INTERNATIONAL STANDARD

ISO/IEC 10967-2

First edition
2001-08-15

Information technology — Language independent arithmetic —

Part 2: Elementary numerical functions

Technologies de l'information — Arithmétique de langage indépendant — Partie 2: Fonctions numériques élémentaires

Reference number
ISO/IEC 10967-2:2001(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this part of ISO/IEC 10967 may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 10967-2 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

ISO/IEC 10967 consists of the following parts, under the general title *Information technology — Language independent arithmetic*:

- *Part 1: Integer and floating point arithmetic*
- *Part 2: Elementary numerical functions*
- *Part 3: Complex integer and floating point arithmetic and complex elementary numerical functions*

Additional parts will specify other arithmetic datatypes or arithmetic operations.

Annex A forms a normative part of this part of ISO/IEC 10967. Annexes B to E are for information only.

Introduction

The aims

Portability is a key issue for scientific and numerical software in today's heterogeneous computing environment. Such software may be required to run on systems ranging from personal computers to high performance pipelined vector processors and massively parallel systems, and the source code may be ported between several programming languages. Part 1 of ISO/IEC 10967 specifies the basic properties of integer and floating point types that can be relied upon in writing portable software.

Programmers writing programs that perform a significant amount of numeric processing have often not been certain how a program will perform when run under a given language processor. Programming language standards have traditionally been somewhat weak in the area of numeric processing, seldom providing an adequate specification of the properties of arithmetic datatypes, particularly floating point numbers. Often they do not even require much in the way of documentation of the actual arithmetic operations by a conforming language processor.

It is the intent of this part to help to redress these shortcomings, by setting out precise definitions of elementary numerical functions, and requirements for documentation.

It is not claimed that this part will ensure complete certainty of arithmetic behaviour in all circumstances; the complexity of numeric software and the difficulties of analysing and proving algorithms are too great for that to be attempted. Rather, this International Standard will provide a firmer basis than hitherto for attempting such analysis.

The aims for this part, part 2 of ISO/IEC 10967, are extensions of the aims for part 1: to ensure adequate accuracy for numerical computation, predictability, notification on the production of exceptional results, and compatibility with programming language standards.

The content

The content of this part is based on part 1, and extends part 1's specifications to specifications for operations approximating real elementary functions, operations often required (usually without a detailed specification) by the standards for programming languages widely used for scientific software. This part also provides specifications for conversions between the "internal" values of an arithmetic datatype, and a very close approximation in, e.g., the decimal radix. It does not cover the further transformation to decimal string format, which is usually provided by language standards. This part also includes specifications for a number of other functions deemed useful, even though they may not be stipulated by programming language standards.

The numerical functions covered by this part are computer approximations to mathematical functions of one or more real arguments. Accuracy versus performance requirements often vary with the application at hand. This is recognised by recommending that implementors support more than one library of these numerical functions. Various documentation and (program available) parameters requirements are specified to assist programmers in the selection of the library best suited to the application at hand.

The benefits

Adoption and proper use of this part can lead to the following benefits.

Language standards will be able to define their arithmetic semantics more precisely without preventing the efficient implementation of their language on a wide range of machine architectures.

Programmers of numeric software will be able to assess the portability of their programs in advance. Programmers will be able to trade off program design requirements for portability in the resulting program.

Programs will be able to determine (at run time) the crucial numeric properties of the implementation. They will be able to reject unsuitable implementations, and (possibly) to correctly characterize the accuracy of their own results. Programs will be able to detect (and possibly correct for) exceptions in arithmetic processing.

End users will find it easier to determine whether a (properly documented) application program is likely to execute satisfactorily on their platform. This can be done by comparing the documented requirements of the program against the documented properties of the platform.

Finally, end users of numeric application packages will be able to rely on the correct execution of those packages. That is, for correctly programmed algorithms, the results are reliable if and only if there is no notification.

Information technology — Language independent arithmetic —

Part 2: Elementary numerical functions

1 Scope

This part of ISO/IEC 10967 defines the properties of numerical approximations for many of the real elementary numerical functions available in standard libraries for a variety of programming languages in common use for mathematical and numerical applications.

An implementor may choose any combination of hardware and software support to meet the specifications of this part. It is the computing environment, as seen by the programmer/user, that does or does not conform to the specifications.

The term *implementation* (of this part) denotes the total computing environment pertinent to this part, including hardware, language processors, subroutine libraries, exception handling facilities, other software, and documentation.

1.1 Inclusions

The specifications of part 1 are included by reference in this part.

This part provides specifications for numerical functions for which all operand values are of integer or floating point datatypes satisfying the requirements of part 1. Boundaries for the occurrence of exceptions and the maximum error allowed are prescribed for each specified operation. Also the result produced by giving a special value operand, such as an infinity, or a NaN, is prescribed for each specified floating point operation.

