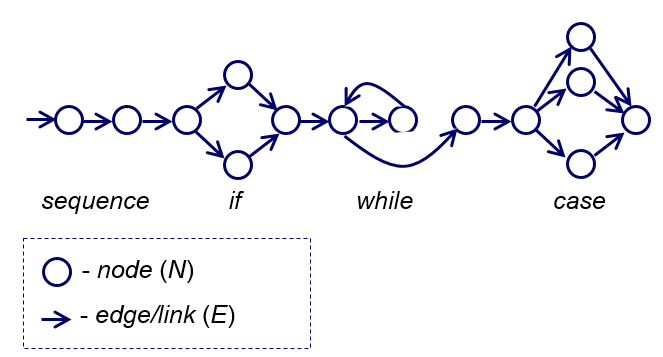
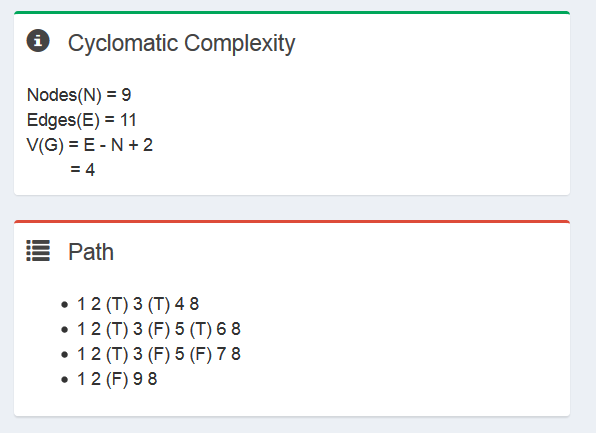
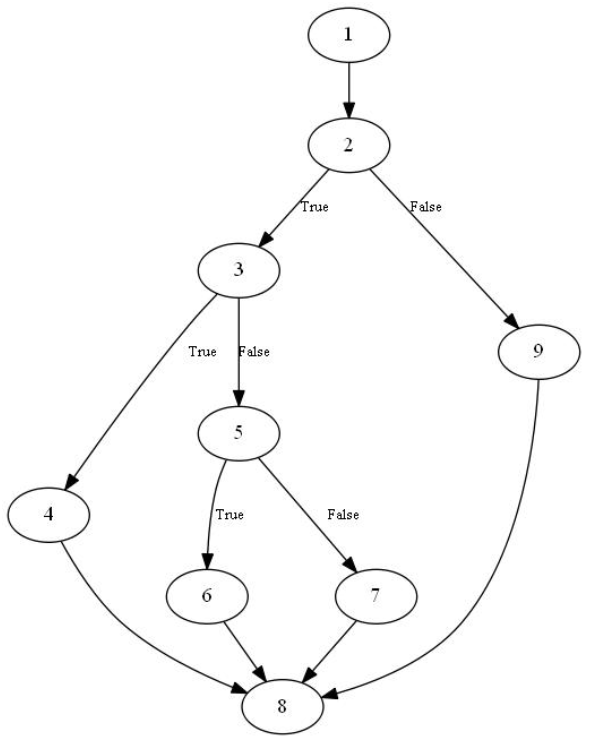
Pengujian Source Code Instrumentation



Contoh:

1. tA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | function type = triangle(sideLengths) **Node 1**  A = sideLengths(1); % First side  B = sideLengths(2); % Second side  C = sideLengths(3); % Third side  if ((A+B > C) && (B+C > A) && (C+A > B)) **Node 2**  if ((A ~= B) && (B ~= C) && (C ~= A)) **Node 3**  type = 'Scalene'; **Node 4**  else  if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ((C == A) && (A ~= B))) **Node 5**  type = 'Isosceles'; **Node 6**  else  type = 'Equilateral'; **Node 7**  end  end  else  type = 'Not a triangle'; **Node 9**  end **Node 8** |



CFG

Hasil Instruumentasi

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | function [traversedPath,type] = triangle(sideLengths)  traversedPath = [];  traversedPath = [traversedPath '1 ' ];  A = sideLengths(1); % First side  B = sideLengths(2); % Second side  C = sideLengths(3); % Third side  % instrument Branch # 1  traversedPath = [traversedPath '2 ' ];  if ((A+B > C) && (B+C > A) && (C+A > B))  traversedPath = [traversedPath '(T) ' ];  % instrument Branch # 2  traversedPath = [traversedPath '3 ' ];  if ((A ~= B) && (B ~= C) && (C ~= A))  traversedPath = [traversedPath '(T) ' ];  traversedPath = [traversedPath '4 ' ];  type = 'Scalene';  else  traversedPath = [traversedPath '(F) ' ];  % instrument Branch # 3  traversedPath = [traversedPath '5 ' ];  if (((A == B) && (B ~= C)) || ((B == C) && (C ~= A)) || ((C == A) && (A ~= B)))  traversedPath = [traversedPath '(T) ' ];  traversedPath = [traversedPath '6 ' ];  type = 'Isosceles';  else  traversedPath = [traversedPath '(F) ' ];  traversedPath = [traversedPath '7 ' ];  type = 'Equilateral';  end  end  else  traversedPath = [traversedPath '(F) ' ];  traversedPath = [traversedPath '9 ' ];  type = 'Not a triangle';  end  traversedPath = [traversedPath '8 ' ]; |

1. mmA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function miniMaxi = minimaxi(num)  numLength = length(num);  mini = num(1);  maxi = num(1);  idx = 2;  while (idx <= numLength) % Branching #1  if maxi < num(idx) % Branching #2  maxi = num(idx);  end  if mini > num(idx) % Branching #3  mini = num(idx);  end  idx = idx+1;  end % while end  miniMaxi = [mini maxi];  end |

