Unidad 6

Descargar estos apunte en pdf o html

Índice

- Índice
- Bucles
  - Definiciones
  - Diseño de un bucle
  - Bucle while (Mientras)
  - Algunos esquemas algorítmicos de utilidad con bucles
    - Contador
    - Ejemplos Estructura Contador
    - Acumulador
    - Ejemplos Estructura Acumulador
  - Bucle do - while (Hacer - Mientras)
    - Ejemplos do-while
  - Esquema algorítmico para obtener máximos y mínimos
  - Bucle for (Para)
    - Ejemplos for
  - Concepto de '*flag*'
    - Ejemplo de uso de '*flags*'
  - Cláusula `break` en bucles
  - Bucles Anidados
- Anexo I: Cláusula `continue` en bucles

Bucles

> **Enlaces**
>
> - Bucles
> - Cláusula while
> - Cláusula for

Definiciones

- Un bucle es la repetición de una secuencia de instrucciones en función de una condición.
- El **cuerpo de un bucle** son las instrucciones incluidas dentro del mismo.
- Una **iteración** es una ejecución del cuerpo del bucle

Diseño de un bucle

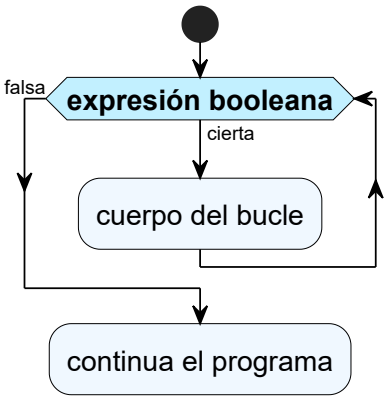
- Deberemos establecer qué ***instrucciones contiene el bucle*** y cuantas **veces se deben repetir**.
- Deberemos llevar especial cuidado en evitar situaciones de **bucle infinito**, debidas a que no se cumpla la condición de salida.

Bucle while (Mientras)

Esquema básico...

```
while (expresión booleana)
    instrucción;

while (expresión booleana)
{
    bloque;
}
```



Se **evalúa la expresión**. Si esta se evalúa a **true** (verdadero), se ejecuta la sentencia o el bloque de código y el control vuelve al principio del ciclo.

Cuando la expresión se evalúa a **false** (falso), termina la ejecución del bucle y se pasa a la sentencia siguiente.

Ejemplo:

Programa que pregunta al usuario... "`Desea continuar [S|N]:`"
Mientras el usuario no introduzca uno de los 2 caracteres válidos volverá a hacer la pregunta.

```
public static void Main()
{
    Console.Write("Desea continuar [S|N]: ");
    char respuesta = char.Parse(Console.ReadLine() ?? "S");

    while (respuesta != 'S' && respuesta != 'N')
    {
        Console.Write("Desea continuar [S|N]:");
        respuesta = char.Parse(Console.ReadLine() ?? "N");
    }
}
```

Ejemplo de ejecución:

```
Desea continuar [S|N]: X
Desea continuar [S|N]: Y
Desea continuar [S|N]: N
```

Algunos esquemas algorítmicos de utilidad con bucles

Antes de continuar, veamos algunos esquemas básicos a través del bucle while.

Contador

- Una variable, normalmente de un tipo numérico, que incrementa o decrementa su valor de forma **FIJA**.
- El proceso se realizará normalmente en un bucle.
- Inicializaremos su valor, para que este quede definido.
- Incrementaremos su valor con una expresión del tipo…

```
idContador = idContador ± VALOR_FIJO;

nivel = nivel + 2;      // → (nivel += 2)
vidas = vidas - 1;      // → (vidas--)
tamaño = tamaño * 2;    // → (tamaño *= 2)
```

Ejemplos Estructura Contador

Ejemplo 1:

Programa que concatena los números del 1 al 15 y los muestra por pantalla.

```
public static void Main()
{
    int i = 1;                          // Inicializamos el contador.
    string texto = (i++).ToString();

    while (i <= 15)                    // Al llegar a 16 salimos.
    {
        texto += $", {i}";
        i++;                            // Lo incrementamos en un valor constante.
    }

    Console.WriteLine(texto);
}
```

Ejemplo de ejecución:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
```

Ejemplo 2:

Programa que dada la base y el exponente me calcule la potencia: multiplicando, mediante un bucle, tantas veces la base como indique el exponente.

