TEMA 03
SENTENCIAS DE CONTROL DE PROGRAMA

3.1.- Introducción.

Las sentencias de control son aquellas que controlan y dirigen la ejecución de un programa. Hasta ahora hemos visto la ejecución de bloques de sentencias desde el principio hasta el final del programa, con un orden. Pero puede que queramos que un bloque de sentencias se ejecute sólo bajo determinadas circunstancias, o que un bloque de sentencias se ejecute varias veces.

Las sentencias pueden ser simples, separadas por el punto y la coma, y compuestas, varias sentencias simples agrupadas entre llaves.

Tendremos en cuenta que para escribir un código fuente mucha más legible iremos colocando las sentencias en niveles de forma que podamos diferenciar con facilidad cada grupo de sentencias.

Las sentencias pueden ser condicionales (if, switch), iterativas (while, for, do), incondicionales (break, continue, goto,return).

La mayoría de las sentencias de control de cualquier lenguaje están basadas en condiciones. Una condición es una expresión cuya resolución da como resultado cierto (true) o falso (false). Muchos lenguajes de programación incorporan los valores true y false; en C cualquier valor distinto de cero es true, y el valor cero es false.

3.2.- Sentencia condicional if.

Permite la ejecución de una sentencia o un bloque de sentencias, en el caso de que se cumpla la expresión lógica que lleva asociada. La sintaxis de la sentencia if es:

if (expresion1) sentencia1

Si la expresion1 es verdadera se ejecutará la sentencia1. Si expresion1 es falsa no se ejecuta sentencia1 y se prosigue el programa.

La sentencia if suele venir acompañada de else. La sintaxis de la sentencia if-else es:

```
if (expresion1) sentencia1
else if (expresion2) sentencia2
.....
else sentenciaN.
```

Si la expresion1 es verdadera se ejecutará la sentencia1. Si expresion1 es falsa se evalúa la expresion2 y si es verdadera se ejecuta la sentencia2. Si la expresion2 es falsa se pasa al siguiente bloque, y así con todos los bloques hasta llegar a la sentenciaN.

Ejemplos:

* Para el caso `if ( 3 < 4 ) x = 2` El valor de x es 2

* Para el caso
```
if ( 3 > 4 ) x = 2
else x = 5
```
El valor de x es 5

* Para el caso
```
if ( 3 > 4 ) x = 2
else if ( 2 < 5 ) x = 8
```
El valor de x es 8

* Para el caso
```
if ( 3 > 4 ) x = 2
else if ( 2 > 5 ) x = 8
else x = 14
```
El valor de x es 14

Programa 006 Ejemplo del uso de if

```
#include<stdio.h>

void main ()
{
      int n;
      printf("Introduzca un numero de una cifra, por favor: \n");
      scanf("%d",&n);
      printf("El numero es %d\n",n);
      if ((n>2)&&(n<6)) /* Con && damos la condición y */
            printf("El numero es 3, 4 o 5");
      else
            printf("El numero es 0, 1, 2, 6, 7, 8 o 9");
}
```

Atención con las sentencias if-else anidadas. Puede que el compilador no de errores y el programa no funcionar correctamente. El compilador asocia cada else con el último if sin else encontrado en el mismo bloque.

El operador condicional ? es una posible alternativa para if en algunas situaciones

Su expresión es: variable = expresión1 ? expresión 2 : expresión 3 ;

Si la expresión 1 es verdad la variable toma el valor de la expresión 2
Si la expresión 1 es falsa la variable toma el valor de la expresión 3

Ejemplo mayor = (a > b) ? 1 : 2 ;
Si a > b mayor = 1
Si a <= b mayor = 2

3.3.- Sentencia condicional switch.

Transfiere el control a un bloque de sentencias o a otro, dependiendo del valor de una expresión. Switch evalúa una expresión. A continuación evalúa cada una de las expresiones constantes hasta que encuentra una que coincida con la primera expresión. Cuando la encuentra ejecuta las sentencias correspondientes a ese case. Si no hay ninguna expresión case que coincida con la primera expresión, se ejecuta las sentencias correspondientes a default que es opcional.

