Proyecto *Torneo de tenis*
Versión 2

Fecha de entrega: ver CV

En la versión 2 del proyecto ampliaremos la versión 1, añadiendo funcionalidad y usando los nuevos conceptos estudiados durante las clases (enumerados, arrays, procedimientos, parámetros por referencia). Se representará gráficamente el juego en la consola, pintando el campo, la posición de la bola y la de los tenistas en cada bola. Además, el partido se jugará al ganador de un set y se generarán estadísticas del partido que se mostrarán tras cada juego.

1. Mejoras en el juego

Las novedades en el desarrollo del partido de tenis son las siguientes:

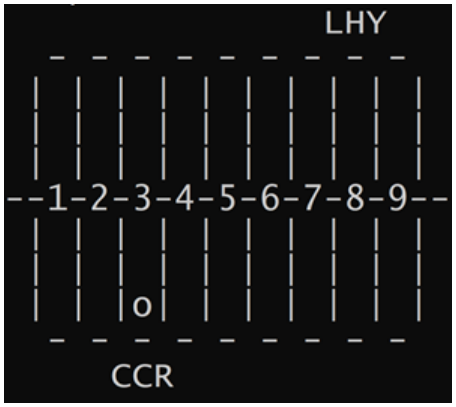
- Se juega un set completo, esto es una serie de juegos, ganando el set el jugador que consiga ganar `JUEGOS_SET` (o más juegos), con al menos dos juegos de ventaja. El valor mínimo y por defecto de la constante `JUEGOS_SET` será 3. Algunos ejemplos de marcadores ganadores serían 3-1, 2-4 o 7-5.
- Se elige de forma aleatoria el jugador que saca en el primer juego, pero en los siguientes juegos del set se van alternando el saque los jugadores. El jugador al que le toca sacar en un juego, sacará en todos los puntos de dicho juego.
- Se representa gráficamente el peloteo de cada punto, indicando la posición (calle) de cada jugador y de la bola al iniciarse el punto y tras cada golpe de la bola.
- Tras cada juego se muestran las estadísticas acumuladas sobre el desempeño de cada jugador hasta ese momento del partido: total de golpes, golpes ganadores, errores no forzados y distribución de los golpes en la pista. Estas estadísticas también se mostrarán tras el último juego del partido.

Además, como se verá en seguida, hay también bastantes novedades en la implementación.

2. Representación del juego

Representación gráfica de la pista

La representación de la pista la realizaremos con guiones, barras verticales y números:



Como sabes, la profundidad de la pista no se contempla en la dinámica del juego, así que la bola, representada por una letra **o** siempre estará en los extremos de la pista. Las siglas de cada jugador, que necesariamente deberán ser de tres letras, estarán situadas fuera de la pista, en la última posición que tuvo o tiene la bola en ese lado de la pista. Como en la versión 1, inicialmente los jugadores y la bola estarán en el medio de la pista (`ANCHO_PISTA / 2 + 1`).

En la red se muestran los números de las calles. Solamente se muestra el punto inicial y final de la bola, es decir, no se muestra la transición de la bola de un campo al contrario tras cada golpe. El largo de la pista solo se muestra a nivel estético (no influye en el *peloteo*, que se mantiene como en la versión 1); se puede ajustar con otra constante (`LARGO_PISTA`).

El marcador

El marcador ahora debe mostrar también los juegos del set que ha ganado cada jugador:

```
LHY  0 : 00 *
CCR  1 : 30
```

Tras las siglas de cada jugador se muestra el número de juegos ganados y a la derecha los puntos que tiene en el juego actual. El asterisco indica el jugador que sirve en el juego actual.

Al final del partido el marcador debe mostrar únicamente el resultado con los juegos ganados por cada jugador.

Las estadísticas

Al final de cada juego se mostrarán las estadísticas acumuladas del desempeño de cada jugador:

```
Estadisticas de LHY
   Golpes totales: 15
   Golpes ganadores: 3
   Errores no forzados: 0
   Distribucion de los golpes en la pista
      Calle      0      1      2      3      4      5      6      7      8
          %      0   26.7   6.67   6.67   6.67   6.67   6.67     40      0
```

Los golpes ganadores son aquellos que entran en la pista y el rival no alcanza.

Los errores no forzados son los envíos fuera de la pista (calles 0 y `ANCHO_PISTA+1`). Opcionalmente se podrá mostrar una representación más gráfica para la distribución de golpes (un histograma de frecuencias relativas).

3. Detalles de implementación

En esta versión vamos a usar un par de tipos enumerados:

- Tipo enumerado `tTenista` con los valores `NADIE`, `TENISTA1` y `TENISTA2`. Debe usarse tanto para saber el jugador que saca o golpea, como el ganador de un juego y el ganador del set.
- Tipo enumerado `tPuntosJuego` con los valores `NADA`, `QUINCE`, `TREINTA`, `CUARENTA` y `VENTAJA` para saber los puntos de cada jugador en un juego.

También vamos a utilizar un tipo de arrays:

- Tipo `tConteoGolpes` para arrays de enteros con un tamaño de `ANCHO_PISTA + 2` (define una constante `DIM_ARRAY_GOLPES` el tamaño de este tipo de arrays). Usaremos un array de este tipo para las estadísticas de los golpes de los jugadores.

