



## Instituto Tecnológico de Tlaxiaco

### Presenta:

Alex Antonio Victoria Vázquez    No. Control: 22026281  
Yeni Daniela Ojeda Gómez        No. Control: C22620155

### Carrera:

Ingeniería en Sistemas Computacionales

### Semestre:

4US ISC

### Asignatura:

Tópicos Avanzados De Programación

### Actividad 02:

Calculadora con librería

### Docente:

Román Cruz José Alfredo

**Tlaxiaco, Oaxaca. 18 de marzo del 2025**

### **Objetivo de la práctica:**

El propósito de esta práctica es doble: primero, adquirir experiencia en la creación y uso de librerías en C#; segundo, aplicar estos conocimientos para desarrollar una calculadora funcional. Se pretende demostrar cómo las librerías pueden mejorar la estructura del código, promover la reutilización y facilitar el desarrollo de aplicaciones más complejas.

### **Descripción:**

Esta práctica consiste en el desarrollo de una aplicación de calculadora en C# utilizando el entorno de desarrollo Visual Studio. El objetivo principal es aprender y aplicar el concepto de librerías en la programación orientada a objetos.

La calculadora implementará las operaciones aritméticas básicas: suma, resta, multiplicación y división. Para lograr esto, se creará una librería personalizada que contendrá las funciones responsables de realizar cada operación. La aplicación principal de la calculadora utilizará esta librería, demostrando así la modularización y reutilización de código.

### **Material:**

- Visual Studio



- GitHub





## Índice de imágenes

Ilustración 1: Creación de biblioteca.....	4
Ilustración 2: Método sumar .....	4
Ilustración 3: Método restar .....	4
Ilustración 4: Método multiplicar .....	5
Ilustración 5: Método dividir.....	5
Ilustración 6: Método de raíz cuadrada .....	5
Ilustración 7: Método potencia.....	6
Ilustración 8: Método porcentaje .....	6
Ilustración 9: Método fracción.....	6
Ilustración 10: Método Seno.....	6
Ilustración 11: Método coseno.....	7
Ilustración 12: Método tangente .....	7
Ilustración 13: Compilación de solución .....	7
Ilustración 14: Verificar estado de compilación .....	7
Ilustración 15: Crear aplicación de consola .....	8
Ilustración 16: Explorador de soluciones .....	8
Ilustración 17: Administrador de referencias .....	8
Ilustración 18: Explorador de archivos.....	9
Ilustración 19: Selección del archivo DLL.....	9
Ilustración 20: Agregar "using" para librería .....	10
Ilustración 21: Creación de variables .....	10
Ilustración 22: Textos del menú.....	10
Ilustración 23: Case 1 .....	11
Ilustración 24: switch completo.....	11
Ilustración 25: Catch.....	12
Ilustración 26: Método para solo 1 número .....	12
Ilustración 27: Método para 2 números .....	12
Ilustración 28: Iniciar consola .....	13
Ilustración 29: Consola ejecutada .....	13

## Procedimiento:

Para comenzar creamos un nuevo proyecto que será la biblioteca de clases, la cuál seleccionamos en las plantillas de Visual Studio

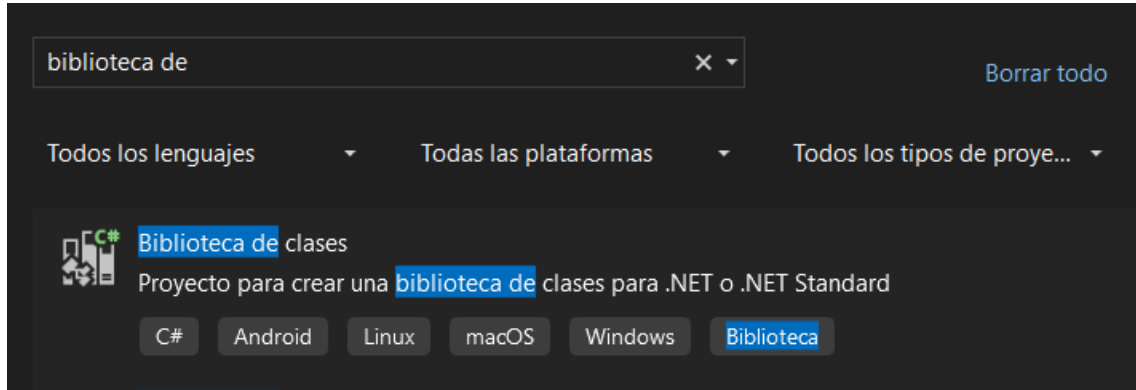


Ilustración 1: Creación de biblioteca

Se crea el método **sumar** el cual regresará los valores sumados de las variables **uno** y **dos**

