

**Instituto Tecnológico de Mexico Campus
Queretaro**

Alumno: Erik Ruben Rodriguez Rodriguez

Profesor: Guillermo Fernandez Romero

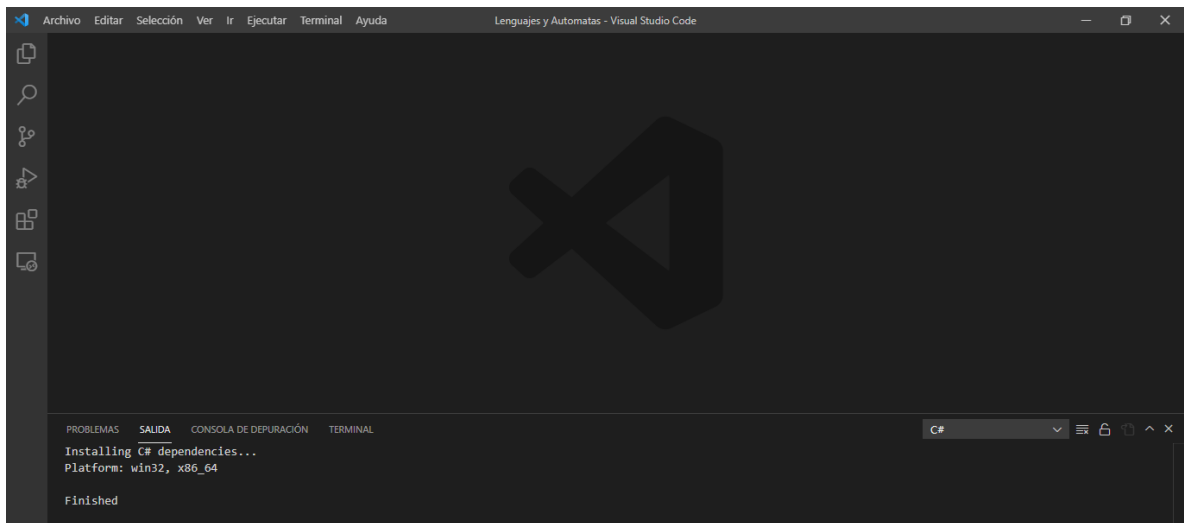
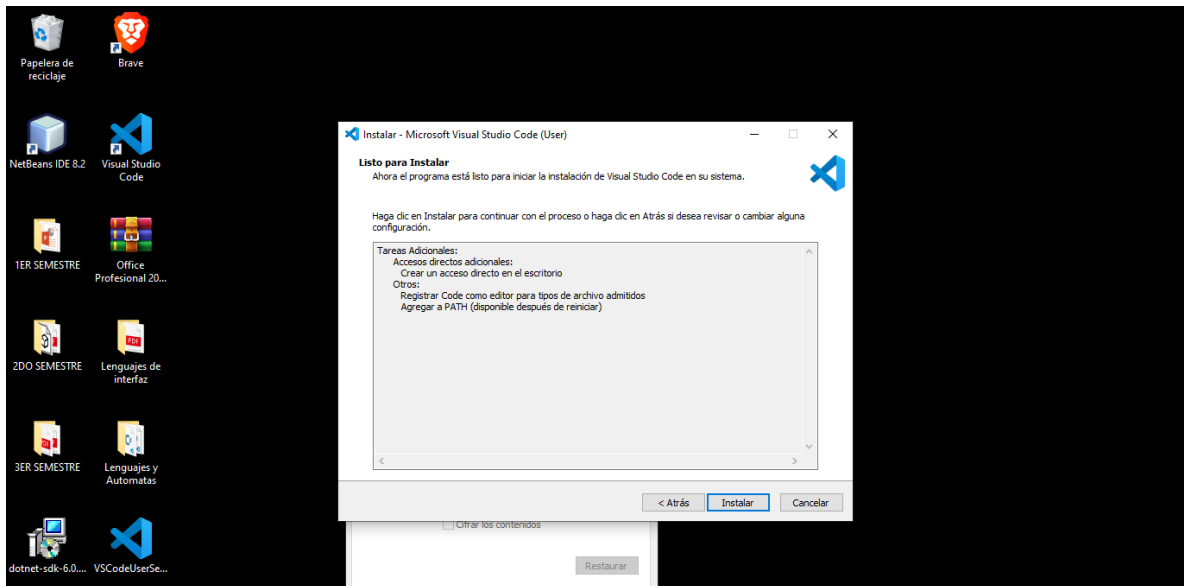
Materia: Automatas 1

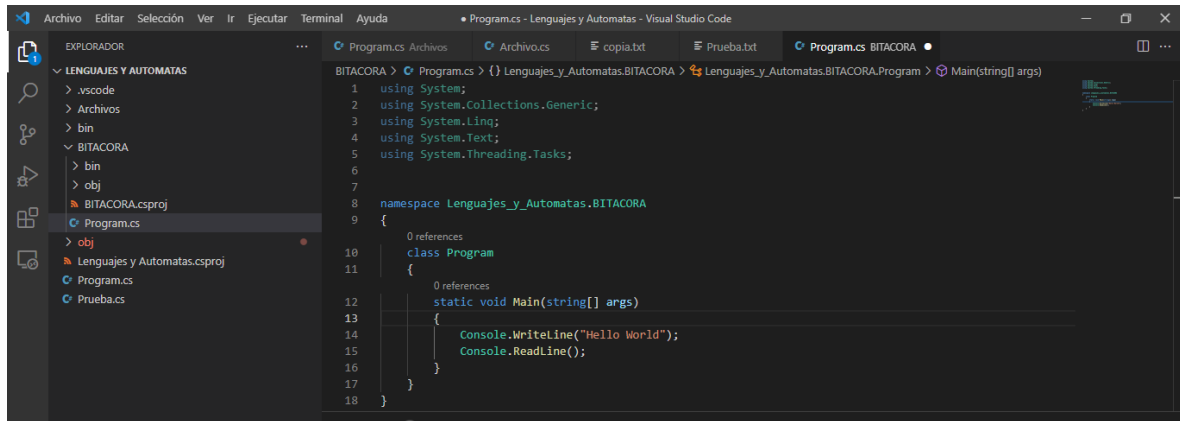
Carrera: Ingeniería en sistemas computacionales

