Ciclos anidados

Un ciclo anidado es un ciclo que se encuentra incluido en el bloque de instrucciones de otro bloque. ***Los ciclos pueden tener cualquier nivel de anidamiento*** (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

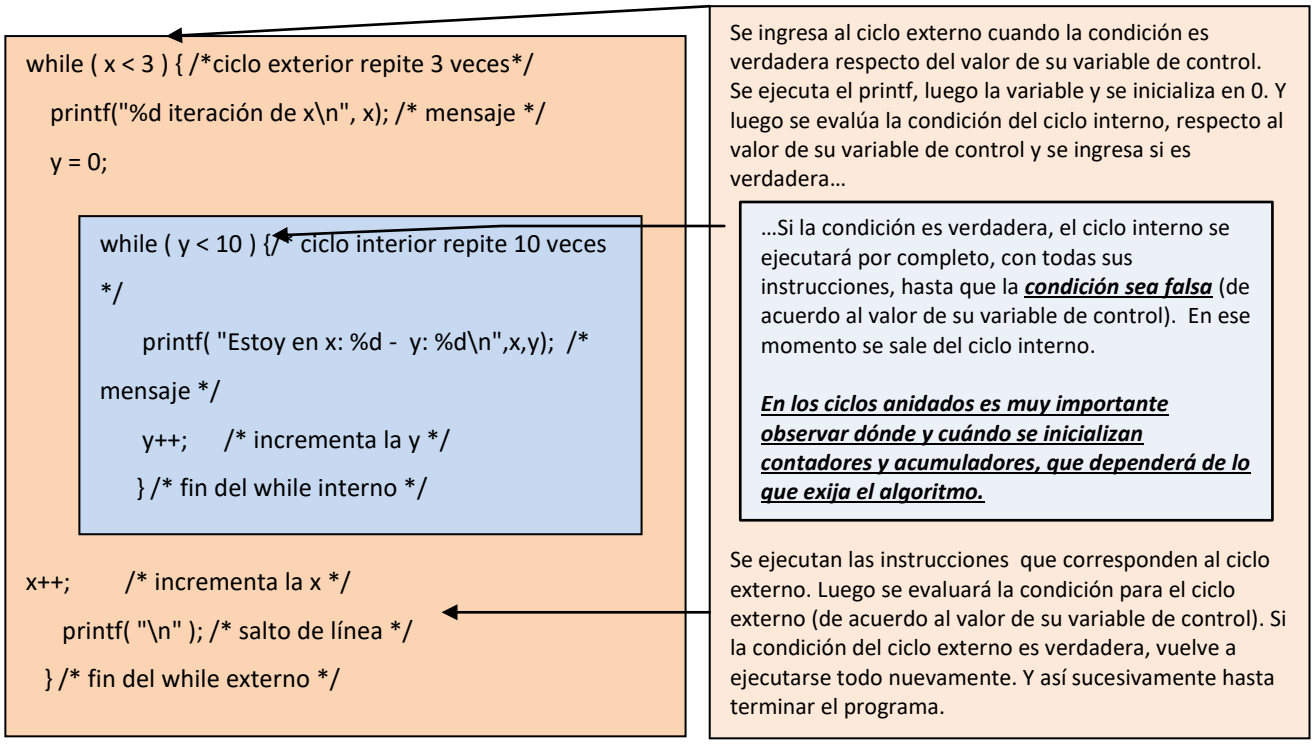
Los ciclos anidados con variables independientes son los ciclos en los que ninguna de las variables de cada uno de los ciclos interviene ni en la condición de continuación ni en la expresión de paso de los otros ciclos.

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

Análisis



Programa (for)	**Resultado**

#include<stdio.h>
int main()
{
int x; /* define x */
int y; /* define y */
 	
printf("----------Ciclo for----------\n");
 	
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 */
return 0; /* indica que el programa terminó con exito */
} /* fin de la función main */
	----------Ciclo for----------
	
	0 iteracion de x
	Estoy en x: 0 y: 0
	Estoy en x: 0 y: 1
	Estoy en x: 0 y: 2
	Estoy en x: 0 y: 3
	Estoy en x: 0 y: 4
	Estoy en x: 0 y: 5
	Estoy en x: 0 y: 6
	Estoy en x: 0 y: 7
	Estoy en x: 0 y: 8
	Estoy en x: 0 y: 9
	
	1 iteracion de x
	Estoy en x: 1 y: 0
	Estoy en x: 1 y: 1
	Estoy en x: 1 y: 2
	Estoy en x: 1 y: 3
	Estoy en x: 1 y: 4
	Estoy en x: 1 y: 5
	Estoy en x: 1 y: 6
	Estoy en x: 1 y: 7
	Estoy en x: 1 y: 8
	Estoy en 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.