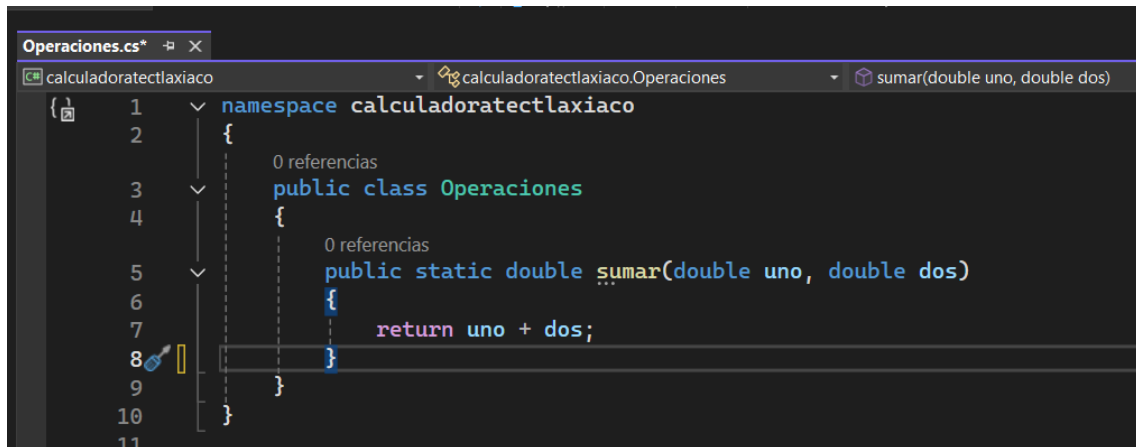


Ilustración 2: Método sumar

Se crea el método **restar** el cual regresará el valor restado de **uno** menos **dos**

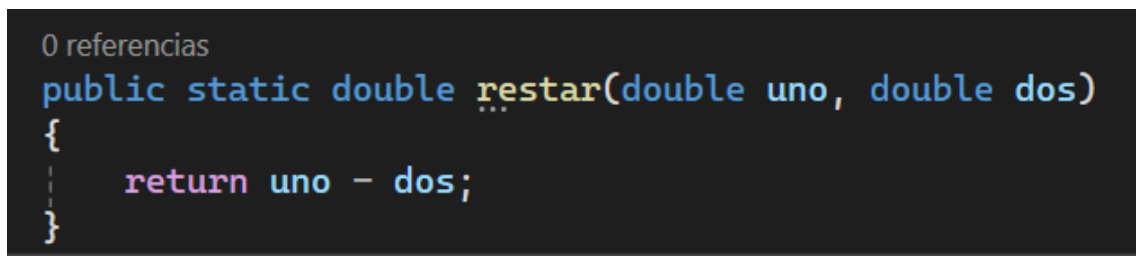


Ilustración 3: Método restar

## Creación del método **multiplicar**

```
0 referencias
public static double multiplicar(double uno, double dos)
{
    return uno * dos;
}
```

*Ilustración 4: Método multiplicar*

Creación del método **dividir** con la condicional de si se intenta dividir entre 0 mostrará una línea de código dando un error ya que no es posible

```
0 referencias
public static double dividir(double uno, double dos)
{
    if (dos == 0)
    {
        Console.WriteLine("Error: División entre cero.");
        return 0;
    }
    return uno / dos;
}
```

*Ilustración 5: Método dividir*

Creación del método **raízCuadrada** con la condicional de que si el número es menor a 0 dará error ya que no se puede calcular la raíz de un número negativo

```
0 referencias
public static double raízCuadrada(double numero)
{
    if (numero < 0)
    {
        Console.WriteLine("Error: No se puede calcular la raíz cuadrada de un número negativo.");
        return 0;
    }
    return Math.Sqrt(numero);
}
```

*Ilustración 6: Método de raíz cuadrada*

### Creación del método **potencia**

```
0 referencias
public static double potencia(double baseNum, double exponente)
{
    return Math.Pow(baseNum, exponente);
}
```

