PRÁCTICA Nº 3: 2 sesiones **(S3: 6, 12, 13, 14 y 15 de Abril)**
(S4: 11, 19, 20, 21 y 22 de Abril)

JUEGO DE SOLITARIO DE CARTAS ESPAÑOLAS

0. OBJETIVOS

El objetivo de esta práctica es la implementación del TAD Pila y su uso en la simulación de un solitario de cartas españolas.

1. ESPECIFICACIÓN DEL PROBLEMA

El juego requiere el manejo de diversas pilas que representarán los diferentes montones de cartas implicados en el juego y consiste en mover una carta de una pila a otra hasta no poder efectuar movimientos válidos o tener todas las cartas del mismo palo amontonadas en su correspondiente pila.

En el juego se empleará una baraja española con 48 cartas (4 palos y 12 cartas por palo), estando cada carta representada por un carácter que indique el palo ('O', 'C', 'E', 'B') y un número. Antes de iniciar el juego, el programa debe generar la baraja (en forma de pila) y proceder a barajar las cartas para que no estén ordenadas. A continuación, el estado inicial del juego requerirá formar los siguientes montones de cartas (pilas), además de la propia baraja:

- 4 montones donde se irán amontonando las cartas de cada uno de los palos: `Oros`, `Copas`, `Espadas` y `Bastos`.
- 7 montones sobre las mesa: `mazo1`, `mazo2`, `mazo3`, `mazo4`, `mazo5`, `mazo6`, `mazo7`.
- 1 montón `Auxiliar`.

El juego se inicia repartiendo cartas de la baraja de la siguiente forma:

- 5 cartas en la pila `Auxiliar`
- 1 carta en cada una de las pilas de la mesa (`mazo1.. mazo7`)

A partir de ahí el programa debe permitir al jugador realizar movimientos de cartas entre las distintas pilas, teniendo en cuenta que en cada movimiento se mueve sólo una carta y que se debe indicar la pila de donde se quita (cima de la pila) y la pila a la que se lleva.

Los movimientos posibles son:

Hacia las pilas de los palos (`Oros`, `Copas`, `Espadas` y `Bastos`):

- Cuando se encuentran vacías solo se puede mover el "as" (B1, C1, E1, O1) del palo correspondiente desde cualquier pila excepto de la `Baraja`.
- Cuando contienen alguna carta solo se puede colocar otra del mismo palo y una unidad superior desde cualquier pila excepto de la `Baraja`.

Hacia las pilas de trabajo (`mazo1.. mazo7`):

- Cuando estan vacías se puede mover cualquier carta desde cualquier pila excepto de la `Baraja`.
- Cuando contienen alguna carta solo se puede colocar otra de distinto palo y una unidad inferior procedente de cualquier pila excepto de la `Baraja`.

Si el jugador intenta realizar un movimiento no válido, se debe impedir el movimiento y avisar al jugador.

El jugador repetirá este tipo de movimientos hasta que considere que con las cartas que tiene en la mesa ya no se pueden hacer más movimientos entre pilas. En ese caso, puede extraer de nuevo 5 cartas de la `Baraja` (las 5 superiores) y ponerlas en la pila `Auxiliar`. Si quedan menos de 5 cartas en la `Baraja` se deben pasar a `Auxiliar` todas las que queden.

Si la `Baraja` se vacía se deben pasar todas las cartas de la pila `Auxiliar` a la `Baraja` y se continúa el juego.

Al iniciar el juego y cuando se produzca algún movimiento se debe visualizar en pantalla el contenido de todas las pilas que intervienen en el juego. Para ello, se debe incluir un método en la clase `Pila` que visualice su contenido siguiendo en siguiente criterio: si la pila está vacía debe mostrar el símbolo `X`, en caso contrario se debe mostrar el contenido entre paréntesis y de izquierda a derecha, siendo la cima de la pila el elemento más a la derecha. Por ejemplo, una situación hipotética del juego podría ser:

```
Tapete
================================
Oros: (O1,O2,O3)
Copas: (C1)
Espadas: X
Bastos: (B1,B2,B3,B4)
================================
Mazo1: (O12,B11)
Mazo2: (C5,E4)
Mazo3: X
Mazo4: X
Mazo5: (E10,O9,C8)
Mazo6: X
Mazo7: (C6)
================================
Auxiliar: (O5,O6,E1,B5)
Baraja: (las restantes cartas separadas por comas)
```

En esta situación:

La pila del palo bastos, contiene el as, el 2, el 3, el 4 de bastos,
La pila del palo copas contiene el as de copas,
La de espadas está vacía,
La de oros contiene el as, el 2 y el 3 de oros
El mazo 1 contiene el Rey de oros y el caballo de bastos.
El mazo 2 contiene el 5 de copas y el 4 de espadas.
Los mazos 3, 4 y 6 estan vacíos.
El mazo 5 contiene las cartas sota de espadas, 9 de oros y 8 de copas.
El mazo 7 contiene el 6 de copas.
La pila auxiliar contiene el 5 de oros, el 6 de oros, el as de espadas y el 5 de bastos.
La baraja contiene las restantes cartas.

