Operadores

Javier Fernández Rivera - www.aurea.es

Los operadores

Son los elementos o caracteres gráficos encargados de manipular los datos, que pueden ser dados por números, caracteres, variables, constantes, matrices, etc.
Hay varias clases de operadores dependiendo de su función.

Tenemos el operador de asignación "="

Con este operador asignamos datos a las variables.
Este operador lo hemos usado en varias ocasiones desde el comienzo del curso por tutoriales. En la practica totalidad de los lenguajes de programación y entre ellos en C no debemos identificar el signo "=" con una equivalencia o igualdad, que es lo que estábamos acostumbrados a asociar. Puesto que desde pequeños nos lo enseñaron así. Debemos asociar este signo como una asignación de valores.

Por ejemplo:

```
numero =  45;      /*Esto asiganaria a la variable numero el dato numérico 45*/
```

También podemos asignar de una sola vez un dato en varios operandos con el mismo operador de asignación

```
numero1 = numero2  = numero3 = 100;                /*Esto metería en las variables
numero1,numero2,numero3 el valor 100*/
```

Podemos asignar el valor de una variable a otras 2, por ejemplo

```
C = 5;
A = B = C;                    /*Las variables A y B tomaran el valor de la variable C y todas valdran 5*/
```

Operadores aritméticos

Son los operadores encargados de realizar las operaciones básicas de aritmética matemática. Estos son: +, -, *, /, %.

SIMBOLO	DESCRIPCION	EJEMPLO
+	suma	a + b
-	resta	a - b
*	multiplicación	a * b
/	división	a / b
%	modulo	a % b
-	negación	-a

1. El + sumara dos operandos.
2. El – restara dos operandos
3. El * multiplicara dos operandos.
4. El / dividira dos operandos.
5. El % tomara el resto en una división. Solo vale para enteros nunca en valores decimales.
6. Signo negativo.

Ejemplo:

```
#include <stdio.h>
main () {
 int resultado,numero1,numero2;
resultado=numero1=numero2=0;              /*asignamos a numero1, numero2 y resultado el valor 0. inicializamos la variable a cero*/
numero1=2;                                /*le damos el valor 2 a numero1*/
numero2=3;                                /*con el operador de asignación de valor introducimos el numero 3 en la variable numero2*/
resultado=numero1+numero2;                /*con el operador de sumar hacemos la suma, luego su valor se lo asignamos a resultado*/
printf("\n[%d + %d] = %d",numero1,numero2,resultado);      /*imprimimos la suma de los 2 números y su resultado*/
resultado=numero1-numero2;                /*con el operador de restar hacemos la resta, luego su valor se lo asignamos a resultado*/
printf("\n[%d - %d] = %d",numero1,numero2,resultado);      /*imprimimos la resta de los 2 números y su resultado*/
printf("\n[%d * %d] = %d",numero1,numero2,numero1*numero2);   /*otra forma de hacerlo prescindiendo de la variable resultado*/
resultado=numero2/numero1;                /*con el operador de dividir hacemos la división, luego su valor se lo asignamos a resultado*/
printf("\n[%d / %d] = %d",numero2,numero1,resultado);      /*imprimimos la division de los 2 números y su resultado*/
resultado=numero2%numero1;
printf("\n[%d % %d] = %d",numero2,numero1,resultado);      /*imprimimos el modulo (resto) de los 2 números y su resultado*/
resultado=-numero2+numero1;               /*negativo de numero2 mas contenido de numero1, se lo añadimos a resultado*/
printf("\n[-%d + %d] = %d",numero2,numero1,resultado);     /*mostramos datos por pantalla*/
printf("\n");                             /*linea en blanco*/
printf("\nFIN DEL PROGRAMA");
printf("\n");                             /*linea en blanco*/
}
```

Este programa usa variables de tipo entero por ello no nos mostrara decimales.

Operador de incremento y decremento

Existe un operador que incrementa el valor de una variable o dato y que por igual puede decrementarlo.

Este operador es representado como

- Incremento: **++**
- Decremento: **--**

```
Numero = 2;      /*variable numero ahora vale 2*/
Numero++;       /*variable numero ha icrementado su valor en uno ahora vale 3*/
```

Este operados es lo mismo que si hiciéramos de forma mas lógica esta operación de asignación

```
Numero = 2;
Numero = Numero + 1;
```

Con esto obtendríamos el mismo resultado y es mas fácil ver su función lógica que en el anterior de los casos. Pero en el caso anterior ahorraremos tiempo y escritura.

Operadores de comparación o relacionales

Son los operadores usados en las estructuras condicionales o de comparación de datos. Estos operadores nos devolveran según el resultado un 0 si la condicion no se cumple o un 1 si se cumple, o lo que es lo mismo, false y true, falso y verdadero.
Los operadores de comparación son:

SIMBOLO	DESCRIPCION	EJEMPLO
<	menor que	(a < b)
>	mayor que	(a >b)
< =	menor o igual que	(a < = b)
>=	mayor o igual que	(a >= b)
= =	igual que	(a = = b)
! =	distinto que	(a != b)

If (A **==** B) Si el operando A es equivalente o igual al operando B.
If (A **!=** B) Si el operando A es distinto al operando B.
If (A **<** B) Si el operando A es menor al operando B.
If (A **>** B) Si el operando A es mayor al operando B.
If (A **<=** B) Si el operando A es menor o igual al operando B.
If (A **>=** B) Si el operando A es mayor o igual al operando B.

