

Universidad Rafael Landívar
Facultad de Ingeniería
Ingeniería Química
Introducción a la Programación – Sección 07
Catedrático: Ing. Edwin Timoteo Chocoy Cordón

PROYECTO PRÁCTICO No. 02
“DADOS”

Amada Jimena García Cordero
1092223

Guatemala, 13 de noviembre de 2023

ÍNDICE

INTRODUCCIÓN	i
ANÁLISIS.....	1
ENTRADAS.....	1
SALIDAS	2
PROCESOS	3
DISEÑO	4
DIAGRAMA DE FLUJO	4
DIAGRAMA DE CLASES	5
CONCLUSIONES	6
RECOMENDACIONES	7
REFERENCIAS	8
LIBRERÍAS UTILIZADAS Y SU UTILIZACIÓN	8
REFERENCIAS BIBLIOGRÁFICAS	8
ANEXOS.....	9
MANUAL DE USUARIO	9

INTRODUCCIÓN

Los juegos de dados han sido una constante fuente de entretenimiento y emoción a lo largo de la historia, cautivando a las personas con su intrigante combinación de suerte y estrategia. Entre estos juegos, destaca uno en particular: el que implica el uso de dos dados de seis caras, cada uno con los números del 1 al 6. En este juego, la acción principal es sumar los números que aparecen en ambos dados, lo que desencadena una serie de reglas que definen el curso del juego. Las posibilidades son diversas, ya que el resultado puede llevar al jugador a la victoria, a enfrentarse con la "Casa" o incluso a perder por varias razones.

Dentro de este programa, se invita al usuario a tomar las riendas, permitiéndole elegir cuántas partidas desea jugar y cuántos tiros se llevarán a cabo en cada una. Lo interesante del programa radica en la generación aleatoria de los números correspondientes a cada lanzamiento de los dados, aplicando las reglas correspondientes al juego en función de estos números. Cada resultado se registra meticulosamente, permitiendo obtener al final datos detallados que serán el corazón del análisis.

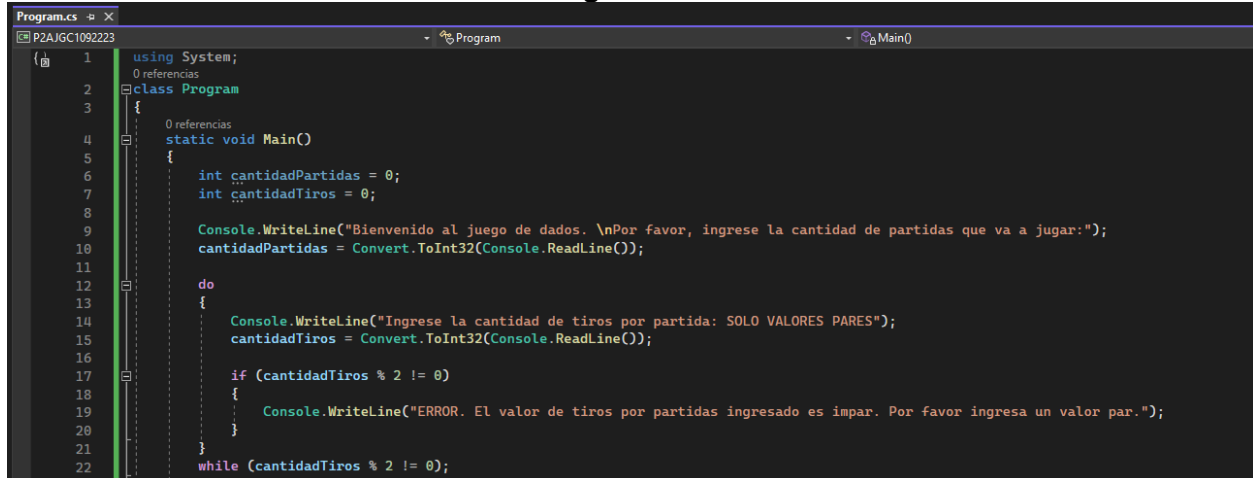
La exploración de los datos recopilados abarcará diversos aspectos cruciales para entender el desenvolvimiento del juego. Se analizará quién resulta ganador en cada partida, se llevará un seguimiento minucioso de los números específicos generados por cada dado, se contabilizará la cantidad de tiros exitosos obtenidos por el jugador, se calculará la probabilidad de su éxito, se estudiará la distribución entre números pares e impares, se evaluará la incidencia de números iguales y, finalmente, se obtendrá el puntaje total al concluir todas las partidas.

