### Ciclos anidados

Un ciclo anidado es un ciclo que se encuentra incluido en el bloque de instrucciones de otro bloque. <u>Los ciclos pueden</u> <u>tener cualquier nivel de anidamiento</u> (un ciclo dentro de otro ciclo dentro de un tercero, etc.). Al ciclo que se encuentra dentro del otro se le denomina ciclo interior o interno. El otro ciclo sería el ciclo exterior o ciclo externo.

En los ciclos anidados es importante utilizar variables de control distintas, para no obtener resultados inesperados, salvo en los casos que el algoritmo lo exija.

## Ciclos anidados con variables independientes

Los ciclos anidados con variables independientes son los ciclos en los que ninguna de las variables de cada uno de los ciclos interviene ni en la condición de continuación ni en la expresión de paso de los otros ciclos.

Se pueden combinar todos los ciclos, por una cuestión didáctica se muestran los siguientes ejemplos utilizando el mismo tipo de ciclo y con el fin de que se compruebe el funcionamiento:

Programa (while)	Resultado
#include <stdio.h></stdio.h>	Ciclo while
int main()	0 iteración de x
int main()	Estoy en x: 0 - y: 0
{	Estoy en x: 0 - y: 1
int x = 0; /* inicializa x */	Estoy en x: 0 - y: 2
	Estoy en x: 0 - y: 3
int y; /* define y */	Estoy en x: 0 - y: 4
	Estoy en x: 0 - y: 5
colors (II)	Estoy en x: 0 - y: 6
printf("Ciclo while");	Estoy en x: 0 - y: 7
	Estoy en x: 0 - y: 8 Estoy en x: 0 - y: 9
	1 iteración de x
	Estoy en x: 1 - y: 0
while ( $x < 3$ ) { /*ciclo exterior repite 3 veces*/	Estoy en x: 1 - y: 1
printf("%d iteración de x\n", x); /* mensaje */	Estoy en x: 1 - y: 2
	Estoy en x: 1 - y: 3
y = 0;	Estoy en x: 1 - y: 4
while ( y < 10 ) {/* ciclo interior repite 10 veces */	Estoy en x: 1 - y: 5
printf( "Estoy en x: %d - y: %d\n",x,y); /* mensaje */	Estoy en x: 1 - y: 6
	Estoy en x: 1 - y: 7
y++; /* incrementa la y */	Estoy en x: 1 - y: 8
} /* fin del while interno */	Estoy en x: 1 - y: 9
x++; /* incrementa la x */	2 iteración de x
printf( "\n" ); /* salto de línea */	Estoy en x: 2 - y: 0
printi( \n ); / · saito de linea · /	Estoy en x: 2 - y: 1
} /* fin del while externo */	Estoy en x: 2 - y: 2
return 0; /* indica que el programa terminó con exito */	Estoy en x: 2 - y: 3
	Estoy en x: 2 - y: 4
} /* fin de la función main */	Estoy en x: 2 - y: 5
	Estoy en x: 2 - y: 6
	Estoy en x: 2 - y: 7
	Estoy en x: 2 - y: 8
	Estoy en x: 2 - y: 9



Se ingresa al ciclo externo cuando la condición es verdadera respecto del valor de su variable de control. Se ejecuta el printf, luego la variable y se inicializa en 0. Y luego se evalúa la condición del ciclo interno, respecto al valor de su variable de control y se ingresa si es verdadera...

...Si la condición es verdadera, el ciclo interno se ejecutará por completo, con todas sus instrucciones, hasta que la *condición sea falsa* (de acuerdo al valor de su variable de control). En ese momento se sale del ciclo interno.

En los ciclos anidados es muy importante
observar dónde y cuándo se inicializan
contadores y acumuladores, que dependerá de lo
que exija el algoritmo.

Se ejecutan las instrucciones que corresponden al ciclo externo. Luego se evaluará la condición para el ciclo externo (de acuerdo al valor de su variable de control). Si la condición del ciclo externo es verdadera, vuelve a ejecutarse todo nuevamente. Y así sucesivamente hasta terminar el programa.

