Práctica 1
Juego de la resta

Fecha de entrega: 18 de diciembre de 2015

1. Descripción del juego

El "Juego de la resta" es un juego para dos personas, muchas veces jugado con calculadora. Se comienza con un número cualquiera, por ejemplo el 3.467 y los jugadores, de manera alterna, le van restando números de tal manera que en cada resta *se cambie únicamente un dígito*.

Así, por ejemplo, al número 3.467 un jugador podría decidir restarle 1.000, 200 o incluso 60, pero no 12 o 90, porque haría cambiar varios dígitos distintos. Tampoco se permite restar un valor que haga que el número pase a ser negativo.

Gana el jugador que *llega a cero*.

2. El programa

En esta práctica tendrás que realizar un programa para jugar al *Juego de la resta* contra el ordenador. Al principio se mostrará una descripción del juego y se preguntará el nivel de dificultad (un valor entre 1 y 3) que se utilizará para elegir el número de dígitos del número inicial; en el nivel de dificultad fácil (1) el número tendrá 3 dígitos, en el intermedio 4, y en el difícil 5.

Luego se preguntará al jugador quién quiere que empiece, él o el ordenador. Finalmente se comenzará el juego, alternando entre el jugador y el ordenador en la elección del siguiente número a restar.

Al acabar la partida, se preguntará si se quiere jugar otra vez.

En cada pregunta al usuario (nivel de dificultad, quién empieza, siguiente número a jugar y otra partida) se forzará al usuario a introducir un valor correcto. Sin embargo, si el usuario escribe una letra cuando debería escribir un número no será necesario garantizar que el programa funcione correctamente. Tampoco será necesario mostrar correctamente las tildes o los símbolos no ingleses.

A continuación se muestra un ejemplo de ejecución. En cursiva y negrita se muestra lo introducido por el usuario.

```
Juego de la resta.

Juego para dos personas. Se comienza con un numero aleatorio.
Cada jugador resta, alternativamente un valor que haga cambiar
unicamente un digito del valor actual.

Gana el jugador que llega a 0.
Que nivel de dificultad quieres?

   1.- Facil
   2.- Medio
   3.- Dificil

   Escoge una opcion: 4
Valor incorrecto.
   Escoge una opcion: 1
Quien empieza? (T/Y): Y
El numero actual es: 698
Que numero restas? 600
El numero actual es: 98
Yo resto 2
El numero actual es: 96
Que numero restas? 70
El numero actual es: 26
Yo resto 1
El numero actual es: 25
Que numero restas? 70
No puedes restar ese numero! Prueba otra vez.
Que numero restas? 11
No puedes restar ese numero! Prueba otra vez.
Que numero restas? 20
El numero actual es: 5
Yo resto 3
El numero actual es: 2
Que numero restas? 2
ME GANASTE!!
Otra partida? (S/N) Y
No te entiendo.
Otra partida? (S/N) N
```

3. Implementación

Para realizar la práctica, divide el programa en *subrutinas* e impleméntalas *y pruébalas* por separado. Define además un tipo enumerado `Turno` cuyos valores posibles sean `Ordenador` y `Humano`.

- ✓ `unsigned int valorDigito(unsigned int numero, unsigned int digito)`: recibe un número positivo y devuelve el valor de uno de sus dígitos. Los dígitos se numeran empezando en 0, contando desde la derecha. Por ejemplo para 1234 y 0 devuelve 4, y para 567 y 2 devuelve 5.
- ✓ `bool jugadaValida(unsigned int tablero, unsigned int jugada)`: comprueba si dado el número actual de la partida ("tablero") una jugada es o no legal. Utiliza `valorDigito(…)` si lo consideras necesario.
- ✓ `unsigned int pedirJugadaHumano(unsigned int tablero)`: pregunta al jugador su siguiente jugada. Si el número proporcionado no es válido en función del "tablero" actual, insistirá en la pregunta.
- ✓ `unsigned int elegirJugadaOrdenador(unsigned int tablero)`: elige el siguiente número jugado por el ordenador a partir del valor del "tablero". El valor se elegirá aleatoriamente, pero se debe garantizar que sea correcto de acuerdo a las reglas del juego. Utiliza `valorDigito(…)` si lo consideras necesario.
- ✓ `unsigned int elegirNumeroInicial(unsigned int digitos)`: elige aleatoriamente un número con el que empezar la partida. Tendrá la cantidad de dígitos especificada en el parámetro, todos ellos diferentes de 0.
- ✓ `void mostrarInstrucciones()`: muestra por pantalla la descripción del juego.
- ✓ `bool preguntarOtraPartida()`: pregunta al jugador si quiere echar otra partida.
- ✓ `unsigned int preguntarNivelDificultad()`: pregunta al jugador el nivel de dificultad (1, 2 o 3).
- ✓ `Turno preguntarQuienEmpieza()`: pregunta al jugador quién quiere que empiece.
- ✓ `void juego(unsigned int numDigitos)`: ejecuta una partida completa usando como número inicial uno aleatorio con el número de dígitos indicado.

Para generar números aleatorios debes utilizar las funciones `rand()` y `srand()` de la biblioteca `cstdlib`. Una secuencia de números aleatorios comienza en un primer número entero que se denomina semilla (*seed*). Una vez iniciada la secuencia, la función `rand()` genera, de forma pseudoaleatoria, otro entero positivo a partir del anterior. Si quieres que la secuencia esté compuesta por números en un determinado intervalo, deberás utilizar el operador %. Por ejemplo, para obtener un entero entre `1` e `INTERVALO` se utiliza `(rand() % INTERVALO) + 1`

Lo que hace que la secuencia se comporte de forma aleatoria es la semilla. Una semilla habitual es el valor de la hora del sistema `time(NULL)`, de la biblioteca `ctime`, ya que es siempre distinta para cada ejecución.

Para establecer la semilla deberás usar: `srand(time(NULL))` *una única vez* en tu programa. Por ejemplo, el principio de la función `main()` es un buen sitio.

4. Partes opcionales

- ✓ Poder salir de una partida: si el jugador elige el número `0` como siguiente jugada, el programa considerará que se desea salir y se terminará la partida en curso, preguntando inmediatamente a continuación si se quiere volver a jugar.
- ✓ Añadir "inteligencia": mejorar el modo de elección de la siguiente jugada del ordenador para que no sea siempre aleatoria. ¿Se te ocurre alguna estrategia para que el ordenador gane *siempre que pueda*?

5. Requisitos de implementación

El código que desarrolles deberá estar bien estructurado y documentado. Pon al principio del fichero de código fuente un comentario con tu nombre.

No utilices variables globales: cada subprograma, además de los parámetros para intercambiar datos, debe declarar las variables locales que necesite usar en el código.

No pueden usarse en la resolución de la práctica elementos de C++ no vistos en clase. El programa no debe utilizar instrucciones de salto no estructuradas como `exit`, `break` (salvo, en su caso, en las cláusulas de la instrucción `switch`) y `return` (salvo en la última instrucción de las funciones que devuelven un valor).

6. Entrega de la práctica

La práctica se entregará a través del Campus Virtual a través de la tarea **Entrega de la Práctica 1** que permitirá subir el archivo con el código fuente.

Fin del plazo de entrega: **18 de diciembre a las 23:55**.