Índice

Ejercicios Unidad 7

Descargar estos ejercicios

> **Antes de empezar**
>
> Para realizar estos ejercicios, deberás descargar los recursos del enlace de proyecto_funciones2. Como puedes ver, la solución está compuesta de varios proyectos. Cada uno de ellos corresponde con un ejercicio, deberás implementar todo el código, tanto de la Main como de los métodos que se piden en cada ejercicio. Cada proyecto contiene el test correspondiente, que deberás pasar para comprobar que has hecho el ejercicio correctamente. ***No olvides comentar los Test de los que todavía no has hecho el código***.

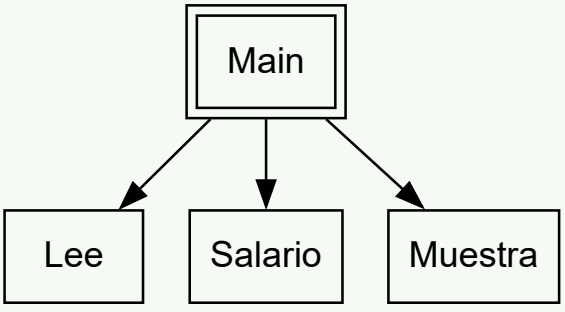
Ejercicio 1. Proyecto calcula salario

Este ejercicio estará formado por varios métodos que definirás en el proyecto ejercicio1.

```
Ejercicio 1: Calculadora de Salario
Introduce el número de departamento: 4
Introduce el coste por hora: 25
Introduce las horas trabajadas: 5

--- INFORMACIÓN DEL EMPLEADO ---
Número de departamento: 4
Coste por hora: 25,00 ?
Horas trabajadas: 5
Salario semanal: 125,00 ?

Presiona cualquier tecla para salir...
```

Requisitos:

- Método llamado **Lee** que recoja los siguientes datos de un usuario: número de departamento (int), coste por hora(float) y horas trabajadas (float) y los devuelva. **Usará tuplas para devolverlos**
- Escribe un método llamado **Salario** que para calcular el salario semanal multiplique el coste por hora por el número de horas trabajadas, que le llegarán como parámetros de entrada. **El método devolverá el salario (float)**.
- Escribe un método llamado **Muestra** que muestre el salario semanal, el número del departamento, el coste por hora y las horas trabajadas, que le llegarán todas por parámetro.
- Podéis fijaros en el siguiente DEM:



Ejercicio 2. Proyecto adivina número

Escribe un programa en el proyecto ejercicio2 para jugar a adivinar números. Los pasos a seguir son los siguientes:

```
Ejercicio 2. Proyecto adivinar el número

--- SELECCIONA EL NIVEL ---
1. Fácil (10 tentativas)
2. Medio (6 tentativas)
3. Difícil (4 tentativas)
Elige una opción (1-3): 4
Opción no válida. Inténtalo de nuevo.

--- SELECCIONA EL NIVEL ---
1. Fácil (10 tentativas)
2. Medio (6 tentativas)
3. Difícil (4 tentativas)
Elige una opción (1-3): 1

¡Adivina el número entre 0 y 100!
Tienes 10 tentativas.

Tentativa 1 de 10:
Introduce tu número: 50
El número es menor.
Te quedan 9 tentativas.

Tentativa 2 de 10:
Introduce tu número: 25
El número es menor.
Te quedan 8 tentativas.

Tentativa 3 de 10:
Introduce tu número: 17
El número es mayor.
Te quedan 7 tentativas.

Tentativa 4 de 10:
Introduce tu número: 20
El número es mayor.
Te quedan 6 tentativas.

Tentativa 5 de 10:
Introduce tu número: 22
¡Felicidades! Has adivinado el número.
¡Excelente! Has adivinado el número en 5 tentativas.

¿Quieres seguir jugando? (S/N): f
Respuesta no válida. Por favor, responde S o N.

¿Quieres seguir jugando? (S/N): n
¡Gracias por jugar!
```

Requisitos:

- Método `NumeroAAdivinar` que devolverá un número entero generado de forma aleatoria. El número debe hallarse entre 0 y 100(ambos inclusive).
- Método `Pista` al que le llega el número a adivinar y se encarga de pedir un número al jugador y dar una **pista** indicando si el número introducido es mayor, igual o menor que el número a adivinar. Devolverá true en caso de que el número se haya adivinado.
- Método `Nivel` que se encargará de mostrar un menú con los tres niveles y devolverá las tentativas (int). Si no se selecciona una opción correcta del menú, se volverá a repetir el proceso. Habrá un máximo de tentativas dependiendo del **nivel** elegido para jugar: `fácil =10, medio = 6, difícil = 4`.
- Método `Juego` este método tendrá la lógica del juego y se encargará de comprobar si se ha acertado el número o si se han acabado las tentativas, repitiendo el proceso hasta que no ocurra esto. Si el jugador acierta el número, la partida terminará indicando la **cantidad de tentativas** hechas por este jugador para acertar.
- El método principal se encargará de preguntar si se desea **seguir jugando**. Si se responde que [S/s] el juego seguirá pidiendo un nuevo nivel y generando otro número para adivinar, si se responde [N/n] se saldrá del programa. Cualquier otra respuesta no será válida y se pedirá que se vuelva a responder.

Ejercicio 3. Proyecto juego

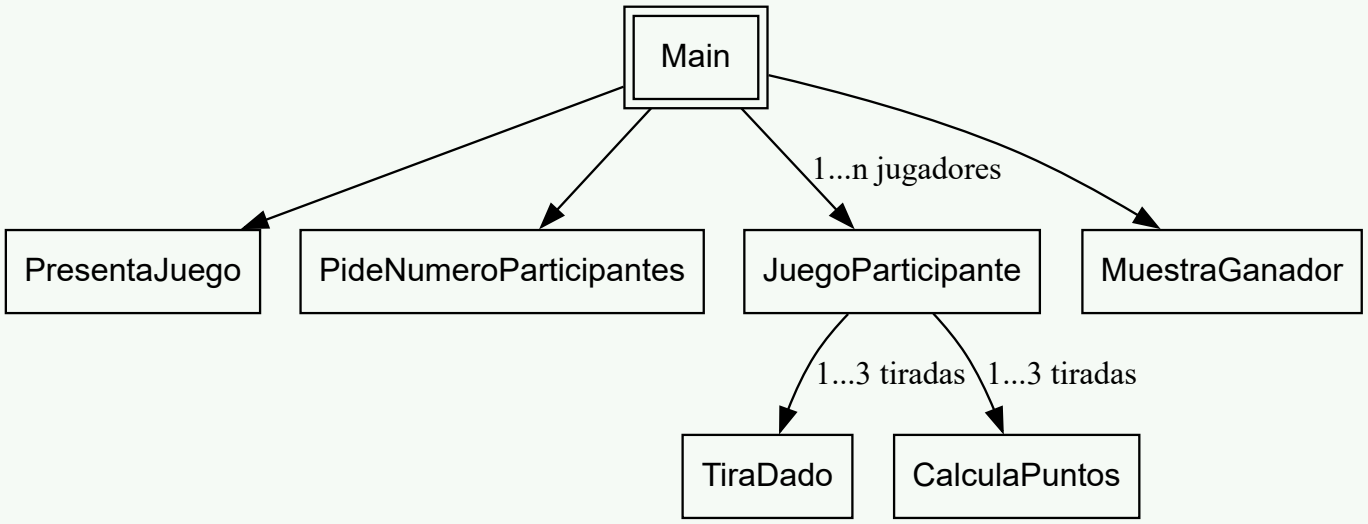
Programa que implementará un juego con las siguientes características:

```
Ejercicio 3: Proyecto juego

Reglas del juego:
* Cada participante tira 3 veces un dado (valores 1-100)
* Se suman las puntuaciones según estas reglas:
  * Múltiplo de 3 o 5: +10 pts
  * Múltiplo de 4 o 6: +5 pts
  * Mayor de 80: +2 pts
  * Mayor de 50: +1 pts
  * Menor de 50: -2 pts
  * Menor de 20: -1 pts
* Gana quien obtenga mayor puntuación total

Introduce el número de participantes: 2

--- PARTICIPANTE 1 ---
Tirada 1: Dado = 78, Puntos = +16
Tirada 2: Dado = 3, Puntos = +7
Tirada 3: Dado = 66, Puntos = +16
Puntuación total del participante 1: 39

--- PARTICIPANTE 2 ---
Tirada 1: Dado = 40, Puntos = +13
Tirada 2: Dado = 13, Puntos = -3
Tirada 3: Dado = 75, Puntos = +11
Puntuación total del participante 2: 21

=== RESULTADO FINAL ===
¡El ganador es el PARTICIPANTE 1 con 39 puntos!

Presiona cualquier tecla para salir...
```