```
public static void Main()
{
    // Entrada: base y exponente
    Console.Write("Base: ");
    int b = int.Parse(Console.ReadLine() ?? "1");
    Console.Write("Exponente: ");
    int e = int.Parse(Console.ReadLine() ?? "1");

    int i = 0; // Inicializo el contador de veces que multiplico el exponente.
    int p = 1; // Inicializo el contador donde voy incrementando la potencia.
    while (i < e)
    {
        p *= b; // potencia = base * base * base * ... * base;
        i++;
    }

    // Salida
    Console.WriteLine($"{b}^{e} = {p}");
}
```

Ejemplo de ejecución:

```
Base: 2
Exponente: 5
2^5 = 32
```

Acumulador

- Una variable, normalmente de un tipo numérico, que incrementará o disminuirá su valor de forma **variable**.
- El proceso se realizará normalmente en un bucle.
- Inicializaremos su valor, para que este quede definido (al neutro de la operación).
- Incrementaremos su valor con una expresión del tipo…

```
idAcumulador = idAcumulador ± VALOR_VARIABLE;

totalFactura = totalFactura + lineaFactura;
```

Ejemplos Estructura Acumulador

Ejemplo 1:

Programa que dado un número me diga su factorial.
Ej. Si el usuario introduce un 4, el factorial sería 24 y se calcularía multiplicando 1 * 2 * 3 * 4

Ampliación: ¿ Serías capaz de componer la cadena de salida "`4! = 4 x 3 x 2 x 1 = 24`" ?

```
public static void Main()
{
    // Entrada: Entrada: Número del que quiero calcular el factorial.
    Console.Write("Número: ");
    int numero = int.Parse(Console.ReadLine() ?? "1");

    int i = 1;           // i: Contador que lleva la factorización.
    long factorial = 1;  // factorial: Donde acumulo el factorial.

    while(i <= numero)
    {
        factorial *= i;  // 1 x 2 x … x N
        i++;
    }
    Console.WriteLine($"{numero}! = {factorial}"; // Salida
}
```

Ejemplo de ejecución:

```
Número: 4
4! = 24
```

Ejemplo 2:

Programa que va pidiendo el **precio** en euros **mientras el precio sea mayor que cero**, y al final de la ejecución me muestre el **total** de precios introducidos.

```
public static void Main()
{
    double precio;

    Console.Write("Precio: ");
    precio = double.Parse(Console.ReadLine() ?? "0");

    double total = precio;
    while (precio > 0)
    {
        Console.Write("Precio: ");
        precio = double.Parse(Console.ReadLine() ?? "0");

        if (precio > 0)
            total += precio;
    }

    Console.WriteLine($"Total: {total:F2} euros");
}
```

Ejemplo de ejecución:

```
Precio: 10.5
Precio: 5.25
Precio: 5
Precio: 0
Total: 20.75 euros
```

🚀 Ampliación opcional:

EN otros lenguajes como **Kotlin** o **Python** también disponemos de este tipo de bucles. Veamos cómo quedaría el ejemplo del factorial en estos lenguajes.

Kotlin:

```kotlin
fun main() {
    // Entrada: Número del que se quiere calcular el factorial.
    print("Número: ")
    val numero = readLine()?.toIntOrNull() ?: 1

    var i = 1            // i: Contador que lleva la factorización.
    var factorial: Long = 1 // factorial: Donde se acumula el factorial. Se usa Long para evitar desborda

    while (i <= numero) {
        factorial *= i   // 1 x 2 x … x N
        i++
    }
    println("$numero! = $factorial") // Salida
}
```

Python:

```python
# Entrada: Número del que se quiere calcular el factorial.
numero = int(input("Número: ") or 1)

i = 1            # i: Contador que lleva la factorización.
factorial = 1    # factorial: Donde se acumula el factorial.

while i <= numero:
    factorial *= i   # 1 x 2 x … x N
    i += 1
print(f"{numero}! = {factorial}")  # Salida
```

Bucle do - while (Hacer - Mientras)

El esquema básico es...

