



Dirección  
Académica

# Programación estructurada

Ciclo Mayo 2022

---

## CONTENIDO

# Orden del día

---

- Repaso clase anterior
- Ejemplo de repaso
- Estructuras de repetición en C#
- Ejemplo en C#
- Ejercicio de repaso

---

## **REPASO CLASE ANTERIOR**

# Repaso clase anterior

---

La pregunta de la semana es:

¿Cómo las sentencias de decisión minimizan el uso de instrucciones secuenciales?

## Repaso clase anterior

### Operadores Lógicos

Los operadores lógicos producen un resultado booleano (verdadero o falso) y sus operandos son también valores lógicos. Nos permiten formular condiciones complejas a partir de condiciones simples.

A continuación se muestra una tabla con las tres compuertas lógicas básicas que nos servirán como operadores lógicos:

OPERADOR	C#	SINTAXIS	COMENTARIO
AND	&&	<u>Exp_Lógica &amp;&amp; Exp_Lógica</u>	Devuelve verdaderos si se cumplen ambas condiciones.
OR		<u>Exp_Lógica    Exp_Lógica</u>	Devuelve verdaderos si se cumple al menos una de las condiciones.
NOT	!	<u>!</u>	Niega la condición.

Juegas futbol && corres  
Juega futbol || corres  
! juegas futbol Verdadero  
No corres  
Falso

Si Acción  
Verdadero  
No Acción  
Falso

Numero 1 mayor

numero 2


C# 7

Exp Lógica

Si ! (numero 1 > numero 2) → Verdadero  
→ Falso

! numero 3 > numero 2

## Repaso clase anterior

Si (Exp Logical 1 && Exp Logical 2)  Dirección Académica

C#

Utilizar las instrucciones de selección **"if"** e **"if..else"** para elegir una de varias acciones alternativas.

Les gusta el futbol? Si me gusta  
NO me gusta

If (gusta el futbol) Si  
else (gusta el basquetbol) Si

if else (gusta voleyball) Si  
No

if (Exp Logical 1)

La Verda Accions

La Falso

Si - No

C# Lo if-else

# Repaso clase anterior

---

Una instrucción if simple responde a la siguiente sintaxis:

```
if (expresión lógica) {
```

Instrucción(es) de condición verdadera.

```
}if else condición es falsa
```



# Repaso clase anterior

---

La **instrucción switch** realiza una de varias acciones distintas, dependiendo del valor de una expresión (expresión de control).

¿Que deporte te gusta dame una opcion? Opcion=1

Switch (opcion):

Case 1 : Futbol

Break;

Case 2 :Basquetbol

Break;

Case 3: Volleybol

Break;

Default: Ninguna

Break;

# Repaso clase anterior

---

## Instrucción Switch

La sintaxis es la siguiente:

```
Switch (expresión de control ){  
case<literal-1>:Instrucción(es)  
break;..  
case<literal-n>:Instrucción(es)  
break; default:Instrucción(es)  
}
```

# Ejemplo de repaso

Diseñar un programa en C# que nos permita saber cuál es el número mayor entre 2 números.

```
class Program
{
    static void Main(string[] args)
    {
        Console.Title="El amyor de dos numeros";
        int x;
        int y;
        Console.WriteLine("Digita el primer numero a comparar");
        Console.WriteLine("Entre el 1 y el 100");
        x = int.Parse(Console.ReadLine());
        Console.WriteLine("Digita el segundo numero a comparar");
        Console.WriteLine("Entre el 1 y el 100");
        y = int.Parse(Console.ReadLine());

        if (x > y) {
            Console.WriteLine("\nEl numero {0} es mayor que {1}", x, y);
        } else {
            Console.WriteLine("\nEl numero {0} es mayor que {1}", y, x);
        }
        Console.WriteLine("\n\n");
        Console.WriteLine("\n----->Fin del programa");
        Console.ReadKey();
    }
}
```

La pregunta sobre la que reflexionarás en esta unidad es:

**¿Cómo sentencias de repetición ayudan a mejorar el código y la lógica del programa?**

En esta unidad aprenderás la función de las estructuras de múltiple, y su aplicación en problemas reales que involucren la elección de un camino entre más de una opción. De igual forma, comprenderás el uso de las tres diferentes instrucciones de repetición que utilizaremos en el curso y la forma en que ayudan a optimizar la escritura de código en el desarrollo de programas.

