Universidad Rafael Landívar
Laboratorio de Programación
Ingeniería Química
Catedrático: Ing. Edwin Chocoy

PROYECTO PRÁCTICO NO. 2
DADOS

Barrientos Mancilla, Jessica Valeria- 1097723

Ciudad de Guatemala, 11 de noviembre de 2023

ÍNDICE

INTRODUCCIÓN..... 3
ANÁLISIS.....4
ENTRADAS..... 4
SALIDAS..... 4
PROCESOS..... 5
RESTRICCIONES..... 7
DISEÑO.....8
DIAGRAMA DE FLUJO..... 8
DIAGRAMA DE CLASE.....9
CONCLUSIONES..... 10
RECOMENDACIONES..... 10
REFERENCIAS..... 11
ANEXOS..... 12
MANUAL DEL USUARIO..... 12

INTRODUCCIÓN

El presente trabajo consiste en la aplicación de habilidades de programación adquiridas a lo largo del semestre para elaborar actividades de entretenimiento que logren ser beneficiadas con el uso de la tecnología. Como resultado de esto, se desarrolló un juego denominado “Dados”, el cual representa una simplificación de su versión en la vida cotidiana donde se emplean dos dados que son lanzados por un jugador, considerando la suma de ambos para ganar o perder el juego.

Profundizando en el trabajo creado, el presente código genera valores arbitrarios para ambos dados, simplificando el trabajo del jugador para que especificara únicamente los parámetros sobre la cantidad de partidas y tiros que desea jugar. De esta manera, se realizan de manera automática los cálculos y análisis correspondientes para que, al final del juego, se indicara de manera sencilla y eficiente el ganador y demás resultados.

Finalmente, se demostró que el uso de la programación en actividades cotidianas, como con juegos, facilita su análisis y funcionamiento, permitiendo implementar la tecnología en ámbitos tanto prácticos como recreativos para mejorar la calidad y experiencia del usuario.

ANÁLISIS

ENTRADAS

Tabla No. 1: Entradas y descripciones

Entrada	Descripción
Cantidad de partidas	El usuario debe ingresar información sobre el número de partidas que desea jugar, utilizando números enteros.
Cantidad de tiros por partidas	El usuario debe ingresar información sobre el número de tiros por partida, es decir, cuántas veces desea que el programa despliegue valores aleatorios de los dos dados en cada partida y utilizando números enteros.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 2”, (2023).

SALIDAS

Tabla No. 2: Salidas y descripciones

Salida	Descripción
Tiros realizados en cada partida- Jugador y casa	Luego de que el usuario ingrese el número de partidas y de tiros, el programa desplegará de manera aleatoria valores para los dos dados en cada tiro para todas las partidas. Además, se indican los tiros que corresponden al jugador y a la casa, repartiéndose de forma par para cada uno de ellos.
Ganador del juego- Partida	En base a la suma de los resultados de cada tiro en las partidas, el programa desplegará como ganador a aquel que consiga la mayor puntuación. Los posibles ganadores son “la casa” o “el jugador”.
Puntos del jugador- Partida	El programa desplegará el número de puntos a favor del jugador, según las reglas indicadas en la tabla de PROCESOS.
Puntos de la casa- Partida	El programa desplegará el número de puntos a favor de la casa, el oponente del jugador/usuario, según las reglas indicadas en la tabla de PROCESOS.
Tiros ganados por el jugador- Partida	El programa desplegará el número de tiros que el jugador ganó según las reglas indicadas en la tabla de PROCESOS, empleando para esto un contador.
Tiros ganados por la casa- Partida	El programa desplegará el número de tiros que la casa ganó según las reglas indicadas en la tabla de PROCESOS, empleando para esto un contador.

Probabilidad de ganar-Partida	El programa desplegará en forma de porcentaje la probabilidad que tiene el usuario de ganar contra la casa según los tiros ganados por el jugador y el número de tiros por partida.
Tiros con números pares- Partida	El programa desplegará el número de tiros que resultaron en valores pares para ambos dados en la partida según su suma.
Tiros números impares-Partida	El programa desplegará el número de tiros que resultaron en valores impares para ambos dados en la partida según su suma.
Tiros con números iguales-Partida	El programa desplegará el número de tiros que resultaron iguales para ambos dados en la partida.
Ganador del juego	El programa, luego de todas las partidas, desplegará el ganador del juego según el número de partidas ganadas, siendo los posibles ganadores el jugador o la casa.
Puntos totales ganados por el jugador/casa	El programa, luego de todas las partidas, desplegará el número de puntos ganados por el jugador y la casa por medio de un contador considerado en cada tiro y partida.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 2”, (2023).

