Universidad Rafael Landívar
Faculta de Ingeniería
Ingeniería en Informática y Sistema
Laboratorio Introducción a la Programación

Universidad Rafael Landívar
Tradición Jesuita en Guatemala

Proyecto de Aplicación

“Proyecto No.01 Juego Snake”

Lara Vásquez, Sergio Daniel [1044418]
Velásquez Villegas, Carlo Antonio Ismael [1264418]

Ciudad de Guatemala, 10 de marzo de 2018.

Introducción

En el proyecto numero uno, que es la elaboración del famoso juego del Snake (culebrita), en el cual se busca la aplicación de las estructuras selectivas, ciclos o bucles, así como el manejo de clases.

El proyecto por sus requerimientos de recibir el tamaño del tablero y mostrarle al usuario luego donde se ubica la cola y cabeza en el plano y validando que la cabeza no tocara ningún borde o chocara consigo misma no era necesario utilizar conocimientos fuera de los ensañados en el curso, pero decidimos emprender y añadirle funcionalidades extra en especial en el modo en que se le muestra al usuario, ya le es mostrado un plano y la culebrita así como su movimiento, para poder hacer esto investigamos por nuestra cuenta, y aplicamos conocimientos nuevos como las estructuras (struc), referencias como “goto” y propiedades más estéticas como “Console.ForegroundColor”

Para que el programa funcionara se tuvo en cuenta un previo análisis acerca de las entradas, salidas, procesos y validaciones que este debía contener. En la entrada solo se necesita que el usuario ingrese el tamaño (altura y largo) del tablero a utilizar para que la culebrita se mueva sobre el como si fuera un plano cartesiano, para lo cual se creó la estructura “coordenadas” que hace que la computadora pueda asimilar un plano cartesiano. Para poder utilizar el plano cartesiano fue se creo dos clases una fue tablero y la otra culebrita, estas con sus respectivos métodos y funciones, en la clase culebrita es donde se validad si esta toco un borde del plano, trato de avanzar sobre si misma o si la cola topa con la cabeza, se calcula el movimiento (coordenadas donde está ubicada) utilizando el objeto “ConsoleKey” y también es la encargada de dibujar la culebrita en la consola; la clase tablero es la que dibuja en tablero al usuario en la consola.

Todo esto al final le muestra al usuario un tablero en las dimensiones que ingreso y la culebrita con su movimiento, gracias a esto determinamos que tanto el uso de ciclos como de estructuras selectivas, a pesar de ser conceptos básicos casi siempre son necesarias para poder crear la solución.

Objetivos

1. Utilizar correctamente las estructuras selectivas y de repetición.

2. Aplicar correctamente el uso de clases, funciones y métodos, tomando en cuenta las buenas prácticas de programación.

3. Resolver la problemática utilizando las fases para la resolución de problemas (análisis, diseño, codificación, etc.).

4. Escribir un algoritmo preciso, definido y finito.

Análisis

Entradas:

- Las dimensiones del plano.
- Las direcciones del movimiento de la serpiente con las flechas del teclado.
- La tecla X para salir;

Procesos:

- Teniendo el ancho y altura del tablero enviarlos a la función DibujarTablero() de la Clase Tablero, para que esta envié varias cadenas de texto y así dibujar el tablero.
- Si el acho y altura no son validos repetir el ingreso de estos.
- En el juego evaluar si cada tecla que se presione sea las flechas o X.
- Si en el movimiento la cabeza toca la cola o si la cabeza toca un borde es igual a intento acabado. Mediante la función EvaluarCabezaIgualCola( ) y EvaluarCabezaMargen( ) de la clase culebrita.
- No puede retroceder la culebrita evaluada mediante EvaluarCabezaIguaCuerpo1()
- Si se presiona X salir de juego

Salidas:

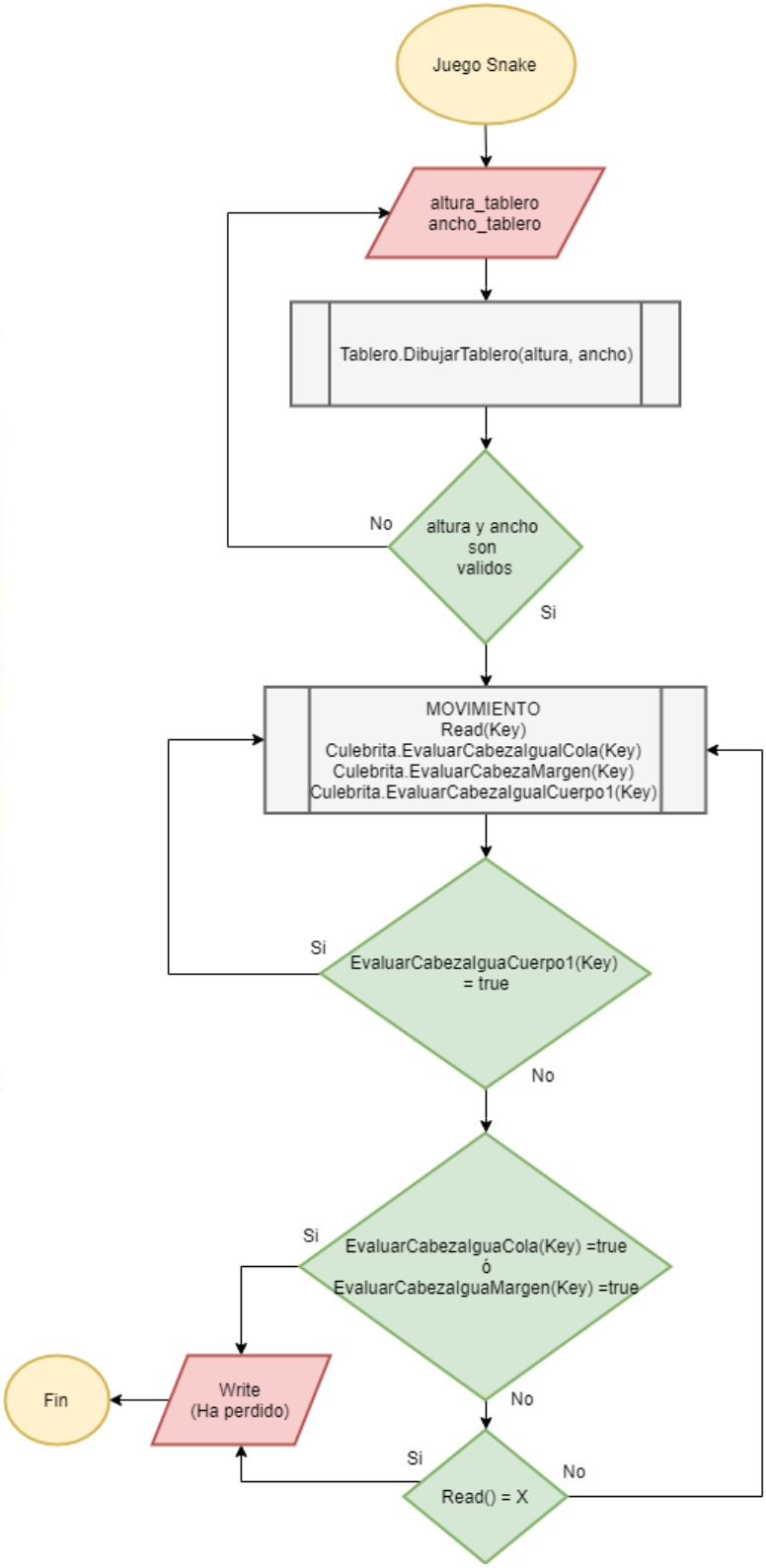
- Luego de cada movimiento mostrar las coordenadas de donde está la cabeza y cola de la culebrita.
- Cuando el movimiento sea en reversa mostrar al usuario que ha sucedido un movimiento invalido.
- Si la cabeza choca con su cola mostrar al usuario un mensaje de que ha perdido el juego porque su cabeza toco su cola.
- Si la cabeza choca con un borde mostrar al usuario un mensaje de que ha perdido el juego porque su cabeza a tocado un límite.

Validaciones:

- El usuario no puede ingresar números negativos, fraccionarios y decimales en el tamaño del plano.
- Los campos no pueden ser vacíos o nulos, en el tamaño del plano.
- Durante el juego solo las flechas (direccionales) y la tecla x ejecutan alguna acción. Las demás estarán deshabilitadas.