- Utilizar las instrucciones de selección **“while”** y **“repeat”** para repetir una cantidad finita de instrucciones de un programa.
- Conocer la sintaxis de C# para las estructuras repetitivas, así como también la utilidad en la programación.
- Aprender a utilizar la estructura y sintaxis para la evaluación de condiciones repetitivas.

# Resultados de aprendizaje

---

- Distinguir la sintaxis de las diferentes sentencias de control de la programación estructurada.
- Elaborar programas basados en sentencias de control de condición múltiple.
- Diseñar soluciones que utilicen instrucciones repetitivas.

- Diseñar programas que requieran la aplicación de sentencias condicionales y repetitivas para resolver situaciones que no se solucionan con controles secuenciales.

# DESARROLLO



# Introducción

---

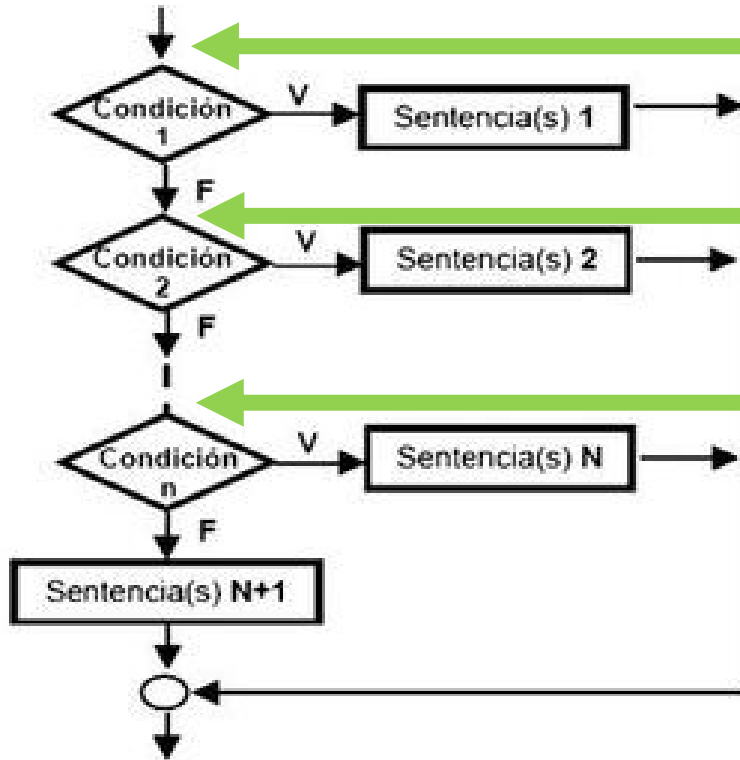
La estructura lógica de un programa y en C#, por lo general, son instrucciones que se ejecutan una después de otra, en el orden en que se escriben.

A este proceso se le conoce como ejecución secuencial.

Varias instrucciones de C# le permiten especificar que la siguiente instrucción a ejecutar no es necesariamente la siguiente en la secuencia.

Y para poder ejecutar repetir una cantidad finita de instrucciones se utiliza las estructuras de repetición.

# Estructuras de repeticion



<expr1> es evaluada una vez antes de entrar al ciclo.

Es utilizada para inicializar los datos del ciclo.

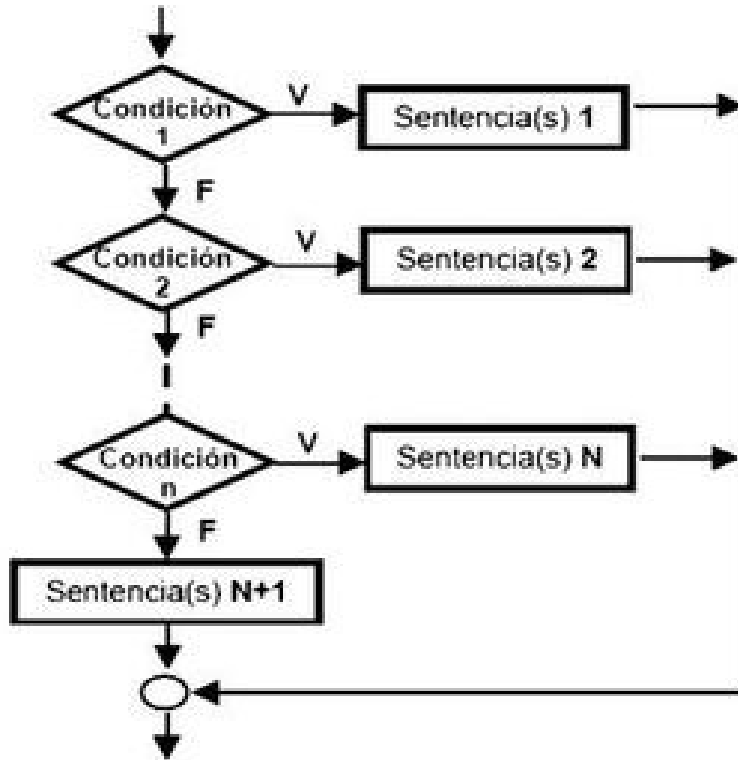
<expr2> es evaluada antes de cada ciclo.