PROCESOS

Tabla No. 3: Procesos y descripciones

Salida	**Descripción**
Valores aleatorios de los dados	El programa utiliza la clase de “Random” para dar a cada dado un valor arbitrario, utilizando un rango de 1 a 6. Es por ello que, en el código de “random.Next()”, se indican números de 1 a 7, ya que el límite de 7 no se incluye.
Suma de dados	Para el puntaje final del jugador y la casa, se suman los valores de ambos dados para cada tiro en las partidas.
Punteo del jugador y la casa	Para establecer si los puntos indicados en cada suma corresponden al jugador o a la casa, se consideran las siguientes reglas: - Si la suma de los dados es 12 o 6 en el primer tiro, el jugador gana 12 puntos.

	- Si la suma es 4 o 10 de la primera tirada el jugador pierde y la "Casa gana" 12 puntos. - Si la suma es 2, 3, 5, 7, 8 o 9 en el tiro, la suma es el puntaje del jugador o la "Casa". - Un jugador pierde si la suma tira un 11 antes de haber ganado un punto, y la "Casa" gana 6 puntos. Además, cada tiro ganado por el jugador se almacena en un contador para la sección de resultados.
Tiros para el jugador y la casa	Para determinar el número de tiros para el jugador y la casa se dividió entre dos el número de tiros por partida ingresado por el usuario, considerando los primeros tiros para el jugador.
Tiros pares, impares o iguales	Para determinar el tipo de tiros en el juego, el programa considera una división entre la suma de ambos dados y un contador. Si el residuo es 0, se considera como número par, y si es diferente de 0, se considera como número impar. Por último, evalúa si en ambos dados se obtuvo el mismo número; de ser verdadero, se considera para el contador.
Tiros ganados por el jugador/casa	Para determinar el número de tiros ganados por el jugador o la casa se empleó un contador ubicado en cada condición de la obtención de puntos, aplicando en el momento en que se cumplía según la suma de los dados.
Ganador del juego- Partida y Juego	Para determinar el ganador del juego entre la casa y el usuario, se utiliza una condición “if”, donde se evalúa el número de puntos que obtuvo cada uno y gana el que tuvo la mayor cantidad.
Probabilidad de ganar- Partida	Para determinar la probabilidad del usuario de ganar contra la casa, se realiza una operación algebráica donde se dividen los tiros ganados por el jugador entre el número de tiros.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 2”, (2023).

