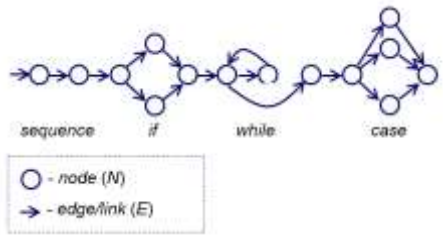
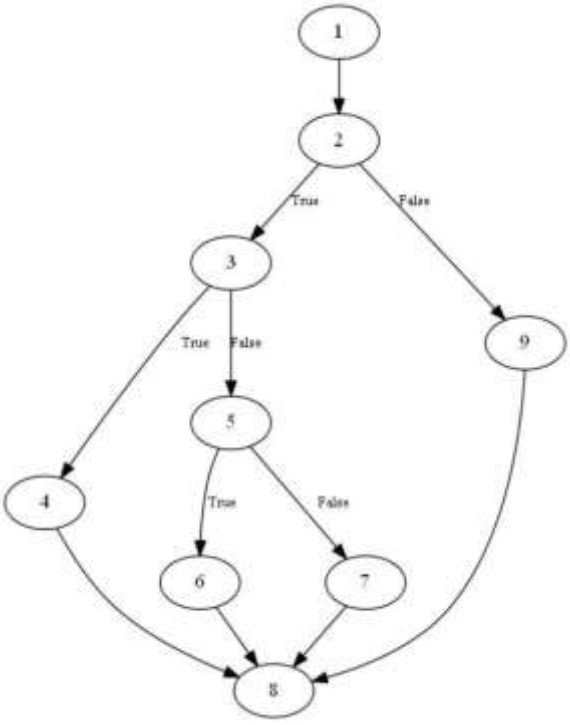
Pengujian Source Code Instrumentation

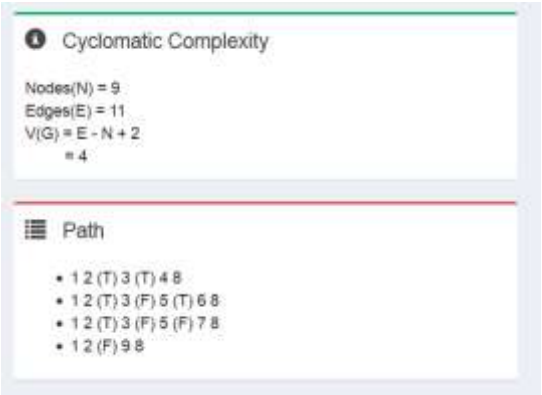


Contoh:

1. tA2008

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





CFG

Hasil Instruumentasi

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

2. mmA2008

```
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
```

3. iA2008

```
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
```

4. binA2008

```
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
```

5. bubA2008

```
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
```

6. gA2008

```
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
```

7. eB2002

```
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

```

8. qB2002

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

9. fB2002

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

10. fmH2014

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