op_increment — Increment and decrement operators

Description

++*i* and *i*++ increment *i*; they perform the operation *i*=*i*+1. ++*i* performs the operation before the evaluation of the expression in which it appears, whereas *i*++ performs the operation afterward.

--*i* and *i*-- decrement *i*; they perform the operation *i*=*i*−1. --*i* performs the operation before the evaluation of the expression in which is appears, whereas *i*-- performs the operation afterward.

Syntax

++*i*	increment before
--*i*	decrement before
i++	increment after
i--	decrement after

where *i* must be a real scalar.

Remarks and examples

These operators are used in code, such as

```
x[i++] = 2
x[--i] = 3
for (i=0; i<100; i++) {
        ...
}
if (++n > 10) {
        ...
}
```

Where these expressions appear, results are as if the current value of `i` were substituted, and in addition, `i` is incremented, either before or after the expression is evaluated. For instance,

```
x[i++] = 2
```

is equivalent to

```
x[i] = 2 ; i = i + 1
```

and

```
x[++i] = 3
```

is equivalent to

```
i = i + 1 ; x[i] = 3
```

Coding

```
for (i=0; i<100; i++) {
        ...
}
```

or

```
for (i=0; i<100; ++i) {
        ...
}
```

is equivalent to

```
for (i=0; i<100; i=i+1) {
        ...
}
```

because it does not matter whether the incrementation is performed before or after the otherwise null expression.

```
if (++n > 10) {
        ...
}
```

is equivalent to

```
n = n + 1
if (n > 10) {
        ...
}
```

whereas

```
if (n++ > 10) {
        ...
}
```

is equivalent to

```
if (n > 10) {
        n = n + 1
        ...
}
else    n = n + 1
```

The ++ and -- operators may be used only with real scalars and are usually associated with indexing or counting. They result in fast and readable code.

Conformability

++*i*, --*i*, *i*++, and *i*--:

i:	1 × 1
result:	1 × 1

Diagnostics

++ and -- are allowed with real scalars only. That is, ++i or i++ is valid, assuming i is a real scalar, but x[i,j]++ is not valid.

++ and -- abort with error if applied to a variable that is not a real scalar.

++i, i++, --i, and i-- should be the only reference to i in the expression. Do not code, for instance,

```
x[i++] = y[i]
x[++i] = y[i]
x[i] = y[i++]
x[i] = y[++i]
```

The value of i in the above expressions is formally undefined; whatever is its value, you cannot depend on that value being obtained by earlier or later versions of the compiler. Instead code

```
i++ ; x[i] = y[i]
```

or code

```
x[i] = y[i] ; i++
```

according to the desired outcome.

It is, however, perfectly reasonable to code

```
x[i++] = y[j++]
```

That is, multiple ++ and -- operators may occur in the same expression; it is multiple references to the target of the ++ and -- that must be avoided.

Also see

[M-2] **exp** — Expressions

[M-2] **Intro** — Language definition

For suggested citations, see the FAQ on citing Stata documentation.