RESTRICCIONES

Tabla No. 4: Restricciones y descripciones

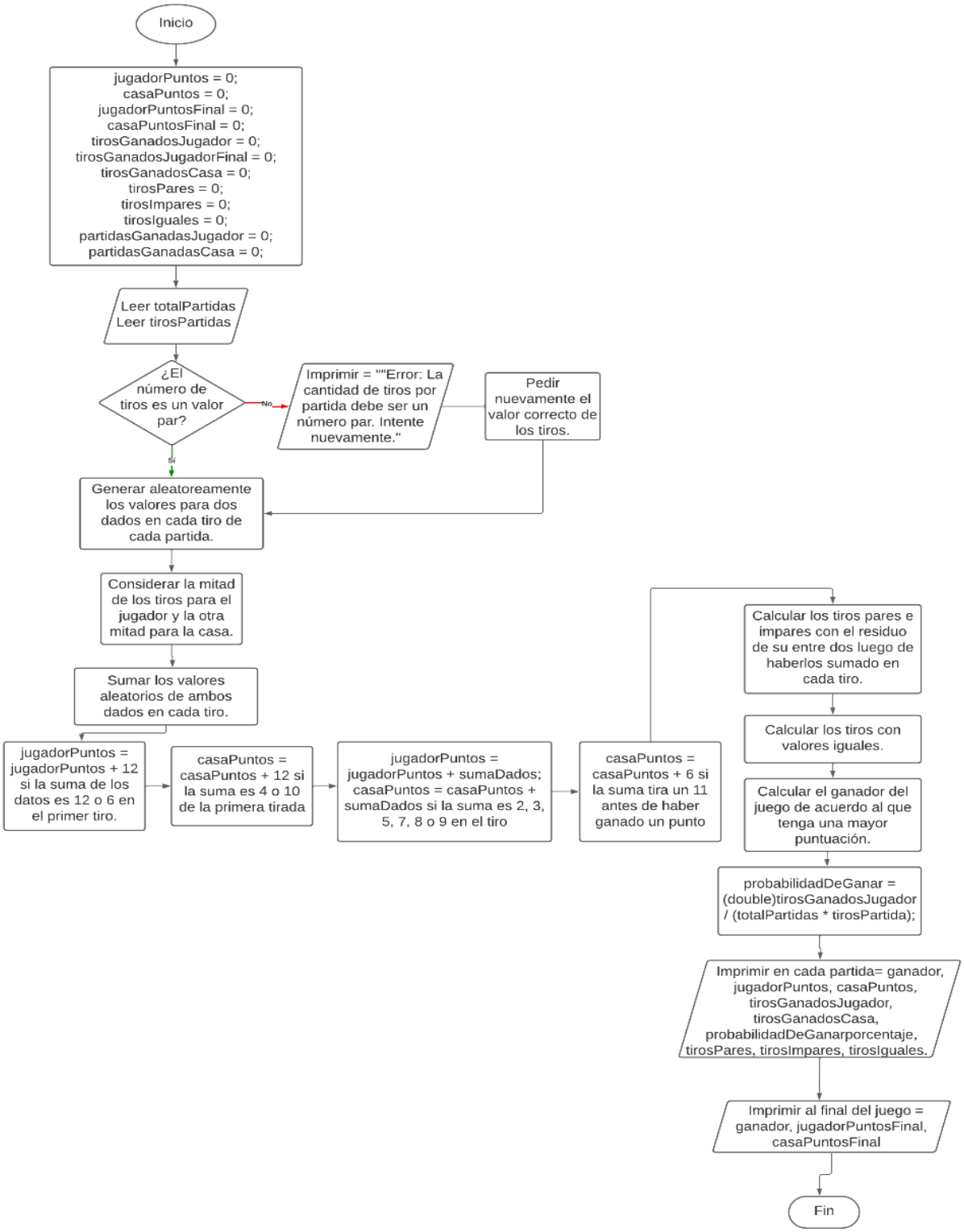
Restricciones	Descripción
Punteo máximo de dos dados	A pesar de que los valores para ambos dados se consideran aleatorios, por su naturaleza solo se toman valores de 1 a 6; es por ello que, en el código de “random”, se indica un rango de 1 a 7, puesto que este último valor se excluye.
Número de tiros	Al establecer que el número de tiros para el jugador y la casa resulta en la mitad de estos, el usuario únicamente puede ingresar valores pares. De lo contrario, se despliega un mensaje de error, indicando al usuario la forma correcta para ingresar dicho parámetro.

Fuente: Elaboración propia en base al “Proyecto Práctico No. 2”, (2023).

DISEÑO

DIAGRAMA DE FLUJO

Diagrama No. 1: Código del juego “dados”



Fuente: Elaboración propia en base a Lucid Chart, (2023).

DIAGRAMA DE CLASE

Diagrama No. 2: Clase del juego “dados”

Clase	Dados
Atributos	
Random random	
int jugadorPuntos	
int casaPuntos	
int jugadorPuntosFinal	
int tirosGanadosJugador	
int tirosGanadosCasa	
int tirosPares	
int tirosImpares	
int tirosIguales	
int partidasGanadasJugador	
int partidasGanadasCasa	
Métodos	
Jugar(int totalPartidas, int tirosPartida)	
JugarPartida(int tirosPartida)	
TiroJugador(int sumaDados, int tiro)	
TiroCasa(int sumaDados2, int tiro)	
Tiro(int dado1, int dado2)	
MostrarResultados(int totalPartidas, int tirosPartida)	
MostrarResultadosFinales(int totalPartidas, int tirosPartida)	

Nota: Cabe destacar que se emplea en los atributos la clase “Random” y el objeto “random = new Random” en su constructor dentro de la clase principal de “Dados”.
Fuente: Elaboración propia en base a Visual Studio en Excel, (2023).

CONCLUSIONES

1. El uso de valores aleatorios dentro del desarrollo de juegos de entretenimiento permite que el usuario tenga una mejor experiencia y una mayor facilidad para jugar un número repetido de veces, permitiendo a su vez que exista una mayor transparencia en los resultados del juego.