*Ilustración 7: Método potencia*

### Creación del método **porcentaje**

```
0 referencias
public static double porcentaje(double total, double porcentaje)
{
    return (total * porcentaje) / 100;
}
```

*Ilustración 8: Método porcentaje*

Creación del método **fracción** con la condicional de que el número no sea igual a 0 ya que no se puede calcular esa fracción

```
0 referencias
public static double fraccion(double numero)
{
    if (numero == 0)
    {
        Console.WriteLine("Error: No se puede calcular la fracción de cero.");
        return 0;
    }
    return 1 / numero;
}
```

*Ilustración 9: Método fracción*

### Creación del método **seno**

```
0 referencias
public static double seno(double angulo)
{
    return Math.Sin(angulo);
}
```

*Ilustración 10: Método Seno*

## Creación método **coseno**

```
0 referencias
public static double coseno(double angulo)
{
    return Math.Cos(angulo);
}
```

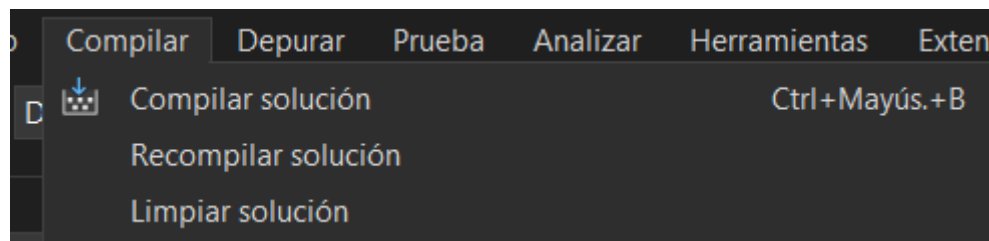
*Ilustración 11: Método coseno*

## Creación del método **tangente**

```
0 referencias
public static double tangente(double angulo)
{
    return Math.Tan(angulo);
}
```

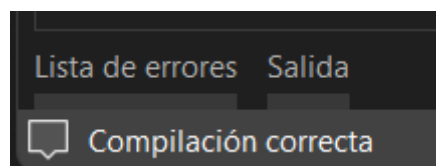
*Ilustración 12: Método tangente*

Con esos métodos quedaría hecha nuestra biblioteca, le da a **compilar** y **compilar solución**



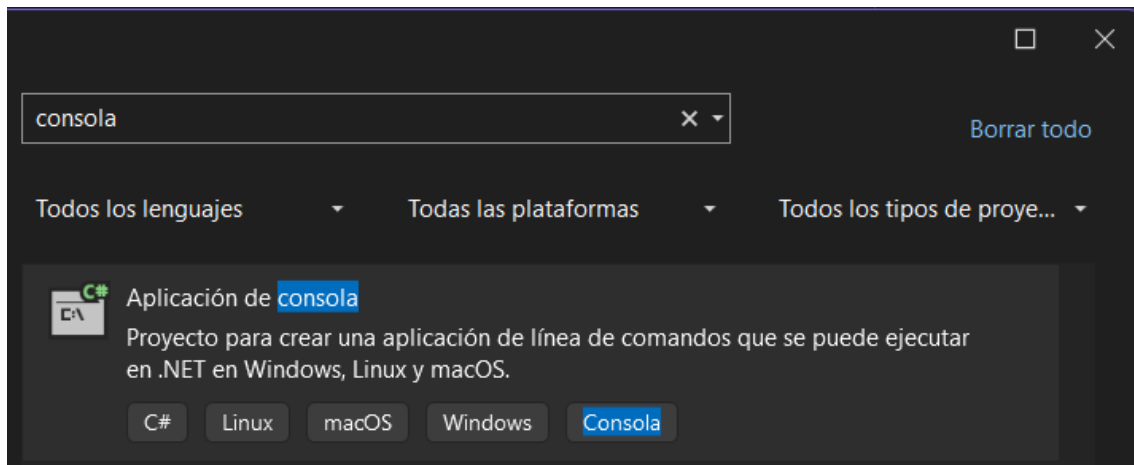
*Ilustración 13: Compilación de solución*

Verificamos que la compilación diga **compilación correcta** en la esquina inferior izquierda y continuamos