```
do
    instrucción;
while (expresión booleana);

do {
    bloque;
} while (expresión booleana);
```

Los bucles **do ... while** son muy similares a los bucles **while**, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio.
La principal diferencia frente a los bucles regulares while es que **se garantiza la ejecución de la primera iteración**.

Ejemplos do-while

Ejemplo 1:

Veamos el mismo ejemplo que hicimos con el bucle while, donde se implementaba un programa que preguntaba al usuario... "`Desea continuar [S|N]:`"
Mientras el usuario no introduzca uno de los 2 caracteres válidos volvería a hacer la pregunta.
Si nos fijamos en este caso el bucle **do...while** nos da lugar a un código más simple.

```
public static void Main()
{
    char respuesta;
    do
    {
        Console.Write("Desea continuar [S|N]:");
        respuesta = char.Parse(Console.ReadLine() ?? "N");
    } while (respuesta != 'S' && respuesta != 'N');
}
```

Ejemplo 2:

Programa que va pidiendo el **precio** en euros **mientras el precio sea mayor que cero**, y al final de la ejecución me muestre el **total** de precios introducidos.

```
public static void Main()
{
    double precio;

    double total = 0;
    do
    {
        Console.Write("Precio: ");
        precio = double.Parse(Console.ReadLine() ?? "0");

        if (precio > 0)
            total += precio;
    }
    while (precio > 0);

    Console.WriteLine($"Total: {total:F2} E");
}
```

Esquema algorítmico para obtener máximos y mínimos

Existen 2 estrategias posibles:

1. El primer valor leído en el bucle erá el máximo y el mínimo. Posteriormente tendré que usar condicionales para establecerlos. (**"La buena"**)
2. Antes de entrar en el bucle, inicializo el **máximo** a un valor muy pequeño `int.MinValue` y el **mínimo** a un valor muy grande `int.MaxValue`.
   Esta última estratégia es menos recomendable.

Ejemplo 1:
Crea un programa en C# de ejecución en terminal, que utilizando una estructura while, tome la temperatura exterior leyéndola de un sensor cada hora durante 24 horas.
Al final del día, la estación meteorológica nos mostrará la temperatura más alta y la más baja a partir de las medidas tomadas cada hora.

📌 **Nota:** Simularemos la lectura del sensor con una entrada por consola del usuario.

```
// Solución usando un bucle While y el primer esquema.
public static void Main()
{
    const uint HORAS_DIA = 24;
    int temperatura, temperaturaMayor, temperaturaMenor;

    Console.Write("Introduce la temperatura?: ");
    temperatura = int.Parse(Console.ReadLine() ?? "0");

    // La temperatura leída a primera hora será la mayor y la menor.
    temperaturaMayor = temperaturaMenor = temperatura;
    uint horaActual = 2;
```

```
    while (horaActual <= HORAS_DIA)
    {
        Console.Write("Introduce la temperatura? ");
        temperatura = int.Parse(Console.ReadLine() ?? "0");

        if (temperatura > temperaturaMayor)
            temperaturaMayor = temperatura;
        if (temperatura < temperaturaMenor)
            temperaturaMenor = temperatura;
        horaActual++;
    }
    Console.WriteLine($"La temperatura mayor es {temperaturaMayor}C y la menor es {temperaturaMenor}C");
}
// Solución usando un bucle do-While y el segundo esquema.
// Al usar un do-while solo tenemos que hacer 1 lectura.
public static void Main()
{
    const uint HORAS_DIA = 24;
    // Inicializaremos el máximo a un valor muy pequeño y hacemos
    // lo contrário para el mínimo.
    int temperaturaMayor = int.MinValue;
    int temperaturaMenor = int.MaxValue;
    uint horaActual = 1;
    do
    {
        Console.Write("Introduce la temperatura? ");
        int temperatura = int.Parse(Console.ReadLine() ?? "0");
        if (temperatura > temperaturaMayor)
            temperaturaMayor = temperatura;
        if (temperatura < temperaturaMenor)
            temperaturaMenor = temperatura;
        horaActual++;
    }
    while (horaActual <= HORAS_DIA);
    Console.WriteLine($"La temperatura mayor es {temperaturaMayor}C y la menor es {temperaturaMenor}C");
}
```

Ampliación:

¿Sabrías utilizar el primer esquema junto con un bucle do-while?

🎓 **Caso de estudio:**

Dada la ecuación cuadrática: $y = -2x^2 + 10x - 5$

Realiza un programa que calcule el **punto donde se encuentra su máximo** para el rango **x = 0** y **x = 5** con incrementos de **x** de **0.5**.

📌 **Nota:** Si es posible, haz que el programa permita introducir diferentes incrementos para ver que sucede.

💡 **Tip:** Para comprobar tu programa ten en cuenta que matemáticamente el **máximo** será el valor de x que cumpla la ecuación donde la primera derivada es igual a 0:

$$y' = -4x + 10 = 0 \rightarrow x = \frac{10}{4} = 2.5$$

