Tema 3.1

Descargar estos apuntes

Índice

Instrucciones

Cada instrucción se separa o va delimitada por el carácter **';'**

Puede contener asignaciones, comentarios e instrucciones de:
Puede contener asignaciones, comentarios e instrucciones de:

- Control / condicionales.
- Iterativas / bucles.

Comentarios

Permiten introducir texto en mitad del código y que sea ignorado.
La sintaxis es parecida en la mayoría de lenguajes.
Deberemos seguir la máxima: `"Si un código hay que comentarlo, posiblemente no esté bien hecho"`

```
// comentarios de una línea
/*
    comentarios de
    varias líneas
*/
```

Bloques de instrucciones

Se usan llaves para delimitar bloques.

```
{
    // Varias instrucciones de código.
}
```

Se pueden **anidar bloques** pero no pueden declararse variables con el mismo nombre en ellos.
Un bloque **anidado** irá tabulado o *'indentado'* dentro del bloque que lo contiene.

```
{
    int i;
    ...
    {
        int i; //ERROR, i esta declarada en un ámbito envolvente
        ...
    }
}
```

Bloques hermanos pueden tener variables con el mismo nombre.

```
{
    int i;
    ...
}
...
{
    int i;
    ...
}
```

Estructuras de control I

Condicionales

Condicional simple y múltiple con if

La expresión debe escribirse **entre paréntesis** y debería ser una expresión que se evalúe a un valor booleano, esto es **true o false**.

> 📌 **Nota:** En ningún caso se permitirá un cast implícito de una expresión que se evalúa como entero a bool como sucede en otros lenguajes.

Estructura **if**

```
if (expresión)
    sentencia;

if (expresión)
{
    bloque;
}
```

Estructura **if - else**

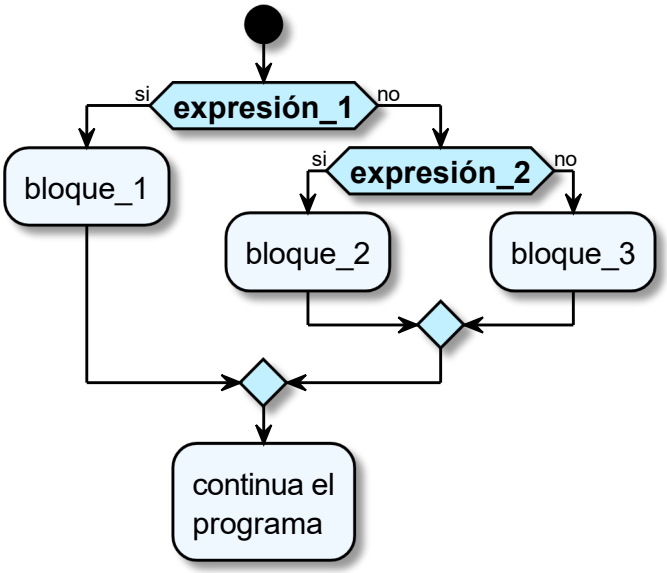
```
if (expresión)
    sentencia_1;
else
    sentencia_2;

if (expresión)
{
    bloque_1;
}
else
{
    bloque_2;
}
```

Estructura **if - else if - else** (Condicional múltiple)

```
if (expresión_1)
{
    bloque_1;
}
else if (expresión_2)
{
    bloque_2;
}
else
{
    bloque_3;
}

// continua el programa;
```



Ejemplos de uso de condicionales

Ejemplo 1

Realiza un programa que calcule la o las soluciones, si las hay, para ecuaciones de 2º grado de la forma:

$$ax^2 + bx^2 + c = 0 \quad donde \quad x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \quad Discriminante \rightarrow \Delta = b^2 - 4ac$$