Diseño Del Proyecto

Diagrama de flujo



Algoritmo En Pseudocodigo

```
Class Program
begin
public struct Coordenadaz
begin
        public int x;
        public int y;
        public override bool Equals(object obj)
        begin
                return obj is Coordenadaz && this == (Coordenadaz)obj;
        end_function
        public override int GetHashCode()
        begin
                return x.GetHashCode() ^ y.GetHashCode();
        end_function
        public static bool operator ==(Coordenadaz c1, Coordenadaz c2)
        begin
                return (c1.x == c2.x) && (c1.y == c2.y);
        end_function
        public static bool operator !=(Coordenadaz c1, Coordenadaz c2)
        begin
                return (c1.x != c2.x) || (c1.y != c2.y);
        end_function
end_Structure

Class Culebrita
        begin
        public Coordenadaz cabeza;
        private Coordenadaz cuerpo1;
        private Coordenadaz cuerpo2;
        public Coordenadaz cola;
        private Coordenadaz borrador;
        private Coordenadaz NewPos;
        begin
        public Culebrita()
        begin
                private cabeza = new Coordenadaz();
                private cuerpo1 = new Coordenadaz();
                private cuerpo2 = new Coordenadaz();
                private cola = new Coordenadaz();
                private borrador = new Coordenadaz();
                private NewPos = new Coordenadaz();
        end_Procedure

        Procedure CulebritaPosicionInicial(Tablero tab)
        begin
                cabeza.x = tab.MargenAncho;
                cabeza.y = tab.MargenAlto / 2;
                cuerpo1.x = cabeza.x - 2;
                cuerpo1.y = cabeza.y;
                cuerpo2.x = cuerpo1.x - 2;
                cuerpo2.y = cabeza.y;
                cola.x = cuerpo2.x - 2;
                cola.y = cabeza.y;
                borrador.x = tab.MargenAncho + 1;
```

```
        borrador.y = 0;
        NewPos.x = 0;
        NewPos.y = 0;
End_Procedure

Procedure Actualizar(ConsoleKey teclaPresionada)
begin
        borrador = cola;
        cola = cuerpo2;
        cuerpo2 = cuerpo1;
        cuerpo1 = cabeza;
        switch (teclaPresionada)
        begin
                case ConsoleKey.UpArrow:
                        cabeza.y--;
                        break;
                case ConsoleKey.DownArrow:
                        cabeza.y++;
                        break;
                case ConsoleKey.LeftArrow:
                        cabeza.x -= 2;
                        break;
                case ConsoleKey.RightArrow:
                        cabeza.x += 2;
                        break;
        end_switch
end_Procedure
bool function EvaluarCabezaIgualCola(ConsoleKey teclaPresionada)
begin
        bool resultado = false;
        NewPos = cabeza;
        switch (teclaPresionada)
        begin
                case ConsoleKey.UpArrow:
                        NewPos.y--;
                        break;
                case ConsoleKey.DownArrow:
                        NewPos.y++;
                        break;
                case ConsoleKey.LeftArrow:
                        NewPos.x -= 2;
                        break;
                case ConsoleKey.RightArrow:
                        NewPos.x += 2;
                        break;
        end_switch
        if (NewPos == cola)
        begin
                resultado = true;
        end_if
        return resultado;
end_function
bool function EvaluarCabezaIgualCuerpo1(ConsoleKey teclaPresionada)
begin
bool resultado = false;
NewPos = cabeza;
```

```
        switch (teclaPresionada)
        begin
                case ConsoleKey.UpArrow:
                        NewPos.y--;
                        break;
                case ConsoleKey.DownArrow:
                        NewPos.y++;
                        break;
                case ConsoleKey.LeftArrow:
                        NewPos.x -= 2;
                        break;
                case ConsoleKey.RightArrow:
                        NewPos.x += 2;
                        break;
                end_switch
                if (NewPos == cuerpo1)
                begin
                        resultado = true;
                end_if
                return resultado;
        end_function
        bool function EvaluarCabezaMargen(ConsoleKey teclaPresionada, Tablero tab)
        begin
                bool resultado = false;
                NewPos = cabeza;
                switch (teclaPresionada)
                begin
                        case ConsoleKey.UpArrow:
                                NewPos.y--;
                                break;
                        case ConsoleKey.DownArrow:
                                NewPos.y++;
                                break;
                        case ConsoleKey.LeftArrow:
                                NewPos.x -= 2;
                                break;
                        case ConsoleKey.RightArrow:
                                NewPos.x += 2;
                                break;
                end_switch
                if (NewPos.x == tab.MargenAncho * 2 - 1 || NewPos.y ==  tab.MargenAlto - 1 ||
NewPos.x == 1 || NewPos.y == 0)
                begin
                resultado = true;
                end_if
                return resultado;
        end_function
        procedure DibujarCulebrita()
        begin
                Console.ForegroundColor = ConsoleColor.Blue;
                Console.SetCursorPosition(cabeza.x, cabeza.y);
                Console.Write("0");
                Console.ForegroundColor = ConsoleColor.DarkBlue;
                Console.SetCursorPosition(cuerpo1.x, cuerpo1.y);
                Console.Write("0");
                Console.SetCursorPosition(cuerpo2.x, cuerpo2.y);
```

```
                Console.Write("0");
                Console.SetCursorPosition(cola.x, cola.y);
                Console.Write("0");
                Console.ForegroundColor = ConsoleColor.Black;
                Console.SetCursorPosition(borrador.x, borrador.y);
                Console.Write("0");
        End_procedure
End_class
class Tablero
        int AnchoCuadrante;
        int AltoCuadrante;