En la elaboración del programa, se optó por emplear Visual Studio, un entorno de desarrollo integrado ampliamente reconocido por su capacidad para que los programadores creen aplicaciones adaptadas a múltiples plataformas. Este software proporciona una extensa gama de herramientas destinadas a agilizar el proceso de desarrollo, facilitando desde la escritura de código hasta la depuración de este, integración con diferentes sistemas, implementación de pruebas automatizadas y una serie de funciones que mejoran la eficiencia en el desarrollo de software.

ANÁLISIS

ENTRADAS

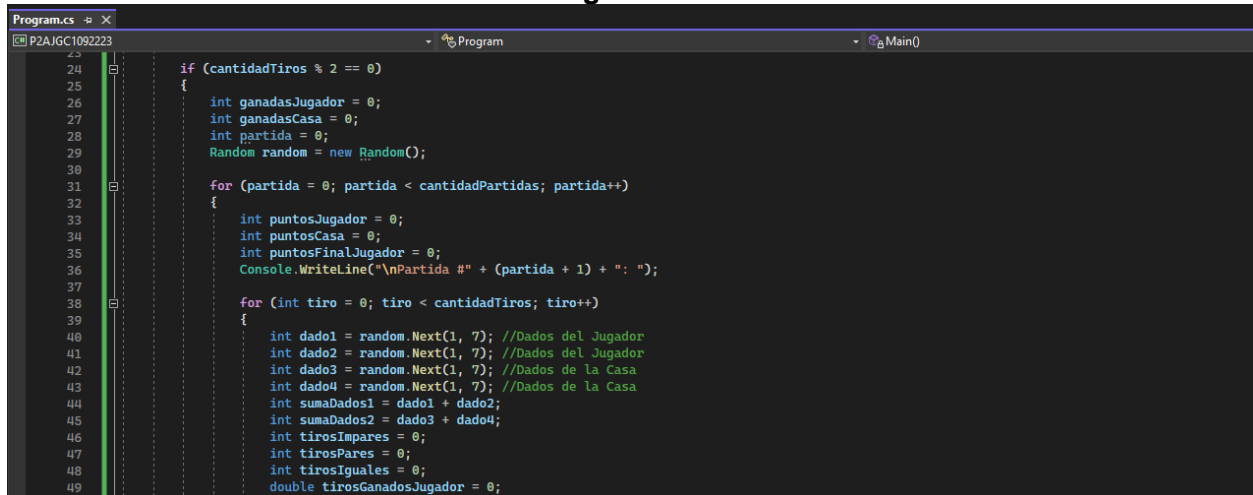
Imagen No. 01



```
1 using System;
2
3 class Program
4 {
5     static void Main()
6     {
7         int cantidadPartidas = 0;
8         int cantidadTiros = 0;
9
10        Console.WriteLine("Bienvenido al juego de dados. \nPor favor, ingrese la cantidad de partidas que va a jugar:");
11        cantidadPartidas = Convert.ToInt32(Console.ReadLine());
12
13        do
14        {
15            Console.WriteLine("Ingrese la cantidad de tiros por partida: SOLO VALORES PARES");
16            cantidadTiros = Convert.ToInt32(Console.ReadLine());
17
18            if (cantidadTiros % 2 != 0)
19            {
20                Console.WriteLine("ERROR. El valor de tiros por partidas ingresado es impar. Por favor ingresa un valor par.");
21            }
22        } while (cantidadTiros % 2 != 0);
23    }
24 }
```

Esta es la entrada principal del código donde se indica que la clase que se usó es “class Program” la cual usa el método “static void Main()”. Después se declararon las variables que se usarán para que el usuario pueda agregar la cantidad de partidas que jugará y los tiros que se harán en cada partida. Aquí podemos encontrar también una condición, la cual es que la cantidad de tiros por partida solo puede ser par y si no se cumple con esta función se muestra un mensaje de error y se le vuelve a pedir al usuario que ingrese un nuevo valor para la cantidad de tiros.

