

NAG Library Routine Document

X07AAF

1 Purpose

X07AAF determines whether a floating-point number is finite.

2 Specification

```
FUNCTION X07AAF (X)
LOGICAL X07AAF
REAL (KIND=nag_wp) X
```

3 Description

X07AAF returns `.TRUE.` if and only if `X` is finite, and returns `.FALSE.` otherwise.

4 References

IEEE (2008) *Standard for Floating-Point Arithmetic* **IEEE Standard 754-2008** IEEE, New York.

5 Parameters

1: `X` – REAL (KIND=`nag_wp`) *Input*
On entry: the number whose status is to be determined.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Further Comments

This routine will return `.FALSE.` if the parameter `X` is either infinite or a NaN (Not A Number).

9 Example

This program creates various infinities, NaNs and normal numbers and distinguishes between them.

9.1 Program Text

```
Program x07aafe
!      X07AAF Example Program Text
!
!      Mark 24 Release. NAG Copyright 2012.
!
!      .. Use Statements ..
!      Use nag_library, Only: nag_wp, x02alf, x07aaf, x07abf, x07baf, x07bbf,    &
!                               x07caf, x07cbf
!      .. Implicit None Statement ..
!      Implicit None
```

```

! .. Parameters ..
Integer, Parameter                :: nout = 6
! .. Local Scalars ..
Real (Kind=nag_wp)                :: neginf, qnan, x, y, zero
! .. Local Arrays ..
Integer                            :: exmode(3), newexmode(3)
! .. Executable Statements ..
Write (nout,*) 'X07AAF Example Program Results'
Write (nout,*)

! Turn exception halting mode off for the three common exceptions
! overflow, division-by-zero, and invalid operation
Write (nout,*) 'Turn exception halting off ...'
exmode = (/0,0,0/)
Call x07cbf(exmode)

! Check that exception halting mode for the three common exceptions
! was really turned off
Call x07caf(newexmode)
Write (nout,99999) 'Exception halting mode is now: ', newexmode

! Look at some ordinary numbers
x = 1.0_nag_wp
Call diagnose('one',x)
x = -2.0_nag_wp
Call diagnose('-two',x)
zero = 0.0_nag_wp
Call diagnose('zero',zero)

! Generate an infinity and a NaN and look at their properties
Call x07baf(-1,neginf)
Call diagnose('-Infinity',neginf)
Call x07bbf(1,qnan)
Call diagnose('Quiet NaN',qnan)

! Do some operations which purposely raise exceptions
Write (nout,*)
Write (nout,*) 'Try to cause overflow - no trap should occur:'
x = x02alf()
y = -x*x
Write (nout,99998) 'y = -huge() * huge() = ', y

Write (nout,*)
Write (nout,*) 'Try to cause NaN - no trap should occur:'
y = zero/zero
Write (nout,99998) 'y = 0.0 / 0.0 = ', y

Write (nout,*)
Write (nout,*) 'Try to cause division by zero - no trap should occur:'
x = -1.0_nag_wp
y = x/zero
Write (nout,99998) 'y = -1.0 / 0.0 = ', y

99999 Format (1X,A,3I3)
99998 Format (1X,A,1P,E12.4)

```

Contains

Subroutine diagnose(c,x)

```

! .. Implicit None Statement ..
Implicit None
! .. Scalar Arguments ..
Real (Kind=nag_wp), Intent (In)    :: x
Character (*), Intent (In)        :: c
! .. Executable Statements ..
Write (nout,*)
Write (nout,99999) c, x

If (x07aaf(x)) Then
  Write (nout,*) '"" // c // "" is finite'

```

```

Else
  Write (nout,*) ''' // c // '" is not finite'
End If
If (x07abf(x)) Then
  Write (nout,*) ''' // c // '" is NaN'
Else
  Write (nout,*) ''' // c // '" is not NaN'
End If

If (x<0.0_nag_wp) Then
  Write (nout,*) ''' // c // '" compares less than zero.'
Else
  Write (nout,*) ''' // c // '" does not compare less than zero.'
End If
If (x==0.0_nag_wp) Then
  Write (nout,*) ''' // c // '" compares equal to zero.'
Else
  Write (nout,*) ''' // c // '" does not compare equal to zero.'
End If
If (x>0.0_nag_wp) Then
  Write (nout,*) ''' // c // '" compares greater than zero.'
Else
  Write (nout,*) ''' // c // '" does not compare greater than zero.'
End If

Return

99999  Format (1X,'Diagnosis of value "',A,'" which prints as ',1P,E12.4)
      End Subroutine diagnose

      End Program x07aafe

```

9.2 Program Data

None.

9.3 Program Results

X07AAF Example Program Results

```

Turn exception halting off ...
Exception halting mode is now:  0  0  0

```

```

Diagnosis of value "one" which prints as  1.0000E+00
"one" is finite
"one" is not NaN
"one" does not compare less than zero.
"one" does not compare equal to zero.
"one" compares greater than zero.

```

```

Diagnosis of value "-two" which prints as -2.0000E+00
"-two" is finite
"-two" is not NaN
"-two" compares less than zero.
"-two" does not compare equal to zero.
"-two" does not compare greater than zero.

```

```

Diagnosis of value "zero" which prints as  0.0000E+00
"zero" is finite
"zero" is not NaN
"zero" does not compare less than zero.
"zero" compares equal to zero.
"zero" does not compare greater than zero.

```

```

Diagnosis of value "-Infinity" which prints as  -Infinity
"-Infinity" is not finite
"-Infinity" is not NaN
"-Infinity" compares less than zero.
"-Infinity" does not compare equal to zero.

```

"-Infinity" does not compare greater than zero.

Diagnosis of value "Quiet NaN" which prints as NaN
"Quiet NaN" is not finite
"Quiet NaN" is NaN
"Quiet NaN" does not compare less than zero.
"Quiet NaN" does not compare equal to zero.
"Quiet NaN" does not compare greater than zero.

Try to cause overflow - no trap should occur:
y = -huge() * huge() = -Infinity

Try to cause NaN - no trap should occur:
y = 0.0 / 0.0 = NaN

Try to cause division by zero - no trap should occur:
y = -1.0 / 0.0 = -Infinity