        int AnchoTotal;
        int AltoTotal;
        public int MargenAncho;
        public int MargenAlto;
        begin
        public Tablero()
        begin
        AnchoCuadrante = 0;
        AltoCuadrante = 0;
        AnchoTotal = 0;
        AltoTotal = 0;
        MargenAncho = 0;
        MargenAlto = 0;
        End_procedure

        Procedure DibujarTablero(int temp1, int temp2)
        begin
                Console.Clear();
                AnchoCuadrante = temp1;
                AltoCuadrante = temp2;
                AnchoTotal = 2 * AnchoCuadrante + 1;
                AltoTotal = 2 * AltoCuadrante + 1;
                MargenAncho = AnchoTotal + 2;
                MargenAlto = AltoTotal + 2;

                for (int filas = 0; filas < MargenAlto; filas++)
                begin
                        for (int columnas = 0; columnas < MargenAncho; columnas++)
                        begin
                                if (columnas == 0 || columnas == MargenAncho - 1 || filas == 0 ||
filas == MargenAlto - 1)
                                begin
                                        Console.ForegroundColor = ConsoleColor.DarkRed;
                                else
                                        Console.ForegroundColor = ConsoleColor.Black;
                                End_if
                                Console.Write(" 0");
                        End_for
                        Console.Write();
                End_for
                End_procedure
        End_clas

        procedure Main()
                string val1, val2;
```

```
		int alto, ancho;
		bool FinDelJuego = false;
		Tablero TableroDeJuego;
		Culebrita CulebritaDeJuego;
		ConsoleKeyInfo teclaPresionada;

		Do
		begin
			TableroDeJuego = new Tablero();
			CulebritaDeJuego = new Culebrita();
			FinDelJuego = false;
			inicioprograma:
			Console.ForegroundColor = ConsoleColor.White;
			Console.Write("Instrucciones:");
			Console.Write("Ingresa los valores del primer cuadrante: ");
			Console.Write("valor minimo 3, valor maximo 15");
			Console.Write("x: ");
			val1 = Console.Read();
			Console.Write("y: ");
			val2 = Console.Read();
			if (int.TryParse(val1, out int temp1) & int.TryParse(val2, out int temp2))
				if ((int.Parse(val1) >= 3) & (int.Parse(val1) <= 15) & (int.Parse(val2)
>= 3) & (int.Parse(val2) <= 15))
					ancho = int.Parse(val1);
					alto = int.Parse(val2);
				else
					Console.Write("Valores fuera de rango, ingresalos
nuevamente.");
					Console.Write("Enter para continuar...");
					Console.Read();
					Console.Clear();
					goto inicioprograma;
				end_if
			else
				Console.Write("Uno o todos los valores no son valios, ingresalos
nuevamente.");
				Console.Write("Enter para continuar...");
				Console.Read();
				Console.Clear();
				goto inicioprograma;
			end_if

			TableroDeJuego.DibujarTablero(ancho, alto);
			CulebritaDeJuego.CulebritaPosicionInicial(TableroDeJuego);
			CulebritaDeJuego.DibujarCulebrita();

			do
			begin
				CulebritaDeJuego.DibujarCulebrita();
				Console.SetCursorPosition(1, TableroDeJuego.MargenAlto + 1);
				Console.Write("                                                  ");
				Console.SetCursorPosition(1, TableroDeJuego.MargenAlto + 1);
				Console.ForegroundColor = ConsoleColor.White;
				Console.Write("Coordenadas: cabeza({0},{1}) cola({2},{3}) ",
(CulebritaDeJuego.cabeza.x - TableroDeJuego.MargenAncho) / 2, (TableroDeJuego.MargenAlto -
```

```
1) / 2 - CulebritaDeJuego.cabeza.y, (CulebritaDeJuego.cola.x - TableroDeJuego.MargenAncho) / 2,
(TableroDeJuego.MargenAlto - 1) / 2 - CulebritaDeJuego.cola.y);
                                    Console.ForegroundColor = ConsoleColor.Black;
                                    teclaPresionada = Console.Read();
                                    Console.SetCursorPosition(1, TableroDeJuego.MargenAlto + 3);
                                    Console.Write("                                                  ");
                                    Console.SetCursorPosition(1, TableroDeJuego.MargenAlto + 3);
                                    Console.ForegroundColor = ConsoleColor.White;
                                    ConsoleKey temp = teclaPresionada.Key;
                                    if (temp == ConsoleKey.UpArrow || temp ==
ConsoleKey.DownArrow || temp == ConsoleKey.LeftArrow || temp == ConsoleKey.RightArrow)
                                        if
(CulebritaDeJuego.EvaluarCabezaIgualCola(teclaPresionada.Key))
                                            Console.Write("Te mordiste la cola");
                                            FinDelJuego = true;
                                        else if
(CulebritaDeJuego.EvaluarCabezaIgualCuerpo1(teclaPresionada.Key))
                                            Console.Write("No puedes regresar");
                                        else if
(CulebritaDeJuego.EvaluarCabezaMargen(teclaPresionada.Key, TableroDeJuego))
                                            Console.Write("Chocaste con la pared");
                                            FinDelJuego = true;
                                        else
                                            CulebritaDeJuego.Actualizar(teclaPresionada.Key);
                                        end_if
                                    End_if
                            end_do          while (teclaPresionada.Key != ConsoleKey.X &&
!FinDelJuego);

                            if (teclaPresionada.Key == ConsoleKey.X)
                                    Console.Write(" Gracias por jugar culebrita");
                            else
                                    Console.Write(" Presione una tecla para reiniciar");
                            End_if

                            Console.ForegroundColor = ConsoleColor.Black;
                            Console.Read();
                            Console.Clear();
                            CulebritaDeJuego = null;
                            TableroDeJuego = null;
                    End_do          while(teclaPresionada.Key != ConsoleKey.X);
            end_procedure_main
end_Class_program
```