Imagen No. 02

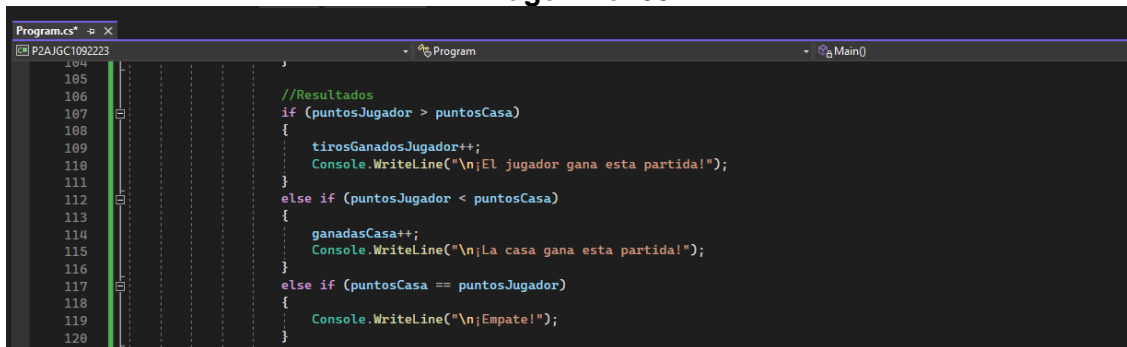


```
24 if (cantidadTiros % 2 == 0)
25 {
26     int ganadasJugador = 0;
27     int ganadasCasa = 0;
28     int partida = 0;
29     Random random = new Random();
30
31     for (partida = 0; partida < cantidadPartidas; partida++)
32     {
33         int puntosJugador = 0;
34         int puntosCasa = 0;
35         int puntosFinalJugador = 0;
36         Console.WriteLine("\nPartida #" + (partida + 1) + ": ");
37
38         for (int tiro = 0; tiro < cantidadTiros; tiro++)
39         {
40             int dado1 = random.Next(1, 7); //Dados del Jugador
41             int dado2 = random.Next(1, 7); //Dados del Jugador
42             int dado3 = random.Next(1, 7); //Dados de la Casa
43             int dado4 = random.Next(1, 7); //Dados de la Casa
44             int sumaDados1 = dado1 + dado2;
45             int sumaDados2 = dado3 + dado4;
46             int tirosImpares = 0;
47             int tirosPares = 0;
48             int tirosIguales = 0;
49             double tirosGanadosJugador = 0;
50         }
51     }
52 }
```

Esta es otra entrada del programa ya que se declaran las demás variables que se usaran dentro del código, aquí también se nombra la función “Random” que hará que los valores de los dados de cada tiro sean aleatorios. Estas variables se encuentran ya dentro de condiciones que más adelante cumplirán las condiciones y operaciones para el desarrollo del juego.

SALIDAS

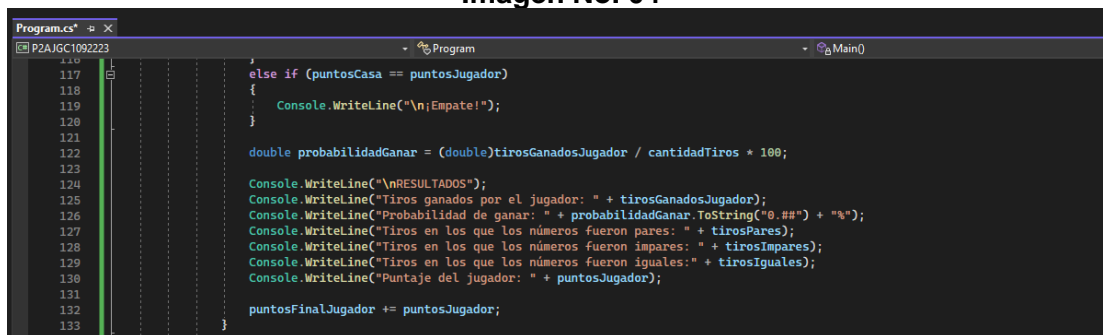
Imagen No. 03



```
Program.cs
P2AJGC1092223
//Resultados
if (puntosJugador > puntosCasa)
{
    tirosGanadosJugador++;
    Console.WriteLine("\nEl jugador gana esta partida!");
}
else if (puntosJugador < puntosCasa)
{
    ganadasCasa++;
    Console.WriteLine("\nLa casa gana esta partida!");
}
else if (puntosCasa == puntosJugador)
{
    Console.WriteLine("\nEmpate!");
}
```

En esta parte del código se muestra la primera salida ya que aquí se determina si la casa o el jugador ganó la partida mediante la comparación de puntos que obtuvo cada uno.