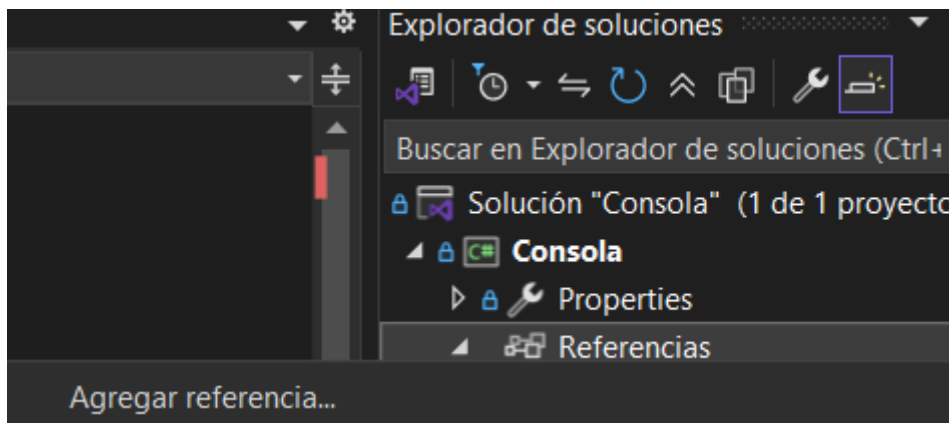
*Ilustración 14: Verificar estado de compilación*

Creamos un nuevo proyecto y en plantillas escogemos **aplicación de consola**



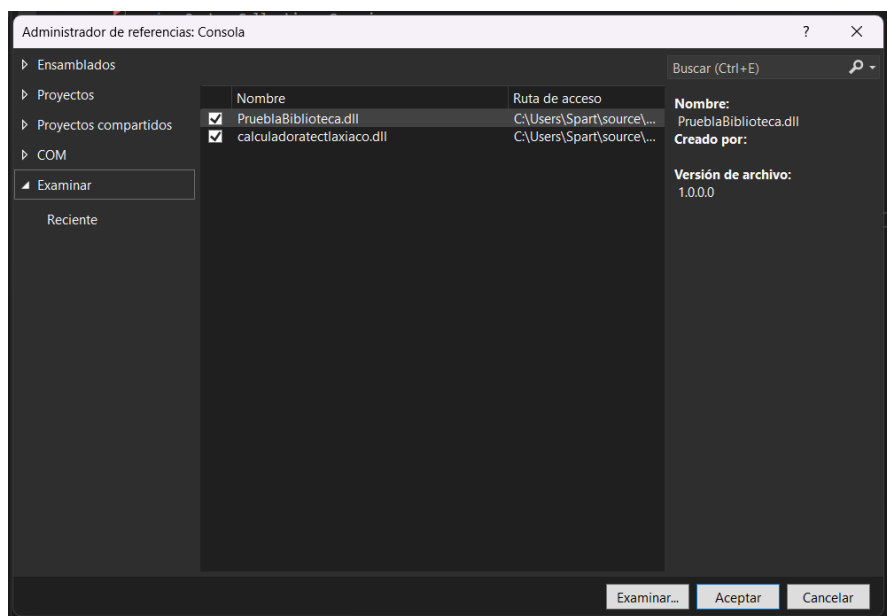
*Ilustración 15: Crear aplicación de consola*

En nuestro nuevo proyecto nos vamos al **explorador de soluciones** y en **referencias** le damos click derecho y presionamos **agregar referencia**



*Ilustración 16: Explorador de soluciones*

Se abrirá el **administrador de referencias** y le damos al botón **examinar**



*Ilustración 17: Administrador de referencias*



De ahí se abrirá el explorador de archivos y entremos a la carpeta de nuestra biblioteca en nuestro caso se llama **calculadortectlaxiaco**

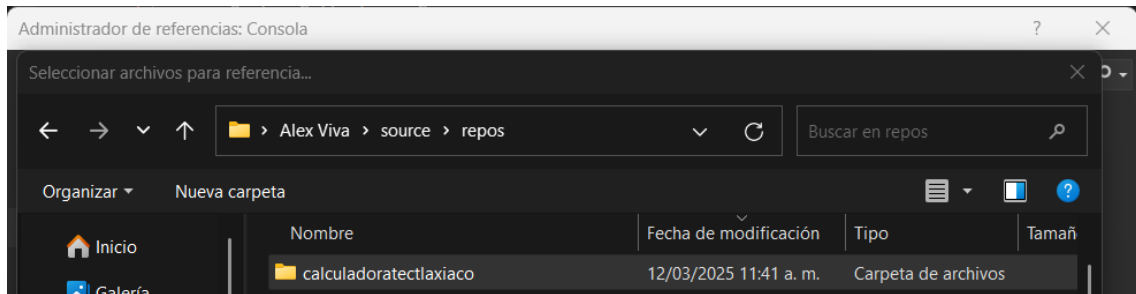


Ilustración 18: Explorador de archivos

Entramos a la carpeta, de a **bin > debug > net8.0** y dentro estará un archivo DLL que es la biblioteca que anteriormente hicimos, seleccionamos el archivo y le damos a **agregar**