2. El uso de clases, métodos, bucles y condiciones permite crear una mejor estructura y legibilidad en el código, tanto para el programador como para el usuario, permitiendo que el banco de memoria sea aprovechado y que exista un mejor orden en cada paso del programa.

3. La programación puede implementarse en diversos ámbitos de la vida, no solo en aspectos prácticos, sino en actividades recreativas que permitan mejorar la calidad de experiencia del usuario en juegos de entretenimiento.

RECOMENDACIONES

1. Familiarizarse con el uso de códigos que simplifiquen el espacio de trabajo, tales como bucles y condiciones, e investigar en páginas confiables el uso de códigos desconocidos para el correcto funcionamiento del programa.

2. Correr el programa un número considerable de veces para asegurarse de que el programa cumpla con los requerimientos de manera correcta, efectiva y sin problemas, verificando que se hayan previsto los posibles errores y restricciones.

3. Previo a la codificación, determinar los puntos más importantes del programa con el uso de diagramas, indicando los objetivos, las entradas y salidas, así como la información que el usuario debe ingresar.

REFERENCIAS

1. Microsoft (2023) *Crear un diagrama de flujo básico en Visio*
   https://support.microsoft.com/es-es/office/crear-un-diagrama-de-flujo-b%C3%A1sico-en-visio-e207d975-4a51-4bfa-a356-eeec314bd276
   Se utilizó para crear el diagrama de flujo del proyecto.

2. Microsoft (2023) *Random Clase*
   https://learn.microsoft.com/es-es/dotnet/api/system.random?view=net-7.0
   Se utilizó para aprender a utilizar el código “random” y obtener valores aleatorios en los dados.

3. StackOverFlow (2017) *¿Cómo especificar la cantidad de decimales de un double?*
   https://es.stackoverflow.com/questions/91442/c%C3%B3mo-especificar-la-cantidad-de-decimales-de-un-double
   Se utilizó para indicar la cantidad de decimales en el porcentaje de probabilidades de ganar.

ANEXOS

MANUAL DEL USUARIO

Forma de uso

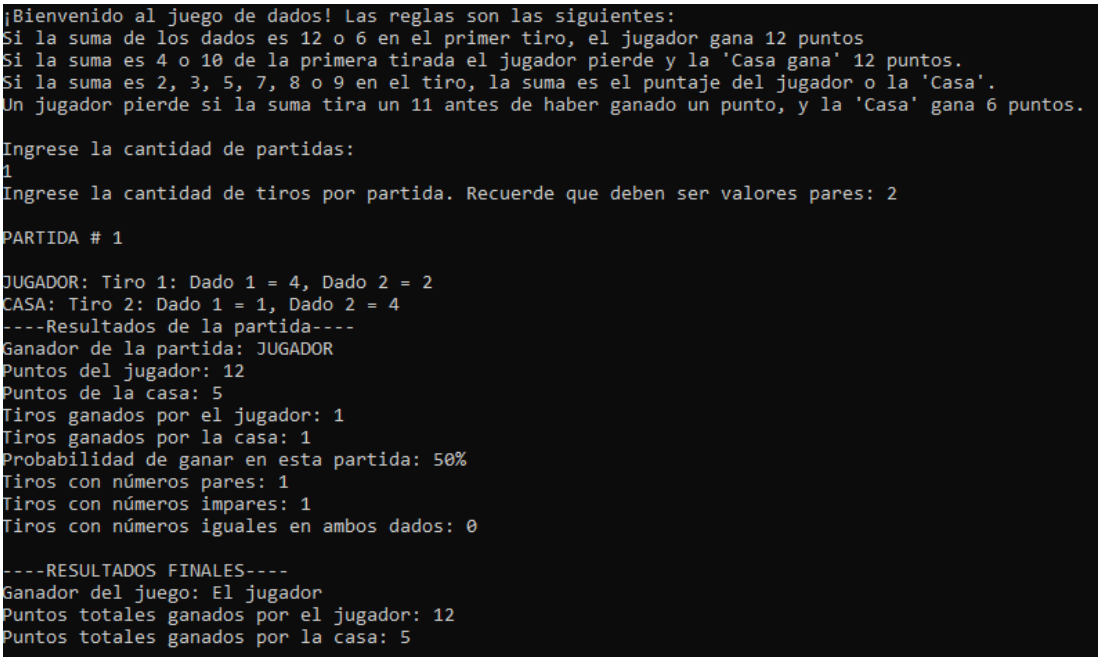
El presente código representa una versión del juego “Dados” en el que el jugador puede lanzar dos dados y considerar la suma de ambos para ganar o perder contra “la casa”. Para que el usuario lo utilice, primero se desplegará una lista de reglas del juego a modo de introducción:

- Si la suma de los dados es 12 o 6 en el primer tiro, el jugador gana 12 puntos
- Si la suma es 4 o 10 de la primera tirada el jugador pierde y la 'Casa gana' 12 puntos.
- Si la suma es 2, 3, 5, 7, 8 o 9 en el tiro, la suma es el puntaje del jugador o la 'Casa'.
- Un jugador pierde si la suma tira un 11 antes de haber ganado, la 'Casa' gana 6 puntos.

Tras esto, el jugador debe ingresar los parámetros del número de partidas y tiros que desee jugar, considerando que únicamente deben ser tiros pares. Luego, el código desplegará una serie de valores aleatorios para ambos dados en cada tiro de las partidas, indicando los correspondientes para el jugador y la casa. Cabe destacar que para este último punto se considera la mitad de tiros para el jugador y la otra mitad para la casa.

Siguiendo con las reglas previamente establecidas, se realizará un cálculo automático de la suma de puntajes, empleando un acumulador para contar las partidas ganadas por parte del jugador y la casa. Finalmente, se desplegarán los resultados indicando primero el ganador del juego, y luego información extra sobre el resultado de la partida, como los tiros ganados, los tiros pares, impares e iguales y el porcentaje de probabilidad de ganar. Tales resultados se indicarán en cada partida, desplegando los resultados finales en relación del juego en su totalidad.

Figura No. 3: Despliegue del código



Fuente: Elaboración propia en base a Visual Studio, (2023).

Códigos empleados

1. **Console WriteLine**
   Empleado para desplegar la información al correr el programa.

   Figura No. 4: ConsoleWriteLine

```
Console.WriteLine("Ganador de la partida: " + ganador);
Console.WriteLine("Puntos del jugador: " + jugadorPuntos);
Console.WriteLine("Puntos de la casa: " + casaPuntos);
Console.WriteLine("Tiros ganados por el jugador: " + tirosGanadosJugador);
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

2. **Console.ReadLine:**
   Empleado para que el usuario escriba información y se utilice en el análisis posterior dentro del código.

   Figura No. 5: ConsoleReadLine

```
Console.WriteLine("\nIngrese la cantidad de partidas: ");
int totalPartidas = int.Parse(Console.ReadLine());
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

3. **Tipos de datos:**
   - **double:** se utilizaron los datos del tipo double para valores decimales, como el porcentaje de la probabilidad de ganar.
     Figura No. 6: Double

```
double probabilidadDeGanar = (double)
double probabilidadDeGanarporcentaje
```

     Fuente: Elaboración propia en base a Visual Studio, (2023).

   - **int:** se emplearon para valores exactos, tales como el número de partidas y tiros.
     Figura No. 7: Int

```
int totalPartidas
```

     Fuente: Elaboración propia en base a Visual Studio, (2023).

   - **string:** se emplearon para desplegar el ganador del juego, “la casa” o “el jugador”.
     Figura No. 8: String

```
string ganador
```

     Fuente: Elaboración propia en base a Visual Studio, (2023).

4. **int.Parse:** se utilizó para realizar conversiones cuando se recibía información por parte del usuario y se requería del tipo entero para su análisis dentro del programa.

   Figura No. 9: int.Parse

```
int.Parse(Console.ReadLine());
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

5. **if**

   Se empleó el código de "if" como bucle para indicar que si la condición se cumplía mostrara las siguientes partes del código que contenía. Por ejemplo, se empleó para determinar si el tiro era par o impar, usando else if para este segundo caso.

   Figura No. 10: if

```
if ((dado1 + dado2) % 2 == 0)
{
    tirosPares++;
}
else if ((dado1 + dado2) % 2 != 0)
{
    tirosImpares++;
}

if (dado1 == dado2)
{
    tirosIguales++;
}
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

6. **For**

   Se empleó el bucle for para establecer que se repitiera el código mientras el argumento se cumpliera. Por ejemplo, para la Figura No. 11, primero se estableció el contador "partida" como valor entero, igualándolo a 1 según el número de partidas y que continúe mientras su valor fuera menor que el número total de partidas.

