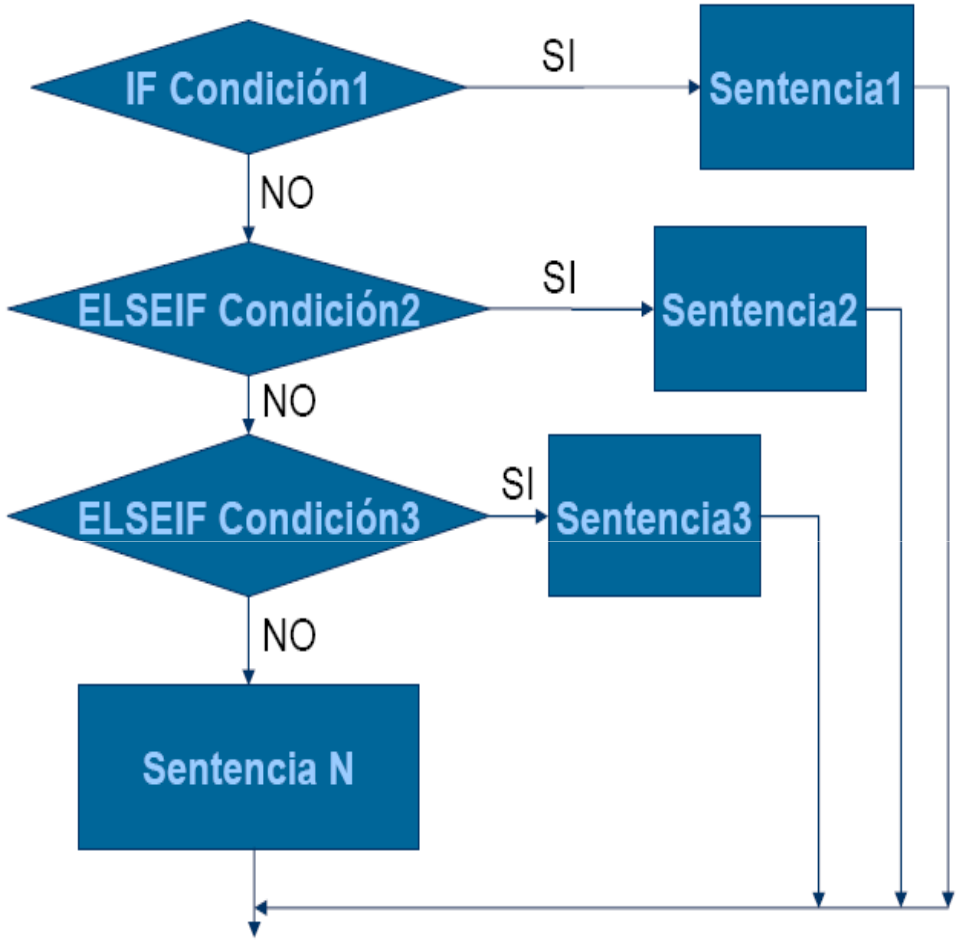
MÚLTIPLES

Estructuras condicionales anidadas

```
if (expresion1 o condición1){
    Sentencias1
}
else {
            if (expresion2){
          Sentencias2
                        }
            else {
                  if (expresion3){
                     Sentencia3
                                  }
                   else {
                   Sentencia 4
                     }
          }
    }
```



SELECCIÓN MÚLTIPLE CON SENTENCIA switch

Permite seleccionar entre varias alternativas posibles

```
switch (expresión) {
    case expr_cte1:
    sentencia1;
     case expr_cte2:
     sentencia2;
      ...
     case expr_cteN:
     sentenciaN;
    default:
    sentencia;
}
```

SWITCH (CONTINUACIÓN)

Se evalúa expresión:

- Si es igual a alguna expresión_cte_#
  - Ejecuta el bloque de código asociado y los siguientes.
  - Si solo se quiere un solo bloque, se coloca *break;* al final del bloque.
- En caso contrario, ejecuta *default*

```
#include <stdio.h>
 int main()
{
    int nota;
   printf(“Calificación: ”);
   scanf(“%d”, &nota);
  switch (nota) {
    case 0:
    case 1:
    case 2:
    case 3:
    case 4:
      printf(“Suspenso”);
      break;
    case 5:
    case 6:
      printf(“Aprobado”);
      break;
    case 7:
    case 8:
      printf(“Notable”);
       break;
    case 9:
      printf(“Sobresaliente”);
      break;
    case 10:
      printf(“Muy bien”);
      break;
    default:
       printf(“Error”);
  }
}
```

ESTRUCTURAS CÍCLICAS O REPETITIVAS

KAREN SAENZ- LAURA SANDOVAL

ASIGNACIÓN (USO DE LAS VARIABLES)

Simples: Consiste en pasar un valor constante a una variable

(a ← 15)

Contador: Consiste en usarla como un verificador del número de veces que se realiza un proceso (a ← a + 1)

Acumulador: Consiste en usarla como un sumador en un proceso (a← a + b)

De trabajo: Donde puede recibir el resultado de una operación matemática que involucre muchas variables (a ← c + b * (2 / 4)).

