Basic Types

Previous Next

On this page:

Built-In Basic Types

Slice provides a number of built-in basic types, as shown in this table:

Type	Range of Mapped Type	Size of Mapped Type
`bool`	`false` or `true`	1bit
`byte`	-128-127 or 0-255 [a]	8 bits
`short`	-2^{15} to $2^{15}-1$	16 bits
`int`	-2^{31} to $2^{31}-1$	32 bits
`long`	-2^{63} to $2^{63}-1$	64 bits
`float`	IEEE single-precision	32 bits
`double`	IEEE double-precision	64 bits
`string`	All Unicode characters, excluding the character with all bits zero.	Variable-length

[a] The range depends on whether `byte` maps to a signed or an unsigned type.

All the basic types (except `byte`) are subject to changes in representation as they are transmitted between clients and servers. For example, a `long` value is byte-swapped when sent from a little-endian to a big-endian machine. Similarly, strings undergo translation in representation if they are sent, for example, from an EBCDIC to an ASCII implementation, and the characters of a string may also change in size. (Not all architectures use 8-bit characters). However, these changes are transparent to the programmer and do exactly what is required.

Back to Top ^

Integer Types

Slice provides integer types `short`, `int`, and `long`, with 16-bit, 32-bit, and 64-bit ranges, respectively. Note that, on some architectures, any of these types may be mapped to a native type that is wider. Also note that no unsigned types are provided. (This choice was made because unsigned types are difficult to map into languages without native unsigned types, such as Java. In addition, the unsigned integers add little value to a language. (See [1] for a good treatment of the topic.)

Back to Top ^

Floating-Point Types

These types follow the IEEE specification for single- and double-precision floating-point representation [2]. If an implementation cannot support IEEE format floating-point values, the Ice run time converts values into the native floating-point representation (possibly at a loss of precision or even magnitude, depending on the capabilities of the native floating-point format).

Back to Top ^

Strings

Slice strings use the Unicode character set. The only character that cannot appear inside a string is the zero character.

This decision was made as a concession to C++, with which it becomes impossibly difficult to manipulate strings with embedded zero characters using standard library routines, such as `strlen` or `strcat`.

The Slice data model does not have the concept of a null string (in the sense of a C++ null pointer). This decision was made because null strings are difficult to map to languages without direct support for this concept (such as Python). Do not design interfaces that depend on a null string to indicate "not there" semantics. If you need the notion of an optional string, use a class, a sequence of strings, or use an empty string to represent the idea of a null string. (Of course, the latter assumes that the empty string is not otherwise used as a legitimate string value by your application.)

Back to Top ^

Booleans

Boolean values can have only the values `false` and `true`. Language mappings use the corresponding native boolean type if one is available.

Back to Top ^

Bytes

The Slice type `byte` is an (at least) 8-bit type that is guaranteed not to undergo any changes in representation as it is transmitted between address spaces. This guarantee permits exchange of binary data such that it is not tampered with in transit. All other Slice types are subject to changes in representation during transmission.

Back to Top ^

See Also

- Sequences
- Classes

References

1. Lakos, J. 1996. Large-Scale C++ Software Design. Reading, MA: Addison-Wesley.
2. Institute of Electrical and Electronics Engineers. 1985. *IEEE 754-1985 Standard for Binary Floating-Point Arithmetic*. Piscataway, NJ: Institute of Electrical and Electronic Engineers.

Previous Next