2. REALIZACIÓN DE LA PRÁCTICA

En esta tercera práctica, se debe implementar una clase en C++ que permita representar el TAD pila, asumiendo la interfaz estándar vista en teoría, y una implementación estática de la pila:

```
class Pila
{
      public:
            Pila ();
            bool Apilar (Carta x);
            bool Desapilar ();
            bool CimaPila (Carta & x);
            bool PilaVacia ();
            void MostrarPila ();
      private:
            typedef Carta Vector[MAX];
            Vector datos;
            int cima;
};
```

Como la pila almacena Cartas es preciso especificar este tipo de datos, para ello, es necesario construir la clase `Carta` que responde al siguiente interfaz:

```
class Carta
{
      public:
            Carta (); //constructor
            bool ponerCarta (char p, int n); //fija los datos de la carta
            //true = los datos son correctos, false = los datos son incorrectos
            char paloCarta (); //devuelve el palo de la carta
            int numeroCarta (); //devuelve el numero de la carta
            void MostrarCarta (); //muestra por pantalla la carta
      private:
            char palo; //'O'=oros, 'C'=copas, 'E'=espadas, 'B'=bastos
            int numero; //1..12
};
```

Se realizará un programa completo llamado `pr03##.cpp`, donde se utilice las clases anteriores, para realizar el juego descrito (`##` indica el número asignado a la pareja en cada grupo de prácticas).

Se crearán, al menos, las funciones necesarias para:

Crear una baraja nueva de 48 cartas: `BarajaNueva`.

Barajar las cartas: `Barajar`. Este proceso se puede realizar de diversas maneras. Un método sencillo y bastante aleatorio puede ser el siguiente:

1.- Volcar los datos de la pila en un vector auxiliar `v` de cartas (la pila queda vacía)

2.- para `fin` = 47 hasta 1 hacer

2.1.- `pos` = generar un nº aleatorio entre 0 y `fin`

2.2.- intercambiar los elementos `v`[`pos`] y `v`[`fin`]

3.- Volcar los datos de `v` de nuevo en la pila

Para facilitar la realización de los movimientos de cartas se crearán dos funciones más:

`MenuOrigen`: que presentará las pilas desde donde se pueden mover las cartas y pedirá que se escoja una de ellas.

`MenuDestino`: que a partir de la opción escogida en `MenuOrigen`, mostrará una lista de pilas posibles de destino y pedirá que se escoja una de ellas.

`Validar`: Se creará otra función que, una vez escogida la pila de destino, validará que la carta de la cima de la pila de origen sigue las reglas del juego para colocarse en la pila destino. Devolverá un valor verdadero o falso según sea el caso.

Finalmente, para mostrar el contenido de todas las pilas que intervienen en el juego se creará una función `MuestraTapete` que realice dicha tarea.

Como el programa debe gestionar 13 pilas, parece conveniente crear un vector de 13 pilas. P[0] será la `Baraja`, P[1] a P[7] las pilas de trabajo `mazo1` a `mazo7` respectivamente, P[8] a P[11] las pilas de los palos (oros, copas, espadas y bastos) y P[12] la pila `auxiliar`.

Con todo esto podemos crear el motor del juego, la función "`jugar`", que, tomando como entrada la pila de la `Baraja` ya barajada, muestre el interfaz de usuario para visualizar la situación de la partida en cada movimiento y permitir realizar los movimientos deseados.

El programa principal (`main`) comenzará presentando las reglas del juego, creando una baraja nueva, la barajará y llamará al motor del juego (`jugar`).

3. ENTREGA DE PROGRAMAS

Cinco días después de realizada la sesión de prácticas se entregará al profesor la siguiente información:

1) Archivo con el programa (**`pr03##.cpp`**)
2) Archivo con el interfaz de la clase *`Carta`* (**`carta##.h`**)
3) Archivo con la implementación de la clase *`Carta`* (**`carta##.cpp`**)
4) Archivo con el interfaz de la clase *`Pila`* (**`pila##.h`**)
5) Archivo con la implementación de la clase *`Pila`* (**`pila##.cpp`**)

dónde `##` es el número asignado a la pareja (01, 02,...) en cada grupo de prácticas.

Al comenzar la siguiente sesión de prácticas se entregará al profesor el formulario con la documentación definitiva de la práctica reflejando todo el trabajo realizado realmente en la práctica.

Nota Muy Importante

Antes de poder empezar a realizar cualquiera de las prácticas es necesario presentar las hojas de especificación de programas (formulario de documentación de programas) con las tareas que se van a realizar en la práctica, explicando brevemente como se van a solucionar los problemas que se plantean.

ENTREGA DE PROGRAMAS:

Los días 16, 24, 25, 26 y 27 de Abril correspondientes con las fechas de realización de la 2ª sesión de la práctica 11, 19, 20, 21 y 22 de Abril

ENTREGA DE FORMULARIOS DEFINITIVOS DE DOCUMENTACIÓN:

Los días 18, 26, 27, 28 y 29 de Abril correspondientes con las fechas de realización de la 2ª sesión de la práctica 11, 19, 20, 21 y 22 de Abril**.**