Imagen No. 04



```
Program.cs
P2AJGC1092223
else if (puntosCasa == puntosJugador)
{
    Console.WriteLine("\nEmpate!");
}

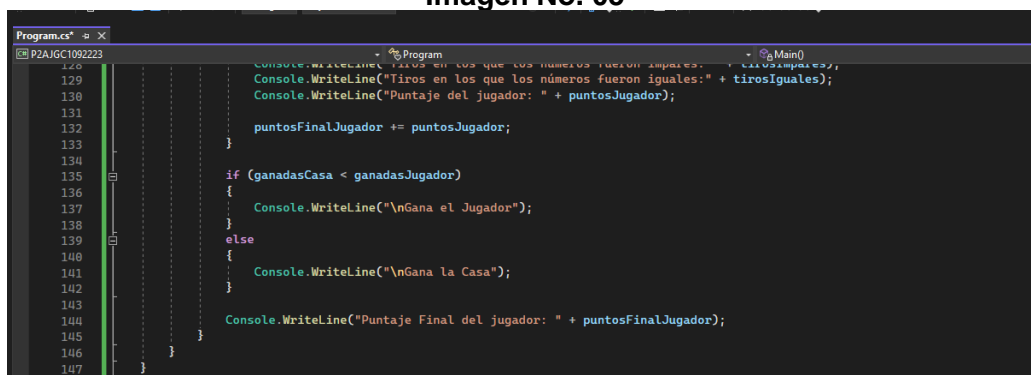
double probabilidadGanar = (double)tirosGanadosJugador / cantidadTiros * 100;

Console.WriteLine("\nRESULTADOS");
Console.WriteLine("Tiros ganados por el jugador: " + tirosGanadosJugador);
Console.WriteLine("Probabilidad de ganar: " + probabilidadGanar.ToString("0.##") + "%");
Console.WriteLine("Tiros en los que los números fueron pares: " + tirosPares);
Console.WriteLine("Tiros en los que los números fueron impares: " + tirosImpares);
Console.WriteLine("Tiros en los que los números fueron iguales: " + tirosIguales);
Console.WriteLine("Puntaje del jugador: " + puntosJugador);

puntosFinalJugador += puntosJugador;
```

Esta es la segunda salida del código, aquí se muestran los resultados más detallados de la partida con información sobre los tiros y si estos fueron pares, impares o iguales, también en cuántos tiros ganó el jugador, la probabilidad que tenía el jugador de ganar y los puntos del jugador.

Imagen No. 05



```
Program.cs
P2AJGC1092223
Console.WriteLine("Tiros en los que los números fueron iguales: " + tirosIguales);
Console.WriteLine("Puntaje del jugador: " + puntosJugador);

puntosFinalJugador += puntosJugador;

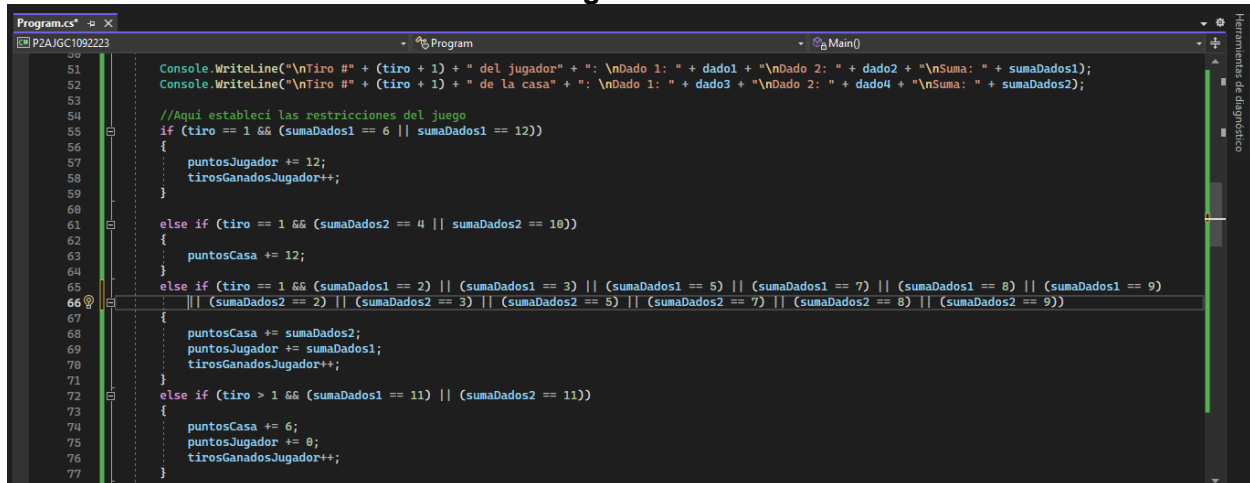
if (ganadasCasa < ganadasJugador)
{
    Console.WriteLine("\nGana el Jugador");
}
else
{
    Console.WriteLine("\nGana la Casa");
}

Console.WriteLine("Puntaje Final del jugador: " + puntosFinalJugador);
```

