VNIVERSITAT DE VALÈNCIA

Laboratorio de PROGRAMACIÓN
Grado en Ingeniería Multimedia (1º)
Curso 2021-22

Práctica Nº 6: Utilización de pilas
Periodo de realización: Semana del **02/05/2022** al **06/05/2022**

Objetivos

- Definir e implementar clases en C++.
- Construir una aplicación con múltiples archivos y clases.
- Aplicar el tipo Pila a la resolución de un problema.

Material a entregar

- **(Repositorio) Proyecto compilable con el código fuente necesario para la práctica.**
- **(Tarea AV) Documentación del código generada por Doxygen.**

Introducción

En esta práctica se va a aplicar el contenedor pila a la resolución de un problema. Para ello, se propone la programación de un juego de cartas sencillo. Estas son sus reglas:

1. Se juega con una baraja española, que posee cuatro palos (Oros, Copas, Espadas y Bastos) con 12 cartas cada uno, numeradas de 1 a 12.
2. En el juego pueden participar dos, tres o cuatro jugadores, de manera que, las cartas de la baraja se repartirán en mazos de igual tamaño entre todos los jugadores (2, 3 o 4 mazos).
3. Las cartas están ocultas hasta que se juegan individualmente.
4. En una mano del juego, cada jugador saca una carta (la primera de su mazo de juego) y gana la mano el jugador con “mejor” carta. El jugador que ha ganado la mano se guarda todas las cartas jugadas por los jugadores en esa mano en un mazo de cartas ganadas (diferente del mazo de juego). Las cartas ganadas ya no se pueden usar para jugar. El mazo de cartas ganadas por cada jugador establecerá su puntuación al finalizar la partida.
5. La partida termina cuando los jugadores ya no tienen cartas para jugar (sus mazos de juego están vacíos).
6. Para asignar un ganador de la partida se cuentan los puntos acumulados por cada jugador, ganando el/los jugador/es con mayor número de puntos (puede haber empates).

- Criterio para establecer la “mejor” carta:

  Para poder comparar las cartas que se juegan en cada mano se establece el criterio de carta más alta. Siempre gana la carta con mayor número. No obstante, para desempatar cuando haya varias cartas con el número más alto, se establece también un criterio de orden entre los palos de la baraja. Se considerará que Oros > Copas > Espadas > Bastos y, por tanto, en caso de empate a puntos, se considerará "mejor carta" la que tenga el palo mayor (Oros gana siempre los desempates, Copas gana a Espadas y Bastos, Espadas gana a Bastos y Bastos pierde siempre los desempates).

  Ejemplo:

  Hay 3 jugadores en la partida, que en una mano del juego sacan las siguientes cartas:

  Jugador 1: *7/Copas*

  Jugador 2: *3/Oros*

  Jugador 3: *7/Espadas*

  Los jugadores 1 y 3 empatan con el número más alto (7), pero la mano la gana el jugador 1, puesto que Copas gana a Espadas.

- Regla para contar los puntos acumulados por cada jugador y establecer el ganador:

  Al final del juego se revisarán las cartas contenidas en el mazo de cartas ganadas por cada jugador y se aplicarán 4 criterios de puntuación, con la siguiente definición y valoración:

  1. Cartas ganadas: El jugador que haya ganado más cartas obtendrá 3 puntos. En caso de empate, cada jugador implicado recibirá 2 puntos.
  2. Suma de números de las cartas: El jugador para el que la suma de los números de las cartas ganadas sea mayor obtendrá 2 puntos. En caso de empate, cada jugador implicado recibirá 1 punto.
  3. Oros: El jugador que haya ganado más cartas del palo Oros obtendrá 1 punto. En caso de empate, cada jugador implicado recibirá 1 punto.
  4. Cartas especiales: El jugador que haya ganado algunas de las cartas consideradas como especiales obtendrá 1 punto por cada una de ellas. Las cartas especiales son: 1 de Copas, 7 de Oros, 10 de Espadas y 12 de Bastos.

Programación del juego

Una vez establecidos los criterios lógicos del juego, se van a establecer los requisitos para su programación en *C++*:

1. Los jugadores participantes en el juego no serán personas. Una vez definido el número de jugadores (2, 3 o 4), será el programa el que los represente y el juego será completamente automático, mostrando la información adecuada para seguir cada mano del juego y el resultado final de la partida. El seguimiento de una partida por pantalla tendrá el siguiente formato:

```
==> Inicio de la partida. Jugadores: 3
*** Mano 1 ***
Jugador 1: 3/Copas
Jugador 2: 5/Bastos
Jugador 3: 3/Espadas
[Gana el Jugador 2]

*** Mano 2 ***
Jugador 1: 12/Oros
Jugador 2: 7/Oros
Jugador 3: 2/Copas
[Gana el Jugador 1]
…
Resultado Final
================
Jugador 1: 2 puntos
Jugador 2: 5 puntos
Jugador 3: 3 puntos

==> Ganador: Jugador 2 <==
```