Es utilizada para decidir si el ciclo continúa o termina.

<expr3> es evaluada al final de cada ciclo.

Es utilizada para asignar el nuevo valor a los datos del ciclo.

# Estructuras de repeticion



Sintaxis:

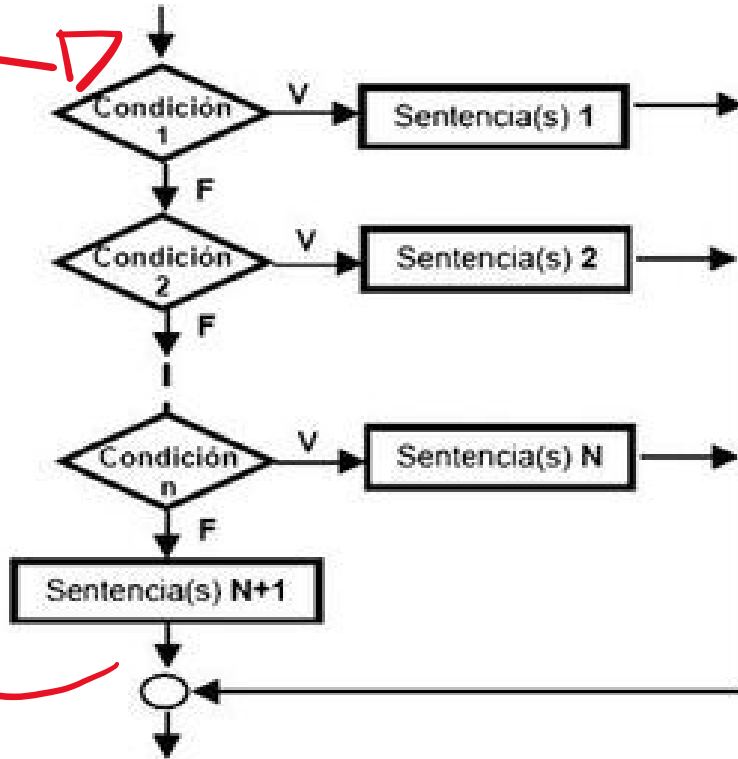
For variable=Valor\_Inicial To Valor\_Final Step

[INCR O DECR]

CONJUNTO DE INSTRUCCION(ES)

Next Variable

# Estructuras de repeticion



Sintaxis:

for (int i=0; i<=10; i++)

Incrementador

Inicialización de Variable

Limite de Variable

## Estructuras de repetición

El ciclo for está conformado por tres partes:

Inicio del ciclo, desde donde comienza.

Condición, límite del ciclo, el cual se repite mientras la condición sea verdadera.

Incremento o decremento, el cual puede ser de 1 en 1 o de x en x, de forma positiva o negativa.



# Ejemplos

Programa ejemplo, que imprime los números comprendidos del 1 al 10:

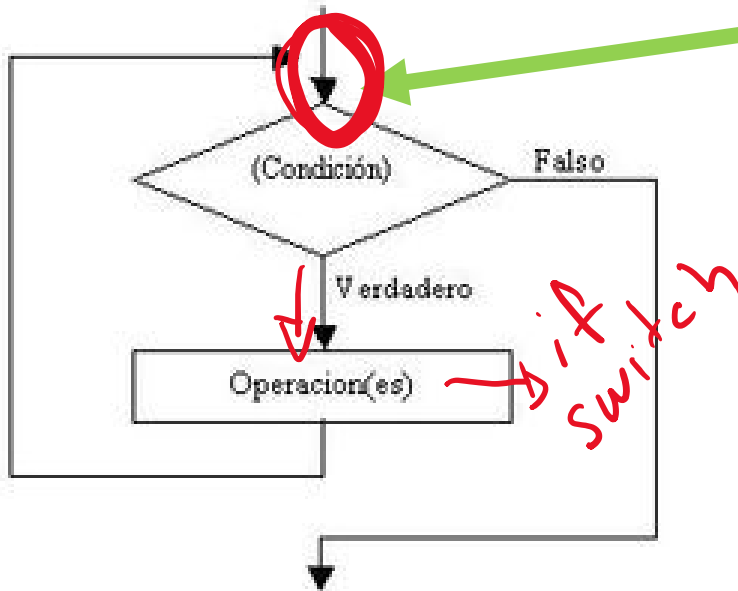
```
Program.cs x
For1 For1.Program
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace For1
8 {
9     class Program
10    {
11        static void Main(string[] args)
12        {
13            Console.WriteLine("Imprime números del 1 al 10");
14            for(int i = 0; i <= 10; i++)
15            {
16                Console.WriteLine("i= " + i);
17            }
18            Console.ReadKey();
19        }
20    }
21 }
```

# Ejemplos

Programa que imprime la numeración del 0 al 25, de 2 en 2:

```
Program.cs x
For1 For1.Program Main(string[] args)
1 using System;
2   using System.Collections.Generic;
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace For1
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              Console.WriteLine("Imprime números del 1 al 25, de 2 en 2");
14              for(int i = 0; i <= 25; i+=2)
15              {
16                  Console.WriteLine("i= " + i);
17              }
18              Console.ReadKey();
19          }
20      }
21  }
```

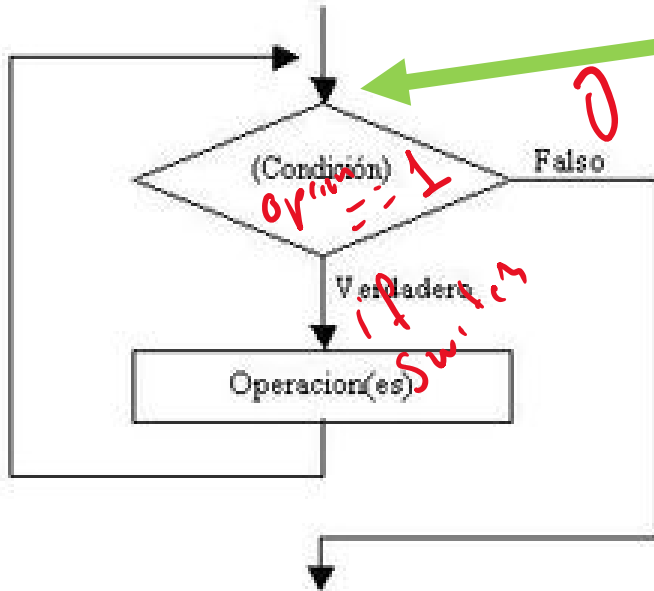
# Estructuras de repeticion



<u>while</u> evalúa la condición antes de ejecutar el bloque de instrucciones.



# Estructuras de repeticion



Sintaxis:

while (condición)  
CONJUNTO DE INSTRUCCION(ES)

# Ejemplos

Un ejemplo que nos diga si cada número que tecleemos es positivo o negativo, y que termine cuando tecleemos el número 0

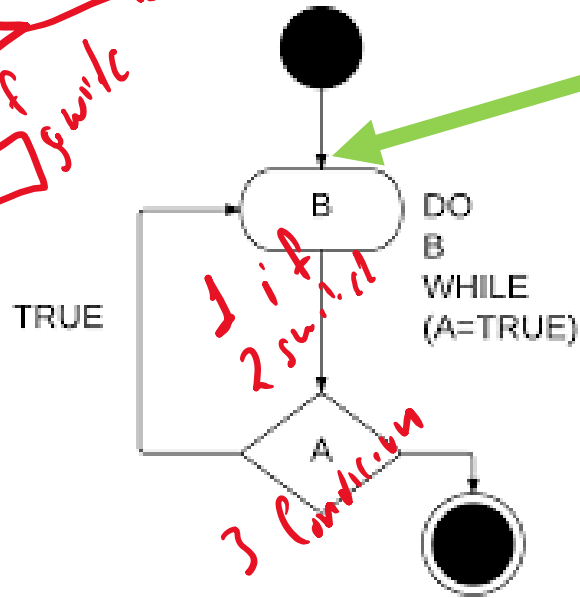
```
using System;
public class Ejemplo {
    public static void Main() {

        int numero;
        Console.Write("Teclea un número (0 para salir): ");
        numero = Convert.ToInt32(Console.ReadLine());
        while (numero != 0) {
            if (numero > 0)
                Console.WriteLine("Es positivo");
            else
                Console.WriteLine("Es negativo");

            Console.WriteLine("Teclea otro número (0 para salir): ");
            numero = Convert.ToInt32(Console.ReadLine());
        }
    }
}
```

