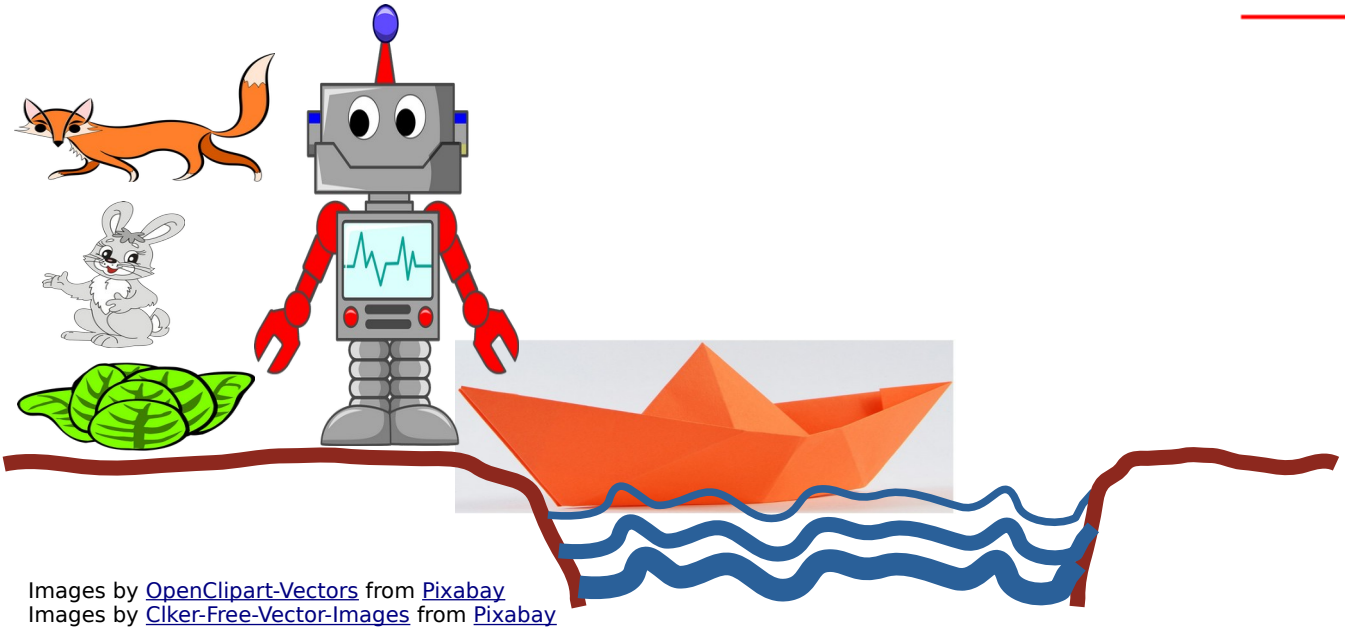
PROYECTO

Asignatura: 750015C - FUNDAMENTOS DE PROGRAMACIÓN ORIENTADA A OBJETOS

Profesor: Ángel E. García Baños
Email: angel.garcia@correounivalle.edu.co
EISC – Universidad del Valle
2022-03-27



Images by OpenClipart-Vectors from Pixabay
Images by Clker-Free-Vector-Images from Pixabay

OBJETIVOS DE APRENDIZAJE:

- **Usar todas las relaciones entre objetos**
- **Polimorfismo** (sirve, dicho rápidamente, para eliminar if)
- **Cambios de propiedad**
- **Clase controladora**
- **Extensibilidad**

Juego: LA BARCA

Objetivo:

Implementar un juego de lógica usando C++ con metodología de programación orientada a objetos.

El juego consiste en lo siguiente: hay un robot que debe transportar un zorro, un conejo y una lechuga desde un lado del río hasta la otra orilla, usando una barca. En la barca solo cabe uno de los tres individuos, además del robot. El problema es que si el robot deja solos al zorro y el conejo, el zorro se comerá el conejo. Y si deja solos al conejo y la lechuga, el conejo se comerá la lechuga. El jugador debe controlar que órdenes dar, para lograr que el robot transporte los tres individuos a la otra orilla, sanos y salvos.