Fecha: 4/09/2022



Instalación de Visual Studio



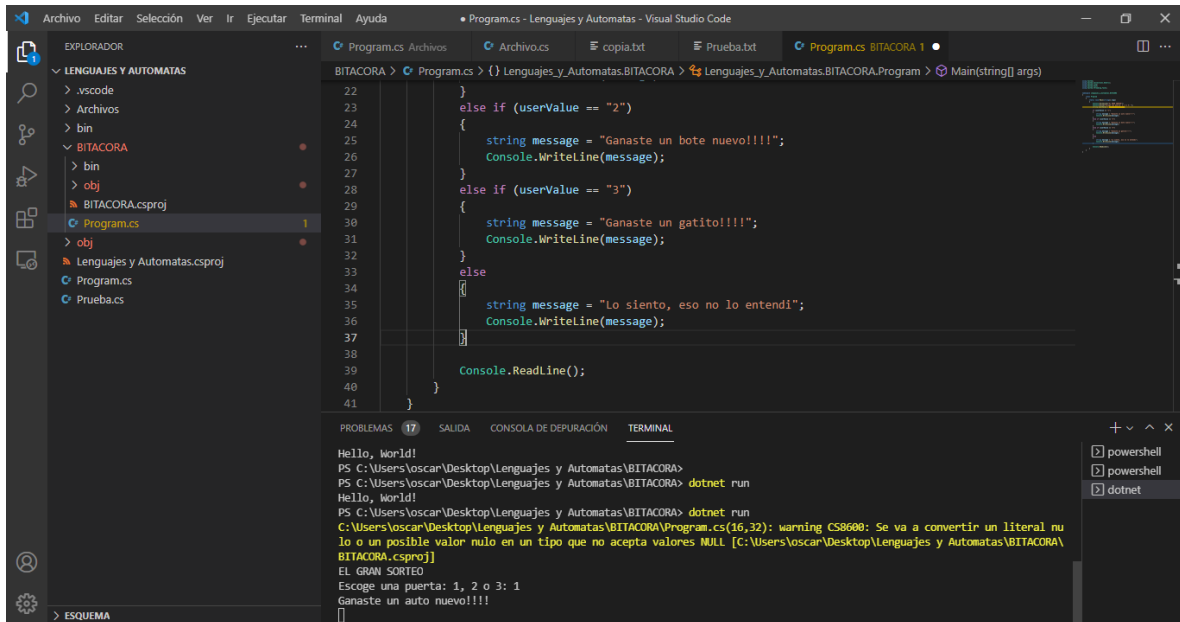


The image shows the Visual Studio Code interface with a C# project named 'Lenguajes y Automatas'. The Explorer sidebar on the left shows the project structure, including folders for '.vscode', 'Archivos', 'bin', 'obj', and files 'BITACORA.csproj', 'Program.cs', and 'Prueba.cs'. The main editor window displays the code for 'Program.cs' within the 'Lenguajes_y_Automatas.BITACORA' namespace. The code defines a 'Program' class with a static 'Main' method that writes 'Hello World' to the console and reads a line of input.

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7
8  namespace Lenguajes_y_Automatas.BITACORA
9  {
10     0 references
11     class Program
12     {
13         0 references
14         static void Main(string[] args)
15         {
16             Console.WriteLine("Hello World");
17             Console.ReadLine();
18         }
19     }
```

Primer programa en Visual “Hello World”

El uso de “IF”:



The screenshot displays the Visual Studio Code interface with a C# program named `Program.cs` open. The program uses `if-else` statements to handle user input. The terminal shows the output of running the program, including a warning about a nullable literal.

```
22     }
23
24     {
25         string message = "Ganaste un bote nuevo!!!!";
26         Console.WriteLine(message);
27     }
28     else if (userValue == "3")
29     {
30         string message = "Ganaste un gatito!!!!";
31         Console.WriteLine(message);
32     }
33     else
34     {
35         string message = "Lo siento, eso no lo entendi";
36         Console.WriteLine(message);
37     }
38
39     Console.ReadLine();
40 }
41 }
```

Terminal Output:

```
Hello, World!
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run
Hello, World!
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run
C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA\Program.cs(16,32): warning CS8600: Se va a convertir un literal nu
lo o un posible valor nulo en un tipo que no acepta valores NULL [C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA\
BITACORA.csproj]
EL GRAN SORTEO
Escoge una puerta: 1, 2 o 3: 1
Ganaste un auto nuevo!!!!
```

Entendiendo las variables