Formato a utilizar es el siguiente

< Variable > ← <valor o expresión >

El símbolo ← debe leerse “asigne”.

OPERADORES DE ASIGNACIÓN

int c=3,d=5,e=4,f=6,g=12

Operador	Ejemplo	Explicación	Asigna
+=	c += 7	c = c + 7	10 a c
-=	d -= 4	d = d – 4	1 a d
*=	e *= 5	e = e * 5	20 a e
/=	f /= 3	f = f / 3	2 a f
%=	g %= 9	g = g % 9	3 a g

OPERADORES INCREMENTALES/DECREMENTALES

Operador	Ejemplo	Explicación
++	++a	Se incremental **a** en 1 y luego se utiliza el nuevo valor de **a** en la expresión en la cual resida **a**.
++	a++	Utilizar el valor actual de **a** en la expresión en la cual reside **a** y después se incrementa **a** en 1
--	--b	Se decrementa **b** en 1 y a continuación se utiliza el nuevo valor de **b** en la expresión en la cual reside **b**.
--	b--	Se utiliza el valor actual de **b** en la expresión en la cual reside **b** y después se decrementa a **b** en 1

EJEMPLO

```
#include<stdio.h>
int main( )
{
int c=5;
printf (" %d \n",c );
printf (" %d \n",c++ );
printf (" %d \n",c );
c=5;
printf (" %d \n",c );
printf (" %d \n",++c );
printf (" %d \n",c );
return 0;
}
```

ESTRUCTURAS CÍCLICAS O REPETITIVAS

Llamadas también bucles o ciclos

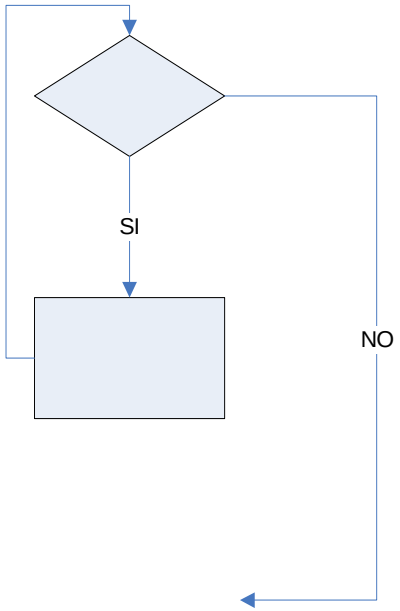
Repiten una secuencia de instrucciones un número determinado de veces

Se llama Iteración al hecho de repetir la ejecución de una secuencia de acciones hasta que se cumpla una condición.

KAREN SAENZ- LAURA SANDOVAL

ESTRUCTURA O BUCLE WHILE

Mientras la condición sea verdadera se realizara un número de instrucciones repetidamente, el ciclo terminará si la condición es falsa.



KAREN SAENZ- LAURA SANDOVAL

WHILE

Sintaxis en el lenguaje C :

```
while(<expresión >){
  instrucciones;
}
```

```
/*Una sola sentencia o instrucción*/
while (<expresión>)
Sentencia o instrucción;
```

EJEMPLO

Imprimir los 10 primeros numeros naturales

```
 #include <stdio.h>
int  main() {  /*inicio main */
  int i;
     i = 1;
      while (i<=10) {  /*  primero se verifica la condición*/
             printf ("\n%d",i);
             i++;
            }
return 0;
  } /*fin main*/
```

EJEMPLO

Calcular la siguiente suma $\sum_{i=1}^{100} 2i = 2+4+6+8+\ldots+$

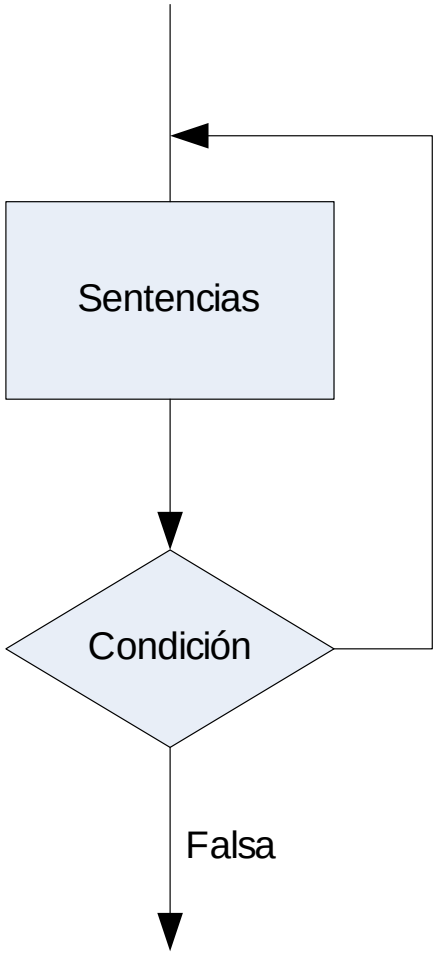
```
#include<stdio.h>
int main(){
   int suma=0,i=1,numero=2;
     while(numero <10){
      numero=i*2;
      suma = suma + numero;
        printf("\n Iteración %d , numero %d  suma %d",i ,numero,suma) ;

       i++;
     }
  printf("la suma es%d",suma);
return 0;
}
```

