

## Integer Component — A Brief Summary

**Range of values:** -2,147,483,648 through +2,147,483,647

**Initial value:** 0

**Assignment operator:** =

### Arithmetic operators:

- + (addition)
- (subtraction)
- \* (multiplication)
- / (integer division — quotient without the remainder)
- mod** (remainder of integer division without the quotient)

Note: When using the mod operation as in `i mod j`, where `i` and `j` are of type Integer, it must be that `j > 0`, while `i` can be any legal Integer value. The result of `i mod j` will always be nonnegative. So, `17 mod 5 = 2` while `-17 mod 5 = 3`. Also, `17 mod -5` is an illegal use of the mod operation.

### Precedence of arithmetic operator evaluation:

Highest: parenthesized subexpressions

↓  
\*, /, mod

lowest: +, -

Note: Consecutive operators of the equal precedence are evaluated left to right.

### Relational operators:

- == (equal)
- != (not equal)
- < (less than and not equal)
- <= (less than or equal)
- > (greater than and not equal)
- >= (greater than or equal)

### Input and output:

Assume that `input` is an object of type `Character_IStream`, that `output` is an object of type `Character_OStream`, and that `j` is an object of type `Integer`.

- To input a value for `j` use `input >> j`.
- To output the value of `j` use `output << j`.

### Conversion operators:

Assume that `j` is an object of type `Integer`.

- To convert the value of `j` to a real value use `To_Real (j)`.
- To convert the value of `j` to a text string use `To_Text (j)`.
- To convert the value of `j` to a character use `To_Character (j)`.

Note: The value of `j` must be in the range 0 through 127 inclusive when using the `To_Character (j)` operation. The result of `To_Character (j)` is the ASCII character whose code corresponds to the value of `j`.