   Figura No. 11: For

```
for (int partida = 1; partida <= totalPartidas; partida++)
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

7. **+=**

   Se empleó para indicar una suma entre la variable de la izquierda más la variable de la derecha.

   Figura No. 12: Sumas

```
jugadorPuntos += 12;
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

8. **Contador**

   Se empleó para llevar un conteo sobre el número de tiros que el jugador ganara.

   Figura No. 13: Contador

```
tirosGanadosJugador++;
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

9. **Random**

   Se utilizó para establecer valores aleatorios en los dos dados. Para ello, primero se estableció el atributo en la clase Dados como “Random random”, indicando lo primero como el tipo de variable. Luego, se creó el objeto “new Random” para generar números aleatorios en todas las partidas. Por último, en el método se empleó como “random.Next(1,7)”, donde los valores entre paréntesis indican el rango de números aleatorios, donde el último representa un extremo que no se incluye.

   Figura No. 14: Random

```
int dado1 = random.Next(1, 7);
```

   Fuente: Elaboración propia en base a Visual Studio, (2023).

10. **Clases**

    Se utilizó para organizar y definir las variables dentro del código, junto con los métodos a emplear. En este caso, se colocaron como privadas, para emplearlas en métodos públicos más adelante.

    Figura No. 15: Atributos en la clase utilizada

```
private Random random;
private int jugadorPuntos;
private int casaPuntos;
private int jugadorPuntosFinal;
private int casaPuntosFinal;
private int tirosGanadosJugador;
private int tirosGanadosCasa;
private int tirosPares;
private int tirosImpares;
private int tirosIguales;
private int partidasGanadasJugador;
private int partidasGanadasCasa;
```

    Fuente: Elaboración propia en base a Visual Studio, (2023).

11. Métodos

Dentro de cada clase existen los métodos, los cuales son las acciones específicas que se llevan a cabo dentro del código con los atributos. En este caso, el método se denominó como “Jugar”, utilizando los atributos de “totalPartidas” y “tirosPartida” para luego poder utilizar los resultados obtenidos en el objeto.

Figura No. 16: Métodos

```
1 referencia
public void Jugar(int totalPartidas, int tirosPartida)
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

12. Objeto

Se utilizó el objeto de juego = new Dados para desplegar los resultados obtenidos luego de que el programa corriera toda la información obtenida, desde el número de partidas y tiros hasta los valores aleatorios de los tiros y los resultados.

Figura No. 17: Objeto

```
Dados juego = new Dados();
juego.Jugar(totalPartidas, tirosPartida);
juego.MostrarResultadosFinales(totalPartidas);
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

13. Switch, case y break

Se utilizó para indicar condiciones al cumplirse una suma de valores en los datos determinada. Dentro de “Switch” se indicó el parámetro a trabajar, la suma de los dados, y luego se estableció cada condición con un “case”, donde una suma correspondía a una acción específica y, al cumplirla, se indicaba con el código “break”.

Figura No. 18: Switch

```
switch (sumaDados)
{
    case 12:
    case 6:
        if (tiro == 1)
        {
            jugadorPuntos += 12;
            tirosGanadosJugador++;
            jugadorPuntosFinal += 12;
        }
        break;
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

14. Do y while

Se empleó “Do” y “While” para indicar que los tiros se consideraran para el código únicamente si contenían valores pares, continuando el bucle mientras se cumpliera dicha condición.

Figura No. 19: Do

```
do
{
    Console.Write("Ingrese la cantidad de tiros por partida. Recuerde que deben ser valores pares: ");
    tirosPartida = int.Parse(Console.ReadLine());

    if (tirosPartida % 2 != 0)
    {
        Console.WriteLine("Error: La cantidad de tiros por partida debe ser un número par. Intente nuevamente.");
    }
} while (tirosPartida % 2 != 0);
```

Fuente: Elaboración propia en base a Visual Studio, (2023).

15. MathRound

Se utilizó para indicar el número de decimales en el porcentaje de las probabilidades de ganar, indicando primero la variable que se desea modificar y luego el número de decimales.

Figura No. 20: MathRound

```
Math.Round(probabilidadDeGanarporcentaje, 2);
```

Fuente: Elaboración propia en base a Visual Studio, (2023).