Estructuras de repetición

- Bucle while
- Bucle do-while
- Bucle for
- Bucle foreach
- Recursión

Bucle while

- El bucle **while** ejecuta reiteradamente un bloque de instrucciones hasta que una expresión concreta de tipo Boolean se evalúa en false.

```
int i = 1;
while (i <= 5)
{
    Console.WriteLine("El valor de i = {0}", i);
    i++;
}
```

Bucle do-while

- El bucle do-while ejecuta reiteradamente un bloque de instrucciones hasta que una expresión concreta de tipo Boolean se evalúa en false. El bucle do-while comprueba la condición que figura al final del bucle.

```
int i = 1;
do
{
    Console.WriteLine("El valor de i = {0}", i);
    i++;
}
while (i <= 5);
```

Bucle for

- El bucle for combina en un código más fácil de leer los tres elementos de iteración, que son: la expresión de inicialización, la expresión de la condición de terminación y la expresión de recuento.

```
for (int i = 1; i <= 5; i++)
{
    Console.WriteLine("El valor de i = {0}", i);
}
```

Bucle foreach

- El bucle **foreach** es una versión mejorada del bucle for para realizar iteraciones mediante colecciones tales como matrices y listas.

```
int[] numbers = { 1, 2, 3, 4, 5 };
foreach (int i in numbers)
{
    Console.WriteLine("El valor de i = {0}", i);
}
```

Recursión

- La recursión es una técnica de programación que hace que un método se llame a sí mismo para calcular un resultado.

```
public static int Factorial(int n)
{
    if (n == 0)
    {
        return 1;  //base case
    }
    else
    {
        return n * Factorial(n - 1);  //recursive case
    }
}
```

Control de excepciones

- Una excepción es una condición de error inesperada que ocurre durante la ejecución de un programa.
- Cuando ocurre la excepción, el tiempo de ejecución crea un objeto de excepción y lo “inicia”.
- A no ser que “atrape” (catch) la excepción, se terminará la ejecución de programa.
- Las excepciones son un objeto de la clase System.Exception o una de sus clases derivadas.
  - Ejemplo: se inicia una excepción **DivideByZeroException** cuando el programa busca dividir por cero.
  - Ejemplo: se inicia un objeto de excepción **FileNotFoundException** cuando el programa no puede encontrar un archivo dado.

Excepciones no controladas

- ¿Qué pasa cuando el archivo c:\data.txt no se encuentra en este código?

```
private static void ExceptionTest()
        {
            StreamReader sr = null;
            sr = File.OpenText(@"c:\data.txt");
            Console.WriteLine(sr.ReadToEnd());
        }
```

Control de excepciones con try-catch

- Coloque el código que inicia las excepciones dentro de un bloque try.
- Coloque el código que controla una excepción dentro de un bloque catch.
- Puede tener más de un bloque catch por cada bloque try. Cada bloque catch controla un tipo de excepción específica.
- Un bloque try debe contar al menos con un bloque catch o un bloque finally asociado a él.

Ejemplo de control de excepciones

```
private static void ExceptionTest()
{
    StreamReader sr = null;
    try
    {
        sr = File.OpenText(@"c:\data.txt");
        Console.WriteLine(sr.ReadToEnd());
    }
    catch (FileNotFoundException fnfe)
    {
        Console.WriteLine(fnfe.Message);
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
}
```

El bloque finally

- El bloque **finally** se usa en asociación con el bloque try.
- El bloque **finally** siempre se ejecuta, independientemente de que se inicie una excepción.
- El bloque **finally** se usa con frecuencia para escribir código de limpieza.

```
StreamReader sr = null;
try
{
    sr = File.OpenText("data.txt");
    Console.WriteLine(sr.ReadToEnd());
}
finally
{
    if (sr != null)
    {
        sr.Close();
    }
}
```

Ejemplo de try-catch-finally

```
StreamReader sr = null;
try
{
    sr = File.OpenText("data.txt");
    Console.WriteLine(sr.ReadToEnd());
}
catch (FileNotFoundException fnfe)
{
    Console.WriteLine(fnfe.Message);
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
finally
{
    if (sr != null)
    {
        sr.Close();
    }
}
```

Resumen

- Algoritmos
  - Diagrama de flujo, tabla de decisión
- El lenguaje de programación C#
  - Variables, constantes, tipos de datos, matrices, operadores, métodos
- Estructuras de decisión
  - if, if-else, switch
- Estructuras de repetición
  - while, do-while, for, foreach, recursión
- Control de excepciones
  - try-catch, try-finally, try-catch