Requisitos:

- El programa pedirá que introduzcas el número de participantes a jugar.
- Cada participante tirará 3 veces un dado con valores entre 1 y 100 (electrónico se entiende), sumándose el valor de las 3 jugadas. Ganará el participante que obtenga mayor puntuación según las siguientes reglas:
  - Si el nº obtenido es múltiplo de 3 o de 5 sumara 10 pts.
  - Si el nº obtenido es múltiplo de 4 o de 6 sumara 5 pts.
  - Si el nº obtenido es mayor de 80 sumara 2 pts.

    - Si el nº obtenido es mayor de 50 sumar 1 pts.
    - Si el nº obtenido es menor de 50 restará 2 pts.
    - Si el nº obtenido es menor de 20 restará 1 pts.
- El DEM para el juego será el siguiente.



Ejercicio 4. Proyecto Conversores

El proyecto 4 tratará de escribe un programa que lea un número en base 10 (decimal) y que posteriormente muestre un menú que nos permita convertirlo a base 2 (binario), base 8 (octal) o base 16 (hexadecimal).

Crea un método llamado **PasaABinario**, otro llamado **PasaAOctal** y otro llamado **PasaAHexadecimal** para realizar dichas operaciones.
Todos los métodos **devolverán un string** que contendrá los dígitos resultantes de la conversión.

- **Nota 1:** No se puede usar el método **Convert** existente ya en C#.
- **Nota 2:** Evita el código repetido.

```
Ejercicio 4. Proyecto conversores

Introduce un número decimal: 15

=== CONVERSOR DE BASES ===
1. Convertir a binario (base 2)
2. Convertir a octal (base 8)
3. Convertir a hexadecimal (base 16)
4. Salir
Selecciona una opción (1-4): 6
Opción no válida. Por favor, selecciona una opción del 1 al 4.

=== CONVERSOR DE BASES ===
1. Convertir a binario (base 2)
2. Convertir a octal (base 8)
3. Convertir a hexadecimal (base 16)
4. Salir
Selecciona una opción (1-4): 1
15 en binario es: 1111

Introduce un número decimal: 15

=== CONVERSOR DE BASES ===
1. Convertir a binario (base 2)
2. Convertir a octal (base 8)
3. Convertir a hexadecimal (base 16)
4. Salir
Selecciona una opción (1-4): 2
15 en octal es: 17

Introduce un número decimal: 15

=== CONVERSOR DE BASES ===
1. Convertir a binario (base 2)
2. Convertir a octal (base 8)
3. Convertir a hexadecimal (base 16)
4. Salir
Selecciona una opción (1-4): 3
15 en hexadecimal es: F

Introduce un número decimal: 4

=== CONVERSOR DE BASES ===
1. Convertir a binario (base 2)
2. Convertir a octal (base 8)
3. Convertir a hexadecimal (base 16)
4. Salir
Selecciona una opción (1-4): 4
¡Hasta luego!
Presiona cualquier tecla para salir...
```

Requisitos:

- Si no sabes como es el proceso de conversión matemático, puedes consultar el proceso de decimal a binario en el siguiente enlace: Decimal a binario
  Para el resto de sistemas es análogo solo que dividiendo entre 8 y 16 respectivamente.
- En todos los casos obtendremos restos **enteros** entre **0 a 15** que deberemos pasar a cadena o a caracter.
  Una de las opciones es pasar a cadena. En este caso cuando el valor esté entre **0 a 9** generaremos una cadena con **"0", "1" ... o , "9"** por ejemplo con **$"{resto}"**. Sin embargo, cuando el valor este entre **10 y 15**, generaremos una cadena con los caracteres **"A", "B", ... o ,"F"** y eso se puede hacer, por ejemplo, con un expresión switch.
  De tal manera que podríamos tener una expresión del tipo ...

```
string textoResto = resto < 10
        ? $"{resto}"
        : resto switch { 10 => "A", 11 => "B", 12 => "C", 13 => "D", 4 => "E", 15 => "F", _ => "?" };
```

