

**break** — Break out of for, while, or do loop

[Description](#)[Syntax](#)[Remarks and examples](#)[Also see](#)

## Description

`break` exits the innermost `for`, `while`, or `do` loop. Execution continues with the statement immediately following the close of the loop, just as if the loop had terminated normally.

`break` nearly always occurs following an `if`.

## Syntax

```

for, while, or do {
    ...
    if (...) {
        ...
        break
    }
}
stmt                ← break jumps here
...

```

## Remarks and examples

stata.com

In the following code,

```

for (i=1; i<=rows(A); i++) {
    for (j=1; j<=cols(A); j++) {
        ...
        if (A[i,j]==0) break
    }
    printf("j = %g\n", j)
}

```

the `break` statement will be executed if any element of `A[i,j]` is zero. Assume that the statement is executed for `i=2` and `j=3`. Execution will continue with the `printf()` statement, which is to say, the `j` loop will be canceled but the `i` loop will continue. The value of `j` upon exiting the loop will be 3; when you break out of the loop, the `j++` is not executed.

## Also see

[M-2] [do](#) — `do ... while (exp)`

[M-2] [for](#) — `for (exp1; exp2; exp3) stmt`

[M-2] [while](#) — `while (exp) stmt`

[M-2] [continue](#) — Continue with next iteration of `for`, `while`, or `do` loop

[M-2] [intro](#) — Language definition