The Goto Statement

The goto statement transfers control to a label. The given label must reside in the same function and can appear before only one statement in the same function.

Syntax

```
statement :
    labeled-statement
    jump-statement

jump-statement :
    goto identifier ;

labeled-statement :
    identifier : statement
```

A statement label is meaningful only to a goto statement; in any other context, a labeled statement is executed without regard to the label.

A jump-statement must reside in the same function and can appear before only one statement in the same function. The set of identifier names following a goto has its own name space so the names do not interfere with other identifiers. Labels cannot be redeclared.

It is good programming style to use the break, continue, and return statement in preference to goto whenever possible. Since the break statement only exits from one level of the loop, a goto may be necessary for exiting a loop from within a deeply nested loop.

This example demonstrates the goto statement:

```
void main()
{

   int i, j;

   for ( i = 0; i < 10; i++ )
{

   printf( "Outer loop executing. i = %d\n", i );
   for ( j = 0; j < 3; j++ )
   {
      printf( " Inner loop executing. j = %d\n", j );
      if ( i == 5 )
      goto stop;
   }
}
   /* This message does not print: */
   printf( "Loop exited. i = %d\n", i );
  stop: printf( "Jumped to stop. i = %d\n", i );
}
```

In this example, a goto statement transfers control to the point labeled stop when i equals 5.