# Estructuras de repetición

1 if  
2 while  
False



do

{

//Codigo a ejecutar

}while(i == 10)

# Ejemplos

Programa ejemplo, que imprime los primeros números comprendidos del 1 al 10:

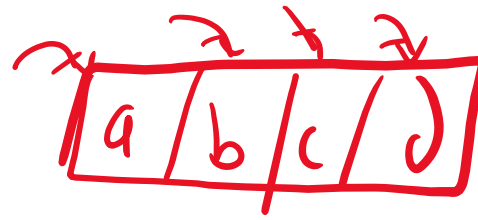
```
/// <summary>  
/// //Do While loop checks the condition at the end of the loop.  
/// Thats why Do while loop guarantees to execute at least once.  
/// </summary>
```

1reference

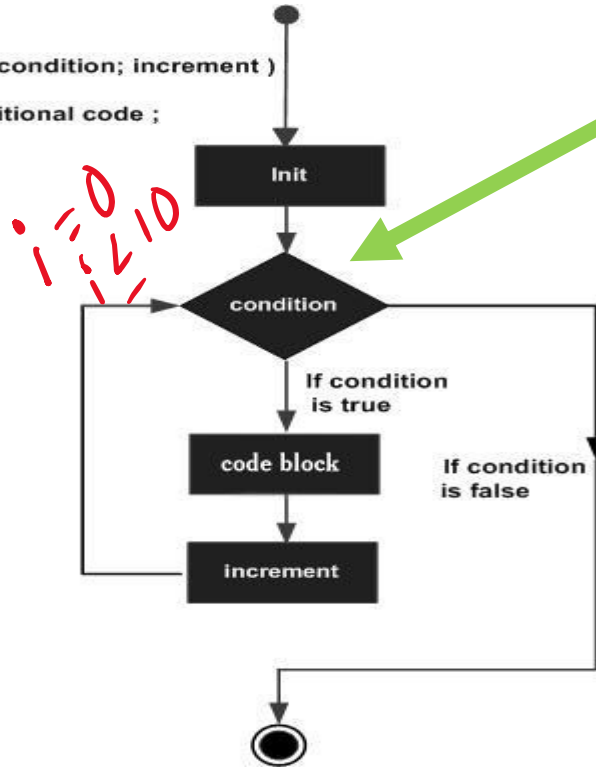
```
public static void Dowhileloop()  
{  
    int[] array = new int[3];  
    array[0] = 4;  
    array[1] = 5;  
    array[2] = 6;  
    //Print above numbers from array using Do While loop.  
    int whileInt = 0;  
    do  
    {  
        Console.WriteLine(array[whileInt]);  
        whileInt++;  
    } while (whileInt < array.Length);  
}
```



# Estructuras de repetición



```
for( init; condition; increment )  
{  
    conditional code ;  
}
```



El foreach es una herramienta utilizada mayoritariamente para recuperar la información de colecciones, arrays o listas, es decir objetos que pueden contener más de un valor almacenado, veamos su sintaxis:

```
foreach(tipoDato nombre in array)  
{  
    ... instrucciones ...  
}
```