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 ***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 (do-while)	**Resultado**

#include<stdio.h>
int main()
{
int x = 0; /* inicializa la x */
int y = 0; /* define la y */
 	
printf("----------Ciclo do-while----------");
 	
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 */
 	
return 0; /* indica que el programa terminó con exito */
} /* fin de la función main */
	----------Ciclo do-while----------
	0 iteracion de x
	Estoy en x: 0 y: 0
	Estoy en x: 0 y: 1
	Estoy en x: 0 y: 2
	Estoy en x: 0 y: 3
	Estoy en x: 0 y: 4
	Estoy en x: 0 y: 5
	Estoy en x: 0 y: 6
	Estoy en x: 0 y: 7
	Estoy en x: 0 y: 8
	Estoy en x: 0 y: 9
	
	1 iteracion de x
	Estoy en x: 1 y: 0
	Estoy en x: 1 y: 1
	Estoy en x: 1 y: 2
	Estoy en x: 1 y: 3
	Estoy en x: 1 y: 4
	Estoy en x: 1 y: 5
	Estoy en x: 1 y: 6
	Estoy en x: 1 y: 7
	Estoy en x: 1 y: 8
	Estoy en 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

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 ***condición evaluada al final del bloqu 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. 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>
int main()
{
int x = 0; /* inicializa la x */
int y = 0; /* define la y */
printf("----------Ciclos do-while/while----------");
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 fin del do-while externo */
 	
return 0; /* indica que el programa terminó con exito */
} /* fin de la función main */
	----------Ciclo do-while/while----------
	0 iteracion de x
	Estoy en x: 0 - y: 0
	Estoy en x: 0 - y: 1
	Estoy en x: 0 - y: 2
	Estoy en x: 0 - y: 3
	Estoy en x: 0 - y: 4
	Estoy en x: 0 - y: 5
	Estoy en x: 0 - y: 6
	Estoy en x: 0 - y: 7
	Estoy en x: 0 - y: 8
	Estoy en x: 0 - y: 9
	
	1 iteracion de x
	Estoy en x: 1 - y: 0
	Estoy en x: 1 - y: 1
	Estoy en x: 1 - y: 2
	Estoy en x: 1 - y: 3
	Estoy en x: 1 - y: 4
	Estoy en x: 1 - y: 5
	Estoy en x: 1 - y: 6
	Estoy en x: 1 - y: 7
	Estoy en x: 1 - y: 8
	Estoy en 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

```
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
fin del do-while externo */
```

Programa	**Resultado**

#include<stdio.h>
int main()
{
int x = 0; /* inicializa la x */
int y; /* define la y */
int z; /* define la z */
printf("----------Ciclo for/do-while/while----------\n");
 	
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 z: %d - x: %d - y: %d",z,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 fin
	----------Ciclo for/do-while/while----------
	
	0 iteracion de z
	0 iteracion de x
	Estoy en z: 0 - x: 0 - y: 0
	Estoy en z: 0 - x: 0 - y: 1
	Estoy en z: 0 - x: 0 - y: 2
	Estoy en z: 0 - x: 0 - y: 3
	Estoy en z: 0 - x: 0 - y: 4
	Estoy en z: 0 - x: 0 - y: 5
	Estoy en z: 0 - x: 0 - y: 6
	Estoy en z: 0 - x: 0 - y: 7
	Estoy en z: 0 - x: 0 - y: 8
	Estoy en z: 0 - x: 0 - y: 9
	
	1 iteracion de x
	Estoy en z: 0 - x: 1 - y: 0
	Estoy en z: 0 - x: 1 - y: 1
	Estoy en z: 0 - x: 1 - y: 2
	Estoy en z: 0 - x: 1 - y: 3
	Estoy en z: 0 - x: 1 - y: 4
	Estoy en z: 0 - x: 1 - y: 5
	Estoy en z: 0 - x: 1 - y: 6
	Estoy en z: 0 - x: 1 - y: 7
	Estoy en z: 0 - x: 1 - y: 8
	Estoy en z: 0 - x: 1 - y: 9
	
	2 iteracion de x
	Estoy en z: 0 - x: 2 - y: 0
	Estoy en z: 0 - x: 2 - y: 1
	Estoy en z: 0 - x: 2 - y: 2
	Estoy en z: 0 - x: 2 - y: 3
	Estoy en z: 0 - x: 2 - y: 4

del do-while externo */ printf("\n"); /* salto de línea */ } return 0; /* indica que el programa terminó con exito */ } /* fin de la función main */	Estoy en z: 0 - x: 2 - y: 5 Estoy en z: 0 - x: 2 - y: 6 Estoy en z: 0 - x: 2 - y: 7 Estoy en z: 0 - x: 2 - y: 8 Estoy en z: 0 - x: 2 - y: 9 1 iteracion de z 3 iteracion de x 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
        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>
int main()
{
int x = 0; /* inicializa la x */
int y; /* define la y */
printf("----------Variable dependiente en el ciclo----------
\n");
for (x = 0; x < 5; x++) { /*ciclo exterior repite 5 veces*/
  printf("\n%d iteracion de x", x); /* mensaje */
  for (y = 0; y < x; y++) { /*se repetirá en función del valor de x*/
  printf("\nEstoy en x: %d y: %d", x, y); /* mensaje */
  } /* fin del for interno */
  printf("\n"); /* salto de línea */
}
return 0; /* indica que el programa terminó con exito */
} /* fin de la función main */
	----------Variable dependiente en el ciclo----------
	
	0 iteracion de x
	
	1 iteracion de x
	Estoy en x: 1 y: 0
	
	2 iteracion de x
	Estoy en x: 2 y: 0
	Estoy en x: 2 y: 1
	
	3 iteracion de x
	Estoy en x: 3 y: 0
	Estoy en x: 3 y: 1
	Estoy en x: 3 y: 2
	
	4 iteracion de x
	Estoy en x: 4 y: 0
	Estoy en x: 4 y: 1
	Estoy en x: 4 y: 2
	Estoy en x: 4 y: 3