This part covers most numerical functions required by the ISO/IEC standards for Ada [11], Basic [16], C [17], C++ [18], Fortran [22], ISLisp [24], Pascal [27], and PL/I [29]. In particular, specifications are provided for:

a) Some additional integer operations.

b) Some additional non-transcendental floating point operations, including maximum and minimum operations.

c) Exponentiations, logarithms, and hyperbolics.

d) Trigonometrics, both in radians and for argument-given angular unit with degrees as a special case.

This part also provides specifications for:

e) Conversions between integer and floating point datatypes (possibly with different radices) conforming to the requirements of part 1, and the conversion operations used, for example, in text input and output of integer and floating point numbers.

f) The results produced by an included floating point operation when one or more argument values are IEC 60559 special values.

g) Program-visible parameters that characterise certain aspects of the operations.

1.2 Exclusions

This part provides no specifications for

a) Numerical functions whose operands are of more than one datatype (with one exception). This part neither requires nor excludes the presence of such "mixed operand" operations.

b) An interval datatype, or the operations on such data. This part neither requires nor excludes such data or operations.

c) A fixed point datatype, or the operations on such data. This part neither requires nor excludes such data or operations.

d) A rational datatype, or the operations on such data. This part neither requires nor excludes such data or operations.

e) Complex, matrix, statistical, or symbolic operations. This part neither requires nor excludes such data or operations.

f) The properties of arithmetic datatypes that are not related to the numerical process, such as the representation of values on physical media.

g) The properties of integer and floating point datatypes that properly belong in programming language standards or other specifications. Examples include

1) the syntax of numerals and expressions in the programming language,
2) the syntax used for parsed (input) or generated (output) character string forms for numerals by any specific programming language or library,
3) the precedence of operators in the programming language,
4) the presence or absence of automatic datatype coercions,
5) the rules for assignment, parameter passing, and returning value,
6) the consequences of applying an operation to values of improper datatype, or to uninitialised data.

Furthermore, this part does not provide specifications for how the operations should be implemented or which algorithms are to be used for the various operations.

2 Conformity

It is expected that the provisions of this part of ISO/IEC 10967 will be incorporated by reference and further defined in other International Standards; specifically in programming language standards and in binding standards.

A binding standard specifies the correspondence between one or more of the parameters and operations specified in this part and the concrete language syntax of some programming language. More generally, a binding standard specifies the correspondence between certain parameters and operations and the elements of some arbitrary computing entity. A language standard that explicitly provides such binding information can serve as a binding standard.

When a binding standard for a language exists, an implementation shall be said to conform to this part if and only if it conforms to the binding standard. In case of conflict between a binding standard and this part, the specification of the binding standard takes precedence.

When a binding standard covers only a subset of the operations specified in this part, an implementation remains free to conform to this part with respect to other operations, independently of that binding standard.

When no binding standard for a language and some operations specified in this part exists, an implementation conforms to this part if and only if it provides one or more operations that together satisfy all the requirements of clauses 5 through 8 that are relevant to those operations. The implementation shall then document the binding.

Conformity to this part is always with respect to a specified set of datatypes and operations. Conformity to this part implies conformity to part 1 for the integer and floating point datatypes used.

An implementation is free to provide operations that do not conform to this part, or that are beyond the scope of this part. The implementation shall not claim or imply conformity to this part with respect to such operations.

An implementation is permitted to have modes of operation that do not conform to this part. A conforming implementation shall specify how to select the modes of operation that ensure conformity. However, a mode of operation that conforms to this part should be the default mode of operation.

NOTES

1 Language bindings are essential. Clause 8 requires an implementation to supply a binding if no binding standard exists. See annex C for suggested language bindings.

2 A complete binding for this part will include (explicitly or by reference) a binding for part 1 as well, which in turn may include (explicitly or by reference) a binding for IEC 60559 as well.

3 This part does not require a particular set of operations to be provided. It is not possible to conform to this part without specifying to which datatypes and set of operations (and modes of operation) conformity is claimed.

3 Normative references

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 10967. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of ISO/IEC 10967 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies. Members of ISO and IEC maintain registers of currently valid International Standards.

IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems.*

ISO/IEC 10967-1:1994, *Information technology – Language independent arithmetic – Part 1: Integer and floating point arithmetic.*

NOTE – See also annex E.

4 Symbols and definitions

4.1 Symbols

4.1.1 Sets and intervals

In this part, $\mathcal{Z}$ denotes the set of mathematical integers, $\mathcal{R}$ denotes the set of classical real numbers, and $\mathcal{C}$ denotes the set of complex numbers over $\mathcal{R}$. Note that $\mathcal{Z} \subset \mathcal{R} \subset \mathcal{C}$.

The conventional notation for set definition and manipulation is used.

