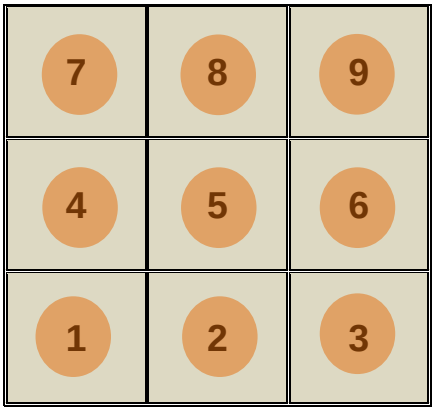
Práctica 1
Pasa la calculadora

Fecha de entrega: 7 de diciembre de 2014

En matematicasdivertidas.com se propone el siguiente juego[1]:

> Dos jugadores **A** y **B** juegan de la manera siguiente: **A** enciende la calculadora y pulsa un dígito, y a continuación pulsa la tecla "**+**". Pasa la calculadora a **B**, que pulsa un dígito que ha de estar en la misma fila o columna que el último dígito pulsado por **A** y ser diferente de éste; a continuación pulsa "**+**" y le devuelve la calculadora a **A**, que repite la operación y así sucesivamente. Pierde el juego el primer jugador que alcanza o supera la suma 31.

La disposición de las teclas de los dígitos en filas y columnas es la siguiente:



1. Descripción de la práctica

En esta práctica tienes que ir desarrollando de manera incremental un programa que permita jugar a *Pasa la calculadora* contra el propio programa y llevar un informe sobre el uso del programa. En cada partida, el programa decidirá quién empieza a jugar de forma aleatoria, y en cada turno leerá el dígito pulsado por el jugador (en caso de que sea el turno del ordenador, el dígito lo generará el propio programa), comprobará que el digito cumple las condiciones de las reglas del juego, y mostrará la suma acumulada. Si la suma alcanza o supera 31 el jugador que acaba de jugar habrá perdido.

[1] http://www.matematicasdivertidas.com/Juegos%20con%20Calculadora/juegos%20con%20calculadora.html#pasa2

1.1. Versión 1 de la práctica (Pasa la calculadora, una partida)

Descripción

En esta primera versión de la práctica solo se permitirá jugar una partida. El programa comienza con un saludo y pidiendo el nombre al usuario. A continuación, el programa decide al azar quien empieza. Cuando le toque el turno al usuario el programa le solicitará que introduzca un dígito, comprobara que el dígito introducido es correcto y mostrará por pantalla la suma. Si el usuario desea abandonar la partida deberá introducir el dígito 0. En el momento en que la suma llegue a 31 o lo supere el programa comunica al usuario quién ha ganado y se despide.

Ejemplo de ejecución de esta primera versión de la práctica:

```
¡Bienvenido a pasa la calculadora!

¿Cómo te llamas? Ana
Hola Ana
Empiezas tú

    7  8 9
    4  5 6
    1  2 3
Introduce un dígito (0 para abandonar): 2
Suma: 2

Yo pulso: 8
Suma: 10

    7  8 9
    4  5 6
    1  2 3
Introduce un dígito (0 para abandonar): 2
Suma: 12

Yo pulso: 8
Suma: 20

    7  8 9
    4  5 6
    1  2 3
Introduce un dígito (0 para abandonar): 8
Error: tiene que ser distinto de 8 y estar en la misma fila o columna
    7  8 9
    4  5 6
    1  2 3
Introduce un dígito (0 para abandonar): 4
Error: tiene que ser distinto de 8 y estar en la misma fila o columna
    7  8 9
    4  5 6
    1  2 3
Introduce un dígito (0 para abandonar): 9
Suma: 29

Yo pulso: 3
Suma: 32

¡Enhorabuena has ganado!
Hasta la próxima Ana (pulsa una tecla)
```

Detalles de implementación

Implementa al menos las siguientes funciones:

- ✓ **`int pasaCalculadora()`**: Conduce el desarrollo del juego y devuelve el ganador (0→Nadie, 1→Autómata, 2→Jugador). Si se abandona devuelve 0 (Nadie). Utiliza, directa o indirectamente, las funciones que siguen.
- ✓ **`int quienEmpieza()`**: Decide aleatoriamente quien empieza (1→Autómata, 2→Jugador).
  - Más abajo explicamos cómo generar números aleatorios.
- ✓ **`bool mismaFila(int ultimo, int nuevo)`**: Devuelve true si nuevo está en la misma fila que ultimo; en caso contrario devuelve false.
  - Observa que la fila que ocupa un dígito, numeradas desde 0, corresponde con (dígito - 1) / 3.
- ✓ **`bool mismaColumna(int ultimo, int nuevo)`**: Devuelve true si nuevo está en la misma columna que ultimo; en caso contrario devuelve false.
  - Observa que la columna que ocupa un dígito, numeradas desde 0, corresponde con (dígito-1) % 3.
- ✓ **`bool digitoValido(int ultimo, int nuevo)`**: Devuelve true si nuevo cumple las reglas del juego con respecto a ultimo; en caso contrario devuelve false.
- ✓ **`int digitoAleatorio()`**: Devuelve un dígito aleatorio.
  - Más abajo explicamos cómo generar números aleatorios.
- ✓ **`int digitoAutomata(int ultimo)`**: Devuelve un dígito generado aleatoriamente que cumpla las reglas del juego con respecto a ultimo. Por ejemplo digitoAutomata(2) puede devolver 1,3,5 u 8.
- ✓ **`int digitoPersona()`**:Pide un dígito al jugador. Devolverá un valor válido (entre 0 y 9). Para cada valor no válido, mostrará un error y solicitará de nuevo el dígito.
- ✓ **`int digitoPersona(int ultimo)`**: Pide un dígito al jugador mostrando el teclado. Sólo devolverá un valor que cumpla las reglas del juego o 0. Para cada valor no válido, mostrará un error y volverá a solicitar el dígito.

Generación de números aleatorios

Para generar números aleatorios utiliza las funciones `rand()` y `srand(semilla)` de la biblioteca `cstdlib`.

Una secuencia de números aleatorios comienza en un primer número entero que se denomina semilla. Para establecer la semilla el programa deberá

invocar a la función `srand` con el argumento deseado. 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 pondremos:

```
srand(time(NULL))
```

La inicialización de la semilla se hace una sola vez nada más comenzar el programa.

Una vez establecida la semilla, 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, tendrás que utilizar el operador % de la siguiente manera:

- Para obtener un entero entre 1 y M: `(rand() % M) + 1`
- Para obtener un entero entre 0 y M: `rand() % (M + 1)`

1.2. Versión 2 de la práctica (Pasa la calculadora con menú)

Descripción

Añade a la versión anterior de la práctica un menú de opciones que permita elegir entre jugar una partida, mostrar la información acerca del programa, o salir del programa. El programa no terminará hasta que se seleccione salir.

Ejemplo de ejecución:

```
¡Bienvenido a pasa la calculadora!
¿Cómo te llamas? Ana
Hola Ana
Selecciona una opción:
      1 - Jugar
      2 - Acerca de
      0 - Salir
Opción: 4
Opción incorrecta. Opción:1

...

Selecciona una opción:
      1 - Jugar
      2 - Acerca de
      0 - Salir
Opción: 0

Hasta la próxima Ana (pulsa una tecla)
```

Ahora el juego se comportará de la siguiente manera según la opción elegida:

