Bisection Algorithm

```plaintext
restart;

bisection := proc(f::algebraic,a::numeric,b::numeric,tol::positive,no::posint,root::name)
local A, B, TOL, C, OK, X, F, FA, FB, I, P, FP;
makes sure endpoints and tolerance are of type floating point
A:=evalf(a);
B:=evalf(b);
TOL:=evalf(tol);
makes sure A<B
if A > B then
  X := A;
  A := B;
  B := X;
end if;
makes f into a function F and evaluate at endpoints
F:=unapply(f,x);
FA:=F(A);
FB:=F(B);
check for errors in choice of endpoints
if A = B then
  ERROR("The two endpoints of the interval \([a,b]\) must be different");
else if FA*FB > 0 then ERROR("The function values at \(a\) and \(b\) must have opposite signs");
end if;
end if;
print table headings
printf(`  i    p                  f(p)`);  
printf(`  -    -                  ----
`);  
execute the algorithm

STEP 1
  I := 1;

STEP 2
  OK := TRUE;
  while I <= no and OK = TRUE do
STEP 3
  Compute \(p_i\)
  C := (B - A) / 2.0;
P := A + C;
STEP 4
  FP := F(P);
  printf(`%3d   %15.8e   %15.7e 
`,I,P,FP);
  if abs(FP) < 1.0e-20 or C < TOL then
procedure completed successfully
```
printf(`The approximate solution is %a`,args[6]);
printf(` = %11.8f 
`,P);
printf(`with f(%a`,args[6]);
printf(`) = %12.8f
`,FP);
root:=P;
OK:=FALSE;
else

STEP 5
   l := l+1;

STEP 6
compute $a_i$ and $b_i$

    if FA*FP > 0 then
       A := P;
       FA := FP;
    else
       B := P;
       FB := FP;
    end if;
    end if;
    end do;
if OK = TRUE then

STEP 7

procedure completed unsuccessfully
    printf(`Iteration number %3d`,no);
    printf(` gave approximation %12.8f
`,P);
    printf(`F(P) = %12.8f not within tolerance : %15.8e
`,FP,TOL);
    RETURN();
else
    P;
end if;
end proc;

Warning, imaginary unit `I` used as a local variable in procedure bisection

bisection := proc(f::algebraic, a::numeric, b::numeric, tol::positive, no::posint, root::name)
local A, B, TOL, C, OK, X, F, FA, FB, I, P, FP;
    A := evalf(a);
    B := evalf(b);
    TOL := evalf(tol);
    if B < A then X := A; A := B; B := X end if
    F := unapply(f, x);
    FA := F(A);
    FB := F(B);
    if A = B then
       ERROR("The two endpoints of the interval [a,b] must be different")
    else
       if 0 < FA*FB then
          ERROR("The function values at a and b must have opposite signs")
       end if
    end if
printf `  i    p                  f(p)`
I := 1;
OK := TRUE;
while I <= no and OK = TRUE do
  C := (B - A)/(2.0);
P := A + C;
FP := F(P);
  printf `%3d   %15.8e   %15.7e`, I, P, FP;
if `abs` FP < 1.0*10^-20 or C < TOL then
  printf `The approximate solution is %a`, args[6]);
  printf( ` = %11.8f `, root := P;
  OK := FALSE
else
  I := I + 1;
  if 0 < FA*FP then A := P; FA := FP else B := P; FB := FP end if
end if
if OK = TRUE then
  printf `Iteration number %3d`, no);
  printf( ` gave approximation %12.8f `, FP, TOL);
end if
end do
if OK = TRUE then
  printf `The approximate solution is %a`, args[6]);
  printf( ` = %11.8f `, root := P;
else
  P
end if
end proc

>bisection_dir:=proc()
printf(`bisection returns a root of the given function.
`); printf(`The arguments for bisection are:
`); printf(`(1)function expression in x
`); printf(`(2)left end point
`); printf(`(3)right end point
`); printf(`(4)tolerance
`); printf(`(5)maximum number of iterations
`); printf(`(6)variable for returning root
`); printf(`If assigning the result to a variable, have the
`); printf(`variable and the 6th argument the same.
`); printf(`If r is the variable for returning the root
`); printf(`and has already been given a value,
`); printf(`the procedure should be preceded by the statement:
`); printf(`r:=r`
`); end:
bisection_dir := proc() 
printf(`bisection returns a root of the given function.
`); printf(`The arguments for bisection are:
`); printf(`(1)function expression in x
`); printf(`(2)left end point
`); printf(`(3)right end point
`);
printf '(4)tolerance';
printf '(5)maximum number of iterations';
printf '(6)variable for returning root.';
printf 'If assigning the result to a variable, have the';
printf 'variable and the 6th argument the same.';
printf 'If r is the variable for returning the root';
printf 'and has already been given a value, ';
printf 'the procedure should be preceded by the statement: ';
printf 'r:=';
end proc