In this part, the following notation for intervals is used

$[x, z]$ designates the interval $\{y \in \mathcal{R} \mid x \leqslant y \leqslant z\}$,
$]x, z]$ designates the interval $\{y \in \mathcal{R} \mid x < y \leqslant z\}$,
$[x, z[$ designates the interval $\{y \in \mathcal{R} \mid x \leqslant y < z\}$, and
$]x, z[$ designates the interval $\{y \in \mathcal{R} \mid x < y < z\}$.

NOTE – The notation using a round bracket for an open end of an interval is not used, for the risk of confusion with the notation for pairs.

4.1.2 Operators and relations

All prefix and infix operators have their conventional (exact) mathematical meaning. In particular this part uses

$\Rightarrow$ and $\Leftrightarrow$ for logical implication and equivalence
$+$, $-$, $/$, $|x|$, $\lfloor x \rfloor$, $\lceil x \rceil$, and round(x) on reals
$\cdot$ for multiplication on reals
$<$, $\leqslant$, $\geqslant$, and $>$ between reals
$=$ and $\neq$ between real as well as special values
max on non-empty upwardly closed sets of reals
min on non-empty downwardly closed sets of reals
$\cup$, $\cap$, $\times$, $\in$, $\notin$, $\subset$, $\subseteq$, $\not\subseteq$, $\neq$, and $=$ with sets
$\times$ for the Cartesian product of sets
$\rightarrow$ for a mapping between sets
$|$ for the divides relation between integers

For $x \in \mathcal{R}$, the notation $\lfloor x \rfloor$ designates the largest integer not greater than x:

$$\lfloor x \rfloor \in \mathcal{Z} \quad \text{and} \quad x - 1 < \lfloor x \rfloor \leqslant x$$

the notation $\lceil x \rceil$ designates the smallest integer not less than x:

$$\lceil x \rceil \in \mathcal{Z} \quad \text{and} \quad x \leqslant \lceil x \rceil < x + 1$$

and the notation round(x) designates the integer closest to x:

$$\text{round}(x) \in \mathcal{Z} \quad \text{and} \quad x - 0.5 \leqslant \text{round}(x) \leqslant x + 0.5$$

where in case x is exactly half-way between two integers, the even integer is the result.

The *divides* relation ($|$) on integers tests whether an integer i divides an integer j exactly:

$$i|j \;\Leftrightarrow\; (i \neq 0 \text{ and } i \cdot n = j \text{ for some } n \in \mathcal{Z})$$

NOTE – $i|j$ is true exactly when j/i is defined and $j/i \in \mathcal{Z}$).

4.1.3 Mathematical functions

This part specifies properties for a number of operations numerically approximating some of the elementary functions. The following ideal mathematical functions are defined in chapter 4 of the *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* [47] (e is the Napierian base)

e^x, x^y, $\sqrt{x}$, ln, $\log_b$,
sin, cos, tan, cot, sec, csc, arcsin, arccos, arctan, arccot, arcsec, arccsc,
sinh, cosh, tanh, coth, sech, csch, arcsinh, arccosh, arctanh, arccoth, arcsech, arccsch.

Many of the inverses above are multi-valued. The selection of which value to return, the principal value, so as to make the inverses into functions, is done in the conventional way. E.g., $\sqrt{x} \in [0, \infty[$ when $x \in [0, \infty[$. The only one over which there is some difference of conventions is the arccot function. Conventions there vary for negative arguments; either a negative value (giving a sign symmetric function), or a positive return value (giving a function that is continuous over zero). In this part, arccot refers to the sign symmetric inverse function (with a branch cut at 0), and arccotc refers to the continuous inverse function.

$\text{arccosh}(x) \geqslant 0$, $\text{arcsech}(x) \geqslant 0$,
$\arcsin(x) \in [-\pi/2, \pi/2]$, $\arccos(x) \in [0, \pi]$, $\arctan(x) \in]-\pi/2, \pi/2[$,
$\text{arccot}(x) \in]-\pi/2, \pi/2]$, $\text{arccotc}(x) \in]0, \pi[$, $\text{arcsec}(x) \in [0, \pi]$, $\text{arccsc}(x) \in [-\pi/2, \pi/2]$.

NOTE – $e = 2.71828...$. e is not in any floating point datatype conforming to part 1, unless added as a special value, which is usually not done.

4.1.4 Exceptional values

The exceptional value **underflow** is used in this part as it is in part 1.

Three new exceptional values, **overflow**, **invalid**, and **infinitary**, are introduced in this part replacing three other exceptional values used in part 1. **invalid** and **infinitary** are in this part used instead of the **undefined** of part 1. **overflow** is used instead of the **integer_overflow** and **floating_overflow** of part 1. Bindings may still distinguish between **integer_overflow** and **floating_overflow**.

One new exceptional value, **absolute_precision_underflow**, is introduced in this part with no correspondence in part 1. The exceptional value **absolute_precision_underflow** is used when the given floating point angle value argument is so big that even a highly accurate result from a trigonometric operation is questionable, due to the fact that the density of floating point values has decreased significantly at these big angle values.