```
7
8 namespace Lenguajes_y_Automatas.BITACORA
9 {
10     class Program
11     {
12         static void Main(string[] args)
13         {
14             int x;
15             int y;
16             x=7;
17             y=x+3;
18             Console.WriteLine(y);
19             Console.ReadLine();
20         }
21     }
22 }
```

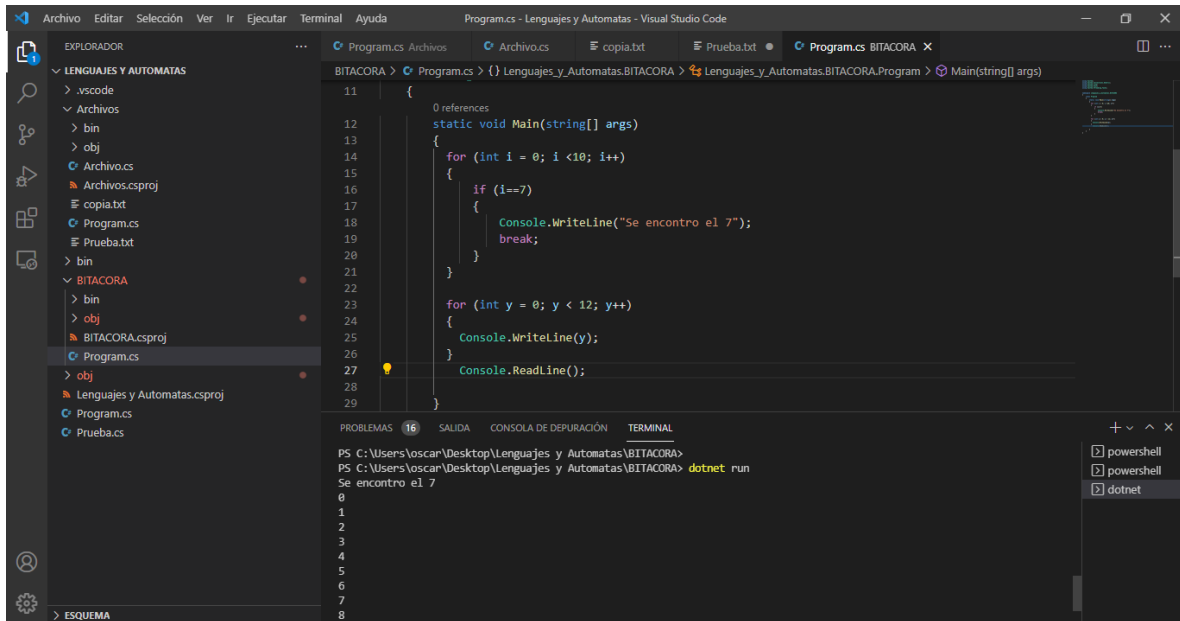
PROBLEMAS (16) SALIDA CONSOLA DE DEPURACIÓN TERMINAL

PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run
C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA\Program.cs(16,32): warning CS8600: Se va a convertir un literal nulo o un posible valor nulo en un tipo que no acepta valores NULL [C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA\BITACORA.csproj]
EL GRAN SORTEO
Escoge una puerta: 1, 2 o 3: 1
Ganaste un auto nuevo!!!!

PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA>
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run
10

```
14 /*int x;
15 int y;
16 x=7;
17 y=x+3;
18 Console.WriteLine(y);
19 Console.ReadLine();*/
20
21 Console.WriteLine("(¿Cual es tu nombre?)");
22 Console.Write("Escribe tu nombre: ");
23 string nombre;
24 nombre = Console.ReadLine();
25
26 string apellido;
27 Console.Write("Escribe tu apellido: ");
28 apellido = Console.ReadLine();
29
30 Console.WriteLine("Hola, "+nombre+" "+apellido);
31 Console.ReadLine();
32
33 }
```

Declaración de Iteraciones.



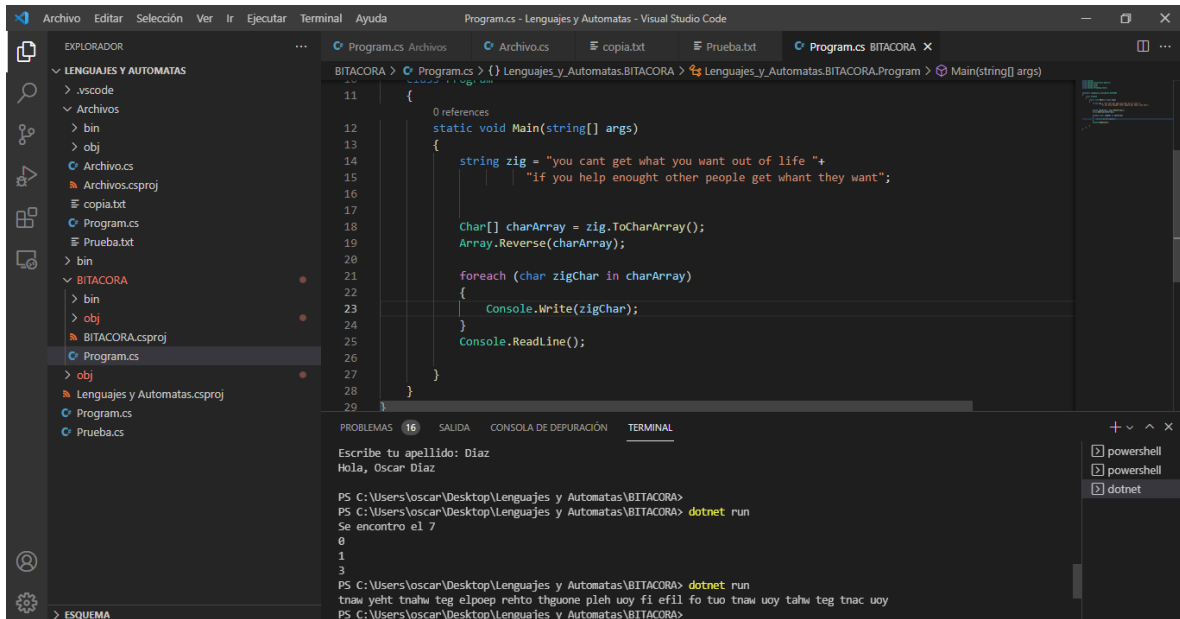
The screenshot shows the Visual Studio Code interface with a C# program named `Program.cs` open. The program contains two nested loops. The first loop iterates over the range `0` to `10` and prints the value of `i` when it reaches `7`. The second loop iterates over the range `0` to `12` and prints the value of `y` for each iteration. The terminal output shows the execution of the program, displaying the message `Se encontro el 7` followed by the numbers `0` through `8`.