Programa (for)	Resultado
#include <stdio.h></stdio.h>	Ciclo for
int main()	
	0 iteracion de x
{	Estoy en x: 0 y: 0
int x; /* define x */	Estoy en x: 0 y: 1 Estoy en x: 0 y: 2
int y; /* define y */	Estoy en x: 0 y: 3
int y, y define y y	Estoy en x: 0 y: 4
	Estoy en x: 0 y: 5
printf("\n");	Estoy en x: 0 y: 6
	Estoy en x: 0 y: 7
	Estoy en x: 0 y: 8
for $(x = 0; x < 3; x++) \{ /*ciclo exterior repite 3 veces*/$	Estoy en x: 0 y: 9
printf("\n%d iteración de x", x); /* mensaje */	1 iteracion de x
for (y = 0; y < 10; y++) { /* ciclo interior repite 10 veces */	Estoy en x: 1 y: 0
	Estoy en x: 1 y: 1
printf("\nEstoy en x: %d y: %d", x, y); /* mensaje */	Estoy en x: 1 y: 2
} /* fin del for interno */	Estoy en x: 1 y: 3
printf("\n"); /* salto de línea */	Estoy en x: 1 y: 4
	Estoy en x: 1 y: 5
}/* fin del for externo */	Estoy en x: 1 y: 6
return 0; /* indica que el programa terminó con exito */	Estoy en x: 1 y: 7
} /* fin de la función main */	Estoy en x: 1 y: 8 Estoy en x: 1 y: 9
	Estoy eli x. 1 y. 9
	2 iteracion de x
	Estoy en x: 2 y: 0
	Estoy en x: 2 y: 1
	Estoy en x: 2 y: 2
	Estoy en x: 2 y: 3
	Estoy en x: 2 y: 4
	Estoy en x: 2 y: 5
	Estoy en x: 2 y: 6
	Estoy en x: 2 y: 7 Estoy en x: 2 y: 8
	Estoy en x: 2 y: 9.
	LJLOY CIT A. Z y. J.

## Análisis

```
for (x = 0; x < 3; x++) { /*ciclo exterior repite 3 veces*/
printf("\n%d iteración de x", x); /* mensaje */

for (y = 0; y < 10; y++) { /* ciclo interior repite

10 veces */
    printf("\nEstoy en x: %d y: %d", x, y); /*
    mensaje */
    } /* fin del for interno */

printf("\n"); /* salto de línea */

} /* fin del for externo */
```

Se ingresa al ciclo externo evaluando la condición respecto del valor de la variable de control. Luego se ejecuta el printf, y luego se ingresa al ciclo interno evaluando variable de control y condición...

..el ciclo interno se ejecutará por completo, con todas sus instrucciones, hasta que la <u>condición</u> <u>sea falsa</u> (de acuerdo al valor de su variable de control). En ese momento se sale del ciclo interno.

En los ciclos anidados es muy importante
observar dónde y cuándo se inicializan
contadores y acumuladores, que dependerá de lo
que exija el algoritmo.

Se ejecutan las instrucciones que corresponden al ciclo externo. Luego se evaluará la condición para el ciclo externo (de acuerdo al valor de su variable de control). Si la condición del ciclo externo es verdadera, vuelve a ejecutarse todo nuevamente. Y así sucesivamente hasta terminar el programa.

Programa (do-while)	Resultado
#include <stdio.h></stdio.h>	Ciclo do-while
int main()	0 iteracion de x
	Estoy en x: 0 y: 0
{	Estoy en x: 0 y: 1
int x = 0; /* inicializa la x */	Estoy en x: 0 y: 2
int y = 0; /* define la y */	Estoy en x: 0 y: 3 Estoy en x: 0 y: 4
int y = 0, / define ia y /	Estoy en x: 0 y: 5
	Estoy en x: 0 y: 6
printf("");	Estoy en x: 0 y: 7
<u> </u>	Estoy en x: 0 y: 8
	Estoy en x: 0 y: 9
do{	
printf("\n%d iteracion de x", x); /* mensaje */	1 iteracion de x
	Estoy en x: 1 y: 0
y = 0;	Estoy en x: 1 y: 1
do {	Estoy en x: 1 y: 2
printf("\nEstoy en x: %d y: %d", x, y); /* mensaje */	Estoy en x: 1 y: 3 Estoy en x: 1 y: 4
	Estoy en x: 1 y: 5
y++; /* incrementa la y */	Estoy en x: 1 y: 6
} while (y < 10); /* ciclo interior repite 10 veces y determina el	Estoy en x: 1 y: 7
fin del do-while interno */	Estoy en x: 1 y: 8
	Estoy en x: 1 y: 9
x++; /* incrementa la x */	
printf("\n"); /* salto de línea */	2 iteracion de x
	Estoy en x: 2 y: 0
}while (x<3); /*ciclo exterior repite 3 veces y determina el fin del	Estoy en x: 2 y: 1
do-while externo */	Estoy en x: 2 y: 2 Estoy en x: 2 y: 3
	Estoy en x: 2 y: 4
	Estoy en x: 2 y: 5
return 0; /* indica que el programa terminó con exito */	Estoy en x: 2 y: 6
} /* fin de la función main */	Estoy en x: 2 y: 7
	Estoy en x: 2 y: 8
	Estoy en x: 2 y: 9

## Análisis

```
do{
    printf("\n%d iteracion de x", x); /* mensaje */
    y = 0;

    do {
        printf("\nEstoy en x: %d y: %d", x, y); /*
        mensaje */
        y++; /* incrementa la y */
        } while (y < 10); /* ciclo interior repite 10 veces
        y determina el fin del do-while interno */

x++; /* incrementa la x */
    printf("\n"); /* salto de línea */
} while (x<3); /*ciclo exterior repite 3 veces y determina el
    fin del do-while externo */
```

Se ingresa al ciclo externo. Se ejecuta el printf, se inicializa la variabla y en 0 y luego se ingresa al ciclo interno...