Esta es la última salida del programa, la cual muestra el resultado final del juego y si ganó el jugador o la casa y el puntaje final del jugador.

PROCESOS

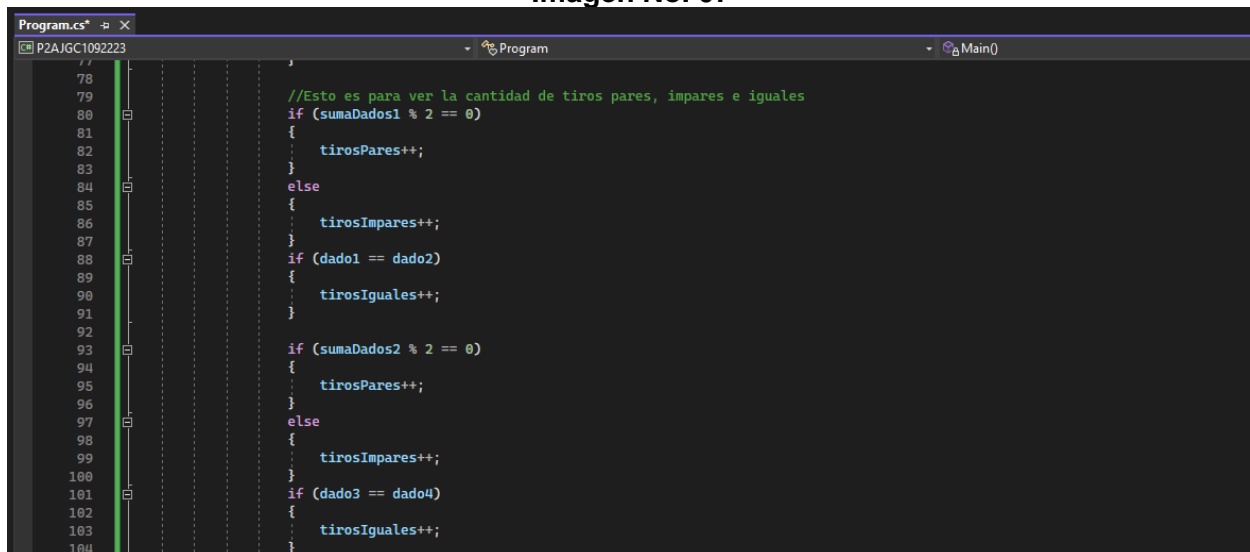
Imagen No. 06

A screenshot of a Visual Studio code editor window titled 'Program.cs' with a file path 'P2AJGC1092223'. The code is in C# and shows a 'Main()' method. It contains several console write lines for displaying game state and a series of conditional statements. A comment at line 54 reads '//Aquí establecí las restricciones del juego'. The code uses 'if' and 'else if' statements to check conditions on 'tiros' (shots) and 'sumas' (sums) to update 'puntosJugador' (player points), 'puntosCasa' (house points), and 'tirosGanadosJugador' (player's winning shots).

```
51 Console.WriteLine("\nTiros #" + (tiros + 1) + " del jugador" + ": \nDado 1: " + dado1 + "\nDado 2: " + dado2 + "\nSuma: " + sumaDados1);
52 Console.WriteLine("\nTiros #" + (tiros + 1) + " de la casa" + ": \nDado 1: " + dado3 + "\nDado 2: " + dado4 + "\nSuma: " + sumaDados2);
53
54 //Aquí establecí las restricciones del juego
55 if (tiros == 1 && (sumaDados1 == 6 || sumaDados1 == 12))
56 {
57     puntosJugador += 12;
58     tirosGanadosJugador++;
59 }
60
61 else if (tiros == 1 && (sumaDados2 == 4 || sumaDados2 == 10))
62 {
63     puntosCasa += 12;
64 }
65
66 else if (tiros == 1 && (sumaDados1 == 2) || (sumaDados1 == 3) || (sumaDados1 == 5) || (sumaDados1 == 7) || (sumaDados1 == 8) || (sumaDados1 == 9)