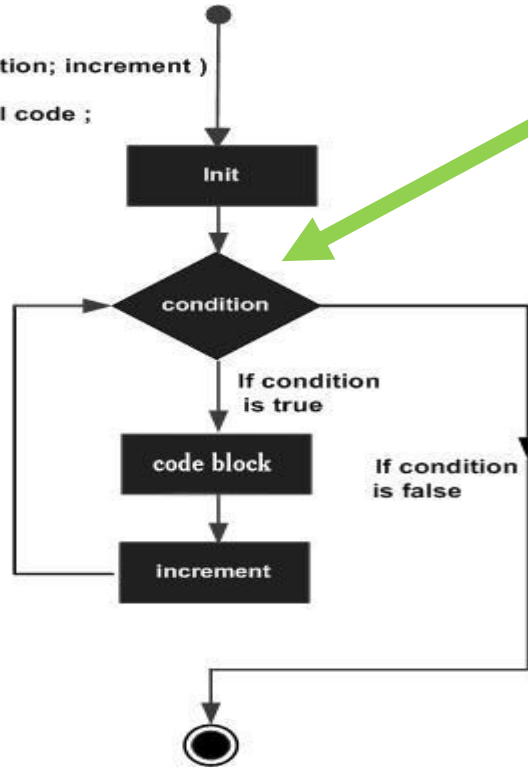
# Ejemplos

Programa ejemplo, que imprime los primeros números comprendidos del 1 al 10:

```
3
4
5
6 using System;
7 using System.Linq;
8 using System.Text;
9 using System.Collections.Generic;
10 using System.Threading.Tasks;
11
12 namespace VarianteCicloForeachRecorridoArrays
13 {
14     public class Program
15     {
16         public static void Main(string[] args)
17         {
18             Random rnd = new Random();
19             int [] valores = new int[10];
20
21             for( int i = 0; i < 10; i++ )
22                 valores[i] = rnd.Next(1, 100);
23
24             foreach (int recorrido in valores)
25                 Console.Write( $"{recorrido}, " );
26
27             Console.ReadKey();
28         }
29     }
30 }
31
```

# Estructuras de repetición

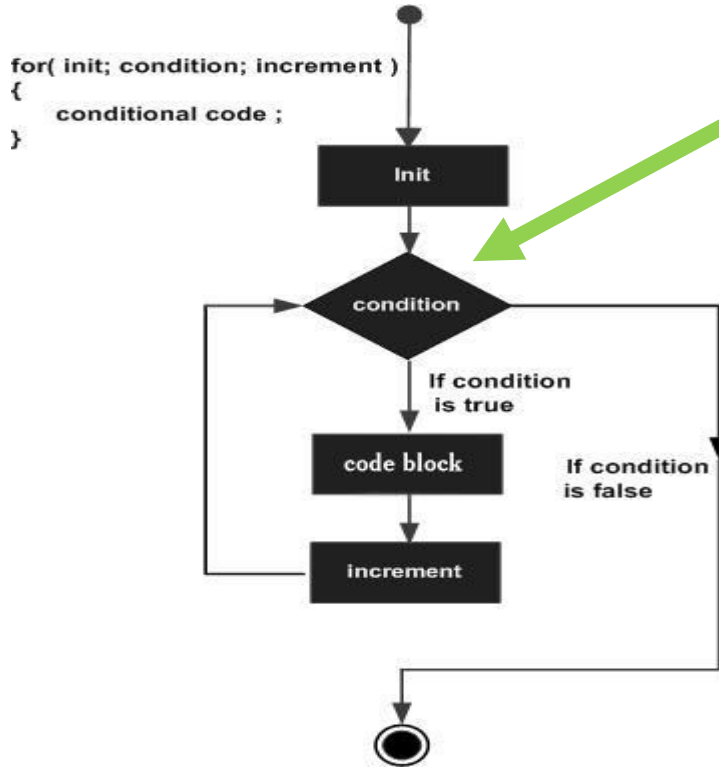
```
for( init; condition; increment )
{
    conditional code ;
}
```



break;

Mediante esta palabra clave, podemos salir de ese bloque de código en cualquier momento. Esta palabra clave provoca que vayamos directamente a la siguiente línea que continúa al bloque donde se encuentre incluida la palabra. Pongamos un ejemplo:

# Estructuras de repetición



```

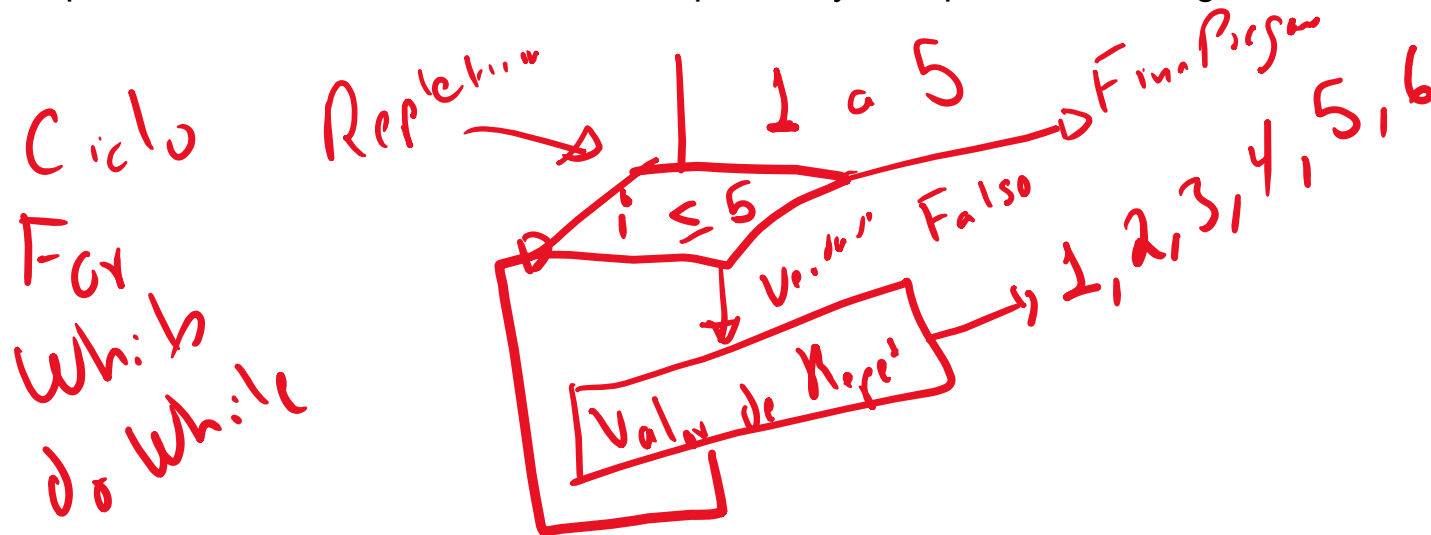
Console.WriteLine(«Buenos dias»);
for(i = 0 ; i < 10 ; i ++ )
{
    Console.WriteLine(i);
    if(i == 5)
    {
        break;
    }
}
Console.ReadKey();
    
```



## **ACTIVIDADES DE REFORZAMIENTO**

# Actividades de reforzamiento

Ejercicio 1. Por ejemplo, si queremos contar del 1 al 5, usaríamos una variable que empezase en 1, que aumentaría una unidad en cada repetición y se repetiría hasta llegar al valor 5.



# Actividades de reforzamiento

---

Ejercicio 1. Por ejemplo, si queremos contar del 1 al 5, usaríamos una variable que empezase en 1, que aumentaría una unidad en cada repetición y se repetiría hasta llegar al valor 5.

# Actividades de reforzamiento

---

```
using System;
```

```
public class Ejemplo {  
    public static void Main() {  
        int n = 1;  
        while (n < 6) {  
            Console.WriteLine(n);  
            n = n + 1;  
        }  
    }  
}
```

## Estructuras Repetitivas

### Introducción

Un ciclo es una estructura que nos permite representar un conjunto de instrucciones que debe repetirse una cantidad limitada de veces, normalmente dependiente de una condición o de una cantidad determinada de repeticiones o iteraciones. Los ciclos permiten iterar todo un proceso tantas veces como el programador (ó el usuario) lo determine.

Es común, que en la solución de muchos problemas algorítmicos, se requiera realizar la repetición de cierto bloque de instrucciones, con el fin de obtener el objetivo buscado por el algoritmo. Para implementar repetición de bloques de instrucciones se utilizan las estructuras de control llamadas ciclos o estructuras repetitivas.

### CONCEPTO GENERAL

Un ciclo puede definirse como una estructura que nos permite repetir o iterar un conjunto de instrucciones y que tiene las siguientes características:



### Tipos de Ciclos

#### Ciclo while (Mientras)

Este ciclo se ejecuta mientras se cumple la condición que se especifica en el cuerpo del ciclo. Si la condición es falsa, se sale del ciclo.

#### Ciclo do while

Este ciclo se ejecuta una o más veces, ya que la condición se evalúa después de haber ejecutado el cuerpo del ciclo.

#### Ciclo for

Este ciclo se ejecuta un número determinado de veces, ya que se especifica el número de iteraciones en el cuerpo del ciclo.

#### Ciclo for each

Este ciclo se ejecuta para cada elemento de un conjunto de datos, ya que se especifica el conjunto de datos en el cuerpo del ciclo.

