FORMAS Y ESTATUTOS DE CONTROL

if-else
else-if
switch
while y for
do-while
break y continue
goto y etiquetas

Las proposiciones de control de flujo de un lenguaje especifican el orden en que se realiza el procesamiento. Una de las actividades que con mayor frecuencia realizan los programas, es la repetición de instrucciones o estatutos. Estos se ejecutan un numero determinado de veces y dependen de que ciertas condiciones se cumplan. Existen muchas situaciones en que los programas necesitan ejecutar la misma tarea repetidas veces, para poder indicarle a la computadora la ejecución de lazos. La cantidad de veces que se ejecuta un lazo deberá indicarse en los comandos que la controlen, ya sea una condición lógica de terminación o un numero determinado de veces.

if-else :

Se utiliza para expresar decisiones, la sintaxis es la siguiente

```
if(expresión)
        proposicion1
else
        proposicion2
```

Donde la parte else es optativa. La expresión se evalúa; si es verdadera (si la expresión tiene un valor diferente de cero), la proposicion1 se ejecuta. Si es falsa (expresión cero), y si existe una parte de else, la proposicion2, se ejecuta en su lugar.

else-if:

La sintaxis es la siguiente

```
if(expresión)
        proposicion1
else if(expression)
        proposicion2
else if(expression)
        proposicion3
else if(expression)
        proposicion4
else
        proposicion5
```

Esta secuencia de proposiciones if es la forma más general de escribir una decisión múltiple. Las expresiones se evalúan en orden; si cualquier expresión es verdadera, la proposición asociada se ejecuta, y esto termina toda la cadena.

switch: Esta es una decisión múltiple que prueba si una expresión coincide con uno de un número de valores constantes enteros y traslada el control adecuadamente.

```
switch(expresión){
        case exp-cte: proposiciones
        case exp-cte: proposiciones
        default: proposiciones
}
```

Cada case se etiqueta con uno o mas valores constantes enteros o expresiones constantes enteras. Si un case coincide con el valor de la expresión, la ejecución comienza en este. Todas las expresiones case deben ser diferentes. El etiquetado como default se ejecuta si ninguno de los otros se satisface (es optativo).

while y do-while: Ciclo que permite la repetición para ejecutar las mismas líneas de código. Por lo general se acompañan de un contador que contabiliza el número de ciclos a realizar.

```
while(expresión)
        proposición      /*verifica al principio */
```

La expresión se evalúa, si es diferente de cero, se ejecuta la proposición y se reevalúa la expresión. este ciclo continua hasta que la expresión se hace cero, punto en el cual se suspende la ejecución para continuar después de la proposición.

La proposición do-while, Prueba al final después de realizar cada paso a través del cuerpo del ciclo, el cual se ejecuta siempre por lo menos una vez.

```
do
        proposición
while(expresión);
```

for:

Esta proposición presenta la siguiente sintaxis:

```
 for(expr1;expr2;expr3)
        proposición;
```

expr1 y expr3 son asignaciones o llamadas a función y expr2 es una expresión de relación. Cualquiera de las tres partes se puede omitir, permaneciendo siempre los punto y coma. Si la prueba expr2 no esta presente, se toma como permanentemente verdadera. Cuando se entra por primera vez al for, el orden de evaluación de las expr es en el orden siguiente: expr1, expr2, si es verdadera la segunda expresión se realiza el contenido del for. Al regresar al for el orden de evaluación de las expr es el siguiente: expr3, expr2, si es verdadera la expr2 se realiza el contenido del for, este procedimiento continua hasta que la expr2 es falsa.

```
for(;;){
...
}
```

es una iteración infinita, que presumiblemente será interrumpida por otros medios, como un break o un return.

break:

Con esta instrucción se puede abandonar el ciclo sin tener la necesidad de probar al inicio o al final. La proposición break proporciona una salida anticipada de un for, while y do, tal como lo hace el switch. Un break provoca que el ciclo o switch más interno que lo encierra termine inmediatamente.

continue:

Cuando se ejecuta esta instrucción dentro de un ciclo, el control del programa regresa a la línea donde inicia el ciclo que lo contiene y continua su ejecución.

goto y etiquetas:

Formalmente, el goto nunca es necesario, y en la práctica es casi siempre más fácil escribir código sin el. Hay situaciones donde los goto pueden encontrar un lugar. La más común es abandonar el procesamiento en alguna estructura profundamente anidada, tal como salir de dos o más ciclos a la vez; break no puede utilizarse directamente, puesto que solo sale del ciclo que lo contiene.