67 || (sumaDados2 == 2) || (sumaDados2 == 3) || (sumaDados2 == 5) || (sumaDados2 == 7) || (sumaDados2 == 8) || (sumaDados2 == 9))
68 {
69     puntosCasa += sumaDados2;
70     puntosJugador += sumaDados1;
71     tirosGanadosJugador++;
72 }
73
74 else if (tiros > 1 && (sumaDados1 == 11) || (sumaDados2 == 11))
75 {
76     puntosCasa += 6;
77     puntosJugador += 0;
78     tirosGanadosJugador++;
79 }
```

En esta parte del código se establecen las restricciones del juego y se asignan los puntos que serán dados a la casa o al jugador dependiendo la regla.

Imagen No. 07

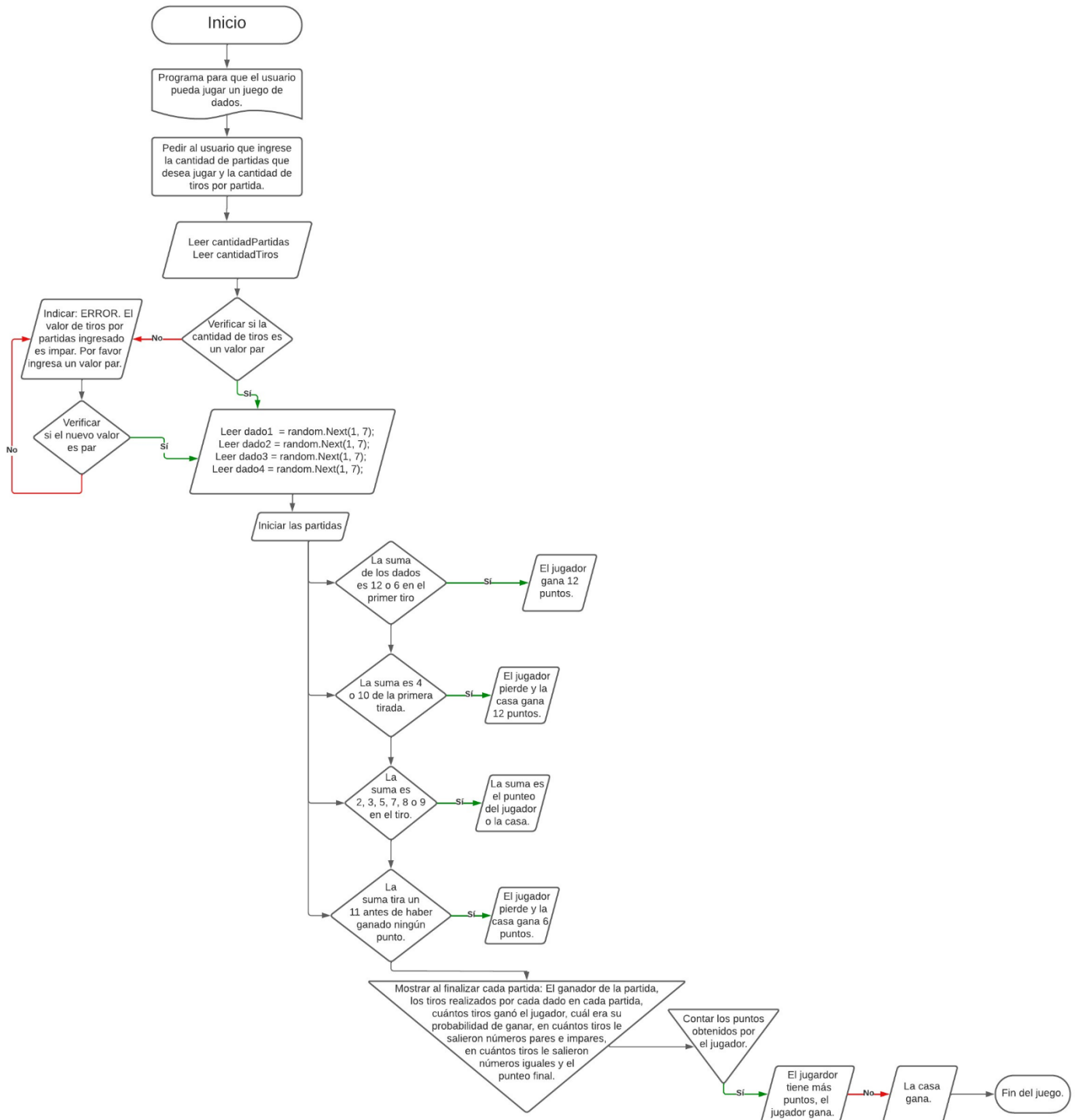
A screenshot of a Visual Studio code editor window titled 'Program.cs' with a file path 'P2AJGC1092223'. The code is in C# and shows a 'Main()' method. It contains conditional statements to count the number of even, odd, and equal shots. A comment at line 78 reads '//Esto es para ver la cantidad de tiros pares, impares e iguales'. The code uses 'if' and 'else' statements to increment 'tirosPares' (even shots), 'tirosImpares' (odd shots), and 'tirosIguales' (equal shots) based on the values of 'dado1', 'dado2', 'dado3', and 'dado4'.

```
78 //Esto es para ver la cantidad de tiros pares, impares e iguales
79 if (sumaDados1 % 2 == 0)
80 {
81     tirosPares++;
82 }
83
84 else
85 {
86     tirosImpares++;
87 }
88
89 if (dado1 == dado2)
90 {
91     tirosIguales++;
92 }
93
94 if (sumaDados2 % 2 == 0)
95 {
96     tirosPares++;
97 }
98
99 else
100 {
101     tirosImpares++;
102 }
103
104 if (dado3 == dado4)
105 {
106     tirosIguales++;
107 }
```