```
public static void Main()
{
    Console.Write("Introduce el principio del rango en X: ");
    double xIni = double.Parse(Console.ReadLine() ?? "0");
    Console.Write("Introduce el final del rango en X: ");
    double xFin = double.Parse(Console.ReadLine() ?? "0");
    Console.Write("Introduce el incremento en X: ");
    double inc = double.Parse(Console.ReadLine() ?? "0");

    double yMax = double.MinValue;
    double yMin = double.MaxValue;
    double x, xMax;
    x=xMax = xIni;

    while ( x <= xFin)
    {
        double y = -2*x*x + 10*x - 5;
        if (y > yMax)
        {
            yMax = y;
            xMax = x;
        }
        if (y < yMin)
            yMin = y;
        x += inc;
    }
    string mensaje = "Ecuación: y = -2x^2 + 10x - 5\n";
    mensaje += $"El valor máximo es y es {yMax:F2} para x = {xMax:F2}";
    Console.WriteLine(mensaje);
}
```

Ejemplo de ejecución:

```
Introduce el principio del rango en X: 0
Introduce el final del rango en X: 5
Introduce el incremento en X: 0.5
Ecuación: y = -2x^2 + 10x - 5
El valor máximo es y es 12.50 para x = 2.50
```

Bucle for (Para)

El esquema básico es...

```
for (expresión1; expresión2; expresión3)
    sentencia;

for (expresión1; expresión2; expresión3)
{
    bloque;
}
```

Podemos decir que *'lleva un contador incorporado'* ya que equivale a una estructura del tipo:

```
expresión1;
while (expresión2)
{
    bloque;
    expresion3;
}
```

Aviso

Un error muy común entre programadores noveles es poner un **';'** al final de la sentencia `for` lo que provocaría que el cuerpo del bucle **solo se ejecutase una vez**.

```
int suma=0;
for (int i=1; i<=10; ++i); // Este punto y coma es un error.
{
    suma += i; // Este bloque no se ejecuta 10 veces, solo una.
}

Console.WriteLine(suma);
```

Ejemplos for

Ejemplo 1:

Programa que visualice la tabla de multiplicar de un número solicitado.

```
public static void Main()
{
    Console.Write("Introduzca un valor entero: ");
    int valor = int.Parse(Console.ReadLine() ?? "0");

    for (int i = 1; i <= 10; ++i)
    {
        Console.WriteLine($"{valor} x {i} = {valor * i}");
    }
}
```

Ejemplo de ejecución:

```
Introduzca un valor entero: 4
4 x 1 = 4
4 x 2 = 8
4 x 3 = 12
4 x 4 = 16
4 x 5 = 20
4 x 6 = 24
4 x 7 = 28
4 x 8 = 32
4 x 9 = 36
4 x 10 = 40
```

Ejemplo 2:

Haz un programa en C# que usando **un único bucle for**, calcule por fuerza bruta todas las combinaciones de dos dados que dan una suma leída por el teclado.

- Las entradas posibles serán de 2 a 12.
- No pueden repetirse combinaciones.

Ejemplo de ejecución:

```
Suma dados: 8
Combinaciones -> (2,6)(3,5)(4,4)(5,3)(6,2)
```

```
public static void Main()
{
    Console.Write("Suma dados: ");
    uint sumaDados = uint.Parse(Console.ReadLine() ?? "0");
    const uint TIRADA_MAXIMA = 6;
    string salida = "Combinaciones -> ";
    for (uint dado1 = 1; dado1 <= TIRADA_MAXIMA; dado1++)
    {
        uint dado2 = sumaDados - dado1;
        if (dado2 <= TIRADA_MAXIMA)
            salida += $"({dado1},{dado2})";
    }
    Console.WriteLine(salida);
}
```

Ejemplo 3:

Crea un programa que utilizando **bucles for (no anidados)** dibuje **un cuadrado** de ceros **'0'** en navegador de **N columnas por N filas**. Siendo N un número introducido por el usuario.
Entre cada cero en la misma fila habrá dos espacios, para que el cuadrado se represente bién en el navegador.

Ejemplo de ejecución:

```
Número de N: 4
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

💡 **Pista:** Puedes hacer un primer bucle `for` que concatene un texto con los ceros de una fila y a continuación un segundo bucle `for` que concatene **N veces** la cadena con los ceros en una fila seguido de un **salto de línea** en la cadena de salida.

```
public static void Main()
{
    Console.Write("Número de N: ");
    int n = int.Parse(Console.ReadLine() ?? "0");

    string textoFila = "";
    for (int i = 0; i < n; i++)
        textoFila += "0 ";

    string salida = "";
    // Podemos volver a usar i como identificador pues
    // la variable dejó de existir y fué retirada del
    // heap después del primer for.
    for (int i = 0; i < n; i++)
        salida += $"{textoFila}\n";
    Console.Write(salida);
}
```

🚀 Ampliación opcional:

En otros lenguajes como **Kotlin**, **Python** o **Java** también disponemos de este tipo de bucles. Veamos cómo quedaría el ejemplo de mostrar la tabla de multiplicar en estos lenguajes.

Kotlin:

```kotlin
fun main() {
    print("Introduzca un valor entero: ")
    val valor = readLine()?.toIntOrNull() ?: 0

    for (i in 1..10) {
        println("$valor x $i = ${valor * i}")
    }
}
```

Python:

```python
valor = int(input("Introduzca un valor entero: ") or 0)
for i in range(1, 11):
    print(f"{valor} x {i} = {valor * i}")
```

Java:

```java
import java.util.Scanner;

public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Introduzca un valor entero: ");
    int valor = scanner.nextInt();

    for (int i = 1; i <= 10; i++) {
        System.out.printf("%d x %d = %d%n", valor, i, (valor * i));
    }
    scanner.close();
}
```

Concepto de '*flag*'

Un **'flag'** es una variable que solo puede tomar **dos valores** diferentes durante el desarrollo de un programa. Se denomina flag o bandera porque estos dos estados equivaldrían a una bandera **subida** 🏳️ o **bajada** 🏴.

Sus dos posibles valores los elige el programador y pueden ser, por ejemplo: `true` o `false`; `0` o `1`; `on` u `off`; `par` o `impar`; `si` o `no`, etc...

El '*flag*' se inicializa con uno de los dos valores elegidos.

```
bool numeroEncontrado = false;
```

A lo largo del proceso, y dependiendo de que una determinada situación se haya producido, se le modifica dicho valor.

```
if (numero == numeroBuscado)
    numeroEncontrado = true;
```

Después, en otro punto del programa, al preguntar por su estado (su valor), se puede detectar si dicha situación ha ocurrido o no.

```
Console.WriteLine((numeroEncontrado ? "Se" : "No se") + " encontró el número");
```

Ejemplo de uso de '*flags*'

Ejemplo 1:

Programa que dado un número introducido, me diga **si es primo o no lo es**.

Pista: El programa dividirá el número introducido **N**, por un contador que irá de **2 a N-1** y si encuentra algún resto 0 pondrá **el flag** `esPrimo` a **false** pues el número ya será divisible.

```
public static void Main()
{
    Console.Write("Introduce un valor: ");
    int valor = int.Parse(Console.ReadLine() ?? "0");

    bool esPrimo = true;
    for (int i = 2; esPrimo && i < valor; ++i)
    {
        if (valor % i == 0)
            esPrimo = false;
    }

    string mensaje = $"El número {(esPrimo?"es":"no es")} primo.";
    Console.WriteLine(mensaje);
}
```

Ejemplo de ejecución:

```
Introduce un valor: 7
El número es primo.
```

Ejemplo 2:

Haz un programa que calcule el número **e** mediante el desarrollo en serie:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots + \frac{1}{50!} = 2,7182...$$

Pediremos al usuario el número de términos de la serie para calcular N,**mínimo 2 y máximo 50**, y filtraremos la entrada a través de un bucle `do - while` y un **flag** que me indique si es válida o no la misma.