Las órdenes que puede dar el jugador son (usando el teclado):
B la Barca se mueve a la otra orilla
R el Robot salta de/a la barca
Z el Zorro salta de/a la barca
C el Conejo salta de/a la barca

L la Lechuga salta de/a la barca

Después de cada orden el jugador debe mostrar en pantalla el nuevo estado del juego, usando 4 columnas. Por ejemplo, el estado inicial es:

IZQUIERDA	BARCA		DERECHA
Robot			
Zorro			
Conejo			
Lechuga			

Y el estado final debe ser:

IZQUIERDA		BARCA	DERECHA
			Robot
			Zorro
			Conejo
			Lechuga

Requerimientos (obligatorios)**:**

1. La **Barca** es un **Lugar**. Y la **Orilla** es un **Lugar**.
2. Si un **Individuo** (**lechuga**, **conejo**, **zorro** o **robot)** intenta saltar a la **barca** y la **barca** no está allí, entonces se cae al agua y el juego termina.
3. Si el **robot** no está en un cierto **Lugar** (o sea en **barca** o en **orillaIzquierda** o en **orillaDerecha**) y allí hay un **Individuo** que se puede comer al otro, entonces se lo come y el juego termina.
4. La **barca** solo se mueve si hay un **robot** en ella.
5. En la **barca** solo caben dos **Individuos**.
6. Al terminar el juego debe de imprimir “GANASTE” o “PERDISTE”. Pueden imprimir todo lo que quieran, pero estas dos palabras son obligatorias.
7. Hay que usar las relaciones “es un”, “contiene un” y “conoce un” en el proyecto.
8. Debe haber una clase **Jugador** que lea el teclado, dé ordenes a los demás objetos y muestre en pantalla el resultado. El **Jugador** es el único que puede usar **cin** y **cout**.

Nota: observe que las palabras en **letra negrita** hay que respetarlas, pues son nombres de clases y objetos. Puede añadir más clases y objetos si lo desea (aunque no es necesario), pero los que están aquí indicados es obligatorio que los ponga en su proyecto.

Dos premios de 0.1 cada uno**:**

- Premio de 0.1 que se suma a la nota de este proyecto: si logra polimorfismo (típicamente sobre la clase **Lugar**).
- Premio de 0.1 que se suma a la nota final de la asignatura si el proyecto tiene nota mayor a 4.0: al finalizar, cambiando solo **main.cpp** se debe poder implementar otra variante del juego, por ejemplo con 2 conejos, 1 lechuga y 1 zorro y 2 robots. Si lo logra, habrá demostrado ser un excelente programador. Para optar por este premio debe entregar el **main.cpp** normal; y luego, en el enlace “premio” del Campus virtual debe subir otro archivo **main.cpp** con otro Conejo (que responda al comando C2 y que también coma lechuga y sea comido por el zorro) y otro Robot (que responda al comando R2).

Entregables

Hay que hacer dos entregas en dos fechas distintas:

- PRIMERA PARTE: Identificación de las clases y relaciones que necesite para resolver este problema. Para ello escriba el archivo .h para cada clase. No necesito código. Basta que pongan los comentarios de autor del archivo y los comentarios de cada clase (lo que hace, que funciones ofrece, que relaciones tiene). El objetivo de ello es inducirle a reflexionar como descomponer un problema grande en clases pequeñas, cada una con una única responsabilidad, y con las relaciones que se necesiten. Por ejemplo, si necesita la clase Aaa que es una Bbb y conoce a una Ccc, debería escribir el archivo Aaa.h siguiente:

```
/*
  Archivo: Aaa.h
  Autor: El nombre e email de ustedes .....
  Fecha creación: .....
  Fecha última modificación: .....
  Versión: .....
  Licencia: GNU-GPL
*/

/**
  CLASE: Aaa
  INTENCIÓN: escriba aquí lo que pretende hacer con esta clase .....
  RELACIONES:
  - es un Bbb
  - contiene .....
  - conoce Ccc
*/


#ifndef AAA_HH
#define AAA_HH

#include "Bbb.h"
#include "Ccc.h"


class Aaa : public Bbb
{
   private:
   Ccc *c;
};

#else
class Aaa; // Declaración adelantada
#endif
```

- SEGUNDA PARTE: Implementación de las clases: escriba el código de cada una de las clases, en los correspondientes archivos .h y .cpp así como el programa principal main.cpp y súbalos al campus virtual. La calificación de esta parte depende de:
  - Estilo correcto.
  - Que haya cumplido los requerimientos.
  - Que compile sin errores.
  - Que funcione sin errores y sin *memory leaks*.

METODOLOGÍA: Dado que estamos cada uno en nuestra casa, aislados para evitar el Covid-19, trabajaremos individualmente o en grupos de 2 o 3 personas que se comunican por medio de herramientas virtuales (chat, repl.it o similares).

ASPECTOS A EVALUAR de la primera parte, que vale un 5%:

- **REQUERIMIENTOS (40%)**
- **ESTILO (40%)**
- **PROGRAMACIÓN (20%)**

ASPECTOS A EVALUAR de la segunda parte, que vale un 95%:

- **REQUERIMIENTOS (10%)**
- **ESTILO (20%)**
- **PROGRAMACIÓN (40%)**
- **COMPILACIÓN (10%)**
- **EJECUCIÓN (20%)**