The cmplx function uses double-precision for its argument. This limits the precision of integer arguments to to cmplx to 53 bits. Consequently, when assigning to a 64-bit complex integer wave, values above 2^53 are imprecise.

In the case of 8-, 16- and 32-bit complex integer waves, math is done in double-precision floating point and then converted to integer. This is fine since double-precision supports up to 2^53 precisely. However, for 64-bit complex integers, doubles are not sufficient. Igor uses special routines for 64-bit complex integer addition and subtraction and returns an error for other operations such as multiplication and division. For example:

```
complex64BitWave += cmplx(-p,-p)    // OK
complex64BitWave *= cmplx(1,1)      // Feature not implemented error
```

Conditional Statements in Functions

Igor Pro supports two basic forms of conditional statements: if-else-endif and if-elseif-endif statements. Igor also supports multiway branching with switch and strswitch statements.

If-Else-Endif

The form of the if-else-endif structure is

```
if ( <expression> )
   <true part>
else
   <false part>
endif
```

<expression> is a numeric expression that is considered true if it evaluates to any nonzero number and false if it evaluates to zero. The true part and the false part may consist of any number of lines. If the expression evaluates to true, only the true part is executed and the false part is skipped. If the expression evaluates to false, only the false part is executed and the true part is skipped. After the true part or false part code is executed, execution continues with any code immediately following the if-else-endif statement.

The keyword “else” and the false part may be omitted to give a simple conditional:

```
if ( <expression> )
   <true part>
endif
```

Because Igor is line-oriented, you may not put the `if`, `else` and `endif` keywords all on the same line. They must each be in separate lines with no other code.

If-Elseif-Endif

The if-elseif-endif statement provides a means for creating nested if structures. It has the form:

```
if ( <expression1> )
   <true part 1>
elseif ( <expression2> )
   <true part 2>
[else
   <false part>]
endif
```

These statements follow similar rules as for if-else-endif statements. When any expression evaluates as true (nonzero), the code immediately following the expression is executed. If all expressions evaluate as false (zero) and there is an else clause, then the statements following the else keyword are executed. Once any code in a true part or the false part is executed, execution continues with any code immediately following the if-elseif-endif statement.