📌 **Nota:** Utilizaremos **un único bucle** `for` para ir acumulando el factorial e ir calculando los números de la serie.

```
bool entradaValida;
int n;
do
{
    Console.Write("Número de términos de la serie [2 a 50]: ");
    n = int.Parse(Console.ReadLine() ?? "2");
    entradaValida = n >= 2 && n <= 50;
    if (!entradaValida)
        Console.WriteLine($"{n} no es un valor válido. Debe estar entre 2 y 50 inclusive.");
}
while (!entradaValida);
double sumaSerie = 1D;
double factorialAcumulado = 1D;
string salida = "1";
for (int i = 1; i <= n; i++)
{
    factorialAcumulado *= i;
    sumaSerie += 1 / factorialAcumulado;
    salida += $" + 1/{i}!";
}
salida += $" = {sumaSerie}";
Console.WriteLine(salida);
```

Ejemplo de ejecución:

```
Número de términos de la serie [2 a 50]: 5
1 + 1/1! + 1/2! + 1/3! + 1/4! = 2.70833333333333
```

Cláusula `break` en bucles

> **Enlaces**
>
> - break

Se usa para ...

1. **forzar la salida inmediata de un bucle**, saltando la evaluación condicional normal.
2. terminar una cláusula `case` de sentencia `switch` y finalizar su ejecución.

Hay que evitarla en la medida de lo posible y si se usa hacerlo con cuidado.

Ejemplo 1:

Ejemplo de programa que pide un número real y muestra su raíz cuadrada, mientras el número introducido no sea negativo.
📌 **Nota:** Fíjate que aunque hemos definido un **bucle infinito**. Usamos una cláusula `break` para salir del mismo.

```
public static void Main()
{
    while(true) // Bucle infinito.
    {
        Console.Write("Introduce un número: ");
        double n = double.Parse(Console.ReadLine() ?? "0");
        if (n < 0d) break;
        Console.WriteLine($"sqrt({n}) = {Math.Sqrt(n):F2}");
    }
}
```

Ejemplo 2:

Vamos a reescribir el programa que me dice si un número es primo o no. Utilizando usando un **bucle while** y una cláusula **break** para salir si el número ya no es primo.

```
public static void Main()
{
    Console.Write("Introduce un valor: ");
    int valor = int.Parse(Console.ReadLine() ?? "0");
    bool esPrimo = true;
    int i = 2;
    while(esPrimo && i < valor)
    {
        if (valor % i == 0)
        {
            esPrimo = false;
            break;
        }
        i++;
    }
    string mensaje = $"El número {(esPrimo?"es":"no es")} primo.";
    Console.WriteLine(mensaje);
}
```

Bucles Anidados

- Formalmente, es cuando creamos un bucle dentro de otro bucle.
- Su uso más habitual es para realizar combinaciones, productos cartesianos, recorridos de espacios N dimensionales.
- Tenemos que usarlos con cuidado pues dan lugar a algoritmos de coste temporal o complejidad exponencial.

Ejemplo 1:

Haz programa en C# que usando **un bucle for anidado**. Calcule por fuerza bruta todas las combinaciones de dos dados que dan una suma leída por el teclado.

- Las entradas posibles serán de 2 a 12.
- No pueden repetirse combinaciones.

Ejemplo de ejecución:

```
Suma dados: 8
Combinaciones -> (2,6)(3,5)(4,4)(5,3)(6,2)
```

📌 **Nota:** Este solución tendrá un coste temporal de **$O(n^2)$** = O(36) mientras que la solución anterior donde hacíamos **un único bucle for** tenía **O(n)** = O(6)

```
public static void Main()
{
    Console.Write("Suma dados: ");
    uint sumaDados = uint.Parse(Console.ReadLine() ?? "0");
    const uint TIRADA_MAXIMA = 6;
    string salida = "Combinaciones -> ";
    for (uint dado1 = 1; dado1 <= TIRADA_MAXIMA; ++dado1)
    {
        for (uint dado2 = 1; dado2 <= TIRADA_MAXIMA; ++dado2)
        {
            if (dado1 + dado2 == sumaDados)
                salida += $"({dado1},{dado2})";
        }
    }
    Console.WriteLine(salida);
}
```

Ejemplo 2:

Crea un programa que utilizando **bucles for anidados** dibuje **un cuadrado** de ceros **'0'** en navegador de **N columnas por N filas**. Siendo N un número introducido por el usuario.
Entre cada cero en la misma fila habrá dos espacios, para que el cuadrado se represente bién en el navegador.

