PROGRAMACIÓN ORIENTADA A OBJETOS
REPORTE FINAL PROYECTO SNAKE

Integrantes:

- Marlon Corona 36505
- Caitlin Murillo 34715
- Raul Cruz 36626

Introducción:
El juego de serpiente es un juego de arcade en el que el jugador controla una “serpiente” que se mueve alrededor de un área cerrada. El objetivo es comer objetos que aparecen aleatoriamente en el área de juego, lo que hace que la serpiente crezca en longitud. El juego termina cuando la serpiente choca contra sí misma o contra los bordes del área de juego. Se ha implementado utilizando c + + basado en la programación orientada a objetos.

Estructura del Código:

El código está dividido en varias clases, el dividir el código en clases nos permitió no tener un código tan repetitivo y facilitar un poco la implementación del código. Las clases son “Juego”, “Serpiente”, “Comida”, “Bonus” y las clases de las dificultades que son fácil, media, difícil.

1. Clase Juego:
   La clase juego es la base de todo el programa y es la que proporciona toda la estructura y lógica para el juego. En ella se incluyen los métodos y funciones principales para el juego. Las más importantes son:
   - GameOver: Usamos el método booleano que indica si el juego ha terminado, este es necesario para que controle el bucle principal del juego.
   - Height y Width: Estas indican las dimensiones que tendrá el mapa del juego.
   - Score: Esta es la puntuación del jugador, debe incrementar cada vez que la serpiente come una comida.

- Speed: Esta es la velocidad del juego.
- dir: Esta es la dirección actual de la serpiente.
- hConsole, buffer, bufferSize, CharacterPos, consoleWriteArea: estas son las variables para manejar la consola y el buffer de la pantalla, esto para poder manejar la pantalla del juego de forma más eficiente.

  Métodos Virtuales:
  - Setup(): Es la configuración inicial del juego, que está sobreescrita en las clases derivadas de la clase base.
  - Draw(): Esta se encarga de dibujar el estado actual del juego en la consola.
  - Input(): Esta es la encargada de capturar y procesar las entradas del usuario.
  - Logic(): Contiene toda la lógica del juego, como el movimiento de la serpiente y la detección de colisiones.
  - run(): Controla el ciclo principal del juego.

incluimos el método para dibujar el mensaje de "Game Over" (isGameOver(), `setGameOver(), setDirection(), getDirection()).

2. Clase Serpiente:
   La clase serpiente está heredada de juego y añadimos la funcionalidad específica de la serpiente.

   - “X”, “Y”: Son las coordenadas de la cabeza de la serpiente.
   - tail, tailY: Estos son arrays que almacenan las coordenadas del cuerpo de la serpiente, esto para poder ver como la serpiente crece.
   - nTail: Este es el tamaño actual que tiene la serpiente, esta aumenta cuando la serpiente come.

     Métodos que están sobreescritos:
     - Setup(), Draw(), Input()
     - Logic(): Incluye la lógica para que la serpiente coma la comida, aumentando la puntuación y el tamaño de la serpiente, y generando nueva comida en una posición aleatoria.

3. Clase comida:
   La clase comida está heredada de Serpiente y añadimos funcionalidades para manejar la comida para que crezca la serpiente.

   Funcionalidades añadidas:
   - FruitX, FruitY: Estas son las coordenadas de la comida en el mapa.

     Métodos que están sobreescritos:
     - Setup(), Draw(), Input() y Logic().

4. Clase Bonus:
   La clase Bonus está heredada de comida y añadimos funcionalidades.
   - Bonus, Bonus Y: Coordenadas del bonus.

     Métodos que están sobreescritos:
     - Setup(), Draw(), Input().
     - Logic(): Incluye la lógica para que la serpiente interactúe con los bonus aumentando la puntuación.

5. Clases de dificultades:
   Estas clases heredan de Bonus y configuran el juego con diferentes niveles de dificultad.

   - Dificultad Fácil: Mapa más pequeño (10x10), la velocidad es más lenta y permite que la serpiente atraviese los bordes.
   - Dificultad Media: Mapa de tamaño medio (20x20), la velocidad es más moderada y la colisión con los bordes finaliza el juego.
   - Dificultad Dificil: Mapa grande (30x30), la velocidad es más alta y también la colisión con los bordes del mapa finaliza el juego.

Cada clase redefine el método `Setup()` para ajustar el tamaño del mapa y la velocidad del juego según la dificultad seleccionada.