..el ciclo interno se ejecutará por completo, con todas sus instrucciones, hasta que la <u>condición</u> <u>evaluada al final del bloqu sea falsa</u> (de acuerdo al valor de su variable de control). En ese momento se sale del ciclo interno.

En los ciclos anidados es muy importante
observar dónde y cuándo se inicializan
contadores y acumuladores, que dependerá de lo
que exija el algoritmo.

Se ejecutan las instrucciones que corresponden al ciclo externo. Al finalizar el bloque externo se evaluará la condición para el ciclo do-while (de acuerdo al valor de su variable de control). Si la condición del es verdadera, vuelve a ejecutarse todo nuevamente. Y así sucesivamente hasta terminar el programa.

## Algunos ejemplos de ciclos combinados

Programa	Resultado
#include <stdio.h></stdio.h>	Ciclo do-while/while
int main()	0 iteracion de x
"	Estoy en x: 0 - y: 0
{	Estoy en x: 0 - y: 1
int x = 0; /* inicializa la x */	Estoy en x: 0 - y: 2
	Estoy en x: 0 - y: 3
int y = 0; /* define la y */	Estoy en x: 0 - y: 4
printf("");	Estoy en x: 0 - y: 5
	Estoy en x: 0 - y: 6
do{	Estoy en x: 0 - y: 7
printf("\n%d iteracion de x", x); /* mensaje */	Estoy en x: 0 - y: 8 Estoy en x: 0 - y: 9
y = 0;	Estay en x. 0 - y. 9
•	1 iteracion de x
while ( y < 10 ) $\{/* \text{ ciclo interior repite 10 veces } */$	Estoy en x: 1 - y: 0
printf( "\nEstoy en x: %d - y: %d",x,y); /* mensaje */	Estoy en x: 1 - y: 1
	Estoy en x: 1 - y: 2
y++; /* incrementa la y */	Estoy en x: 1 - y: 3
} /* fin del while interno */	Estoy en x: 1 - y: 4
v /* ingramenta la v.*/	Estoy en x: 1 - y: 5
x++; /* incrementa la x */	Estoy en x: 1 - y: 6
printf("\n"); /* salto de línea */	Estoy en x: 1 - y: 7
}while (x<3); /*ciclo exterior repite 3 veces y determina el fin del	Estoy en x: 1 - y: 8
	Estoy en x: 1 - y: 9
do-while externo */	
	2 iteracion de x
	Estoy en x: 2 - y: 0
return 0; /* indica que el programa terminó con exito */	Estoy en x: 2 - y: 1
} /* fin de la función main */	Estoy en x: 2 - y: 2
•	Estoy en x: 2 - y: 3
	Estoy en x: 2 - y: 4
	Estoy en x: 2 - y: 5 Estoy en x: 2 - y: 6
	ESLUY EIT X. Z - Y. O

```
Estoy en x: 2 - y: 7
Estoy en x: 2 - y: 8
Estoy en x: 2 - y: 9
```