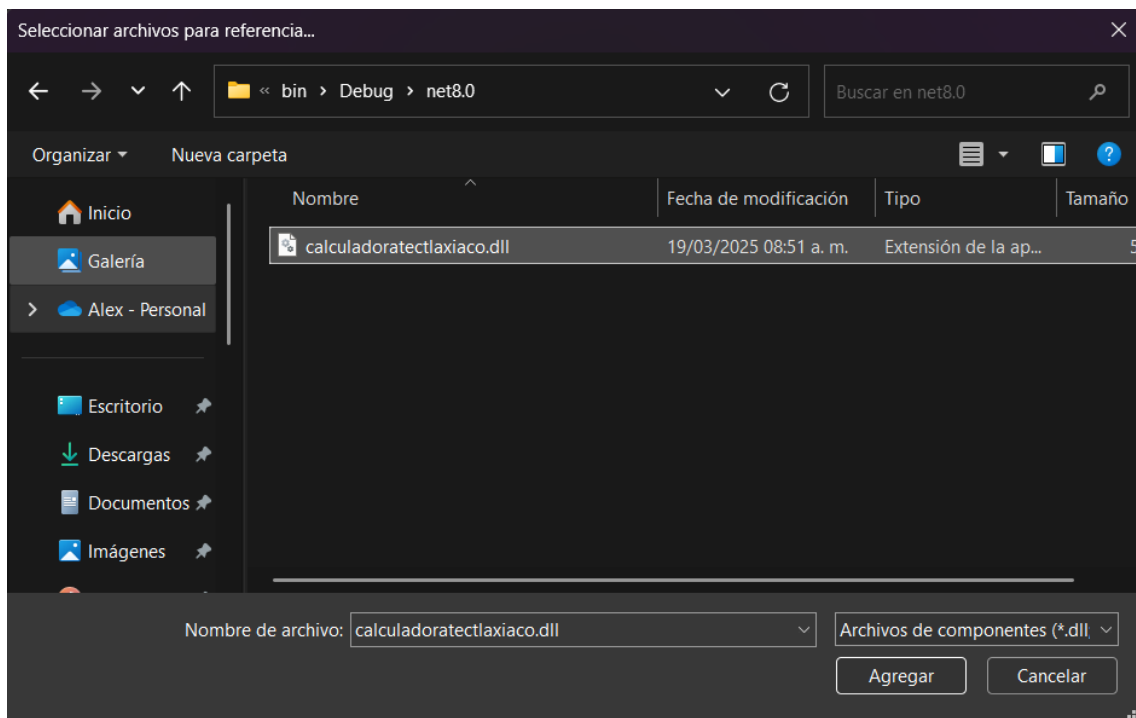


Ilustración 19: Selección del archivo DLL

En la parte superior del código agregamos una línea que diga **using calculadortectlaxiaco**; (el nombre de nuestra librería)

```
Program.cs*  + X
C# Consola  Consola.Program
1  using System;
2  using calculadortectlaxiaco;
3  using System.Collections.Generic;
4  using System.Linq;
5  using System.Text;
6  using System.Threading.Tasks;
7
```

Ilustración 20: Agregar "using" para librería

Creamos las variables para ocupar en nuestra consola

```
0 referencias
class Program
{
    private static double num1 = 0, num2 = 0, num0 = 0;
0 referencias
```