En esta parte del proceso se estableció cuando los tiros eran pares, impares o iguales para que estos se mostraran en los resultados finales.

DISEÑO

DIAGRAMA DE FLUJO



Fuente: Elaboración Propia (2023)

DIAGRAMA DE CLASES

En el código proporcionado, se emplea una única clase denominada "Program", que actúa como el punto de entrada principal. Por convención, esta clase suele contener un método especial llamado "Main". En este contexto, la clase "Program" encapsula el código esencial para iniciar la ejecución del programa.

La utilidad de las clases radica en la capacidad para organizar el código de forma estructurada y modular. Al agrupar propiedades y comportamientos afines en una entidad única, se facilita el mantenimiento y la comprensión del código. Este enfoque contribuye a una mejor organización visual, ya que las variables, funciones y operaciones se distribuyen de manera ordenada en bloques definidos por las clases.

Es importante señalar que el código puede funcionar sin la presencia explícita de clases adicionales. En este caso, solo se emplea la clase "Program" como elemento fundamental, marcando el punto de inicio del programa. Aunque las clases enriquecen la estructura del código, el programa demuestra que, en ciertos casos, es posible omitir el uso de clases específicas, enfocándose en una implementación más simplificada.

En el caso de este código en particular, la creación de un diagrama de clases no resulta aplicable, ya que no se requirió la implementación de clases adicionales. En los escenarios donde se utilizaron clases, su propósito principal era proporcionar una estructura organizativa más que cumplir funciones esenciales en la lógica del programa, es decir, si bien se pudieron haber empleado clases con el fin de mantener un orden estructurado en el código, su presencia no fue determinante para la funcionalidad principal del programa.

.

