```fortran
program bissection_method
  implicit none
  real :: a, b, c, f, tol, f_a, f_b, f_c
  integer :: getNIt, i, n
  logical :: verifyInterval, isValidInterval

  print *, "Type a precision"
  read (*,*) tol
  if ( tol<=0 ) then
    print *, "You should provide a positive tolerance value"
    stop
  end if

  call getInterval(a, b)
  isValidInterval = verifyInterval(f(a), f(b))
  do while(.not.isValidInterval)
    print *, "You should provide an interval where both f(a) and f(b) have different sign"
    call getInterval(a,b)
    isValidInterval = verifyInterval(f(a), f(b))
  end do

  n = getNIt()
  do i = 1, n
    c = (a+b)/2
    f_c = f(c)
    print *, "c: ", c
    print *, "f(c) = ", f_c
    print *, "Iteraction: ", i
    if ( (abs(f_c).eq.0).or.(abs(f_c)<=tol) ) then
      print *, "-------------\\-------------"
      print *, "The found root is ", c
      print *, "f(x = root) = ", f_c
      print *, "Number of iteraction used:", i
      stop
    end if
    f_a = f(a)
    f_b = f(b)
    if ( ((f_a>0).and.(f_c>0)).or.((f_a<0).and.(f_c<0)) ) then
      a = c
    else
      b = c
    end if
  end do

  if ( f_c>tol ) then
    print *, "The simulation require more number of interactions."
    print *, " Please, restart the program and type a greater number of iteractions"
  end if

end program bissection_method

subroutine getInterval(a, b)
implicit none
real :: a, b
print *, "Type a value for 'a'"
read (*,*) a

print *, "type a value for 'b'"
```

```fortran
read (*,*) b

end subroutine getInterval

function getNIt() result(n)
  implicit none
  integer :: n
  print *, "Type a number of interactions"
  read (*,*) n
  if ( n<=0 ) then
    print *, "You should provide a number of interaction more than zero"
    stop
  end if
end function getNIt

function f(x) result(y)
  implicit none
  real, intent(in) :: x
  real :: y
  y = sin(x*cos(x)-sin(x))
end function f

function verifyInterval(f_a, f_b) result(isValid)
  implicit none
  real, intent(in) :: f_a, f_b
  logical :: isValid
  isValid = .true.
  print *, f_a, f_b
  if ( f_a.eq.0 ) then
    print *, "The interval 'a' is a root of the equation"
    stop
  end if
  if ( f_b.eq.0 ) then
    print *, "The interval 'b' is a root of the equation"
    stop
  end if
  if(((f_a>0).and.(f_b>0)).or.((f_a<0).and.(f_b<0))) then
    isValid = .false.
  end if
end function verifyInterval
```