Ejemplo:

```
#include <stdio.h>
main () {
 int resultado,numero1,numero2,numero3;
numero1=2;
numero2=3;
numero3=2;
if (numero1 < numero2) { printf("\n >> numero1 es menor que numero2 [ %d < %d]",numero1,numero2);  }
if (numero2 > numero3) { printf("\n >> numero2 es mayor que numero3 [ %d > %d]",numero2,numero3);  }
if (numero1 <= numero3) { printf("\n >> numero1 es menor o igual que numero3 [ %d <=
%d]",numero1,numero3);  }
if (numero2 >= numero3) { printf("\n >> numero2 es mayor o igual que numero3 [ %d >
%d]",numero2,numero3);  }
if (numero3 == numero1) { printf("\n >> numero3 es igual que numero1 [ %d == %d]",numero3,numero1);
}
if (numero1 != numero2) { printf("\n >> numero1 es distinto que numero2 [ %d != %d]",numero1,numero2);
}
printf("\n");                                    /*linea en blanco*/
printf("\nFIN DEL PROGRAMA");
printf("\n");                                    /*linea en blanco*/
}
```

Uno de los errores mas comunes dentro de los operadores relacionales es confundir el "==" con el "=". Recordemos que en el primer caso "==" estamos haciendo una comparación mediante un operador relacional y en el segundo caso "=" estamos dando un valor mediante un operador de asignación.

Operadores lógicos

Son los operadores que nos permiten unir varias condiciones.

Estamos hablando de operadores tales como:

SIMBOLO	DESCRIPCION	EJEMPLO

&&	Y (AND)	(a>b) && (c < d)
||	O (OR)	(a>b) || (c < d)
!	NEGACION (NOT)	!(a>b)

Y (&&) Este operador une condiciones: a == b && b == c /* si a es igual a b **Y** b es igual a c */
O (||) Este operador es disyuntivo: a == b || b == c /* si a es igual a b **O** b es igual a c */
No (!) Este operador niega los operandos u operadores.

Además de estos operadores existen otros muchos pero ya son los menos usados o utilizados en casos muy específicos. Los citados aquí son los mas usuales comunes y que es necesario saber para programar adecuadamente en C.

Ejemplo:

```
#include <stdio.h>
main () {
 int  a,b,c;
a=2;
b=3;
c=2;
if (a==c) && (b>a) { printf("\n>>>> a es igual que c b y b es mayor que a. [ (%d == %d) Y (%d > %d)
]",a,c,b,a);   }
if (a>c) || (b>a) { printf("\n>>>> a no es mayor que c, pero b si es mayor que a. [ (%d == %d) Y (%d > %d)
]",a,c,b,a);   }
printf("\n");                                   /*linea en blanco*/
printf("\nFIN DEL PROGRAMA");
printf("\n");                                   /*linea en blanco*/
}
```

Fijaros que si usamos el operador && (y) para que la sentencia se cumpla se deben ser ciertas todas las condiciones expuestas. En caso contrario si usamos el operador || (o), una puede ser cierta y la otra no y en tal caso también se cumpliría la condición y se ejecutaría.

Operadores de manejo de bits

Estos operadores muestran una de las armas más potentes del lenguaje C , la de poder manipulear INTERNAMENTE , es decir bit a bit , las variables .
Debemos anticipar que estos operadores sólo se aplican a variables del tipo char , short , int y long y NO pueden ser usados con float ó double ,
Sabemos que las computadoras guardan los datos organizados en forma digital , en bytes , formado por números binarios de 8 bits y como se vió anteriormente cuando se analizó el tamaño de las variables , un char ocupará un byte de 8 bits , mientras que los short e int se forman con dos bytes (16 bits) y los long por cuatro bytes (32 bits).
Para el manejo de dichos bits , contamos con los operadores descriptos en la tabla:

SIMBOLO	DESCRIPCION	EJEMPLO
&	Y ó AND (bit a bit)	a & b
|	O ú OR INCLUSIVA	a | b
^	O ú OR EXCLUSIVA	a ^ b
<<	ROTACION A LA IZQUIER	a << b
>>	ROTACION A LA DERECHA	a >> b
~	COMPLEMENTO A UNO	~a

Preferencia de análisis de los operadores

El compilador de C da una prioridad al análisis de ciertos operadores sobre otros que subordina. En muchos casos esto nos lleva a cometer errores en nuestros códigos.

El orden de precedencia que da el lenguaje C es:

1. Paréntesis
2. Potencias
3. Productos y divisiones de izquierda a derecha
4. Sumas y restas
5. Concatenación (suma de caracteres “strings”)
6. Operadores relacionales
7. Negación
8. Conjunción (y).
9. Disyunción (o)

Pienso que no es necesario saber estas preferencia. Pero si es necesario obedecer y dar prioridad a ciertos operadores sobre otros y esto lo podemos lograr estableciendo las operaciones entre paréntesis. Agrupando los operandos con sus operadores en paréntesis nos ahorramos el trabajo de saber estas prioridades de memoria y clarificamos mucho mas el código. O sea que ya sabes ante la duda, abre y cierra paréntesis ☺. Es la mejor solución agrupar en llaves y poner cuantos paréntesis te parezca bueno.