ESTRUCTURA DO ... WHILE

Sintaxis

```
 do {
Instrucción
} while (exp2);
```



EJEMPLO

```
#include <stdio.h>

int main() {
int  i = 0;

do {
printf ("\n%d",i);  /* siempre realiza  por lo menos
  una vez las       instricciones*/
 i++;
    } while (i < 10 );  /* una vez realizada  la o las
            instrucciones,
                                    se revisa la condición */
}
```

EJERCICIO

Realizar la suma

Utilizando do-while

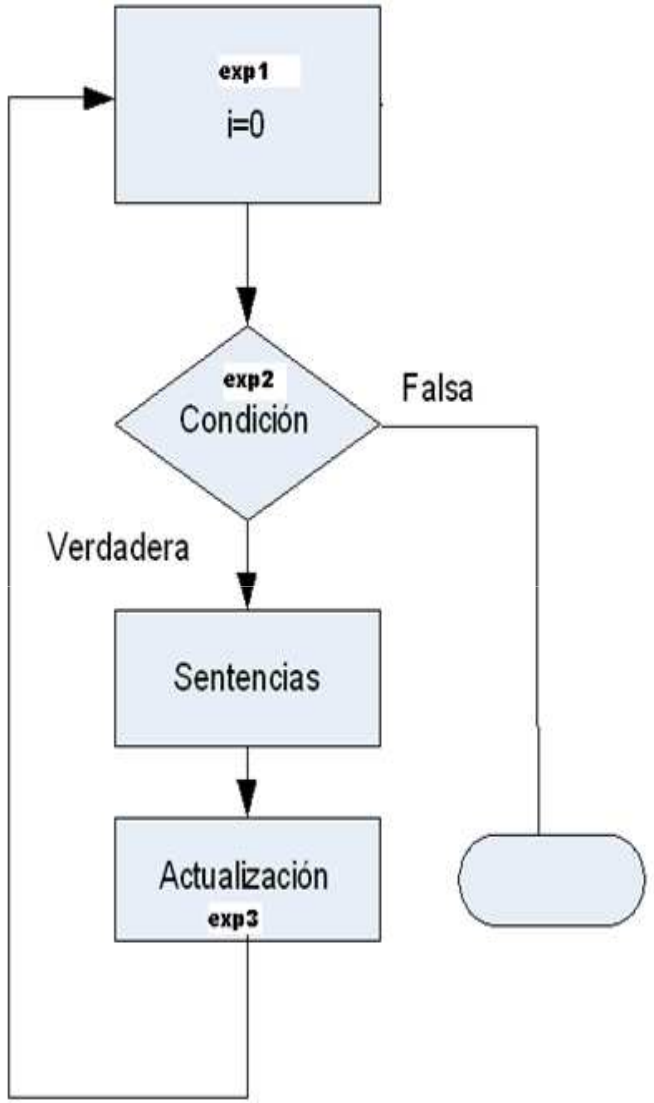
$$\sum_{i=1}^{100} 2i = 2 + 4 + 6 + 8 + \ldots +$$

KAREN SAENZ- LAURA SANDOVAL

ESTRUCTURA FOR

Sintaxis en C

```
for (exp1;exp2;exp3) {
   varias_instrucciones;
                        }
```



EJEMPLO

```
#include <stdio.h>
main ( )
{
 int i;   /* Esta variable la utilizaremos como contador*/
   for (i=10; i>0;i–)  /*Realiza desde i=10 hasta i<0,i se decrementa i=i-1*/
    {
     printf ("\n%d",i);
    }
}
```

FACTORIAL DE UN NÚMERO

n!=1x2x3 . . . . .x(n-1)xn

5!=1x2x3x4x5

KAREN SAENZ- LAURA SANDOVAL

EJEMPLO

```
#include <stdio.h>
void main()
{
long i, n, factorial;
printf ("Introduzca un número: ");
scanf ("%ld", &n);
  factorial = 1;
    for (i=1; i<=n; i++) {
      factorial *= i;
     }
printf ("factorial(%ld) = %ld", n, factorial);
}
```

CRITERIOS DE UTILIZACIÓN DE LOS BUCLES

while: Si hay casos en los que el bucle no se ejecute

do-while: Si la parte de ejecución del bucle se ha de hacer al menos una vez

for: Si se sabe el número de veces que se ha de repetir el bucle.