Unidad 2
RECURSIVIDAD

Si una expresión en el cuerpo de una función (método), se llama a sí misma, es una función recursiva.

Cuando una función se llama así misma se asigna espacio en la pila (de la computadora) para nuevas variables locales y parámetros, el código de la función se ejecuta con esas nuevas variables desde el principio, no se hace una nueva copia de la función. Al volver de una llamada recursiva, las variables locales y los parámetros antiguos se borrar de la pila y la ejecución se reanuda en el punto de la llamada de la función.

Cuando se escriben funciones recursivas se debe contar con una sentencia if en algún sitio que force a la función a volver sin que se ejecute la llamada recursiva. Si no se hace esto la función nunca devolverá el control una vez que se le ha llamado.

Una función recursiva debe constar de dos partes:

1.- La base.

2.- La fórmula recursiva.

La fórmula recursiva es la que se ejecuta **n** veces, hasta que se llegue a la base. Este es normalmente el punto de regreso, y a partir de este punto es donde se empiezan a ejecutar las operaciones.

Cuando se prueba una función recursiva es conveniente colocar sentencias **cout** para analizar la secuencia de avance y si es necesario pararla en un momento dado. Aunque es mas conveniente utilizar el debug (depurador) del compilador para realizar dichas pruebas.

Ejemplo: La siguiente clase implementa un método recursivo para calcular el factorial de un numero entero N.

Primero analicemos la definición matemática del factorial de un número entero:

$$n! \begin{cases} 1 & \text{si } n = 0 \\ n * (n-1) * (n-2) * \ldots * 1 & \text{si } n > 0 \end{cases}$$

```
class Matematicas
{
    public:
    unsigned int Factorial(int n);
};


unsigned int Matematicas::Factorial(int n)
{
  int f = 1;
  if(n == 0 ||  n == 1)
       return f;
  else
        f = n * Factorial( n – 1 );
  return f;
}

// Aplicación sencilla para probar nuestro método recursivo
int main( )
{

   Matematicas numero;
   unsigned int fact =  numero.Factorial( 4 );
   cout << "\n El factorial de 4 es : "<< fact;
   cin.get();
   return 0;

}
```

- **TRANSFORMACION DE ALGORITMOS RECURSIVOS A ITERATIVOS.**

La recursividad es una herramienta muy poderosa para resolver problemas, los cuales resueltos iterativamente podrían tener una solución muy compleja, sobre todos aquellos cuya propia definición es recursiva, aunque esto no asegura que dichos algoritmos recursivos posean una eficiencia alta, ya que suelen consumir un mayor tiempo de cálculo. Una solución recursiva debe ser evitada cuando implique más inconvenientes que ventajas.

En general, si un problema se puede describir en términos de versiones más pequeñas de él mismo, entonces la recursividad permite expresarlo, en la mayoría de los casos, y por tanto implementarlo más fácilmente. De igual forma, cuando la recursividad nos permita solucionar problemas cuyas soluciones iterativas sean difíciles de implementar, utilizaremos esta primera técnica.

Eliminación de la recursión mediante el uso de pilas.

Cuando se produce una llamada a una rutina, se introducen en la pila los valores de las variables locales, la dirección de la siguiente instrucción a ejecutar una vez que finalice la llamada, se asignan los parámetros actuales a los formales y se comienza la ejecución por la primera instrucción de la rutina llamada. Cuando acaba el método o función, se saca de la pila la dirección de retorno y los valores de las variables locales, y por último se ejecuta la siguiente instrucción.

En esta técnica de eliminación de recursividad, el programador implementa una estructura de datos con la que simulará la pila del ordenador, introduciendo los valores de las variables locales y parámetros en la pila, en lugar de hacer las llamadas recursivas, y sacándolos para comprobar si se cumplen los casos bases. Se añade un bucle, generalmente while, que iterará mientras la pila no esté vacía, hecho que significa que se han finalizado las llamadas recursivas. Al meter en la pila los valores de las variables locales y los parámetros formales, así como en algunos casos la dirección de retorno (el lugar donde deberá volver el control cuando la invocación de la rutina termine), se simula la invocación recursiva. Cuando se saca de la pila, se está simulando el comienzo de la rutina recursiva, ya que se recuperan los valores de las variables locales y parámetros formales con los que se haría la llamada recursiva.

Ejercicio: Aplicar la alternativa anterior al método recursivo que calcula los Coeficientes Binomiales.