1. iA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | function sortedArray = insertion(anyArray)  k = 1; % The smallest integer increment  n = length(anyArray);  I = 2;  for i=2:n  x = anyArray(i);  j = I – 1;  while ((j > 0) & (anyArray(j) > x)),  anyArray(j+1) = anyArray(j);  j = j - 1;  end  anyArray(j+1) = x;  end  sortedArray = anyArray;  end |

1. binA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | function itemIndex = binary(itemNumbers)  item = itemNumbers(1);  numbers = itemNumbers(1,2:end);  lowerIdx = 1;  upperIdx = length(numbers);  while (lowerIdx ~= upperIdx), % Branch # 1  temp = lowerIdx + upperIdx; % additional statement  if (mod(temp, 2) ~= 0),  temp = temp - 1;  end % additional statement  idx = temp / 2;  if (numbers(idx) < item), % Branch # 2  lowerIdx = idx + 1;  else  upperIdx = idx;  end  end  % Additional code that returns -1 if the item is not found  if (item == numbers(lowerIdx)),  temIndex = lowerIdx;  else  itemIndex = -1;  end  end |

1. bubA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | function sortedArray = bubble(anyArray)  sorted = 0; % 0 means false  i = 1; n = length(anyArray);  while ((i <= (n-1)) && ~sorted), % Branch # 1  sorted = 1;  j = n;  for j=n:-1:i+1 % Branch # 2  if (anyArray(j) < anyArray(j-1)) % Branch # 3  %exchange(anyArray(j), anyArray(j-1));  temp = anyArray(j);  anyArray(j) = anyArray(j-1);  anyArray(j-1) = temp;  sorted = 0;  end  end  i = i + 1;  end  sortedArray = anyArray;  end |

1. gA2008

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | function y = gcd(number)  a = number(1);  b = number(2);  if (a == 0),  y = b;  else  while b ~= 0  if a > b  a = a - b;  else  b = b - a;  end  end  y = a;  end  end |

1. eB2002

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | function result = expintBueno2002(numbersIn)  n = numbersIn(1); % integer  x = numbersIn(2); % floa  MAXIT = 100;  EULER = 0.5772156649;  FPMIN = 1.0e-30;  EPS = 1.0e-7;  nm1 = n - 1;  if (n < 0 || x < 0.0 || (x == 0.0 && (n == 0.0 || n==1)))  result = 0;  % disp(‘bad arguments in expintBueno2002’);  elseif (n == 0)  result = exp(-x)/x;  elseif (x == 0.0)  result = 1.0/nm1; % strangy: what is nm1?  elseif (x > 1.0)  b = x + n;  c = 1.0 / FPMIN;  d = 1.0 / b;  h = d;  for i=1 : MAXIT  a = -i \* (nm1 + i);  b = b + 2.0;  d = 1.0 / (a\*d+b);  c = b + a / c;  del = c \* d;  h = h \* del;  if (abs(del-1.0) < EPS) % abs is fabs in C  result = h \* exp(-x);  return;  end  end  disp(‘continuated fraction failed in expint’);  else  % ans = (nm1!=0 ? 1.0/nm1 : -log(x)-EULER);  % is interpreted as follows  if (nm1 ~= 0)  result = 1.0 / nm1;  else  result = -log(x)-EULER;  end  fact = 1.0;  for i = 1 : MAXIT  fact = fact \* (-x / i);  if (i ~= nm1)  del = -fact / (i - nm1);  else  psi = -EULER;  for ii = 1 : nm1  psi = psi + (1/ii);  end  del = fact \* (-log(x) + psi);  end  result = result + del;  if (abs(del) < abs(result) \* EPS) % abs is fabs in C  return;  end  end  disp(‘series failed in expint’);  end  end |

1. qB2002

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | function [q, r] = quotientBueno2002(operands)  n = operands(1); % First number  d = operands(2); % Second number  q = 0;  if (d ~= 0)  if ( (d > 0) && (n > 0) )  q = 0;  r = n;  t = d;  while (r >= t)  t = t \* 2;  end  while (t ~= d)  q = q \* 2;  t = t / 2;  if (t <= r)  r = r - t;  q = q + 1;  end  end  end  end  end |

1. fB2002

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47 | function a = findBueno2002(numbersIn)  f = numbersIn(1); % key or index  a = numbersIn(2:end); % an array of integers to be re-arranged  % n = length(numbers);  b = 0;  m = 1;  ns = length(a);  % Probe added on 02.09.2010  if f > ns  f = mod(ns,f);  end  i = 1;  while ((m < ns) || b)  if (~b)  i = m;  j = ns;  else  b = 0;  end  if (i > j)  if (f > j)  if (i > f)  m = ns;  else  m = i;  end  else  ns = j;  end  else  while (a(i) < a(f))  i = i + 1  end  while (a(f) < a(j))  j = j - 1 ;  end  if (i <= j)  w = a(i);  a(i) = a(j);  a(j) = w;  i = i + 1;  j = j - 1;  end  b = 1;  end  end  end |

1. fmH2014

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | function branchVal = fitnessMiniMaxi(branchNo, predicate)  k = 1; % the smallest step for integer  switch (branchNo)  case 1,  % branch #1: (idx <= numLength)  branchVal = predicate(1) - predicate(2);  case 2,  % branch #2: (maxi < num(idx))  branchVal = predicate(1) - predicate(2);  case 3,  % branch #3: (mini > num(idx))  branchVal = predicate(2) - predicate(1);  end  if ((branchNo == 2) || (branchNo == 3)),  if (branchVal < 0)  branchVal = branchVal - k;  else  branchVal = branchVal + k;  end  end  end |