Operador Condicional / Operador Ternario

La forma general de éste operador es:

```
expresion1 ? expresion2 : expresion3
```

Donde el operador `?  :` puede usarse para sustituir ciertas sentencias del tipo **if-then-else**, aunque puede conducir a expresiones más compactas que las correspondientes **if...else .** La gran importancia de tener el operador ternario (o coloquialmente hablando), condicional, se verá luego en su uso en macro reemplazos, pero nunca viene mal introducir el tema de forma anticipada.

Volviendo al tema anterior, el operador ternario funciona de la siguiente manera:

Se evalúa **expresion1**, si el resultado de la evaluación es verdadero (diferente de cero), se ejecuta **expresion2**; si el resultado de la evaluación de expresion1 es falso (igual a cero), se ejecuta **expresion3**.

Ejemplo:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int calificacion;
    printf("Ingrese una nota de parcial:");
    scanf("%d",&calificacion);
    printf("%s\n", calificacion >=7 ? "Aprobado" : "Reprobado");
    return 0;
}
```

UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

En el ejemplo anterior, si el valor de la variable **`calificacion`** es mayor o igual a **`7`**, se imprime la cadena **`"Aprobado"`**, en caso contrario se imprime la cadena **`"Reprobado"`**.

Una de las ventajas que tiene el operador ternario es que puede estar anidado. Por ejemplo, el siguiente código determina el mayor de tres números:

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int a=6,b=2,c=8;
    int mayor;
    mayor= a > b ? (a > c ? a : c)
           : (b > c ? b : c);

    printf("El mayor es: %d",mayor);
    return 0;
}
```

El siguiente ejemplo escribe enteros pares en un archivo y enteros impares en otro archivo:

```c
#include<stdio.h>

int main()
{
    FILE *pares, *impares;
    int n = 10;
    int k = 0;

    pares = fopen("pares.txt", "w");
    impares = fopen("impares.txt", "w");

    for(k = 1; k <= n; k++)
    {
        k%2==0 ? fprintf(pares, "\t%d\n", k)
               : fprintf(impares, "\t%d\n", k);
    }
    fclose(pares);
    fclose(impares);

    return 0;
}
```

Algunas consideraciones:

- El operador condicional se asocia de derecha a izquierda. Considera lo siguiente:

`exp1 ? exp2 : exp3 ? exp4 : exp5`

Como la asociación es de derecha a izquierda, la expresión anterior se evalúa como

`exp1 ? exp2 : ( exp3 ? exp4 : exp5 )`

- Aunque no estamos limitados a asignación de variables, podríamos por ejemplo hacer lo siguiente:

```c
#include <stdio.h>

int main()
{
  int a=10, b=12;

  printf ("El mayor es: %d\n", (a>b)?a:b);
  return 0;
}
```

Podemos incluso, hacer llamadas a funciones dependiendo de si se cumple o no la condición:

```c
#include <stdio.h>

void funcion1()
{
  printf ("Has llamado a la funcion 1\n");
}

void funcion2()
{
  printf ("Has llamado a la funcion 2\n");
}

int main()
{
  int a=10, b=12;

  ((a>b)?funcion1:funcion2)();
  return 0;
}
```

En este último ejemplo es importante el paréntesis que rodea a la expresión, ya que el resultado de `(a > b) ? funcion1 : funcion2;` es lo que tiene que ejecutarse. En definitiva, esto tiene muchas posibilidades y podemos usarlo de muchas maneras. Aunque la evaluación de la expresión siempre será de la misma manera.

Hay que tener en cuenta que la expresión a la izquierda puede ser cualquier tipo de expresión, y que se evalúa siguiendo la precedencia de los operadores intervinientes, veamos una asignación de una operación matemática...

-p. ej.: **c = a + b ? j : k**; se evalúa en primer lugar **a + b**, si el resultado es verdadero (distinto de cero) se evalúa la segunda expresión (**j**), para ser asignada a la variable c. De ser falso, a c se asigna el resultado de la tercera expresión (**k**). (tal vez yo esté repitiendo algo que indicaste implícitamente al comienzo).

Referencias:

[1] B. W. Kernighan y D. M. Ritchie, El lenguaje de programación C, Pearson Educación, 1991.

[2] H. M. Deitel y P. J. Deitel, Cómo programar en C/C+, Pearson Educación, 1995.