow)	{(5,6),(5,7)}	(2,(5,6),pow) (2,(5,7),pow)	2,4,5,6 2,4,5,7
(3,pow)	{6}	(3,6,pow)	{5,6),(5,7)}	(3,(5,6),pow) (3,(5,7),pow)	3,4,5,6 3,4,5,7
(4,z)	{6,8,9}	(4,6,z) (4,8,z) (4,9,z)	∅	∅	4,5,6 4,5,7,8 4,5,7,9
(6,z)	{6,8,9}	(6,8,z)	∅	∅	6,5,7,8,9 6,5,7,9
(6,pow)	{6}	(6,6,pow)	{(6,6)}	∅	6
(8,z)	{9}	(8,9,z)	∅	∅	8,9

Notações:
{*i*,*j*} nodos em uso;
(*i*, *j*, **var**) e (*i*,(*j*, *k*),**var**) indicam que a variável **var** é definida no nó *i* e existe um uso computacional de var no nó *j*; ou um uso predicativo de **var** no *arco (j, k).*

Todas-definições:
1,3,4,5,6,5,7,8,9
1,2,4,5,6,5,7,8,9
1,3,4,5,7,8,9
1,2,4,5,7,8,9
1,3,4,5,7,9
1,2,4,5,7,9

Referências:

Maldonado, José Carlos et all. INTRODUÇÃO AO TESTE DE SOFTWARE (Versão 2004-01), ICMC/USP.

Arndt von Staa, Departamento de Informática, PUC-Rio, Outubro 2008

Purdue University, Ramkumar Natarajan, Baskar Sridharan,Guidant Corporation, August 12-16, Minneapolis/St Paul, MN

Algoritmos para Análise com a variável *length* com critério genérico

```
#include <stdio.h>
int valid_s(char ch);
int valid_f(char ch);
main (){
/* 1 */ char achar;
/* 1 */ int length, valid_id;
/* 1 */ length = 0;
/* 1 */ valid_id = 1;
/* 1 */ printf ("Identificador: ");
/* 1 */ achar = fgetc (stdin);
/* 1 */ valid_id = valid_s(achar);
/* 1 */ if(valid_id){
/* 2 */ length = 1;
/* 2 */ }
/* 3 */ achar = fgetc (stdin);
/* 4 */ while(achar != '\n'){
/* 5 */ if(!(valid_f(achar))){
/* 6 */ valid_id = 0;
/* 6 */ }
/* 7 */ length++;
/* 7 */ achar = fgetc (stdin);
/* 7 */ }
/* 8 */ if(valid_id && (length >= 1) && (length < 6)){
/* 9 */ printf ("Valido\n"); }
/* 10 */ else {
/* 10 */ printf ("Invalid\n");}
/* 11 */ }
```

```
int valid_s(char ch) {
/* 1 */ if(((ch >= 'A') && (ch <= 'Z')) ||
          ((ch >= 'a') && (ch <= 'z'))) {
/* 2 */ return (1);
/* 2 */ }
/* 3 */ else
/* 3 */ {
/* 3 */ return (0);
/* 3 */ }
/* 4 */ }

int valid_f(char ch) {
/* 1 */ if(((ch >= 'A') && (ch <= 'Z')) ||
          ((ch >= 'a') && (ch <= 'z')) ||
          ((ch >= '0') && (ch <= '9'))) {
/* 2 */ return (1);
/* 2 */ }
/* 3 */ else
/* 3 */ {
/* 3 */ return (0);
/* 3 */ }
/* 4 */ }
```

Algoritmos para Análise com os critério Genérico e Critério Todas-Definições e Todos-Usos

```
#include <iostream>
using namespace std;
int main(){
int x,y; float z,pow;
cin >> x >> y;
if (y<0)
   pow=0-y;
else
   pow=y;
z=1.0;
while (pow !=0){
    z=z*x;
    pow=pow-1;
}
if (y<0)
    z=1.0/z;
cout << "Saida: " << z;
}
```