



Nombre

**Instrucciones para la realización de la prueba:**

- El proyecto se realizará durante el periodo de vacaciones de Navidad
- Lee atentamente el enunciado y las posibles preguntas realízalas en el foro para que así entre todos vayamos aclarando dudas. Debes realizarlas dándole a responder en el hilo "Proyecto carrera caballos"
- El proyecto puntuará como si fuera un control más.
- Se restará **0,1 décimas** en cada uno de los siguientes casos:
  - Llaves mal puestas e instrucciones mal tabuladas dentro de un bloque.
  - Identificador de variable que a juicio del profesor no esté bien puesto porque no sigue las reglas descritas en clase.
  - Variables declaradas en lugares inadecuados
- En caso de descubrir a algún alumno que ha copiado el proyecto se le suspenderá el mismo.

Descargar este enunciado [pdf](#) y [html](#)

## Especificaciones del programa

### Consideraciones iniciales

Supondremos que la consola es un terminal de en modo texto de **24 filas x 80 columnas**.

- En todas las carreras correrán **8 caballos**.
- Dispondremos de un **saldo inicial de 1000 €**.

### Menú

Aparecerá un menú con las siguientes opciones...

Elige una opción:

1. Jugar apuesta caballos.
2. Ver apuestas jugadas.
3. Salir.

Pulsa una opción:

### Opción 1: Jugar apuesta caballos

Me **pedirá** los datos indicados con interrogaciones:

```
Saldo actual: 1000,00
Cuanto quieres apostar: ???
Introduce el número del caballo (de 1 a 8): ?
La apuesta se paga ? a 1
```

Con ellos crearemos un **ticket de apuesta** que guardaremos como **tupla** de la siguiente forma:

```
(ushort caballo, float cantidadApostada, ushort eurosPagadosPorUnoApostado) ticketApuesta;
```

Tras apostar se disputará la carrera tal y como se muestra en la siguiente imagen:

17 col	8 col	17 col	10 col	8 col	17 col
Saldo	C.A.	Apuesta	Se Paga	C.G.	Ganado
1.000,00 ?	8	200,00 ?	4 a 1	8	800,00 ?
					~(1)o
					~(2)o
					~(3)o
					~(4)o
					~(5)o
					~(6)o
					~(7)o
					~(8)o
Pulsa una tecla...					

Mientras se disputa la carrera se mostrará información del saldo y el ticket de apuesta, tal y como se muestra en la imagen.

Al finalizar la carrera se indicará la información de final de carrera, con el caballo ganador (C.G.) y la cantidad ganada si el caballo ganador coincide con el caballo apostado (C.A.). Si no coinciden en ganado se mostrará 0,00 €.

Tras al finalizar la carrera se esperará una pausa pulsando una tecla y se volverá al menú.

## Opción 2: Ver apuestas jugadas

Mostrara información sobre las partidas jugadas en el siguiente formato:

17 col	8 col	17 col	10 col	8 col	17 col
Saldo	C.A.	Apuesta	Se Paga	C.G.	Ganado
1.000,00 ?	3	50,00 ?	6 a 1	8	0,00 ?
950,00 ?	7	7,00 ?	7 a 1	5	0,00 ?
943,00 ?	2	300,00 ?	9 a 1	7	0,00 ?
643,00 ?	5	200,00 ?	5 a 1	7	0,00 ?
443,00 ?	5	100,00 ?	3 a 1	5	300,00 ?
643,00 ?	6	40,00 ?	9 a 1	6	360,00 ?
963,00 ?	1	900,00 ?	2 a 1	7	0,00 ?
63,00 ?	3	63,00 ?	7 a 1	3	441,00 ?
Pulsa una tecla...					

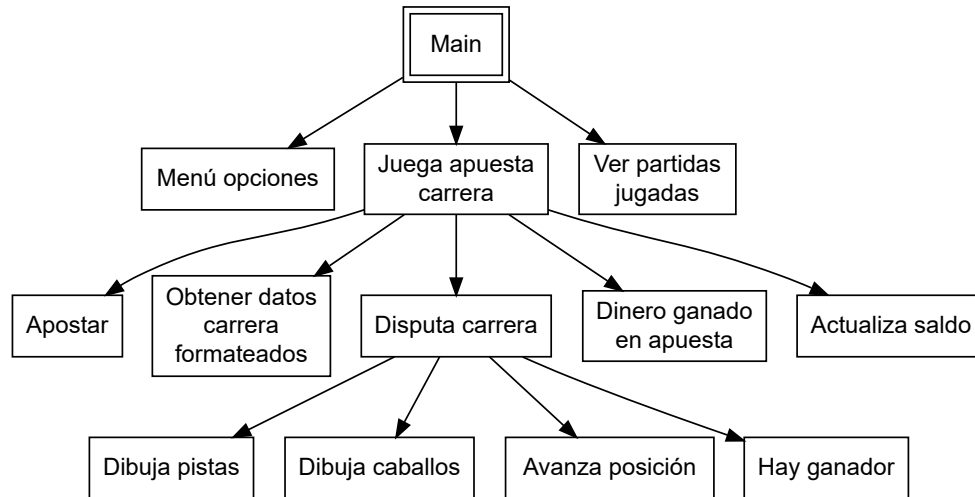
Si al llegar el número de partidas no cabe en una sola pantalla, se mostrarán en tantas pantallas como sea necesario. Siempre precedido de la pulsación de una tecla para continuar. Tras Finalizar volveremos al menú principal.