Las expresiones a evaluar solo pueden ser de tipo entero o de tipo carácter. La sintaxis es:

```
switch (expresion)
{
    case expresion1:
        sentencia1
        break;
    case expresion2:
        sentencia2
        break;
    .....
    default:
        sentenciadefault
}
```

Programa 007 Ejemplo del uso de switch

```
#include<stdio.h>

void main ()
{
      int num1, num2;
      char op;
      printf("Introduce el primer numero: ");
      scanf("%d",&num1);
      printf("Introduce el segundo numero: ");
      scanf("%d",&num2);
      printf("¨ Que operacion queremos hacer con ellos ? ");
      scanf("%c",&op);
            * Con este recogemos el INTRO del numero anterior */
      scanf("%c",&op);
      switch(op)
            {
            case '+':
                  printf("El resultado de sumar %d y %d es
                  %d",num1,num2,num1+num2);
                  break;
            case '-':
                  printf("El resultado de restar %d y %d es
                  %d",num1,num2,num1-num2);
            break;
            default:
            printf("Operacion no valida");
            }
}
```

3.4.- Sentencia iterativa while.

Permite la ejecución de un bloque de sentencias sí se evalúa verdadera una expresión lógica. La expresión lógica aparece al principio del bloque de sentencias. La sintaxis es:

```
while ( expresión lógica )
{
      .....
}
```

El bloque delimitado por las llaves puede reducirse a una sentencia, y en este caso se pueden suprimir las llaves.

Cuando el programa llega a una sentencia while, sigue los siguientes pasos:

- Evalúa la expresión lógica.
- Si es falsa, continua la ejecución tras el bloque de sentencias.
- Si es verdadera entra en el bloque de sentencias asociado al while. Ejecuta dicho bloque de sentencias, y cuando finaliza vuelve al primer paso, evaluando de nuevo la expresión y actuando en consecuencia.

Si la primera evaluación resulta falsa, el bloque de sentencias no se ejecuta nunca.

Programa 008 Ejemplo del uso de while

```
#include<stdio.h>

void main ()

{
      int num1,num2,num3,suma, resta,multi,resto;
      printf("Introduzca la primera cifra entre 0 y 9: ");
      scanf("%d",&num1);
      printf("Introduzca la segunda cifra entre 0 y 9: ");
      scanf("%d",&num2);
      printf("Introduzca la tercera cifra entre 0 y 9: ");
      scanf("%d",&num3);
      while(num1>5)
      {
            suma=num1+num2;
            multi=num1*num2;
            printf("La suma de %d y %d es %d ",num1,num2,suma);
            printf("\nLa   multiplicacion   de   %d   y   %d   es
            %d",num1,num2,multi);
            num1=2; /* para romper el while */
      }
      resta=num3-num2;
      resto=num3%num2;
 printf("\n\nLa resta de %d y %d es %d ",num3,num2,resta);
 printf("\nEl resto de dividir %d y %d es %d ",num3,num2,resto);
}
```

3.5.- Sentencia iterativa do-while.

Ejecuta un bloque de sentencias mientras se cumpla una expresión lógica. La expresión lógica aparece al final del bloque de sentencias. Igual que en la sentencia while, el bloque delimitado por las llaves puede reducirse a una sentencia, y en este caso se pueden suprimir las llaves. El programa ejecuta el bloque de sentencias entre llaves y cuando llega al final del bloque, evalúa la expresión lógica.

- Si es falsa, continua la ejecución del programa tras el bloque do .... while.
    1. Si es verdadera, retorna a la sentencia do. Vuelve a ejecutar el bloque y a evaluar la expresión lógica al final de este, actuando en consecuencia.

Vemos, que al contrario de la sentencia while, el bloque de sentencias asociado a una sentencia do-while se ejecuta al menos una vez.

Programa 009 Ejemplo de la sentencia do-while.

```
#include<stdio.h>
void main ()
{
      int i;
      i=1;
      do
      {
            printf("\n%d",i++);
      }while(i<=10);
}
```

3.6.- Sentencia iterativa for.

