Ekaba Bisong
Programming in C++
University of Calabar

Lesson Note #40
June 15, 2015

Adapted from C++ How To Program edited for our own purposes

Recursion

For some problems, it's useful to have functions call themselves. A recursive function is a function that calls itself, either directly, or indirectly (through another function).

Recursive problem-solving approaches have a number of elements in common. A recursive function is called to solve a problem. The function knows how to solve only the simplest case(s), or so-called base case(s). If the function is called with a base case, the function simply returns a result.

If the function is called with a more complex problem, it typically divides the problem into two conceptual pieces — a piece that the function knows how to do and a piece that it does not know how to do.

To make recursion feasible, the latter piece must resemble the original problem, but be a slightly simpler or smaller version. This new problem looks like the original, so the function calls a copy of itself to work on the smaller problem—this is referred to as a recursive call and is also called the recursion step. The recursion step often includes the keyword return, because its result will be combined with the portion of the problem the function knew how to solve to form the result passed back to the original caller, possibly main.

The recursion step can result in many more such recursive calls, as the function keeps dividing each new subproblem with which the function is called into two conceptual pieces. In order for the recursion to eventually terminate, each time the function calls itself with a slightly simpler version of the original problem, this sequence of smaller and smaller problems must eventually converge on the base case. At that point, the function recognizes the base case and returns a result to the previous copy of the function, and a sequence of returns ensues up the line until the original call eventually returns the final result to main.

```cpp
// Demonstrating the recursive function factorial.
#include <iostream>
#include <iomanip>
using namespace std;

unsigned long factorial( unsigned long ); // function prototype

int main()
{
   // calculate the factorials of 0 through 10
   for ( int counter = 0; counter <= 10; ++counter )
      cout << setw( 2 ) << counter << "! = " << factorial( counter )
         << endl;
} // end main

// recursive definition of function factorial
unsigned long factorial( unsigned long number )
{
   if ( number <= 1 ) // test for base case
      return 1; // base cases: 0! = 1 and 1! = 1
   else // recursion step
      return number * factorial( number - 1 );
} // end function factorial
```

```
 0! = 1
 1! = 1
 2! = 2
 3! = 6
 4! = 24
 5! = 120
 6! = 720
 7! = 5040
 8! = 40320
 9! = 362880
10! = 3628800
```