- Mas **eficiente** y tradicional de programadores de C y C++ es usar la tabla UTF-8 o su subconjunto ASCII para en lugar de pasar a cadena pasar a **carácter**. El planteamiento sería el siguiente:
  - El carácter '0' internamente se guarda con el valor 48 (Decimal) y el resto de caracteres que representan dígitos decimales **van seguidos en esa tabla**. Por tanto, si cuando el resto es menor que 10 hago...

```
const int INCIO_CARCACTER_0 = 48;
char simbolo = (char)(INCIO_CARCACTER_0 + resto);
```

  - El carácter mayúscula 'A' se guarda internamente con el valor 65 y la 'B', 'C',..., 'F' van seguidos en esa tabla luego...

  Puedes utilizar este último planteamiento si lo consideras más adecuado.
- En el proceso de crear la cadena con la conversión, igual se te plantea la necesidad de invertir una cadena. Veamos un ejemplo...

> **Ejemplo:**
>
> Supongamos que queremos pasar el 6 en decimal a binario (`110`)
>
> En algún momento inicializarás la cadena de conversión.
> `cadenaConLaConversion = "";`
>
> En la **iteración 1**:
> `6 % 2 = 0` entonces `cadenaConLaConversion += simboloDelResto;`
> Ahora `cadenaConLaConversion = "0"` y obtendremos el cociente para siguiente iteración con `6 / 2 = 3`

En la **iteración 2**:
`3 % 2 = 1` entonces `cadenaConLaConversion += simboloDelResto;`
Ahora `cadenaConLaConversion = "01"` y obtendremos el cociente para siguiente iteración con `2 / 2 = 1`

Como `cociente < base` entonces `cadenaConLaConversion += simboloDelCociente;`
Ahora cadenaConLaConversion final valdrá `"011"`

Para tener la conversión correcta `"011"` os puede surgir la necesidad de invertir la cadena para obtener `"110"` que es el **6 en binario**. De momento solo sabemos concatenarlas con el operador `+` o `+=` por tanto una forma de resolver esto sería la siguiente ...

En la **iteración 1**:
`6 % 2 = 0` entonces `cadenaConLaConversion = simboloDelResto + cadenaConLaConversion;`
Ahora `cadenaConLaConversion = "0"` y obtendremos el cociente para siguiente iteración con `6 / 2 = 3`

En la **iteración 2**:
`3 % 2 = 1` entonces `cadenaConLaConversion = simboloDelResto + cadenaConLaConversion;`
Ahora `cadenaConLaConversion = "10"` y obtendremos el cociente para siguiente iteración con `2 / 2 = 1`

Como `cociente < base` entonces
`cadenaConLaConversion = simboloDelCociente + cadenaConLaConversion;`

Ahora cadenaConLaConversion final valdrá `"110"`

De esta manera no necesito invertir porque ya la tengo en el orden correcto. La idea es, en lugar de concatenar por la derecha, **concatenar** por la izquierda.

Ejercicio 5. Funciones con cuerpo de expresión

Crea un conjunto de funciones matemáticas simples **usando sintaxis de expresión**.

```
Ejercicio 5: Funciones con cuerpo de expresión
Introduce un número: 5
Introduce otro número: 3

Número absoluto de -5: 5
¿5 es par? False
¿3 es primo? True
Máximo entre 5 y 3: 5
Mínimo entre 5 y 3: 3
```

Requisitos:

- Define las siguientes funciones usando cuerpo de expresión (`=>`):
  - `int ValorAbsoluto(int n)` que retorne el valor absoluto
  - `bool EsPar(int n)` que determine si es par
  - `bool EsPrimo(int n)` que determine si es primo (solo para números pequeños)
  - `int Maximo(int a, int b)` que retorne el mayor
  - `int Minimo(int a, int b)` que retorne el menor