```
11 {
12     0 references
13     static void Main(string[] args)
14     {
15         for (int i = 0; i < 10; i++)
16         {
17             if (i==7)
18             {
19                 Console.WriteLine("Se encontro el 7");
20                 break;
21             }
22         }
23         for (int y = 0; y < 12; y++)
24         {
25             Console.WriteLine(y);
26         }
27         Console.ReadLine();
28     }
29 }
```

PROBLEMAS 16 SALIDA CONSOLA DE DEPURACIÓN TERMINAL

```
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA>
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run
Se encontro el 7
0
1
2
3
4
5
6
7
8
```

Entendiendo los arreglos



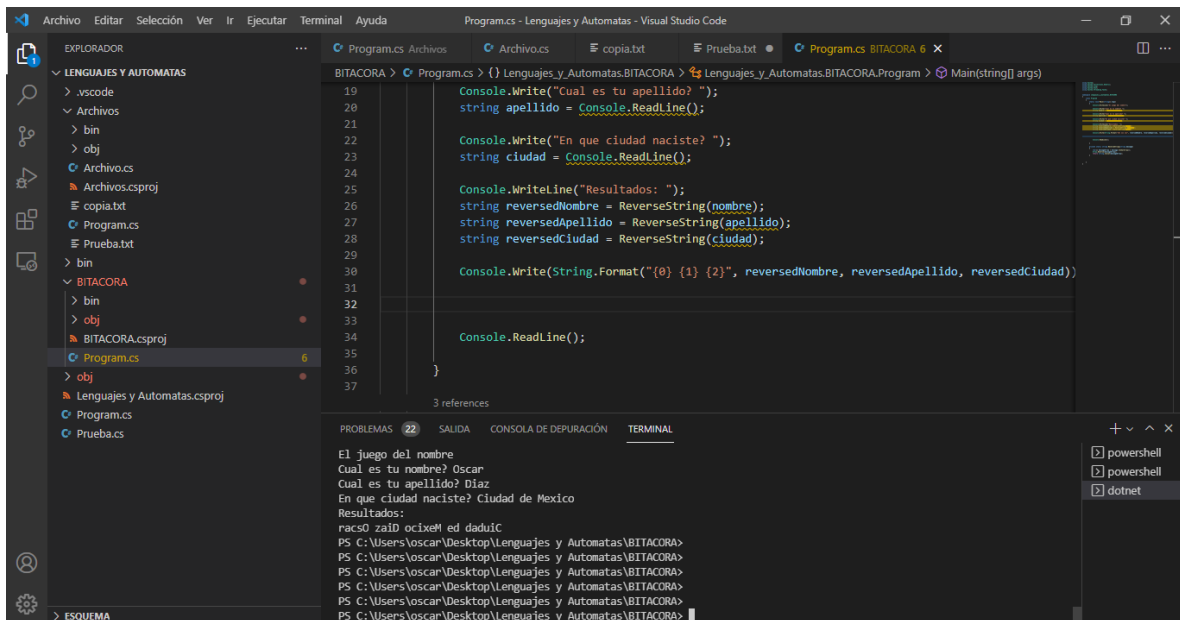
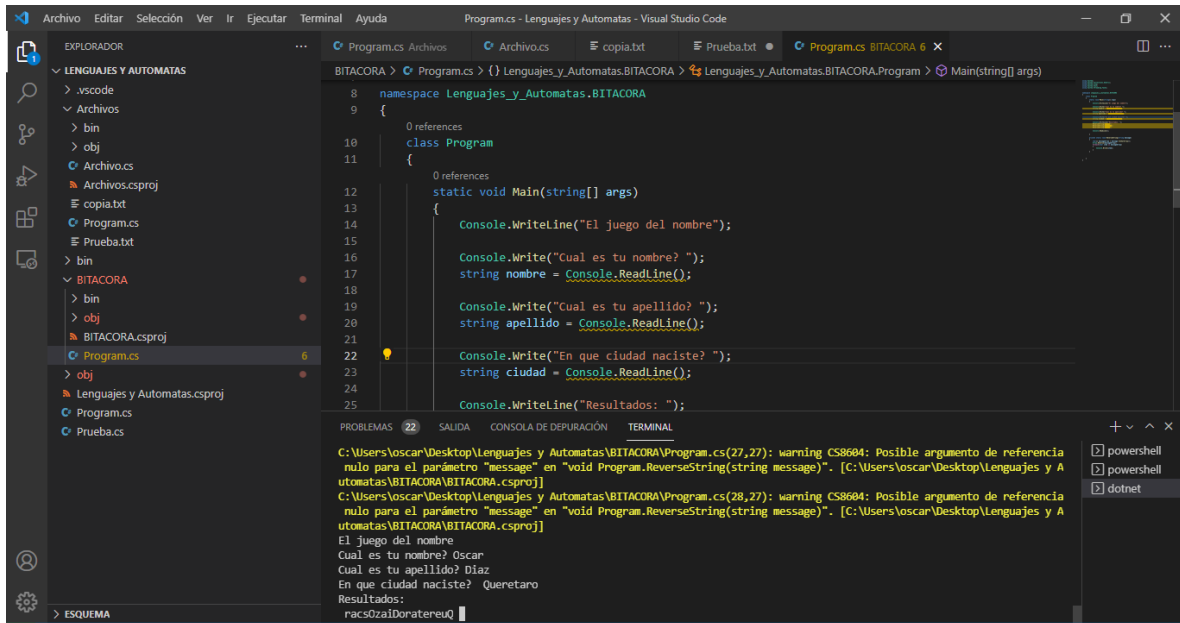
The screenshot shows the Visual Studio Code interface with a C# project named "Lenguajes y Automatas". The file explorer on the left shows the project structure, including a "BITACORA" folder. The main editor displays the code for "Program.cs", which is a C# program that demonstrates array reversal. The code is as follows:

```
11 {  
12     0 references  
13     static void Main(string[] args)  
14     {  
15         string zig = "you cant get what you want out of life "+  
16             "if you help enough other people get whant they want";  
17  
18         Char[] charArray = zig.ToCharArray();  
19         Array.Reverse(charArray);  
20  
21         foreach (char zigChar in charArray)  
22         {  
23             Console.Write(zigChar);  
24         }  
25         Console.ReadLine();  
26     }  
27 }  
28  
29
```

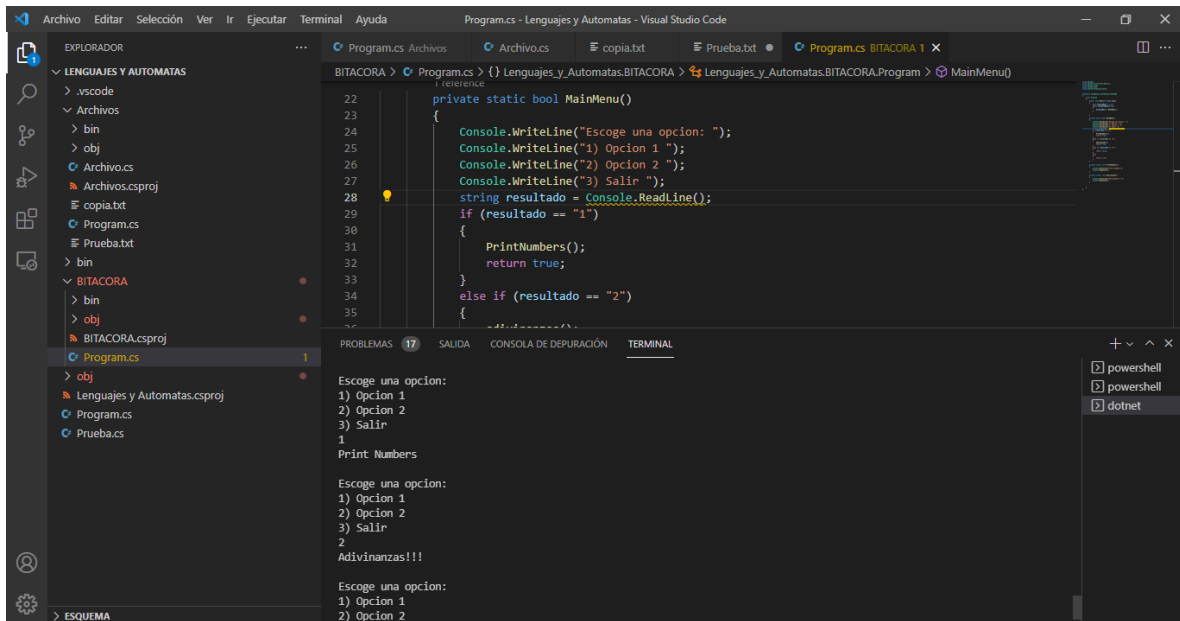
The terminal at the bottom shows the execution of the program using the command `dotnet run`. The output is:

```
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA>  
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run  
Se encontro el 7  
0  
1  
3  
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA> dotnet run  
tnaw yeht tnahw teg elpoeep rehto thguone pieh uoy fi efil fo tuo tnaw uoy tahw teg tnac uoy  
PS C:\Users\oscar\Desktop\Lenguajes y Automatas\BITACORA>
```

Definiendo y llamando métodos



While Iteration Statement



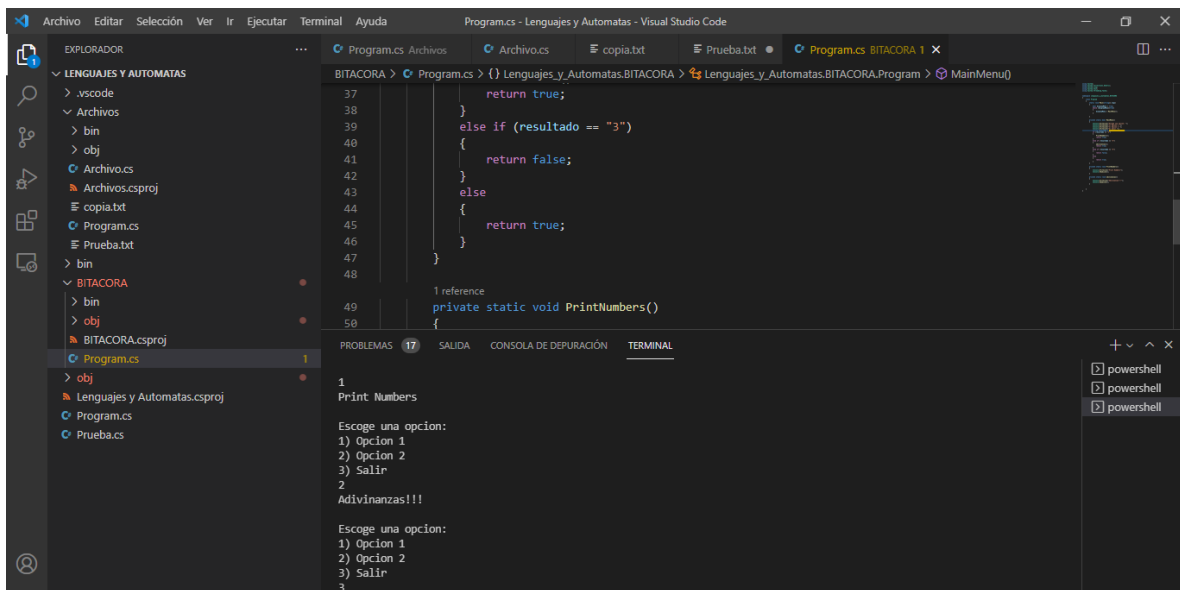
```
22 private static bool MainMenu()  
23 {  
24     Console.WriteLine("Escribe una opcion: ");  
25     Console.WriteLine("1) Opcion 1 ");  
26     Console.WriteLine("2) Opcion 2 ");  
27     Console.WriteLine("3) Salir ");  
28     string resultado = Console.ReadLine();  
29     if (resultado == "1")  
30     {  
31         PrintNumbers();  
32         return true;  
33     }  
34     else if (resultado == "2")  
35     {  
36         PrintNumbers();  
37         return true;  
38     }  
39     else if (resultado == "3")  
40     {  
41         return false;  
42     }  
43     else  
44     {  
45         return true;  
46     }  
47 }  
48  
49 1 reference  
50 private static void PrintNumbers()  
51 {  
52     Console.WriteLine("Print Numbers");  
53 }
```

PROBLEMAS (17) SALIDA CONSOLA DE DEPURACIÓN TERMINAL

Escoge una opcion:
1) Opcion 1
2) Opcion 2
3) Salir
1
Print Numbers

Escoge una opcion:
1) Opcion 1
2) Opcion 2
3) Salir
2
Adivinanzas!!!

Escoge una opcion:
1) Opcion 1
2) Opcion 2