Se han de implementar, al menos, los siguientes subprogramas:

- `string puntosAstring(tPuntosJuego puntuacion)`: recibe una puntuación de tipo `tPuntosJuego` y devuelve la cadena de texto que representa la puntuación; sustituye a la función `marcador()` de la versión 1.
- `void introducirTenista(string& iniciales, int& habilidad, int& velocidad)`: pide al usuario todos los datos del tenista: iniciales del nombre (3 letras, ni más ni menos), habilidad y velocidad (usará la función `introducirDato()`) y devuelve los tres valores.
- `tTenista actualizarMarcador(tTenista ganador_punto, tPuntosJuego &puntos1, tPuntosJuego &puntos2, int& juegos1, int& juegos2)`: dado el tenista ganador del punto actualiza las puntuaciones de los jugadores (sus puntos y sus juegos). En el caso de que un jugador haya ganado el juego devuelve el tenista ganador (`TENISTA1` o `TENISTA2`); en otro caso devuelve `NADIE`.
- `void pintarMarcador(string nombre1, string nombre2, tPuntosJuego puntos1, tPuntosJuego puntos2, int juegos1, int juegos2, tTenista servicio_para)`: muestra en la consola el marcador (en la sección anterior se explica en qué forma).
- `void pintarPeloteo(string nombre1, string nombre2, int pos_t1, int pos_t2, tTenista bola_jugador, int pos_bola)`: dados los nombres de los tenistas, sus posiciones, el jugador que tiene la bola y la posición de ésta, dibuja el campo en la consola tal como se explicó en la sección anterior. Invoca procedimientos auxiliares para pintar el tenista en el lugar adecuado y pintar la pista (pintar las líneas de fondo, pintar el fondo de la pista según la posición de la bola y pintar el resto de la pista).
- `tTenista lance(tTenista tenista_golpea, string nombre, int habilidad, tConteoGolpes golpes, int &golpes_ganados, int velocidad, int &pos_recibe, int& pos_bola)`: dado el tenista que golpea la bola, su nombre y su habilidad, la velocidad del que recibe la bola y su posición, y la posición de la bola, realiza el golpe de la bola usando `golpeBola()`, actualizando la posición de esta. Si el golpe entra en la pista, actualiza la posición del jugador que recibe la bola usando `correTenista()`. En `golpes`

debe añadirse el golpe realizado y, en el caso de ganar el punto, incrementar el valor de `golpes_ganados`. La función devuelve si hay ganador del lance: en caso de que el golpe vaya fuera establece que el ganador del punto es el tenista que recibe y si éste no llega a la bola, el que golpea, en cualquier otro caso devuelve `NADIE`.

- `tTenista jugarPunto(tTenista servicio, string nombre1, int habilidad1, int velocidad1, tConteoGolpes golpes1, int& golpes_ganados1, string nombre2, int habilidad2, int velocidad2, tConteoGolpes golpes2, int &golpes_ganados2)`: dados los datos de los tenistas y el tenista que sirve (`servicio`), conduce el juego de un punto, colocando a los jugadores y la bola en el centro del campo, pintando el campo y, mientras no haya ganador del punto, realizando un `lance()` del partido e intercambiando quién golpea la bola tras el lance. Además, llama a `pintarPeloteo()` para mostrar gráficamente el estado del partido. La función devuelve el ganador del punto (`TENISTA1` o `TENISTA2`).
- `tTenista jugarJuego(tTenista servicio, string nombre1, int habilidad1, int velocidad1, int& juegos1, tConteoGolpes golpes1, int &golpes_ganados1, string nombre2, int habilidad2, int velocidad2, int& juegos2, tConteoGolpes golpes2, int &golpes_ganados2)`: dados los datos de los tenistas y el que sirve (`servicio`), conduce un juego del set. Mientras no haya ganador del juego, muestra el marcador con `pintarMarcador()` y llama a `jugarPunto()`. Al acabar el punto, debe mostrar quién ha sido el ganador del punto y actualizar el marcador usando `actualizarMarcador()`. La función devuelve el jugador que gana el juego.

Cambios en el programa principal, es decir, en la función `main()`:

- Necesitarás añadir variables para contabilizar los juegos ganados por cada jugador y para las estadísticas del partido: los golpes de ambos jugadores (array de tipo `tConteoGolpes`) y los golpes ganadores. En esos arrays se almacenarán las calles a las que golpea cada uno de los tenistas (también los golpeos que van fuera). Los arrays para almacenar los datos de los golpeos deben inicializarse para que todos sus valores comiencen en 0. Un golpe ganador es aquel golpeo al que, entrando la bola dentro del campo rival (calles 1 hasta ANCHO_PISTA), el tenista rival no consigue llegar a la bola.

  En primer lugar, se sorteará el tenista que comienza sirviendo. A continuación, será necesario usar un bucle `while` para controlar el ganador del set. En el bucle se mostrará quién es el tenista que tiene el servicio y se llamará a la función `jugarJuego()`, mostrando el ganador de cada juego y las estadísticas acumuladas en el partido para cada jugador. Para esto último, crea un procedimiento `mostrarEstadisticas()` que muestre esa información para un jugador con el formato que se mostró anteriormente. Debes invocar el procedimiento dos veces (una por jugador). Observa que los errores no forzados podrás calcularlos con el array de golpeos contabilizando las bolas enviadas fuera de la pista. En el bucle también debe comprobar si hay ganador del set (función `hayGanadorSet()`). En caso de que no haya ganador del set, debe alternar el turno de saque en cada juego y mostrar las estadísticas del partido.

  Al acabar el set se debe mostrar el resultado final y el nombre del jugador que ha ganado.

En el modo de depuración (`MODO_DEBUG` igual a `true`) no será necesario mostrar la posición de los jugadores, ya que esa información ya se muestra al pintar el campo en la consola. Además, en este modo, se generará una semilla aleatoria idéntica en cada ejecución.

En archivos aparte puedes ver ejemplos de ejecución para los dos valores de `MODO_DEBUG`.