Nota: Si el **discriminante** es mayor que 0 tiene 2 soluciones. Si es igual a 0 tiene 1 solución y si es menor que cero no tiene soluciones reales.

```
static void Main()
{
    Console.WriteLine("Dame el valor a:");
    int a = int.Parse(Console.ReadLine());
    Console.WriteLine("Dame el valor b:");
    int b = int.Parse(Console.ReadLine());
    Console.WriteLine("Dame el valor c:");
    int c = int.Parse(Console.ReadLine());

    double discriminante = Math.Pow(b, 2) - 4 * a * c;

    string salida;
    if (discriminante < 0)
        salida = "No hay solución";
    else if (discriminante == 0)
    {
        double x = -b / (2 * a);
        salida = $"El resultado final es: {x:F2}";
    }
    else
    {
        double x1 = (-b + Math.Sqrt(discriminante)) / (2 * a);
        double x2 = (-b - Math.Sqrt(discriminante)) / (2 * a);
        salida = $"Tu operación tiene dos soluciones: {x1:F2} y {x2:F2}";
    }

    Console.WriteLine(salida);
}
```

Ejemplo 2

Se nos pide hacer un sencillo programa de terminal en C# que calcule el máximo recomendable de pulsaciones durante el ejercicio físico de una persona. Para ello se nos dan las siguientes reglas:
Si una persona hace ejercicio con regularidad y esta en buen estado físico, para calcular sus pulsaciones restaremos a 200 puntos la edad del individuo. Además si tiene entre 30 y 39 deberemos descontar siempre 5 puntos, para los que tengan entre 40 y 49, 15 puntos, y los que se hallen por encima de los cincuenta años 20 puntos.
Si la persona no esta en buen estado físico o empieza ha hacer ejercicios deberemos restar además de las cantidades anteriores 40 puntos a cualquier edad.

```
static void Main()
{
    Console.Write("Introduzca su edad: ");
    uint edad = ushort.Parse(Console.ReadLine());
    uint pulso = 200u - edad;

    if (edad >= 30 && edad < 40)
        pulso -= 5;
    else if (edad >= 40 && edad < 50)
        pulso -= 15;
    else if (edad >= 50)
        pulso -= 20;

    Console.Write("Hace ejercicio físico con frecuencia S/N: ");
    bool haceEjercicio = char.ToLower(char.Parse(Console.ReadLine())) == 's';
    if (!haceEjercicio)
        pulso -= 40;

    Console.WriteLine($"Su pulso idóneo para hacer ejercicio es de {pulso} ppm");
}
```

Ejemplo 3

Realiza un programa que te ayude a tomar decisiones jugando a **Piedra - Papel - Tijera**
Para ello...

1. Pedirá al usuario una cadena con su elección y él elegirá un número al azar y lo transformará a una jugada.
2. La expresión / función para generar un valor entero aleatorio en C#

   ```
   Random semilla = new Random();
   int jugadaOrdenador = semilla.Next(1, 4); // Devuelve un valor entero aleatorio entre 1 y 3
   ```
3. El programa los comparará, mostrará ambas elecciones y dirá quien ha perdido.

```
static void Main()
{
    Console.Write("Introduce tu jugada (Piedra, Papel, Tijera): ");
    string jugadaUsuario = Console.ReadLine().ToUpper();

    const string PIEDRA = "PIEDRA";
    const string PAPEL = "PAPEL";
    const string TIJERA = "TIJERA";

    Random semilla = new Random();
    int valorJugadaOrdenador = semilla.Next(1, 4);

    string jugadaOrdenador;

    if (valorJugadaOrdenador == 1)
        jugadaOrdenador = PIEDRA;
    else if (valorJugadaOrdenador == 2)
        jugadaOrdenador = PAPEL;
    else
        jugadaOrdenador = TIJERA;

    string mensaje;
    if (jugadaUsuario == jugadaOrdenador)
        mensaje = "Empate";
    else if (jugadaUsuario == PIEDRA && jugadaOrdenador == TIJERA
                ||
                jugadaUsuario == PAPEL && jugadaOrdenador == PIEDRA
                ||
                jugadaUsuario == TIJERA && jugadaOrdenador == PAPEL)
        mensaje = "Ganas";
    else
        mensaje = "Pierdes";


    Console.WriteLine($"Tu jugada ha sido: {jugadaUsuario}");
    Console.WriteLine($"La jugada del ordenador ha sido: {jugadaOrdenador}");
    Console.WriteLine(mensaje.ToUpper() + " !!!");
}
```

Condicional múltiple con switch

La sintaxis básica de la **instrucción o sentencia switch** será la siguiente:

```
switch (expresión no booleana)
{
    case literal1:
    // No se permiten instrucciones ni bucles sin un break;
    case literal2:
    case literal3:
        instrucción;
        break;
    case literal4: {
        bloque
        break;
    }
    ...
    default:
        sentencia;
        break;
}
```

Existe con sintaxis similar en casi todos los lenguajes y es la que deberíamos usar en la mayoría de los casos, pues es reconocida por todos los programadores.