```
37     return true;  
38 }  
39 else if (resultado == "3")  
40 {  
41     return false;  
42 }  
43 else  
44 {  
45     return true;  
46 }  
47 }  
48  
49 1 reference  
50 private static void PrintNumbers()  
51 {  
52     Console.WriteLine("Print Numbers");  
53 }
```

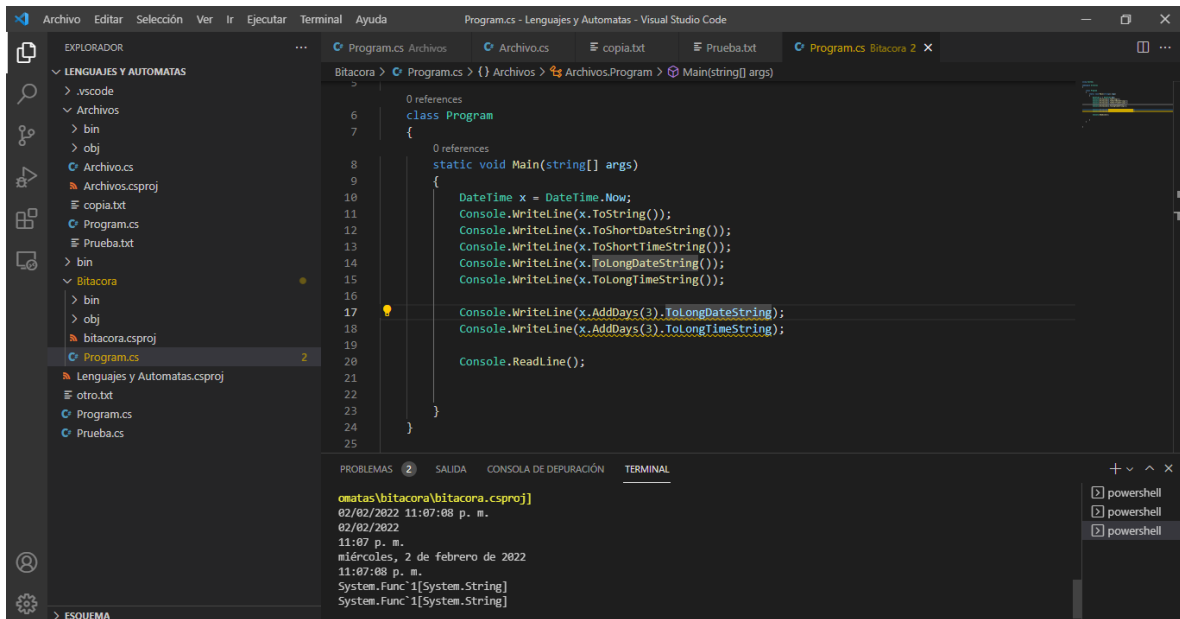
PROBLEMAS (17) SALIDA CONSOLA DE DEPURACIÓN TERMINAL

1
Print Numbers

Escoge una opcion:
1) Opcion 1
2) Opcion 2
3) Salir
2
Adivinanzas!!!

Escoge una opcion:
1) Opcion 1
2) Opcion 2
3) Salir
3

Uso de horas y fechas.



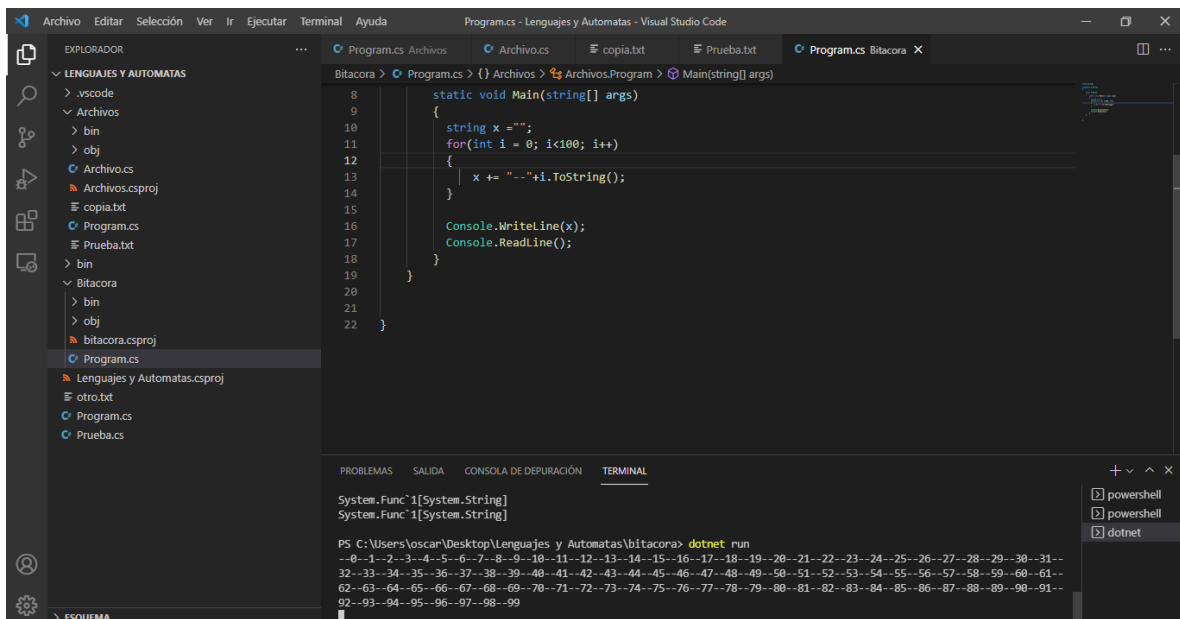
The screenshot shows the Visual Studio Code interface with a C# file named `Program.cs` open. The file contains a `Program` class with a `Main` method that demonstrates various `DateTime` methods for working with dates and times.

```
5 0 references
6 class Program
7 {
8     0 references
9     static void Main(string[] args)
10    {
11        DateTime x = DateTime.Now;
12        Console.WriteLine(x.ToString());
13        Console.WriteLine(x.ToShortDateString());
14        Console.WriteLine(x.ToShortTimeString());
15        Console.WriteLine(x.ToLongDateString());
16        Console.WriteLine(x.ToLongTimeString());
17        Console.WriteLine(x.AddDays(3).ToLongDateString());
18        Console.WriteLine(x.AddDays(3).ToLongTimeString());
19
20        Console.ReadLine();
21    }
22 }
23
24
25
```

The bottom panel shows the `TERMINAL` output, which displays the execution results of the program:

```
omatas\bitacora\bitacora.csproj]
02/02/2022 11:07:08 p. m.
02/02/2022
11:07 p. m.
miércoles, 2 de febrero de 2022
11:07:08 p. m.
System.Func`1[System.String]
System.Func`1[System.String]
```

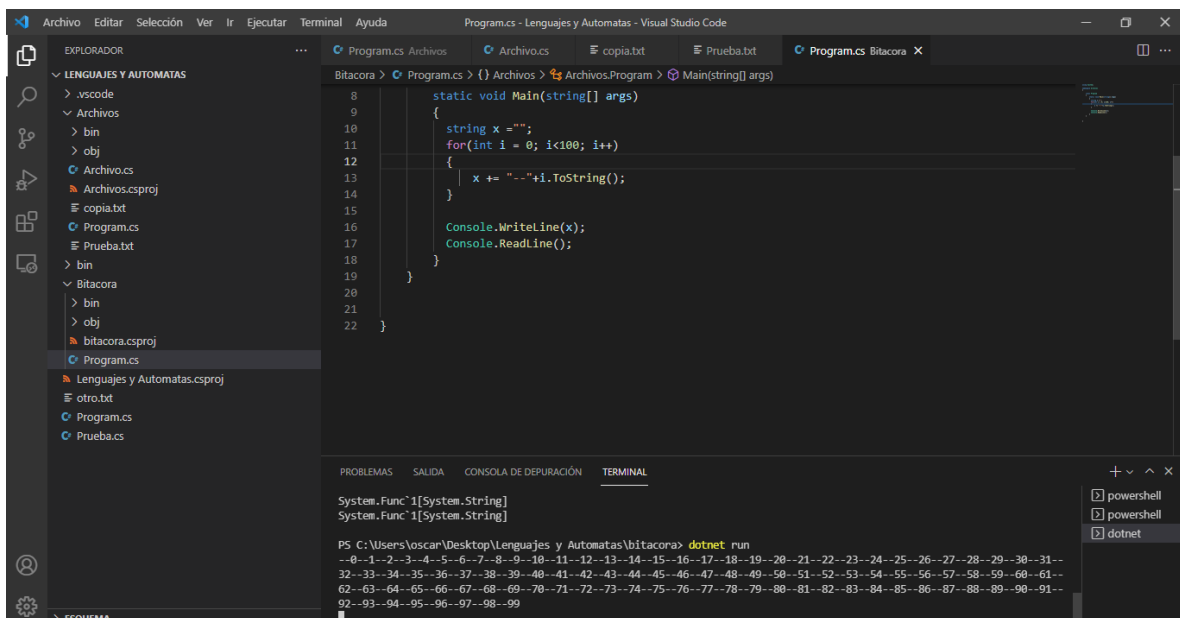
Trabajando con Strings



The screenshot shows the Visual Studio Code interface with a C# file named `Program.cs` open. The code defines a `Main` method that builds a string `x` containing numbers from 0 to 99, separated by spaces. The string is then printed to the console and the user is prompted to press Enter.