6. Ejecución:

   El juego es ejecutado en la función main(), que muestra el menú principal donde el jugador puede seleccionar la dificultad del juego. Dependiendo de que seleccione se usa la clase correspondiente (Fácil, Media, Difícil) y llama al método run, que es la encargada de controlar la ejecución del juego.

   Menú principal:
   - Muestra las opciones de dificultades y se puede seleccionar la de preferencia.
   - Esta selección es por medio de un cin.

     Bucle de juego:
     - El run() se ejecuta en un bucle hasta que la declaración de GameOver sea true para detenerse.
     - En cada iteración se guardan las entradas del usuario, también se actualiza la lógica del juego, y se dibuja el estado actual del juego en la consola.
     - Cuando se termina el juego muestra un mensaje de “GameOver” y hay una espera de 2 segundos para volver a mostrar el menú principal.
7. Conclusiones:

   Equipo:
   Algunos de los retos y dificultades fue el manejo de la consola ya que poder lograr dibujar en la consola de una manera eficiente en la pantalla del usuario fue un desafío importante ya que se necesito arreglar que no se viera como movida. El estar testeando el código también fue un desafío ya que era encontrar errores a la hora de jugar para así poder modificar esto en el código en su lógica o donde fuera necesario. De los desafíos más significativos fue la comunicación del equipo y la coordinación ya que teníamos que asegurarnos que todos los miembros del equipo estuvieran entendiendo la estructura de código que creamos entre nosotros para así poder seguir desarrollando el programa.

Por otro lado, se logró implementar un juego de la serpiente funcional con diferentes niveles de dificultad. El juego incluye todas las características básicas del juego de la serpiente que son el movimiento de la serpiente, la comida, el crecimiento, y la detección de colisiones.

Individuales:

- **Marlon Corona:** Me parece que la idea del proyecto fue muy buena para demostrar los temas aprendidos en clase, ya que es un juego sencillo donde los elementos o “objetos” son fáciles de visualizar y existen múltiples funciones que se pueden añadir a partir de herencias de clase. La parte más difícil en mi opinión fue la parte de implementar las funciones como clases y desarrollar las herencias, a pesar de que el programa era relativamente sencillo, al momento de crear las clases y objetos se me dificulta visualizar cómo cada objeto funcionaba dentro del programa y al momento de llamar a las funciones en el main. Los logros fundamentales del programa (fuera de que funcionara apropiadamente como un juego de Snake) fueron cumplidos, que eran agregar dificultades y comidas “bonus” a través de herencia, a su vez de la creación de un pequeño menú. Nos hubiera gustado hacerle un menú más completo al juego que incluyera las instrucciones y créditos, además de mejorar el funcionamiento del juego en general y algunos bugs.
- **Caitlin Murillo:** En este proyecto pude poner en práctica los conceptos que fueron vistos en clase, la idea del proyecto en general se veía muy sencilla para implementarla, ya que consideraba que era fácil identificar las clases más fundamentales para el desarrollo, La parte más desafiante para mi considera fue el cómo empezar a implementar la lógica del juego basica para así poder pasarla a clases con funciones y métodos, también el tener que checar los errores que iban surgiendo mientras íbamos desarrollando el programa, el tener que visualizar que modificar para arreglar el error, el tener que estar agregando o borrando cosas, investigando sobre el tema. Fuera de todo eso considero que fue logrado el objetivo principal que era hacer

el juego de snake funcional. Este proyecto me permitió aplicar los conceptos de programación orientada a objetos en un contexto diferente y así poder desarrollar más mis habilidades en esta área.

- **Raul Cruz:** En este proyecto pude visualizar de una manera diferente como se maneja el uso de las clases con respecto a la programación orientada objetos, a lo cual durante todo el semestre estuvimos canalizando de diferentes maneras como funcionan las clases heredadas, lo que beneficia y afecta el que las declaraciones de variables estén protegidas, privadas y públicas. Lo visto en este código fue como pudimos comenzar con nuestra clase principal y como pudimos irla desarrollando para que nos elaborara una buena forma de interactuar con la serpiente, y demostrar que los diferentes tipos de dificultades también, como elaborar un tipo de comida el cual hace que afecte a la serpiente y la vaya haciendo crecer en la parte de su colita, el cómo elaborar un tipo de comida especial que solamente lo ponemos en las dificultades media y difícil, como mandar a llamar cada función ya sea tipos virtuales, new, void, y también la elaboración de los constructores y destructores.