1. Jugar: Se jugará una partida tal y cómo se hacía en la versión 1 de la práctica.

2. Acerca de: Se mostrarán los créditos e instrucciones que se encuentren en el archivo de texto versionPC.txt. Ejemplo de ejecución:

```
Selecciona una opción:
      1 - Jugar
      2 - Acerca de
      0 - Salir
Opción: 2

------------------------------------------------------
                Acerca de Pasa la calculadora

Práctica 1 Versión 3.1 (20/10/2014)

Fundamentos de Programación
Facultad de Informática
Universidad Complutense de Madrid

Autores:
       nombre y apellidos (grupo)
       nombre y apellidos (grupo)

Instrucciones: ...
------------------------------------------------------

Selecciona una opción:
      1 - Jugar
      2 - Acerca de
      0 - Salir
Opción: _
```

   Crea tu propio archivo versionPC.txt con los nombres de los autores e incluye unas sencillas instrucciones del juego. El archivo debe de terminar con el centinela XXX.

3. Salir: El programa termina.

Detalles de implementación

Añade a la versión 1 al menos las siguientes funciones:

- ✓ **`int menu()`**: Muestra el menú, pide la opción y la devuelve como resultado. Sólo devolverá una opción válida. Para cada valor no válido, mostrará un error y volverá a pedir una opción válida.
- ✓ **`bool mostrar(string nombArch)`**: Muestra en la consola el contenido del archivo de texto *nombArch*. Si el archivo no se encuentra, devuelve false, en otro caso true.
  - ➢ Recuerda que el archivo debe de terminar con el centinela XXX para que el programa sepa cuando parar de leer del archivo.

1.3. Versión 3 (Pasa la calculadora con menú e informe)

Descripción

Ahora se quiere tener información sobre el uso del programa, para lo cual se guardarán en el archivo informePC.txt los siguientes datos:

- Número de veces que se ha utilizado el programa.
- Número total de partidas jugadas (incluidos los abandonos).
- Número total de partidas ganadas por el programa.
- Número total de abandonos.

Cada uno de estos datos se guardará en una línea, y en el orden indicado. Ejemplo de archivo informePC.txt:

```
8
4
0
2
```

Para mantener el informe, al terminar el programa se actualizará el archivo sumando a los datos que ya había los datos de la última ejecución.

Detalles de implementación

Añade a la versión 2 del programa al menos la siguiente función:

- ✓ **`void actInforme(int jugadas, int ganadas, int abandonos)`**: Actualiza la información del archivo informePC.txt con los tres argumentos. Si el archivo ya estaba creado, suma la información de los tres argumentos a la que ya había en el archivo, si el archivo no estaba creado, lo crea con la información de los tres argumentos.

1.4. Versión 4 (Opcional)

- Haz a tu programa más inteligente, mejorando la función `digitoAutomata`.
- Añade al menú la opción de seleccionar el nivel de juego. Se podrá elegir el nivel 1 (el programa juega con la función `digitoAutomata` descrita en la versión 1) o el nivel 2 (el programa juega con una función de `digitoAutomata` mejorada, más inteligente).

2. Requisitos de implementación

No olvides declarar las constantes necesarias para hacer tu código reutilizable y fácil de modificar.

No olvides incluir los prototipos de tus funciones.

No utilices variables globales: cada función, además de los parámetros con los que se le pasa información en las llamadas, debe declarar localmente las variables que requiera.

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

3. Entrega de la práctica

La práctica se entregará a través del Campus Virtual.

Se habilitará una nueva tarea: **Entrega de la Práctica 1** que permitirá subir un fichero comprimido que debe incluir el código fuente de la última versión (**practica1.cpp**) y el archivo **versionPC.txt**.

El fichero subido deberá tener el siguiente nombre: **Practica1GrupoXXX**, siendo XXX el número de grupo. Uno sólo de los miembros del grupo (no los dos) será el encargado de subir la práctica.

La práctica debe funcionar usando Visual Studio, que es la herramienta vista en clase. La práctica subida debe compilar y cumplir con la funcionalidad descrita en el enunciado.

No se corregirá ninguna práctica que no cumpla estos requisitos.