Permite la ejecución de un bloque de sentencias sí se evalúa verdadera la expresión lógica, al igual que la sentencia while. La característica que diferencia a la sentencia for de la sentencia while, es que realiza tres operaciones diferentes dentro de una sola sentencia:

- Sentencias ejecutadas antes de entrar en el bloque delimitado por la sentencia for.
- Expresión lógica que indica si se repetirá o no la ejecución del bloque de sentencias.
- Sentencias ejecutadas al finalizar cada ejecución del bloque de sentencias.

La sintaxis de la sentencia for es:

```
for ( sentencia1; expresión lógica; sentencia2)
{
.....................
}
```

El programa al llegar a la sentencia for, ejecuta la sentencia1. Después evalúa la expresión lógica, y si es verdadera, ejecuta el bloque de sentencias entre llaves. en este caso, al llegar la final del bloque, ejecuta la sentencia2, y vuelve al principio del for. Evalúa la expresión lógica y actúa en consecuencia.

```
Programa 010 Ejemplo de la sentencia for.

#include<stdio.h>

void main ()
{
      int x;
      for(x=1;x<=10;x++)
      {
            printf("\n%d",x);
      }
}
```

3.7.- Sentencia de salto break.

El uso de esta sentencia no es recomendable.

Esta sentencia provoca la salida inmediata de las sentencias swich, while, for o do-while. Por lo tanto su uso solo es correcto dentro de un bloque de una de estas sentencias.

Programa 011 Ejemplo de la sentencia break

```
#include<stdio.h>

void main ()

{
      int num1,num2,suma, resta,multi,resto;
      printf("Introduzca la primera cifra entre 0 y 9: ");
      scanf("%d",&num1);
      printf("Introduzca la segunda cifra entre 0 y 9: ");
      scanf("%d",&num2);
      while(num1>5)
            {
            suma=num1+num2;
            multi=num1*num2;
            printf("La suma de %d y %d es %d ",num1,num2,suma);
            printf("\nLa   multiplicacion   de   %d   y   %d   es
            %d",num1,num2,multi);
            break;
            }
      resta=num1-num2;
      resto=num1%num2;
      printf("\n\nLa resta de %d y %d es %d ",num1,num2,resta);
      printf("\nEl   resto   de   dividir   %d   y   %d   es   %d
      ",num1,num2,resto);
}
```

3.8.- Sentencia de salto continue.

El uso de esta sentencia no es recomendable.

Funciona algo similar a break. Solo puede ser empleada en el interior de una sentencia de repetición. Es un salto incondicional al final del bucle. No es un salto al exterior del bucle, sino al final de su bloque de sentencias. En vez de forzar la terminación, continue fuerza una nueva iteración del bucle y salta cualquier código que exista entre medias.

Programa 012 Ejemplo de la sentencia continue.

```
#include<stdio.h>

void main ()
{
      int i;
      printf("Los numeros pares menores 25 son: ");
      for(i=0;i<25;i++)
      {
            if((i%2)!=0)
                  continue;
            else
                  printf("\n%d",i);
      }
}
```

3.9.- Sentencia de salto goto.

El uso de esta sentencia no es recomendable.

La sentencia goto provoca un salto incondicional a una etiqueta que se encuentra en la misma función. Las etiquetas son identificadores seguidos de dos puntos (:) y poseen su propio espacio nominal (dentro del mismo bloque no pueden aparecer dos etiquetas iguales).

El compilador no se encarga de realizar comprobaciones respecto al lugar hacia donde se salta. Por ejemplo, si el salto se produce al interior de un bucle, pueden originarse problemas debido a que salta la inicialización de dicho bucle.

3.10.- Sentencia de salida return.

Es un salto incondicional al final del cuerpo de una función, a la llave de fin de bloque. Es de uso obligado en las funciones que devuelven un valor. Hablaremos de esta sentencia cuando veamos el tema de las funciones.