Ejercicio 6. Polimorfismo funcional. Gestión Taxi

Para practicar el concepto de **polimorfismo funcional o sobrecarga**, vamos a suponer que tenemos un método de utilidad que nos permite calcular el coste de la carrera de un taxi.

```
Ejercicio 6. Polimorfismo funcional. Gestión Taxi

Coste carrera lunes mañana -> 21,38
Coste carrera lunes noche -> 22,10
Coste carrera lunes con mi mascota Dogo -> 22,38
Coste carrera Domingo de Ramos -> 29,94
Coste carrera Domingo noche -> 25,66
Coste carrera Domingo de Ramos noche con Dogo y Minina -> 31,94

Pulse una tecla para finalizar...
```

Requisitos

Hay cuatro conceptos básicos que conforman el precio de la carrera. Dos de ellos fijos, que son la bajada de bandera y la carrera mínima, y dos conceptos variables en cada trayecto: los kilómetros recorridos y el tiempo de espera.

Además de los cuatro conceptos básicos, existen algunos recargos o suplementos que deben pagarse en determinadas circunstancias: por día festivo o domingo, por horario nocturno, por mascotas u otros conceptos de ocupación extra.

A partir del método `CosteCarrera` con **parámetros opcionales** que se muestra a continuación, refactoríza el código para quitar los parámetros opcionales de métodos públicos sobrecargando `CosteCarrera` y que se ofrezcan sobrecargas con el menor número de parámetros posibles.

> 📌 **Nota:** Modificaremos el `Main` para evitar llamadas donde se pasen los valores a `0` o a `false` en las llamadas a `CosteCarrera`, al realizar estos cambios deberás modificar el tipo de `ocupacionExtra` a `uint`, para que no se produzca ambigüedad en la llamada.

```
public static class Taxi
{
    const float BAJADA_BANDERA = 1.82F;
    const float CARRERA_MINIMA = 3.63F;
    const float COSTE_KM = 0.9F;
    const float ESPERA_POR_HORA = 18.77F;
    const short PORCENTAJE_NOCTURNO = 30;

    public static double CosteCarrera(
                    float kilometrosRecorridos, float minutosEspera,
                    bool nocturno = false, int porcentajeFestivo = 0, int ocupacionExtra = 0)
    {
        float costeCarrera = BAJADA_BANDERA + kilometrosRecorridos * COSTE_KM
                                + minutosEspera * (ESPERA_POR_HORA / 60);
        costeCarrera = costeCarrera < CARRERA_MINIMA ? CARRERA_MINIMA : costeCarrera;
        float incrementoNocturno = nocturno ? costeCarrera * PORCENTAJE_NOCTURNO / 100f : 0;
        float incrementoFestivo = porcentajeFestivo != 0 ? costeCarrera * porcentajeFestivo / 100f : 0;
        costeCarrera += incrementoFestivo >= incrementoNocturno ? incrementoFestivo : incrementoNocturno;
        costeCarrera += ocupacionExtra;
        return costeCarrera;
    }
}
```

```
class Program
{
    static void Main()
    {
        Console.WriteLine("Ejercicio 6. Polimorfismo funcional. Gestión Taxi\n");
        Console.WriteLine($"Coste carrera lunes mañana -> {Taxi.CosteCarrera(20, 5):f2}");
        Console.WriteLine($"Coste carrera lunes noche -> {Taxi.CosteCarrera(20, 5, true):f2}");
        Console.WriteLine($"Coste carrera lunes con mi mascota Dogo -> " +
                        $"{Taxi.CosteCarrera(20, 5, 1u):f2}");
        Console.WriteLine($"Coste carrera Domingo de Ramos -> {Taxi.CosteCarrera(20, 5, 40):f2}");
        Console.WriteLine($"Coste carrera Domingo noche -> {Taxi.CosteCarrera(20, 5, true, 20):f2}");
        Console.WriteLine($"Coste carrera Domingo de Ramos noche con Dogo y Minina -> " +
                        $"{Taxi.CosteCarrera(20, 5, true, 40, 2u):f2}");
        Console.WriteLine("\nPulse una tecla para finalizar...");
        Console.ReadKey(true);
    }
}
```