

Trabajo Práctico nº 6

Polimorfismo - Relaciones entre clases en Visual Studio C++

Trabajo Preliminar

Crear una solución llamada **TrabajoPractico-6** que contendrá los proyectos: **Ejercicio-0, Ejercicio-1, Ejercicio-2, Ejercicio-3.**

Ejercicio 0

Crear las siguientes clases:

```
class FiguraGeometrica{
public:
    void dibujar(){ cout<<"no se que soy!"<<endl;}
};
class Cuadrado: public FiguraGeometrica{
public:
    void dibujar(){ cout<<"soy un cuadrado"<<endl;}
};
class Triangulo: public FiguraGeometrica{
public:
    void dibujar(){ cout<<"soy un triangulo"<<endl;}
};
```

Crear un programa tal que cree dos variables de `FiguraGeometrica` y reciba un objeto de cada clase y las muestre utilizando el método `dibujar` tal que la salida sea:

La salida será:

```
no se que soy!
no se que soy!
```

Colocar `virtual` delante del método `dibujar` en la clase base y volver a compilar, linkear y correr

La salida será:

```
soy un cuadrado
soy un triangulo
```

Ejercicio 1

Reemplazar el método `dibujar` por el siguiente

```
virtual void dibujar()=0; // es un metodo virtual puro
```

Crear la siguiente clase:

```
class Circulo: public FiguraGeometrica{
public:

}
```

Al compilar da un error explicar porque y como se soluciona

Crear un programa tal que cree un array de **FiguraGeometrica** y lo cargue con tres objetos uno de cada clase y las muestre utilizando el método `dibujar` tal que la salida sea:

```
soy un cuadrado
soy un triangulo
soy un circulo
```

Ejercicio 2

Crear las clases Personaje, Mago, Caballero y Arquero según los siguientes diagramas:

La clase **Mago** hereda de **Personaje** y posea un atributo **int _tipoVarita** validando que sus valores solo pueden ser (1 principiante, 2 avanzado 3 maestro).

Deberá tener 2 constructores, un destructor, los setters, getters correspondientes y verTodo.



Crear las clases que heredan de Personaje

Clase	Posee al crearse		Ataca restando	
	vida	magia	vida	magia
"Caballero"	5	2	6	0
"Arquero"	3	5	2	4
"Mago"	4	15	3	3

Tal que deba codificar obligatoriamente los métodos virtuales puros Atacar y getTipo de cada clase.

Crear el archivo **Ejer-2.cpp** tal que cree un array de 3 personajes que se cargará con 3 objetos uno de cada clase recorra el arreglo e imprima sus vidas y magia tal que la salida sea:

```
El Caballero posee 5 vidas y 2 de magia  
El Arquero posee 3 vidas y 5 de magia  
El Caballero posee 4 vidas y 15 de magia
```

Nota: Para enviar un string a cout debe usar el método `c_str()` ya que cout no puede recibir objetos a imprimir por ejemplo: `cout<<p->getTipo().c_str();`

Ejercicio 3

Crear en Ejer-3.cpp un videojuego con las clases Personaje, Caballero, Arquero y Mago. Cada uno debe guardar sus cantidades de vida y magia. Todos los personajes tienen un método Atacar() que recibe por referencia otro personaje al que le quitará puntos de vida y magia. Sin embargo cada personaje produce distintas cantidades de daño:

- el caballero quita 6 unidades de vida y ninguna de magia al personaje que ataca.
- el arquero resta 2 puntos de vida y 4 de magia al atacar.
- el mago quita 3 puntos de magia y 3 de vida al adversario al que ataca.

Cada personaje posee además un método estaVivo() que permite saber si el personaje aún tiene algo de vida y una función getTipo() que devuelve un string con un texto indicando la clase del personaje ("Caballero", "Arquero" o "Mago").

Cree una clase Juego para representar un videojuego simple que contenga 9 personajes de distinto tipo en único arreglo(3 de cada clase). Cada personaje deberá elegir a otro al azar y atacarlo (recuerde que ningún personaje puede atacar si no está vivo y tampoco puede atacarse a sí mismo).

Esto se debe repetir automáticamente hasta que solo quede un personaje vivo.

Mostrar por pantalla

Durante el juego:

- El personaje, posición y a quien mató, su posición y tipo de personaje.

Ejemplo

Caballero 2 mato al personaje 6 Mago

- Cuantos personajes quedan luego que alguno muera.

Ejemplo

Quedan 6 Personajes

- El personaje su posición a quien ataca, su posición y cuantas vidas le queda.