2. Todos los mazos de cartas necesarios para el juego se deberán representar mediante una Pila de Cartas. Por lo tanto, cada jugador deberá manejar dos pilas: una para las cartas que se juegan, que va disminuyendo en tamaño, y otra para las cartas ganadas, que va creciendo.
3. Como consecuencia del punto anterior será necesario especificar, declarar e implementar dos clases, Pila y Carta. Para la clase Pila se debe utilizar la implementación proporcionada en el material de la práctica, la cual consiste en una plantilla de clases (*template*) de este tipo de contenedor. La especificación de la clase Carta se describe más adelante en este mismo documento.
4. La programación del juego responderá al siguiente esquema algorítmico:
    1. Preguntar y establecer el número de jugadores.
    2. Barajar todas las cartas y repartirlas en tantos mazos como jugadores, todos de igual tamaño.
    3. Mientras haya cartas en los mazos de los jugadores hacer:
        a. Sacar carta de cada jugador
        b. Establecer carta y jugador ganador
        c. Las cartas jugadas se añaden al mazo de cartas ganadas del jugador ganador
    4. Fin_mientras
    5. Contar puntuación de cada jugador
    6. Establecer jugador ganador de la partida

Ejercicios

SCV: Debes trabajar en la carpeta "Pr6" del repositorio que ya tienes creado. Descarga allí los archivos "`Pr6_Carta.h`", "`Pr6_Carta.cpp`", "`Pr6_Pila.h`" y "`Pr6_Prueba.cpp`" que ya tienes disponible en el Aula Virtual. Haz un primer *commit* ("Pr.6: Versión inicial") y *push* con este material inicial.

Tarea 1

Los archivos `Pr6_Carta.h` y `Pr6_Carta.cpp` implementan la siguiente especificación formal de la clase Carta (excepto las operaciones marcadas en negrita):

TAD Carta

Dominio: Representa una carta de la baraja española, caracterizada por la siguiente información:

- número: el número de la carta entre 1 y 12.
- palo: el palo de la carta (Oros, Copas, Espadas y Bastos).

Operaciones:

- *CrearCarta () -> Carta*
- *CrearCarta (Entero, Carácter) -> Carta*
- *EstableceNumero (Carta, Entero) -> Carta*
- *EstablecePalo (Carta, Carácter) -> Carta*
- *DevuelveNumero (Carta) -> Entero*
- *DevuelvePalo (Carta) -> Carácter*
- *Mostrar (Carta)*
- ***EsMayorQue (Carta, Carta) -> Booleano***
- ***EsMenorQue (Carta, Carta) -> Booleano***

Axiomas:

Sea c, c1 $\in$ Carta, n $\in$ Entero, p $\in$ Caracter

CrearCarta ()	*Crea una carta con valores por defecto (1 de Oros).*
CrearCarta (n, p)	*Crea una carta con el número n y el palo p ('O' – Oros, 'C' – Copas, 'E' – Espadas, 'B' – Bastos).*
EstableceNumero (c, n)	*Establece el número de la carta.*
EstablecePalo (c, p)	*Establece el palo de la carta.*
DevuelveNumero (c)	*Devuelve el número de la carta.*
DevuelvePalo (c)	*Devuelve el palo de la carta.*
Mostrar (c)	*Muestra por pantalla el valor de la carta con el formato "número/palo" (por ejemplo, "5/Copas").*
EsMayorQue (c, c1)	***Devuelve cierto si la carta c es más "alta" (según el criterio del juego) que la carta c1, y falso en caso contrario.***
EsMenorQue (c, c1)	***Devuelve cierto si la carta c1 es más "alta" (según el criterio del juego) que la carta c, y falso en caso contrario.***

Implementa las operaciones faltantes *EsMayorQue y EsMenorQue* mediante la sobrecarga de los operadores "mayor que" (>) y "menor que" (<), respectivamente.

SCV: Registra los cambios de "`Pr6_Carta.h`" y "`Pr6_Carta.cpp`" (*commit*) con el mensaje "Pr.6: Completada Tarea 1".

Tarea 2

Crea un proyecto en Dev-C++ ("`Pr6_Prueba.dev`") para probar la clase Carta implementada. Añade los archivos que has modificado en la tarea anterior. Añade también el archivo "`Pr6_Prueba.cpp`" del material de la práctica.

El fichero "`Pr6_Prueba.cpp`" contiene un programa de prueba que maneja una baraja de cartas. A partir de este programa:

- Implementa la función `IniciarBaraja`. Se trata de colocar las 48 cartas de la baraja española en el vector de cartas (baraja) en el siguiente orden: del 1 al 12 de Oros, del 1 al 12 de Copas, del 1 al 12 de Espadas y del 1 al 12 de Bastos.
- Completa el programa para que extraiga dos cartas aleatorias de la baraja, imprima por pantalla las cartas extraídas, las compare (utilizando las operaciones implementadas en la tarea anterior), y muestre cuál es la carta más alta. Comprueba que se calcula correctamente la mejor carta. El programa debe repetir la secuencia anterior hasta que el usuario indique que desea terminar.

