NAG Library Function Document

nag_ip_read (h02xyc)

1 Purpose

nag_ip_read (h02xyc) reads a set of optional argument values from a file and assigns those values to a given options structure of type Nag_H02_Opt. Values supplied are checked as being of correct type and within range for the specified optional argument.

2 Specification

#include <nag.h>
#include <nagh.h>

void nag_ip_read (const char *name, const char *opt_file,
    Nag_H02_Opt *options, Nag_Boolean print, const char *outfile,
    NagError *fail)

3 Description

The integer programming (IP) functions of the h Chapter Introduction have a number of optional arguments, which are set by means of a structure of type Nag_H02_Opt. Optional argument values may be assigned to members of the options structure directly in the program text and/or by supplying the optional values in a file which can be read by the function nag_ip_read (h02xyc).

When optional argument values are read from a file using nag_ip_read (h02xyc) then the options structure will be initialized automatically if this has not already been done. It is only necessary to call nag_ip_init (h02xxc) if direct assignments to the options structure are made in your program before calling nag_ip_read (h02xyc).

As well as reading from a file, nag_ip_read (h02xyc) will also read from stdin. This allows redirection to be used to supply the file; it also permits nag_ip_read (h02xyc) to be used interactively with you supplying values from the keyboard.

Checks are made that the values read in are of valid type for the optional argument specified and that the value is within the valid range for that argument. If a value is accepted, a printed confirmation of the setting of the relevant argument will be output if print = Nag_TRUE. An unacceptable argument name or value will give an error message if print = Nag_TRUE.

4 References

None.

5 Arguments

1: name – const char * Input

On entry: a character string specifying either the NAG six character name or the NAG long name of the proposed IP function. The case of the character string is disregarded.

2: opt_file – const char * Input

On entry: the name of the file which specifies the optional argument values. If stdin is to be used, the string "stdin" should be supplied. The set of option values must be preceded by the keyword begin followed by the function name for which the set of options is being supplied. The function name may be the six character NAG name of an IP function or its associated long name.
Each option value specified in the file must be preceded by the name of the optional argument. The argument name and value must be separated by at least one blank space or an equals symbol. nag_ip_read (h02xyc) will read to the end of file or until the keyword end is found or until another begin is found. C style comments may be placed within a set of option values to aid your documentation. Outside the option value sets, text need not be within C style comment delimiters.

N.B. Assignment to function pointers in the options structure, memory allocation to array pointers and assignment of trailing array dimensions cannot be performed from an options file. These must be assigned directly to the options structure in your calling program.

3: **options** – Nag_H02_Opt *  
   *Input/Output*  
   *On entry:* the options structure may or may not have previously been initialized, and had values assigned to its members.  
   *On exit:* the options structure, initialized and with values assigned according to the values found in the options file.

4: **print** – Nag_Boolean  
   *Input*  
   *On entry:* if Nag_TRUE a message confirming the setting of each option will be output.

5: **outfile** – const char *  
   *Input*  
   *On entry:* a character string specifying the name of the file to which confirmation messages should be output. If `stdout` is required then the string "stdout" should be given. When `print = Nag_FALSE` the empty string "" can be supplied as outfile will be ignored.

6: **fail** – NagError *  
   *Input/Output*  
   The NAG error argument (see Section 3.6 in the Essential Introduction).

### 6 Error Indicators and Warnings

**NE_FIELD_UNKNOWN**  
(line `value`) ‘`(string)`’ is not a permitted structure member or option for `(string)`.

**NE_INVALID_BEGIN**  
The Begin statement occurring in data file from which options are being read is not valid.

**NE_INVALID_ENUM_RANGE**  
Enum value `(string)` given to `(option)` is not valid for this function.

**NE_INVALID_INT_RANGE_1**  
(line `value`) ‘`(string)`’ cannot be assigned to using an options file.

**NE_INVALID_OPTION**  
(line `value`) ‘`(string)`’ is not a valid name for a structure member or option.  
This error message is output if, for example, the specified string contains characters which are not permitted in a variable name in the C programming language.

**NE_INVALID_REAL_RANGE_CONS**  
Value `value` given to `(option)` is not valid. The argument `(option)` must satisfy `(constraint)`.
7 Accuracy

Not applicable.

8 Parallelism and Performance

Not applicable.

9 Further Comments

nag_ip_read (h02xyc) may be used to read the optional ‘MPSX names’ (e.g., prob_name, obj_name) for nag_ip_mps_read (h02buc). However, although this function allows the names to contain non-leading blank characters, nag_ip_read (h02xyc) will not read such names correctly from an options file since blank spaces are assumed to denote the end of an option name or value within the file. All other valid MPSX names (see the documentation for nag_ip_mps_read (h02buc) for details) will be read correctly by nag_ip_read (h02xyc).
10 Example

See Section 10 in nag_ip_mps_read (h02buc).