INTRODUCCIÓN A LA PROGRAMACIÓN Nivel 1

ESTRUCTURAS ITERATIVAS

Capacítate para el empleo | FUNDACIÓN Carlos Slim

Cuando en un programa se necesitan ejecutar declaraciones o instrucciones en más de una ocasión, se utilizan estructuras iterativas o bucles que lo hacen más sencillo. Para poder hacerlo, debes emplear un contador **(c)**, que es una variable int de valor cero, que se incrementará una unidad cada vez que las acciones se repitan **(c=c+1 ó c++)**.

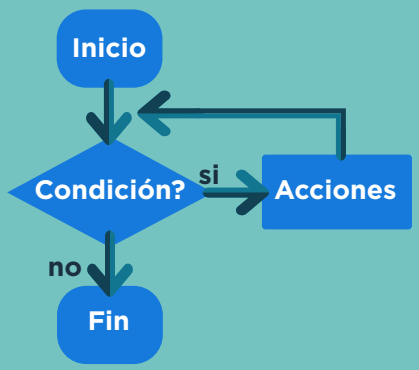
Las estructuras iterativas más comunes son:

WHILE

USO

Mientras una condición se evalúe como verdadera, las acciones asociadas serán ejecutadas.
Esta estructura también permite hacer iteraciones infinitas si se desarrolla un while con condición 1.

DIAGRAMA



EJEMPLO

```
int c=0,n=0;

while (c<3)
{
c=c+1;
n+=c;
}
```

EXPLICACIÓN

El contador "c" aumentará uno cada vez que no sea mayor a 3, es decir:

- **Al ingresar: c=0 y n = 0, por lo que c<3, después c=c+1=0+1=1, por lo que n=n+c=0+1=1 quedando: c=1 y n = 1**
- **Después del segundo pase: c=2 y n = 3**
- **Después del tercer pase: c=3 y n = 6**

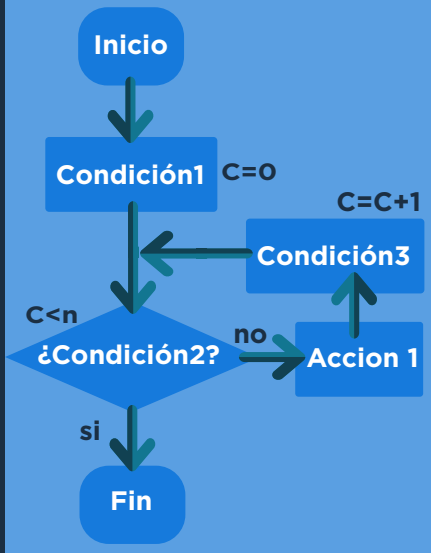
Después de repetir tres veces, la condición c < 3 no será verdadera más tiempo, por lo que se termina el while.

FOR

USO

Mediante éste se puede especificar el número de veces que debes repetir las acciones.
Donde la condición 1 hace referencia al valor del contador,
la condición 2 es la que se debe cumplir para que se repita el ciclo
y la condición 3 se refiere al incremento del contador.

DIAGRAMA



EJEMPLO

```
int c,n=0;

for(c=0;c<3;c++)
{
n+=c;
}
```

EXPLICACIÓN

La variable n=0 y “c” son enteros. Se realiza la declaración for donde la condición 1 es que c=0, la condición 2 es que se realizará mientras “c” sea menor a 3
y la condición 3 es que el contador se aumentara una unidad, por lo que:

- Después del primer pase: c=1 y n = 1
- Después del segundo pase: c=2 y n = 3
- Después del tercer pase: c=3 y n = 6

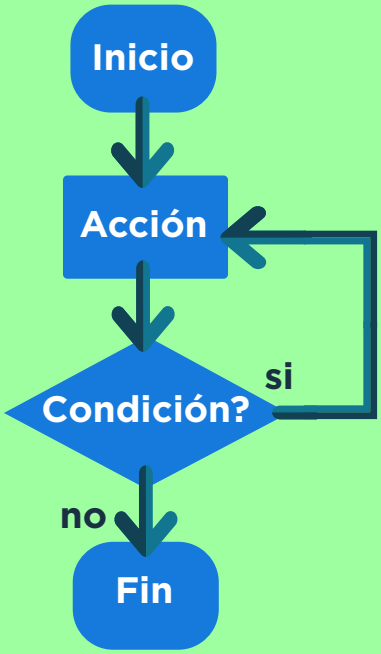
Después de esto, la condición n < 3 no será verdadera, por lo que se termina el proceso.

DO-WHILE

USO

Permite realizar una acción específica mientras no se cumpla una condición.

DIAGRAMA



EJEMPLO

```
int c=0,n=0;

do
{
c=c+1;
n+=c;
} while (c<3)
```

EXPLICACIÓN

El contador “c” aumentará uno y la variable “n” hará n=n+c cada vez que c no sea mayor a 3; es decir:

- Al ingresar:

c=0 y n = 0 por lo que, c=c+1=0+1=1 y n=n+c=0+1=1, después se evalúa la condición c<3 quedando: c=1 y n = 1

- Después del segundo pase:

c=2 y n = 3

- Después del tercer pase:

c=3 y n = 6

Posterior a esta evaluación, se determina que la condición c < 3 ya no se cumple por lo que se detienen las repeticiones.

Existen dos instrucciones que permiten a las estructuras iterativas dar opciones:

BREAK

Usada cuando se quiere dar por terminada una iteración, aunque la condición se siga cumpliendo. Por ejemplo:

```
int c=0;  ---------- 1. Se declara el contador igual a 0.

while(c<4)  ---------- 2. Mientras el contador sea menor a 4,
                          se ejecutarán las acciones.
{

c++  ---------- 3. Se le agrega una unidad al contador

if(c==2)break; ---------- 4. Si el contador es igual a 2,
                            el proceso termina pese a que tenga
                            un valor menor a 3.

printf("Iteración %d\n",i)  ---------- 5. Se muestra el texto entre ""
                                          con el número de iteraciones realizadas.
                                          En este caso, se verá:
}                                         "Iteración 1", "Iteración 2".
```

CONTINUE

Ésta regresa el programa a la siguiente iteración sin completar las acciones para la que está escrito, es decir, las salta.

```
int c=0; ----------------------
```
1. Se declara el contador igual a 0.

```
while(c<4) ------------------
```
2. Mientras el contador sea menor a 3, se ejecutarán las acciones.

```
{
```

```
c++ ----------------------
```
3. Se le agrega una unidad al contador.

```
if(c==2)continue; ------------
```
4. Si el contador es igual a 2, el proceso se salta este paso y continua en el siguiente.

```
printf("Iteración %d\n",i) ------
```
5. Se muestra el texto entre "" con el número de iteraciones realizadas. En este caso, se verá: "Iteración 1", "Iteración 3", "Iteración 4".

```
}
```

CONTINUE

Ésta regresa el programa a la siguiente iteración sin completar las acciones para la que está escrito, es decir, las salta.

```
int c=0; ----------------------
```
1. Se declara el contador igual a 0.

```
while(c<4) ------------------
```
2. Mientras el contador sea menor a 3, se ejecutarán las acciones.

```
{
```

```
c++ ----------------------
```
3. Se le agrega una unidad al contador.

```
if(c==2)continue; ------------
```
4. Si el contador es igual a 2, el proceso se salta este paso y continua en el siguiente.

```
printf("Iteración %d\n",i) ------
```
5. Se muestra el texto entre "" con el número de iteraciones realizadas. En este caso, se verá: "Iteración 1", "Iteración 3", "Iteración 4".

```
}
```