**The *break* Statement**

We often come across situations where we want to jump out of a loop instantly, without waiting to get back to the conditional test.

The keyword break allows us to do this. When break is encountered inside any loop, control automatically passes to the first statement after the loop.

A break is usually associated with an if. As an example, let’s consider the following example.

Example: Write a program to determine whether a number is prime or not. A prime number is one, which is divisible only by 1 or itself.

All we have to do to test whether a number is prime or not, is to divide it successively by all numbers from 2 to one less than itself. If remainder of any of these divisions is zero, the number is not a prime. If no division yields a zero then the number is a prime number. Following program implements this logic.

main( )

{

int num, i ;

printf ( "Enter a number " ) ;

scanf ( "%d", &num ) ;

i = 2 ;

while ( i <= num - 1 )

{

if ( num % i == 0 )

{

printf ( "Not a prime number" ) ;

break ;

}

i++ ;

}

if ( i == num )

printf ( "Prime number" ) ;

}

In this program the moment num % i turns out to be zero, (i.e. num is exactly divisible by i) the message “Not a prime number” is printed and the control breaks out of the while loop. Why does the program require the if statement after the while loop at all? Well, there are two ways the control could have reached outside the while loop:

* It jumped out because the number proved to be not a prime.
* The loop came to an end because the value of i became equal to num.

When the loop terminates in the second case, it means that there was no number between 2 to num - 1 that could exactly divide num. That is, num is indeed a prime. If this is true, the program should print out the message “Prime number”.

The keyword break, breaks the control only from the while in which it is placed. Consider the following program, which illustrates this fact.

main( )

{

int i = 1 , j = 1 ;

while ( i++ <= 100 )

{

while ( j++ <= 200 )

{

if ( j == 150 )

break ;

else

printf ( "%d %d\n", i, j ) ;

}

}

}

In this program when j equals 150, break takes the control outside the inner while only, since it is placed inside the inner while.