¿Cómo evalúa C# la siguiente sentencia?...

1. Se evalúa la expresión que debería dar un resultado no booleano.
2. Se ejecuta la claúsula `case` cuyo literal se corresponda con el valor obtenido en el punto 1.
   Si no se encuentra correspondencia, se ejecuta el caso `default`; y si no hay `default`, termina la ejecución de `switch`.
3. Termina cuando se encuentra una sentencia `break` esta deberíamos ponerla para evitar caídas al *'vacío'*. (💀 **fall through** 💀)
   Solo será válido no ponerla para **agrupar** literales.

Ejemplos condicional múltiple con switch

> **Ejemplo 1**
>
> Veamos un sencillo ejemplo en el que vamos a asignar a una variable **presupuesto** el presupuesto de un departamento para un año. Teniendo en cuenta que el departamento de matemáticas tiene asignados 200€, el de lengua francesa y castellano 500€ y el de informática 100€.

```
string departamento = "INFORMATICA";
int? presupuesto;
switch (departamento)
{
    case "MATEMATICAS":
            presupuesto = 200;
            // Fall thought al no poner break.
            // C# nos generará un error de compilación aunque otros lenguajes no.
    case "CASTELLANO":
    case "FRANCES":
            // Agrupación de varios casos (Esto es válido).
            presupuesto = 500;
            break;
    case "INFORMATICA":
            presupuesto = 100;
            break;
    default:
            presupuesto = null;
            break;
}
string mensaje = $"El departamento de {departamento.ToLower()} " +
                ((presupuesto != null)
                ? $"tiene {presupuesto}€ de presupuesto"
                : "no tiene presupuesto");
Console.WriteLine(mensaje);
```

Ejemplo 2

¿Cómo sería la selección de la jugada en el ordenador para nuestro juego de **piedra – papel – tijera** si usáramos un switch?

```
switch(valorJugadaOrdenador)
{
    case 1: jugadaOrdenador = PIEDRA;
        break;
    case 2: jugadaOrdenador = PAPEL;
        break;
    case 3: jugadaOrdenador = TIJERA;
        break;
    default: jugadaOrdenador = "Jugada no válida";
        break;
}
```

Ejemplo 3

Realiza un programa de terminal en C# que dado un año y un mes de ese año, ambos en formato numérico. Me diga cuantos días tiene ese mes.
Utiliza una estructura switch para hacerlo.

```
static void Main()
{
    int? dias;

    Console.Write("Dime un año: ");
    uint año = uint.Parse(Console.ReadLine());
    Console.Write("Dime un número de mes de ese año: ");
    uint mes = uint.Parse(Console.ReadLine());

    switch (mes)
    {
        case 1:
        case 3:
        case 5:
        case 7:
        case 8:
        case 10:
        case 12:
            dias = 31;
            break;
        case 2:
            dias = año % 4 == 0 && año % 100 != 0
                    ||
                    año % 400 == 0 ? 28 : 29;
            break;
        case 4:
        case 6:
        case 9:
        case 11:
            dias = 30;
            break;
        default:
            dias = 0;
            break;
    }

    string mensaje = dias != null ?
                    $"El {mes} de {año} tiene {dias} días." :
                    "Número de mes incorrecto.";
    Console.WriteLine(mensaje);
}
```

Cláusula when junto con instrucción switch

Desde **C#7** el lenguaje permite añadir condiciones a un mismo caso. A través de la evaluación de una expresión booleana. Para ello usaremos la palabra reservada del lenguaje **when**