Ilustración 21: Creación de variables

Creamos un menú con **do while** con una condición boolean

```
bool salir = false;
while (!salir)
{
    try
    {
        Console.WriteLine();
        Console.WriteLine(" Menu Principal de Calculadora del Tecnológico de Tlaxiaco");
        Console.WriteLine("1. Sumar dos numeros");
        Console.WriteLine("2. Restar dos numeros");
        Console.WriteLine("3. Multiplicar");
        Console.WriteLine("4. Dividir");
        Console.WriteLine("5. Raiz Cuadrada");
        Console.WriteLine("6. Potencia");
        Console.WriteLine("7. Porcentaje");
        Console.WriteLine("8. Fraccion");
        Console.WriteLine("9. Calcular Seno");
        Console.WriteLine("10. Calcular Coseno");
        Console.WriteLine("11. Calcular Tangente");
        Console.WriteLine("12. Salir");
        Console.WriteLine("-----");
        Console.WriteLine("Elige una de las opciones");
        int opcion = Convert.ToInt32(Console.ReadLine());
```

Ilustración 22: Textos del menú

De ahí creamos un **switch** para poner un **case** para las opciones de nuestra calculadora, ocupamos el nombre de nuestra clase para referenciar la biblioteca y con un punto seleccionamos el método que deseamos usar

```
switch (opcion)
{
    case 1:
        teclado1();
        Console.WriteLine("El resultado de la suma es: " + Operaciones.sumar(num1, num2));
        break;
    case 2:
```

Ilustración 23: Case 1

Repitiendo esos mismos pasos creamos todos los **case** restantes que se ocuparán para nuestra calculadora

```
Program.cs  + x
[+] Consola  Console.Program  Main(string[] args)
38  switch (opcion)
39  {
40  case 1:
41      teclado1();
42      Console.WriteLine("El resultado de la suma es: " + Operaciones.sumar(num1, num2));
43      break;
44  case 2:
45      teclado1();
46      Console.WriteLine("El resultado de la Resta es: " + Operaciones.restar(num1, num2));
47      break;
48  case 3:
49      teclado1();
50      Console.WriteLine("El resultado de la Multiplicacion es: " + Operaciones.multiplicar(num1, num2));
51      break;
52  case 4:
53      teclado1();
54      Console.WriteLine("El resultado de la Division es: " + Operaciones.dividir(num1, num2));
55      break;
56  case 5:
57      teclado0();
58      Console.WriteLine("El resultado de la Raiz cuadrada es: " + Operaciones.raizCuadrada(num0));
59      break;
60  case 6:
61      teclado1();
62      Console.WriteLine("El resultado de la Potencia es: " + Operaciones.potencia(num1, num2));
63      break;
64  case 7:
65      teclado1();
66      Console.WriteLine("El resultado del porcentaje es: " + Operaciones.porcentaje(num1, num2));
67      break;
68  case 8:
69      teclado0();
70      Console.WriteLine("El resultado de la fraccion es: " + Operaciones.fraccion(num0));
71      break;
72  case 9:
73      Console.WriteLine("Has elegido salir de la aplicación");
74      Environment.Exit(1);
75      salir = true;
76      break;
77  default:
78      Console.WriteLine("Elige una opcion entre 1 y 9");
```

Ilustración 24: swich completo

Creamos un bloque **catch** para captura de excepciones del tipo `FormatException`, que ocurren cuando el usuario ingresa un valor no válido (por ejemplo, letras en lugar de números).

```
    }  
    catch (FormatException e)  
    {  
        Console.WriteLine("Error al ingresar!!");  
    }  
}
```

*Ilustración 25: Catch*

Creamos un método para cuando el usuario solo deba escribir un número

```
2 referencias  
private static void teclado0()  
{  
    Console.WriteLine("Introduzca un numero");  
    num0 = double.Parse(Console.ReadLine());  
}
```

*Ilustración 26: Método para solo 1 número*

Creamos un método para cuando el usuario solo deba escribir dos números

```
6 referencias  
private static void teclado1()  
{  
    Console.WriteLine("Introduzca el primer numero");  
    num1 = double.Parse(Console.ReadLine());  
    Console.WriteLine("Introduzca el segundo numero");  
    num2 = double.Parse(Console.ReadLine());  
}
```