Ejemplo de ejecución:

```
Número de N: 4
0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0
```

📌 **Nota:** Este solución tendrá un coste temporal de **$O(n^2)$** = O(16) mientras que la solución anterior donde no anidábamos bucles tenía **O(2n)** = O(8)

```
public static void Main()
{
    Console.Write("Número de N: ");
    int n = int.Parse(Console.ReadLine() ?? "0");

    string salida = "";
    for (int i = 0; i < n; i++)
    {
        // Aquí estámos dentro del ámbito de primer bucle
        // por lo que deberemos definir otro identificador
        // para el contador del bucle anidado.
        for (int j = 0; j < n; j++)
            salida += "0 ";
        salida += "\n";
    }
    Console.Write(salida);
}
```

Ejemplo 3:

Crea un programa que utilizando **bucles for anidados** dibuje **uns diagonal** de ceros **'0'** en navegador de **N columnas por N filas**. Siendo N un número introducido por el usuario.
Entre cada cero en la misma fila habrá dos espacios, para que el cuadrado se represente bién en el navegador.

Ejemplo de ejecución:

```
Número de N: 4
0
  0
    0
      0
```

📌 **Nota:** En este caso la solución más sencilla sería el bucle anidado, independientemente del coste temporal.

```
public static void Main()
{
    Console.Write("Número de N: ");
    int n = int.Parse(Console.ReadLine() ?? "0");

    string salida = "";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            salida += i == j ? "0 " : "  ";
        salida += "\n";
    }
    Console.Write(salida);
}
```

Anexo I: Cláusula `continue` en bucles

Existen otras cláusulas de salto como: `goto` y `continue`. Qué existen en C# pero no se usan en la práctica, ya que pueden dar lugar a código difícil de mantener y entender. Si tienes tiempo, puedes ver en este anexo la cláusula `continue` por si pudiera aparecer en algún código legado.

> **Enlaces**
>
> - continue

- Para detener la iteración actual de un bucle e iniciar la siguiente de inmediato.
- **Se usa muy ráramente**, pero podría usarse para obviar la ejecución del cuerpo del bucle en ciertos casos.
- 💀 **Deberemos evitarla SIEMPRE**, pues normalmente se puede sustituir por un condicional y puede producir efectos colaterales, además de dar lugar a código ofuscado.

Ejemplo 1:

Programa que lee 10 caracteres por teclado y solo los visualizará si no es un dígito del 0 al 9.

```
public static void Main()
{
    Console.
    Console.Write("Introduce 10 caracteres: ");
    for (int i = 0; i < 10; ++i)
    {
        char car = Console.ReadKey(intercept:true).KeyChar;
        if (car >= '0' && car <= '9')
            continue;
        Console.Write(car);
    }
}
```

Ejemplo de ejecución:

Con la entrada: `s8J-P/5qf*`

```
Introduce 10 caracteres: sJ-P/qf*
```

Ejemplo 2:

Programa que muestra aquello número que sean múltiplos de 5 entre 0 y 50.

```
public static void Main()
{
    for (int n = 0; n <= 50; n++)
    {
        if (n % 5 != 0)
            continue;
        Console.Write($"{n} ");
    }
}
```

Ejemplo de ejecución:

```
0 5 10 15 20 25 30 35 40 45 50
```

🎓 **Caso de estudio:**

¿Serías capaz de indicar lo que va ha mostrar en la consola este programa haciendo una traza manual y **sin ejecutarlo**?

```
int a = 10;
int b;

while (a != 0)
{
    if (a == 3)
        break;
    else
        b = a + 1;

    while (b > 0)
    {
        b--;
        if (b == 5)
            continue;
        else
            Console.Write(b);
    }
    Console.Write("\n");
    a--;
}
```

Respuesta:

```
10987643210
987643210
87643210
7643210
643210
43210
43210
```