Veamos un **ejemplo básico** de sintaxis, aunque su utilidad real la veremos más adelante al ver otros casos de uso de la instrucción switch dentro de la Programación Orientada a Objetos (POO).

```
switch (expresión no booleana)
{
    case literal1 when <expresión booleana>:
    case literal2:
        instrucción;
        break;
    case literal3 when <expresión booleana>:
        break;
    case literal4 when <expresión booleana>: {
        bloque
        break;
    }
    ...
    default:
        sentencia;
        break;
}
```

> **Ejemplo**
>
> ¿Cómo sería la selección del resultado del juego de **piedra – papel – tijera** si usáramos un switch con un when?

```
string mensaje;
switch (jugadaUsuario)
{
    case PIEDRA when jugadaOrdenador == TIJERA:
    case PAPEL when jugadaOrdenador == PIEDRA:
    case TIJERA when jugadaOrdenador == PAPEL:
        mensaje = "Ganas";
        break;
    case PIEDRA when jugadaOrdenador == PAPEL:
    case PAPEL when jugadaOrdenador == TIJERA:
    case TIJERA when jugadaOrdenador == PIEDRA:
        mensaje = "Pierdes";
        break;
    default:
        mensaje = "Empate";
        break;
}
```

Expresiones condicionales switch

Además, de como instrucción, a partir de **C#8** podemos, usar la cláusula switch a modo expresión, evaluándose esta a un resultado.

Este esquema switch como expresión se ha incorporado de forma similar a otros lenguajes como Java14, Kotlin o Rust.

Las expresiones switch permiten usar una sintaxis de expresiones más concisa. Hay **menos palabras clave** case y break repetitivas.

La sintaxis básica podría ser:

```
TipoDato dato = <patrón> switch
{
    <caso 1 patrón> => <expresión que se evalúe a TipoDato>,
    <caso 2 patrón> when <expresión booleana> => <expresión que se evalúe a TipoDato>,
    ...
    <caso n patrón> => <expresión que se evalúe a TipoDato>,
    _               => <expresión que se evalúe a TipoDato o Error> // Caso por defecto
};
```

> **Ejemplo**
>
> ¿Cómo sería la selección de la jugada en el ordenador para nuestro juego de **piedra – papel – tijera** si usáramos una **expresión switch**?
> Si lo comparamos con el que hicimos con la **instrucción o sentencia switch**, vemos que queda más claro y simplificado ahorrándonos clausulas case y break.

```
string jugadaOrdenador = valorJugadaOrdenador switch
{
    1 => PIEDRA,
    2 => PAPEL,
    3 => TIJERA,
    _ => "Jugada no válida"
};
```

Existe una variante donde el patrón a buscar sea un **tupla** de 2 o n valores:

```
TipoDato dato = (e1,e2,...,en) switch
{
    (d1t1,d2t1,...,dnt1) => <expresión que se evalúe a TipoDato>,
    (d1t2,d2t2,...,dnt2) => <expresión que se evalúe a TipoDato>,
    ...
    (d1tn,d2tn,...,dntn) => <expresión que se evalúe a TipoDato>,
    (_,_,...,_)          => <expresión que se evalúe a TipoDato o Error> // Caso por defecto
};
```

Ejemplo

Así podría ser la selección del resultado del juego de **piedra – papel – tijera** si usáramos: una **expresión switch** con un patrón tupla dentro, a su vez, de una expresión más compleja. En el ejemplo las tuplas serán de 2 valores solo.

📌 **Nota:** Este ejemplo es algo enrevesado pero nos ayuda a entender una expresión puede formar parte de otra aún más compleja. Obviamente no sería la forma más recomendable pues al usarse con una **ternaria** y un **??**, la expresión resultante es algo ofuscada.

```
string mensaje = jugadaUsuario == jugadaOrdenador
? "Empate"
: (jugadaUsuario, jugadaOrdenador) switch
{
    (PIEDRA, TIJERA) => "Ganas",
    (PAPEL, PIEDRA) => "Ganas",
    (TIJERA, PAPEL) => "Ganas",
    (_, _) => null
}
?? "Pierdes";
```