Detalles de clases

Class Program
Public Strut Coodernadaz() Class Culebrita Class Tablero Public Static Void Main()

Class Culebrita
Public Coordenadaz cabeza Private Coordenadaz cuerpo1 Private Coordenadaz cuerpo2 Public Coordenadaz cola Private Coordenadaz borrador Private Coordenadaz NewPos
Public Culebrita() Public Void PosiscionInicial() Public Void Actualizar(); Public Bool EvaluarCabezaIgualCola() Public Bool EvaluarCabezaIgualCuerpo1() Public Bool EvaluarCabezaIguaMargen(); Public Void DibujarCulebrita()

Class Tablero
Private int AnchoCuadrante Private int AltoCuadrante Private int AnchoTotal Private int AltoTotal Public int MargenAncho Public int MargenAlto
Public Void dibujar tablero()

Conclusiones

- Las estructuras repetitivas se pueden crear anidadas para la resolución de repetir algo dentro de una repetición.

- La programación Orientada a objetos tiene la facilidad de las clases y funciones que nos permiten desarrollar d forma modular, así ahorrando tiempo.

- La facilidad que se tiene al trabajar con objetos es que se puede trabajar con varios, por ejemplo, estructuras.

Referencias

Electrónicas

- Microsoft, (s.f.) Console.ForegrounColor Disponible en: https://msdn.microsoft.com/es-es/library/system.console.foregroundcolor(v=vs.100).aspx

- Microsoft (s.f.) GoTo Disponible en: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/goto

- Microsoft (s.f.) Struc Disponible en: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/using-structs

- Microsoft (s.f.) Equals Method Disponible en: https://msdn.microsoft.com/en-us/library/336aedhh(v=vs.100).aspx

- Microsoft (s.f.) Override Disponible en: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/override

- Microsoft (s.f.) Objet.GetHashCode Disponible en: https://msdn.microsoft.com/es-es/library/system.object.gethashcode(v=vs.110).aspx