*Ilustración 27: Método para 2 números*

Le damos a **iniciar** para confirmar que nuestro programa ejecute correctamente

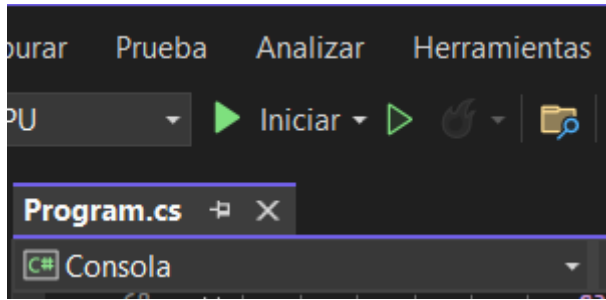


Ilustración 28: Iniciar consola

Con eso nuestra consola ejecutaría correctamente

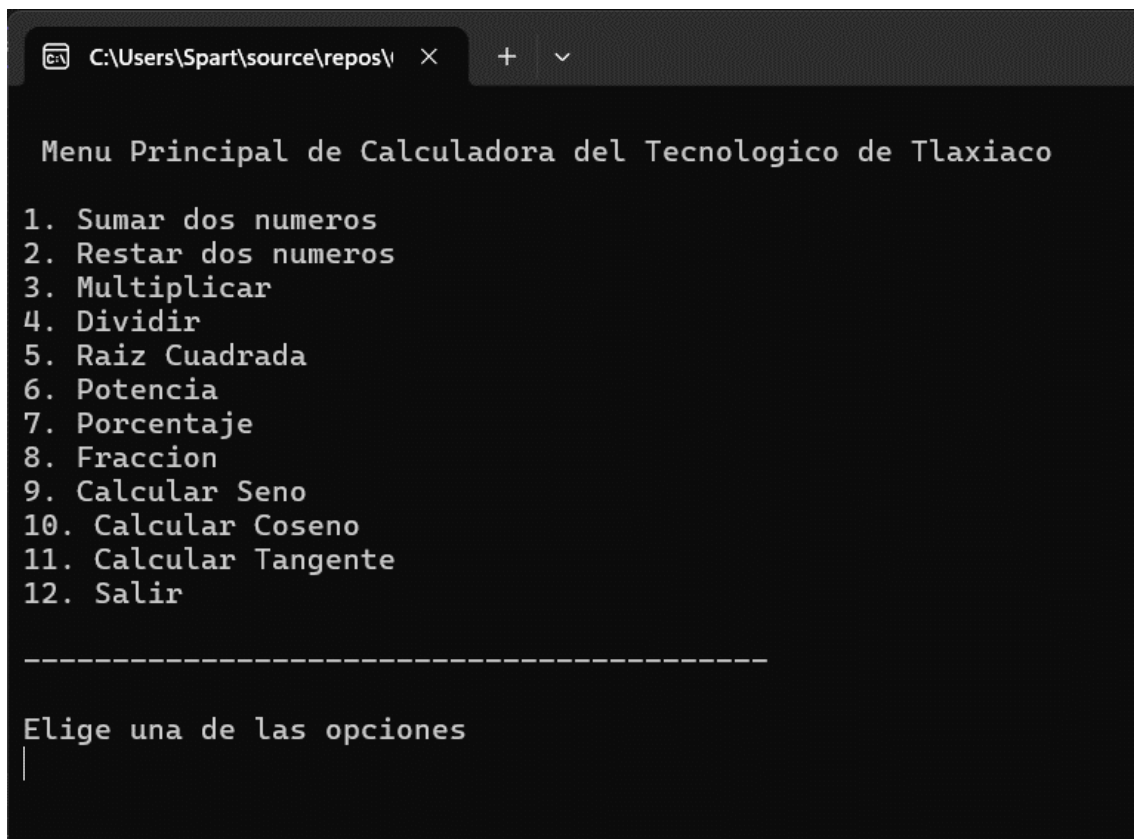


Ilustración 29: Consola ejecutada

Con eso nuestra app de consola quedaría correcta y totalmente funcional ocupando una librería de clases en la cual vienen almacenados los métodos



## Conclusión

En esta práctica, se logró cumplir con el objetivo de crear y utilizar una librería personalizada en C# para desarrollar una calculadora funcional. La implementación de las operaciones aritméticas básicas y funciones adicionales como potencia, raíz cuadrada, porcentaje, fracciones y funciones trigonométricas permitió reforzar los conceptos de modularización y reutilización de código.

Además, el uso de una biblioteca externa facilitó la separación de responsabilidades, haciendo el código más limpio, mantenible y eficiente. Esta experiencia también brindó una comprensión más profunda de la programación orientada a objetos, así como de la interacción entre proyectos en Visual Studio.

En conclusión, la práctica demostró la importancia de las librerías en el desarrollo de aplicaciones escalables y organizadas. Esta habilidad es fundamental para abordar proyectos más complejos en el futuro, promoviendo buenas prácticas de desarrollo de software.