NAG Library Function Document

nag_is_finite (x07aac)

1 Purpose

nag_is_finite (x07aac) determines whether a floating-point number is finite.

2 Specification

```
#include <nag.h>
#include <nagx07.h>
Nag_Boolean nag_is_finite (double x)
```

3 Description

nag_is_finite (x07aac) returns Nag_TRUE if and only if **x** is finite, and returns Nag_FALSE otherwise.

4 References

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

5 Arguments

1: **x** – double *Input*

On entry: the number whose status is to be determined.

6 Error Indicators and Warnings

None.

7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

This function will return Nag_FALSE if the argument **x** is either infinite or a NaN (Not A Number).

10 Example

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

10.1 Program Text

```
/* nag_is_finite (x07aac) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 24, 2013.
 */

#include <nag.h>
#include <nagx02.h>
#include <nagx07.h>
#include <stdio.h>

static void diagnose(const char *c, double x)
{
  Nag_Boolean isfin, isnan;
  printf("\nDiagnosis of value \"%s\" which prints as %12.4e\n", c, x);

  /* nag_is_finite (x07aac).
   * Determines whether its argument has a finite value. */
  isfin = nag_is_finite(x);
  if (isfin)
    printf("\"%s\" is finite\n", c);
  else
    printf("\"%s\" is not finite\n", c);

  /* nag_is_nan (x07abc).
   * Determines whether its argument is a NaN (Not A Number). */
  isnan = nag_is_nan(x);
  if (isnan)
    printf("\"%s\" is NaN\n", c);
  else
    printf("\"%s\" is not NaN\n", c);

  if (x < 0.0)
    printf("\"%s\" compares less than zero.\n", c);
  else
    printf("\"%s\" does not compare less than zero.\n", c);

  if (x == 0.0)
    printf("\"%s\" compares equal to zero.\n", c);
  else
    printf("\"%s\" does not compare equal to zero.\n", c);

  if (x > 0.0)
    printf("\"%s\" compares greater than zero.\n", c);
  else
    printf("\"%s\" does not compare greater than zero.\n", c);

}

int main(void)
{
  Integer exit_status = 0;
  double neginf, qnan, x, y, zero;
  Integer exmode[3], newexmode[3];

  printf("nag_is_finite (x07aac) Example Program Results\n\n");

  /* Turn exception halting mode off for the three common exceptions
   * overflow, division-by-zero, and invalid operation. */
  printf("Turn exception halting off ...\n");
  exmode[0] = exmode[1] = exmode[2] = 0;
  /* nag_set_ieee_exception_mode (x07cbc).
   * Sets behaviour of floating point exceptions. */
  nag_set_ieee_exception_mode(exmode);

  /* Check that exception halting mode for the three common exceptions
   * was really turned off. */
  /* nag_get_ieee_exception_mode (x07cac).
```

```
   * Gets current behaviour of floating point exceptions. */
  nag_get_ieee_exception_mode(newexmode);
  printf("Exception halting mode is now: %"NAG_IFMT" %"NAG_IFMT" %"NAG_IFMT"\n",
         newexmode[0], newexmode[1], newexmode[2]);

  /* Look at some ordinary numbers. */
  x = 1.0;
  diagnose("one", x);
  x = -2.0;
  diagnose("-two", x);
  zero = 0.0;
  diagnose("zero", zero);

  /* Generate an infinity and a NaN and look at their properties. */
  /* nag_create_infinity (x07bac). Creates a signed infinite value. */
  nag_create_infinity(-1, &neginf);
  diagnose("-Infinity", neginf);

  /* nag_create_nan (x07bbc). Creates a NaN (Not A Number). */
  nag_create_nan(1, &qnan);
  diagnose("Quiet NaN", qnan);

  /* Do some operations which purposely raise exceptions. */
  printf("\nTry to cause overflow - no trap should occur:\n");
  /* nag_real_largest_number (X02ALC). The largest positive model number. */
  x = nag_real_largest_number;
  y = -x * x;
  printf("y = -huge * huge = %12.4e\n\n", y);

  printf("Try to cause NaN - no trap should occur:\n");
  y = zero / zero;
  printf("y = 0.0 / 0.0 = %12.4e\n\n", y);

  printf("Try to cause division by zero - no trap should occur:\n");
  x = -1.0;
  y = x / zero;
  printf("y = -1.0 / 0.0 = %12.4e\n", y);

  return exit_status;
}
```

10.2 Program Data

None.

10.3 Program Results

```
nag_is_finite (x07aac) 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         -inf
"-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 =         -inf

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 =         -inf
```