Ejemplo

Arquero 3 ataco al Arquero 4, al Arquero 4 le quedan 1 vidas

Al finalizar:

- El tipo de personaje y la posición que ocupa en el array del ganador del juego.
El Ganador del Juego Fue el Personaje numero: 2 Profesion: Caballero

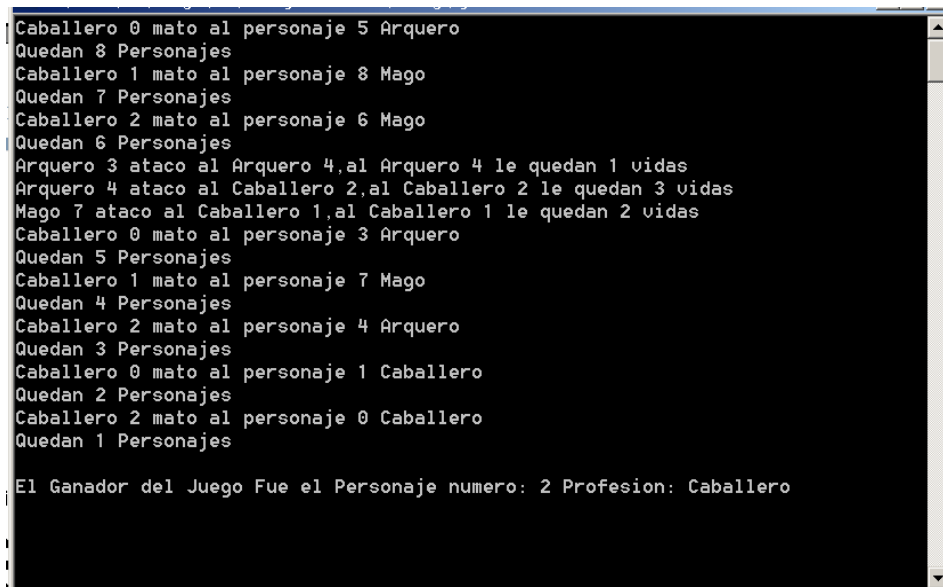
```
#define TOPE 9

class Juego{
    Personaje* _personajes[TOPE]; // array de punteros a Personaje
    bool _gameOver;                // indica si terminó el juego
public:
    Juego();                       // crea los 9 personajes
    ~Juego();                      // libera memoria
    void play();                   // juega el juego
    bool gameOver();              // devuelve true si solo queda un personaje con vida
    ...
};
```

Nota:

Para utilizar la función random hay que ejecutar `srand(time(0))` por única vez al principio del programa para generar la semilla del random. Si no se utiliza esta función los números random serán iguales en cada ejecución. Se puede observar que consulta la hora del sistema para así obtener un número distinto cada vez que se ejecuta. Luego se puede ejecutar todas la veces que sea necesario `rand()` como por ejemplo `rand() % 9` para obtener un numero al azar entre 1 y 9.

Ejemplo de una salida:



```
Caballero 0 mato al personaje 5 Arquero
Quedan 8 Personajes
Caballero 1 mato al personaje 8 Mago
Quedan 7 Personajes
Caballero 2 mato al personaje 6 Mago
Quedan 6 Personajes
Arquero 3 ataco al Arquero 4, al Arquero 4 le quedan 1 vidas
Arquero 4 ataco al Caballero 2, al Caballero 2 le quedan 3 vidas
Mago 7 ataco al Caballero 1, al Caballero 1 le quedan 2 vidas
Caballero 0 mato al personaje 3 Arquero
Quedan 5 Personajes
Caballero 1 mato al personaje 7 Mago
Quedan 4 Personajes
Caballero 2 mato al personaje 4 Arquero
Quedan 3 Personajes
Caballero 0 mato al personaje 1 Caballero
Quedan 2 Personajes
Caballero 2 mato al personaje 0 Caballero
Quedan 1 Personajes

El Ganador del Juego Fue el Personaje numero: 2 Profesion: Caballero
```

Preguntas

- Ejercicio 1: comente la instrucción donde crea el objeto Circulo sin tocar las demás instrucciones. ¿Qué sucede con la ejecución? Explicar.
- Ejercicio 2: comente las instrucciones que inicializan el array en NULL, comente la instrucción donde crea el objeto Circulo sin tocar las demás instrucciones. ¿Qué sucede con la ejecución? Explicar.
- Ejercicio 3: intente crear un objeto de clase Personaje. ¿Qué sucede? Explicar.
- Ejercicio 4: crear 8 personajes pero recorrer el array para liberar la memoria hasta 9. ¿Qué sucede? Explicar.