CONCLUSIONES

1. El código realiza una validación de la entrada de datos para la cantidad de tiros por partida, asegurándose de que solo se ingresen valores pares. Sin embargo, la validación se realiza dentro de un bucle do-while, lo que garantiza que el usuario ingrese un valor par antes de continuar. Esto para evitar posibles errores relacionados con valores impares.
2. El código proporciona una salida detallada después de cada partida, mostrando información relevante como los resultados de los dados, el puntaje de cada jugador, la probabilidad de ganar y la cantidad de tiros pares, impares e iguales. Esta salida detallada es útil para comprender el progreso del juego y los resultados obtenidos en cada partida.
3. En el código, la clase única "Program" actúa como el punto de entrada principal, encapsulando el código esencial para iniciar la ejecución del programa. Aunque las clases son herramientas poderosas para organizar el código de manera estructurada y modular, este ejemplo demuestra que, en ciertos casos, es posible omitir el uso de clases adicionales. La implementación se centra en una estructura más simplificada, utilizando la clase "Program" como elemento fundamental. Aunque las clases podrían haberse utilizado para mantener un orden estructurado, su ausencia no afecta la funcionalidad principal del programa.

RECOMENDACIONES

1. Se recomienda dividir el código en métodos más pequeños y específicos, cada uno encargado de una tarea particular. Esto mejoraría la legibilidad del código y facilitaría su mantenimiento.
2. Añadir comentarios explicativos a lo largo del código puede hacer que sea más comprensible para otros desarrolladores. Además, se podría incluir documentación en el código para explicar las reglas del juego y la lógica detrás de las decisiones tomadas en cada paso.
3. En situaciones más complejas, considerar la introducción de clases adicionales, incluso si no son esenciales para la funcionalidad principal. Esto puede mejorar la claridad y mantenibilidad a medida que el programa evoluciona. La modularidad que ofrecen las clases facilita la expansión y modificación del código en el futuro, proporcionando una base más sólida para posibles actualizaciones o incorporación de nuevas funcionalidades.

REFERENCIAS

LIBRERÍAS UTILIZADAS Y SU UTILIZACIÓN

using System: Es un espacio de nombres en el que se definen varias clases útiles, significa que se está utilizando la Systembiblioteca en el proyecto. Lo que le brinda algunas clases útiles como Console y funciones/métodos como WriteLine.

class Program: Es el nombre de la clase de punto de entrada. A diferencia de Java, que requiere que le pongas un nombre Main, se puede nombrar como quieras en C#.

static void Main(): Es el método de punto de entrada de un programa. Este método se llama antes que nada dentro del programa.

REFERENCIAS BIBLIOGRÁFICAS

EDraw. (2023). *Cómo crear un Diagrama de Flujo*. Wondershare.

<https://www.edrawsoft.com/es/create-programming-flowchart.html>

APR. (s.f.) *Tipos de variables en Visual Basic*.

https://www.aprenderaprogramar.com/index.php?option=com_content&view=article&id=140:tipos-de-variables-en-visual-basic-integer-single-double-string-object-boolean-etc-ejemplos-cu00308a&catid=37&Itemid=61

DevCamp. (s.f.). *Qué es una librería en programación*. <https://devcamp.es/que-es-libreria-programacion/>

Armetrics. (s.f.). *Qué es Visual Studio*. <https://www.armetrics.com/glosario-digital/visual-studio>

ANEXOS

MANUAL DE USUARIO

1. El usuario debe ingresar al programa si desea jugar el juego de los dados.
2. Se deben conocer las reglas del juego y las restricciones de este:
 - Se deben ingresar la cantidad de partidas que el usuario desee jugar e ingresar los tiros que se realizarán por cada partida, este último valor debe ser par de lo contrario el programa no avanzará.
 - Cuando comienza el juego, al lanzar los dados la suma de ambos resultados implica diferentes restricciones, las cuales son:
 - 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 punteo del jugador o la “Casa”.
 - Un jugador puede perder si la suma tira un 11 antes de haber ganado ningún punto, para este caso la “Casa” gana 6 puntos.
 - Para que gane el jugar este debe tener más puntos que la casa al finalizar las partidas que se definieron al inicio del juego.
3. Los resultados se mostrarán al final de cada una de las partidas jugadas, donde se encontrará la siguiente información :
 - El ganador de la partida: El jugador o la casa.
 - Los tiros realizados en cada partida, cada dado por separado.
 - En cuántos tiros ganó puntos el jugador.
 - Cuál era la probabilidad del jugador de ganar.
 - En cuántos tiros le salieron números pares e impares.
 - En cuántos tiros le salieron números iguales.
 - Punteo final.