LogicalExpression can use any comparison or logical operator in the expression.

Flags

/FREE	Creates a free destWave (see **Free Waves** on page IV-91). /FREE is allowed only in functions and only if a simple name or wave reference structure field is specified for *destWave*.
/INDX	Stores the index in *destWave* instead of data from *srcWave*.
/O	Allows *destWave* to be the same as *srcWave* (overwrite source).

Type Flags *(used only in functions)*

Extract also can use various type flags in user functions to specify the type of destination wave reference variables. These type flags do not need to be used except when it needed to match another wave reference variable of the same name or to identify what kind of expression to compile for a wave assignment. See **WAVE Reference Types** on page IV-73 and **WAVE Reference Type Flags** on page IV-74 for a complete list of type flags and further details.

Details

srcWave may be of any type including text.

destWave has the same type as *srcWave*, but it is always one dimensional. With /INDX, the *destWave* type is set to unsigned 32-bit integer and the values represent a linear index into *srcWave* regardless of its dimensionality.

Example

```
Make/O source= x
Extract/O source,dest,source>10 && source<20
print dest
```

Prints the following to the history area:

```
  dest[0]= {11,12,13,14,15,16,17,18,19}
```

See Also

The **Duplicate** operation.

factorial

`factorial(`*`n`*`)`

The factorial function returns *n*!, where *n* is assumed to be a positive integer.

Note that while factorial is an integer-valued function, a double-precision number has 53 bits for the mantissa. This means that numbers over 2^{52} will be accurate to about one part in about $2x10^{16}$. Values of *n* greater than 170 result in overflow and return Inf.

If you encounter overflow you can use the **APMath** operation to obtain the result. For example:

```
• Print factorial(1000)
  inf
• APMath/V result = factorial(1000)
  4.02387260077093773543702433923003985719374864210715E+2567
```

Faddeeva

`Faddeeva(`*`z`*`)`

Faddeeva computes the Faddeeva function, also commonly denoted as "w", using an approximation that has, as described by the author, "accuracy is typically at at least 13 significant digits". Both the input and the output are complex numbers.

The Faddeeva function was added in Igor Pro 8.00.

The Faddeeva function is the basis of the Voigt function, implemented in Igor as **VoigtFunc**. VoigtFunc(X, Y) = real(Faddeeva(cmplx(X, Y)).

References

The code used to compute the VoigtFunc was written by Steven G. Johnson of MIT. You can learn more about it at http://ab-initio.mit.edu/Faddeeva.