12. Input and output (I/O) of intervals

12.1. Overview. This standard specifies conversion from a text string that holds an interval literal to an interval internal to a program (input), and the reverse (output). The method by which strings are read from, or written to, a character stream is language- or implementation-defined.

Containment shall hold on input and output so that, when a program computes an enclosure of some quantity given an enclosure of the data, a correctly written program can ensure this holds all the way from text data to text results.

In addition to normal I/O, the standard requires each interval type $\mathbb{T}$ to have a *public representation*. This has two parts: operations to convert any internal $\mathbb{T}$-interval $\boldsymbol{x}$ to a string $\boldsymbol{s}$, and back again to recover $\boldsymbol{x}$ exactly; and documentation of how to convert $\boldsymbol{s}$ to the Level 1 interval represented by $\boldsymbol{x}$. For proprietary types in particular, this makes explicit the mathematical definition of the type, while letting its Level 3 implementation remain private.

12.2. Input. Input is provided for each supported bare or decorated interval type $\mathbb{T}$ by the $\mathbb{T}$-version of `text2interval` ($\boldsymbol{s}$), where $\boldsymbol{s}$ is a string, as specified in §11.11.8.

12.3. Output. Implementations shall provide a function

$$\texttt{interval2text}(\boldsymbol{x}, cs)$$

where $\boldsymbol{x}$ is a bare or decorated interval of any supported type $\mathbb{T}$ and cs is a string, the conversion specifier. It converts $\boldsymbol{x}$ to a string, in a way specified by cs.

The allowed forms of cs are language-defined, and may depend on $\mathbb{T}$, but shall let the user specify output in any of the forms of interval literal in §11.11.1, namely:

(i) Inf-sup form `[` l `,` u `]`, where the layouts of l and u can be specified independently.
(ii) Uncertain form such as m `?` r.

⚠ Say what layout means. There should be ways to control how Empty and Entire are output, e.g., whether Entire becomes `[Entire]` or `[-Inf, Inf]`. In all cases the resulting string shall be a valid interval literal that may be read by `text2interval`. In the bare interval case, its Level 1 value $\boldsymbol{y}$ shall contain $\boldsymbol{x}$. In the decorated case, the interval part of $\boldsymbol{y}$ shall contain that of $\boldsymbol{x}$, and the decoration part of $\boldsymbol{y}$ shall equal that of $\boldsymbol{x}$.

It shall be possible to specify output of l, u, m and r to a given number of places after the point or to a given number of significant figures, and to specify their field width. (For instance, by conversion specifiers like `f12.5` and `e12.5` in Fortran, or `%12.5f` and `%12.5e` in C.)

If $\mathbb{T}$ is a 754-conforming type, the enclosure represented by $\boldsymbol{s}$ shall be tightest possible. Namely let $\boldsymbol{x} = [\underline{x}, \overline{x}]$ be a $\mathbb{T}$-interval. For inf-sup form, l is the largest number of the specified layout that is $\leq \underline{x}$ and u is the smallest number of the specified layout that is $\geq \overline{x}$; either may be infinite in case of overflow. For uncertain $m?r$ form, m is the number of the specified layout that is closest to the exact midpoint; then r is the smallest number of the specified layout such that the exact interval $[m - r, m + r]$ contains $\boldsymbol{x}$. ⚠ Do $m?rd$ form, etc. The treatment of infinite values, overflow and tie-breaking shall follow that of the `inf`, `sup`, `mid` and `rad` functions in §11.11.9.

For other types the tightness of enclosure of $\boldsymbol{x}$ by $\boldsymbol{s}$ is language- or implementation-defined.

12.4. Public representation. For any supported bare interval type $\mathbb{T}$ an implementations shall provide functions `interval2public` and `public2interval`, as follows.

- For any $\mathbb{T}$-interval datum $\boldsymbol{x}$ the value `interval2public`($\boldsymbol{x}$) is a string $\boldsymbol{s}$, the **public representation** of $\boldsymbol{x}$, such that `public2interval`($\boldsymbol{s}$) $= \boldsymbol{x}$.
- The implementation's documentation shall describe how to convert $\boldsymbol{s}$ to the mathematical interval $[l, u]$ represented by the datum $\boldsymbol{x}$, by means of an algorithm for obtaining l and u as decimal, hexadecimal or binary numbers, exactly or to any desired accuracy.

  If $\mathbb{T}$ is a 754-conforming type, the public representation $\boldsymbol{s}$ of $\boldsymbol{x}$ shall be as an interval literal (§11.11.1) that, for nonempty $\boldsymbol{x}$, is of inf-sup form. Its bounds l, u if finite shall be represented exactly, as decimal numbers if $\mathbb{T}$ is a decimal type, or in the hexadecimal-significand form of 754§5.12.3 if $\mathbb{T}$ is a binary type.

  [*Note. A public representation should aim for simplicity. For instance if $\boldsymbol{x}$ represents an interval with small integer bounds such as* $[1, 2]$*, it should be straightforward to convert s by hand or with the help of a pocket calculator. A good public representation exposes the values of the parameters on which the mathematical model of the type is based.*]