SCV: Registra los cambios de "`Pr6_Prueba.dev`" y "`Pr6_Prueba.cpp`" (*commit*) con el mensaje "Pr.6: Completada Tarea 2".

Tarea 3

Ahora vamos a construir el juego propiamente dicho. Reutiliza los conceptos y código de interés del programa de la tarea anterior.

Crea un nuevo proyecto en Dev-C++ ("`Pr6.dev`"). Añade los archivos de la clase Carta y el archivo "`Pr6_Pila.h`". Añade también un nuevo archivo "`Pr6_JuegoCartas.cpp`" al proyecto.

El programa del juego ("`Pr6_JuegoCartas.cpp`") debe utilizar la misma estructura de datos para la baraja (vector de cartas) del programa de pruebas, así como las funciones `InicializarBaraja` y `Barajar`.

A continuación:

- Define las estructuras que consideres adecuadas para definir a los participantes del juego. Recuerda que cada jugador debe manejar dos mazos de cartas y su puntuación obtenida en el juego.
- Implementa la función `RepartirCartas`. Esta función debe repartir todas las cartas de la baraja entre los participantes del juego. Para ello, apila cada una de las cartas repartidas en el mazo correspondiente de cada jugador.
- Crea un programa que, en primer lugar, pregunte el número de participantes del juego (2, 3 o 4 jugadores). Después, que inicialice y baraje la baraja de cartas (como el programa de pruebas). Por último, debe repartir las cartas entre los participantes del juego.

Utiliza el operador de salida (<<) de la clase Pila para mostrar el contenido de los mazos de los jugadores después de repartir las cartas, y comprueba que el reparto se hace correctamente para cualquier número de jugadores.

SCV: Registra los cambios de "`Pr6.dev`" y "`Pr6_JuegoCartas.cpp`" (*commit*) con el mensaje "Pr.6: Completada Tarea 3".

Tarea 4

Implementa el bucle principal del juego para jugar todas las manos de la partida. Según el esquema algorítmico propuesto en la introducción, dicho bucle debe sacar una carta del mazo de cada jugador, calcular la carta más alta, y establecer el jugador ganador de la mano. Seguidamente, se deben añadir las cartas jugadas al mazo de las cartas ganadas del jugador ganador.

El programa resultante debe preguntar el número de participantes del juego, inicializar y barajar la baraja de cartas, repartir las cartas entre los jugadores, y realizar el bucle principal del juego. Comprueba que el programa sigue el funcionamiento descrito del juego.

Utiliza el operador de salida (<<) de la clase Pila para mostrar el contenido de los mazos de los jugadores, y comprobar el funcionamiento del juego en cualquier momento: en el inicio (después de repartir las cartas), después de cada mano, o al final de la partida (mostrando el mazo de las cartas ganadas). Ten en cuenta que, la versión final del juego no debe mostrar esta información de depuración.

SCV: Registra los cambios de "`Pr6_JuegoCartas.cpp`" (*commit*) con el mensaje "Pr.6: Completada Tarea 4".

Tarea 5

Por último, completa la programación del juego:

- Implementa la función `Puntuacion`, que calcula los puntos obtenidos por cada jugador al finalizar el juego, de acuerdo con los criterios especificados al inicio de este documento. En este punto, lo más adecuado será definir nuevas funciones para calcular las puntuaciones de cada uno de los 4 criterios establecidos en el juego.
- Implementa la función `MostrarResultados`, que muestra los puntos de cada jugador e identifica al ganador. Se debe tener en cuenta que los empates son posibles y que puede haber más de un ganador.

Utiliza estas funciones para calcular las puntuaciones finales y mostrar el jugador ganador al terminar la partida.

SCV: Registra los últimos cambios de "`Pr6_JuegoCartas.cpp`" (*commit*) con el mensaje "Pr.6: Versión final, completada Tarea 5". Haz *push*.

Tarea 6: Generación de documentación

El código realizado deberá estar correctamente documentado con Doxygen, de acuerdo con la guía de estilo. Una vez realizada la versión final del programa, se deberá ejecutar Doxygen para generar toda

la documentación definitiva. La documentación deberá ser generada exclusivamente en formato en *html* (utilizar como referencia, convenientemente personalizado para esta práctica, el archivo de configuración de Doxygen disponible en el Aula Virtual). Solo la documentación deberá subirse a Aula Virtual (como archivo comprimido). **El programa NO se debe subir al AV, ya que será evaluado directamente desde el repositorio de control de versiones**.

La documentación generada con Doxygen NO debe subirse al repositorio de control de versiones Se debe mantener exclusivamente en el ordenador personal.