#### Ejemplo

```
1. Inicializar una variable a 1.  
2. Mientras a sea menor o igual a 10:  
3.   Imprimir a  
4.   Incrementar a en 1
```

Este ejemplo muestra un ciclo while que imprime los números del 1 al 10. La condición del ciclo es 'a sea menor o igual a 10'. Cada vez que se ejecuta el cuerpo del ciclo, se imprime el valor de 'a' y se incrementa en 1.

---

## **PREGUNTA DE INVESTIGACION**

# Pregunta de investigación

---

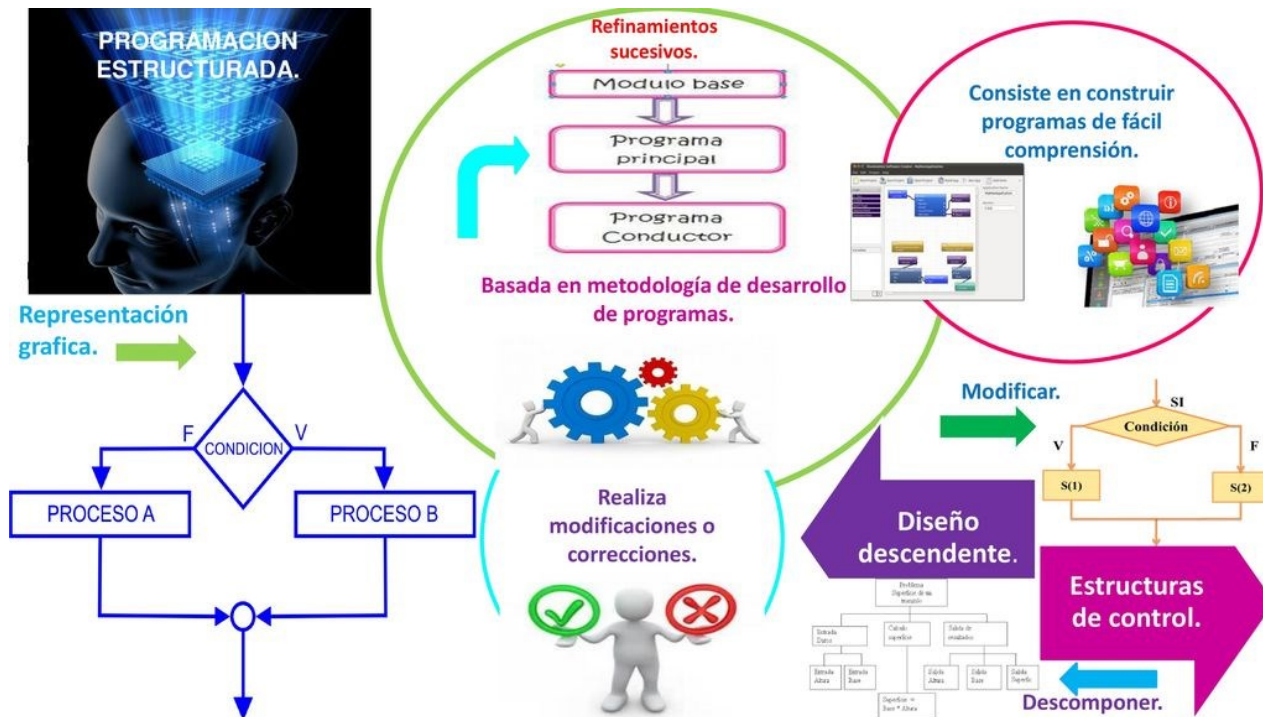
¿De qué forma los arreglos permiten el almacenamiento masivo de datos?

---

**CIERRE**

---





# BIBLIOGRAFÍA COMPLEMENTARIA

<http://www.msdn.microsoft.com/net/ecma>.

“A programmer’s introduction to C#” escrito por Eric Gunnerson y publicado por Apress en 2000.

C# and the .NET Framework”, escrito por Andrew Troelsen y publicado por Apress en 2001

“C# Essentials”, escrito por Beb Albahari, Peter Drayton y Brand Merril y publicado por O’Reilly en 2000.

“C# Programming with the Public Beta”, escrito por Burton Harvey, Simon Robinson, Julian Templeman y Karli Watson y publicado por Wrox Press en 2000.

“Inside C#”, escrito por Tom Archer y publicado por Microsoft en 2000 • “Presenting C#”, escrito por Christoph Wille y publicado por Sams Publishing en 2000.

[ÍNDICE](#)



---

**DUDAS**