**Tip:** Puedes usar un **StringBuilder** en el programa principal, para guardar la información de la partida. Se puede usar un carácter específico para separar las distintas partidas y después antes de visualizarlas hacer **Split** sobre la cadena guardada.

# Propuesta de diseño

Según las especificaciones anteriormente descritas una propuesta de diseño para el programa puede ser la siguiente:

## Esquema de diagrama de estructura de módulos



Se recomienda empezar la implementación de los mismos de abajo a arriba. (Bottom-Up)

Con los siguientes interfaces de métodos estáticos a implementar...

```
static bool HayGanador(
    ushort[] posiciones, ushort longitudPistaEnColumnas,
    out ushort ganador)

// Solo avanzará uno de los caballos.
static void AvanzaPosicion(ushort[] posiciones, Random semillaAleatoria)

static void DibujaCaballos(ushort[] posiciones)
static void DibujaPistas(ushort numCaballos, ushort longitudPistaEnColumnas)
static (ushort caballo,
    float cantidadApostada,
    ushort eurosPagadosPorUnoApostado) Apostar(float saldo, ushort numCaballos)
static float DineroGanadoEnApuesta(
    ushort caballoGanador,
    (ushort caballo, float cantidadApostada, ushort eurosPagadosPorUnoApostado) ticketApuesta)
static ushort DisputaCarrera(ushort numCaballos)
static string CabecerasDatosCarrera()
static string ObtenDatosCarreraFormateados(
    float saldo,
    (ushort caballo, float cantidadApostada, ushort eurosPagadosPorUnoApostado) ticketApuesta,
    string caballoGanador, float ganado)
static float ActualizaSaldo(float saldo, float apostado, float ganado)
static void Pausa()
static string JuegaApuestaACarrera(ref float saldo)
static void MenuOpciones()
static void VerPartidasJugadas(string datosCarrerasAcabadas)
```

# Recomensaciones y tips de implementación

1. Se debe ejecutar el programa en un terminal externo. Esto lo puedes configurar en la `launch.json`
2. Para establecer el `Console.SetCursorPosition` no se salga del tamaño de la ventana del terminal. Puedes configurar sus dimensiones de la siguiente manera.

```
public void InicializaVentana()
{
    const int COLUMNAS_TERMINAL = 30;
    const int FILAS_TERMINAL = 90;
    Console.SetWindowSize(COLUMNAS_TERMINAL, FILAS_TERMINAL);
    Console.SetBufferSize(COLUMNAS_TERMINAL, FILAS_TERMINAL);
    Console.Clear();
}
```

3. En el módulo `DisputaCarrera`, para evitar efectos secundarios y parpademos del cursor al situarlo por la pantalla. Puedes ocultarlo al principio y mostrarlo al final de la siguiente forma.

```
static ushort DisputaCarrera(ushort numCaballos) {
    Console.CursorVisible = false;
    // ...
    Console.CursorVisible = true;
    return ganador;
}
```

4. En el módulo `DisputaCarrera`, tras actualizar de forma aleatoria las posiciones que han avanzado los caballos y dibujarlas. Deberíamos realizar una pequeña pausa para evitar que la carrera se dispute muy rápido. Para ello puedes usar el siguiente método:

```
Thread.Sleep(8); // Pausará 8 milisegundos.
```

5. Puedes guardar la posición donde se encuentran los caballos, puedes cambiar el color de primer plano al dibujar mediante el enumerado `Console.ForegroundColor`. **No olvides restablecerla tras el dibujo.**

```
Console.ForegroundColor = ConsoleColor.DarkGreen;
```

6. Para que no quede rastro de los caballos mientras corren. Deberías borrar su última posición, pintando en su lugar espacios donde se encontraban.

# Memoria de trabajo


La carpeta del proyecto llevará asociado **obligatoriamente** un **repositorio local de git**. En caso de no tenerlo no se corregirá el mismo. Para ello deberemos utilizar la herramienta

Para crearlo ejecutaremos en la carpeta donde esté la solución nada más crear la misma los siguientes comandos del CLI.

```
git init
dotnet new gitignore
```

Tras acabar la implementación de un método realizaremos un commit del mismo con los siguientes comandos.

```
git add .
git commit -m "Nombre módulo finalizado"
```

 **Nota:** También será posible realizar commit intermedios durante el desarrollo de un módulo. Pero en ningún caso menos de uno.

## Criterios de calificación

La rúbrica para la calificación del trabajo será la siguiente. Para obtener la calificación de **Apto** en cada item, se deberá ajustar a las especificaciones de diseño y el vídeo de ejecución junto a este enunciado.

Items	Peso	Apto	No Apto
Sigue DEM especificado	20,0%		
Sigue la especificación de las interfaces del enunciado	20,0%		
Implementacion y código según las especificaciones	15,0%		
Interfaz y presentación por pantalla ajustada a la especificación	15,0%		
Funcionamiento correcto	15,0%		
Ha realizado un commit por módulo con la implementación propuesta.	15,0%		