Programa	Resultado
#include <stdio.h></stdio.h>	Ciclo for/do-while/while
int main()	Otherseign de s
{	0 iteracion de z 0 iteracion de x
	Estoy en z: 0 - x: 0 - y: 0
int $x = 0$ ; /* inicializa la $x *$ /	Estoy en z: 0 - x: 0 - y: 1
int y; /* define la y */	Estoy en z: 0 - x: 0 - y: 2
int z; /* define la z */	Estoy en z: 0 - x: 0 - y: 3
	Estoy en z: 0 - x: 0 - y: 4
printf("\n");	Estoy en z: 0 - x: 0 - y: 5
	Estoy en z: 0 - x: 0 - y: 6 Estoy en z: 0 - x: 0 - y: 7
for (z = 0; z <3; z++) { /*ciclo exterior repite 3 veces*/	Estoy en z: 0 - x: 0 - y: 8
	Estoy en z: 0 - x: 0 - y: 9
printf("\n%d iteracion de z", z); /* mensaje */	
do{	1 iteracion de x
printf("\n%d iteracion de x", x); /* mensaje */	Estoy en z: 0 - x: 1 - y: 0
	Estoy en z: 0 - x: 1 - y: 1
y = 0;	Estoy en z: 0 - x: 1 - y: 2 Estoy en z: 0 - x: 1 - y: 3
while ( y < 10 ) $\{/* \text{ ciclo interior repite 10 veces */}$	Estoy en z: 0 - x: 1 - y: 5 Estoy en z: 0 - x: 1 - y: 4
printf( "\nEstoy en z: %d - x: %d - y: %d",z,x,y); /* mensaje	Estoy en z: 0 - x: 1 - y: 5
	Estoy en z: 0 - x: 1 - y: 6
*/	Estoy en z: 0 - x: 1 - y: 7
y++; /* incrementa la y */	Estoy en z: 0 - x: 1 - y: 8
} /* fin del while interno */	Estoy en z: 0 - x: 1 - y: 9
	2 iteracion de x
x++; /* incrementa la x */	Estoy en z: 0 - x: 2 - y: 0
	Estoy en z: 0 - x: 2 - y: 1
printf("\n"); /* salto de línea */	Estoy en z: 0 - x: 2 - y: 2
}while (x<3); /*ciclo exterior repite 3 veces y determina el fin	Estoy en z: 0 - x: 2 - y: 3
•	Estoy en z: 0 - x: 2 - y: 4

```
del do-while externo */
                                                                          Estoy en z: 0 - x: 2 - y: 5
                                                                          Estoy en z: 0 - x: 2 - y: 6
printf("\n"); /* salto de línea */
                                                                          Estoy en z: 0 - x: 2 - y: 7
}
                                                                          Estoy en z: 0 - x: 2 - y: 8
                                                                          Estoy en z: 0 - x: 2 - y: 9
return 0; /* indica que el programa terminó con exito */
                                                                          1 iteracion de z
                                                                          3 iteracion de x
} /* fin de la función main */
                                                                          Estoy en z: 1 - x: 3 - y: 0
                                                                          Estoy en z: 1 - x: 3 - y: 1
                                                                          Estoy en z: 1 - x: 3 - y: 2
                                                                          Estoy en z: 1 - x: 3 - y: 3
                                                                          Estoy en z: 1 - x: 3 - y: 4
                                                                          Estoy en z: 1 - x: 3 - y: 5
                                                                          Estoy en z: 1 - x: 3 - y: 6
                                                                          Estoy en z: 1 - x: 3 - y: 7
                                                                          Estoy en z: 1 - x: 3 - y: 8
                                                                          Estoy en z: 1 - x: 3 - y: 9
```

```
for (z = 0; z < 3; z++) \{ /*ciclo exterior repite 3 veces*/
  printf("\n%d iteracion de z", z); /* mensaje */
           do{
            printf("\n%d iteracion de x", x); /* mensaje */
            y = 0;
                     while (y < 10) {/* ciclo interior repite 10
                  veces */
                      printf( "\nEstoy en x: %d - y: %d",x,y); /*
                  mensaje */
                      y++;
                                           /* incrementa la y */
                      } /* fin del while interno */
           x++; /* incrementa la x */
           printf("\n"); /* salto de línea */
           }while (x<3); /*ciclo exterior repite 3 veces y determina el</pre>
           fin del do-while externo */
printf("\n"); /* salto de línea */
```

Como se observa en los ejemplos anteriores se pueden realizar distintas combinaciones según lo demande el algoritmo.

De acuerdo a lo mencionado anteriormente, es muy importante tener en cuenta la inicialización de las variables de control, como así también dónde y cuándo se inicializan contadores y acumuladores.

# Ciclos anidados con variables dependiente

Los ciclos anidados con variables dependientes son los ciclos en los que la variable de uno de los ciclos interviene en la condición de continuación o en la expresión de paso de los otros ciclos. Un ejemplo de ciclos anidados con variables independientes sería el siguiente:

Programa	Resultado
#include <stdio.h></stdio.h>	Variable dependiente en el ciclo
int main()	0 iteracion de x
{	
int x = 0; /* inicializa la x */	1 iteracion de x
int y; /* define la y */	Estoy en x: 1 y: 0
printf("Variable dependiente en el ciclo	2 iteracion de x
\n");	Estoy en x: 2 y: 0 Estoy en x: 2 y: 1
for $(\underline{x} = 0; x < 5; x++)$ { /*ciclo exterior repite 5 veces*/	3 iteracion de x
printf("\n%d iteracion de x", x); /* mensaje */	Estoy en x: 3 y: 0
for (y = 0; y < $\underline{x}$ ; y++) { /*se repetirá en función del valor de x*/	Estoy en x: 3 y: 1 Estoy en x: 3 y: 2
printf("\nEstoy en x: %d y: %d", x, y); /* mensaje */	2567 8.1. 3.7. 2
} /* fin del for interno */	4 iteracion de x Estoy en x: 4 y: 0
printf("\n"); /* salto de línea */	Estoy en x: 4 y: 1
}	Estoy en x: 4 y: 2
return 0; /* indica que el programa terminó con exito */	Estoy en x: 4 y: 3
} /* fin de la función main */	