```
8 static void Main(string[] args)
9 {
10     string x = "";
11     for(int i = 0; i<100; i++)
12     {
13         x += " "+i.ToString();
14     }
15
16     Console.WriteLine(x);
17     Console.ReadLine();
18 }
19
20
21
22 }
```

The terminal output shows the execution of the program, displaying the string `System.Func`1[System.String]` and the prompt `PS C:\Users\oscar\Desktop\Lenguajes y Automatas\bitacora> dotnet run`.



This screenshot is identical to the one above, showing the same C# code and terminal output in Visual Studio Code.

Entendiendo las Clases.

The screenshot shows the Visual Studio Code editor with a C# file named 'Program.cs'. The code is as follows:

```

12
13     {
14         car carro = new car();
15         carro.Make = "Auto viejo";
16         carro.Model = "Cutlas Supreme";
17         carro.Year = 1986;
18         carro.Color = "Silver";
19
20         Console.WriteLine("{0} {1} {2} {3}", carro.Make, carro.Model, carro.Year, carro.Color);
21         Console.ReadLine();
22     }
23
24     2 references
25     class car
26     {
27         2 references
28         public string Make {get; set; }
29         2 references
30         public string Model {get; set; }
31         2 references
32         public int Year {get; set; }
33         2 references
34         public string Color {get; set; }
35     }
36
37     2 references
38     static void Main(string[] args)
39     {
40         // TODO: Complete this method
41     }
42 }

```

The output in the terminal is:

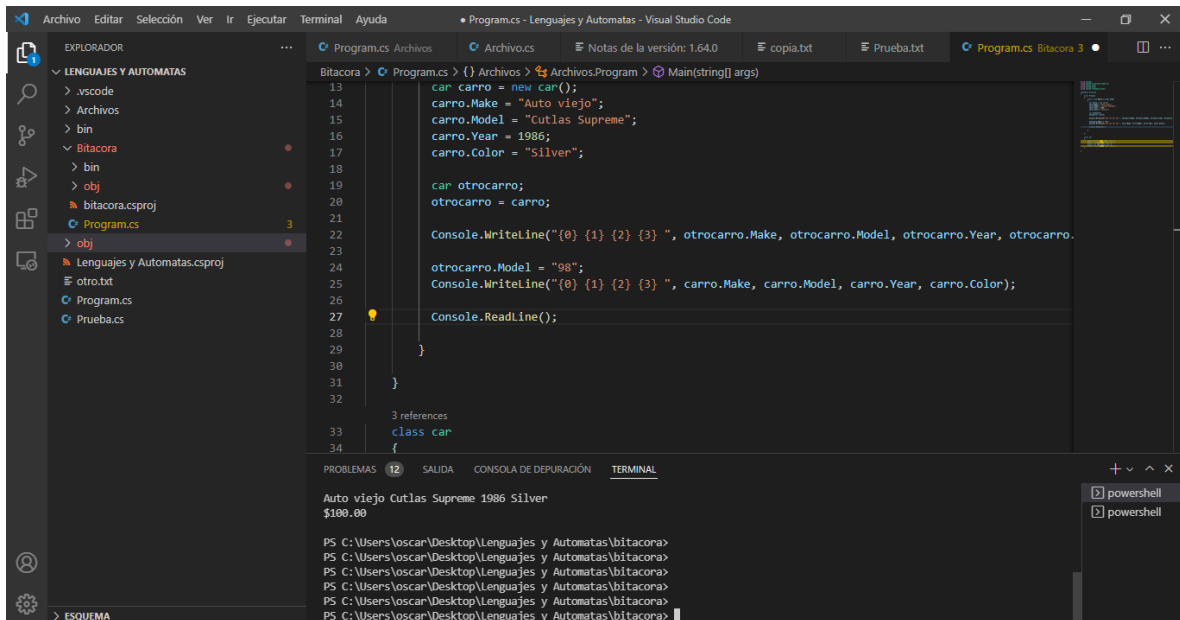
```

Auto viejo Cutlas Supreme 1986 Silver

```

[illegible]

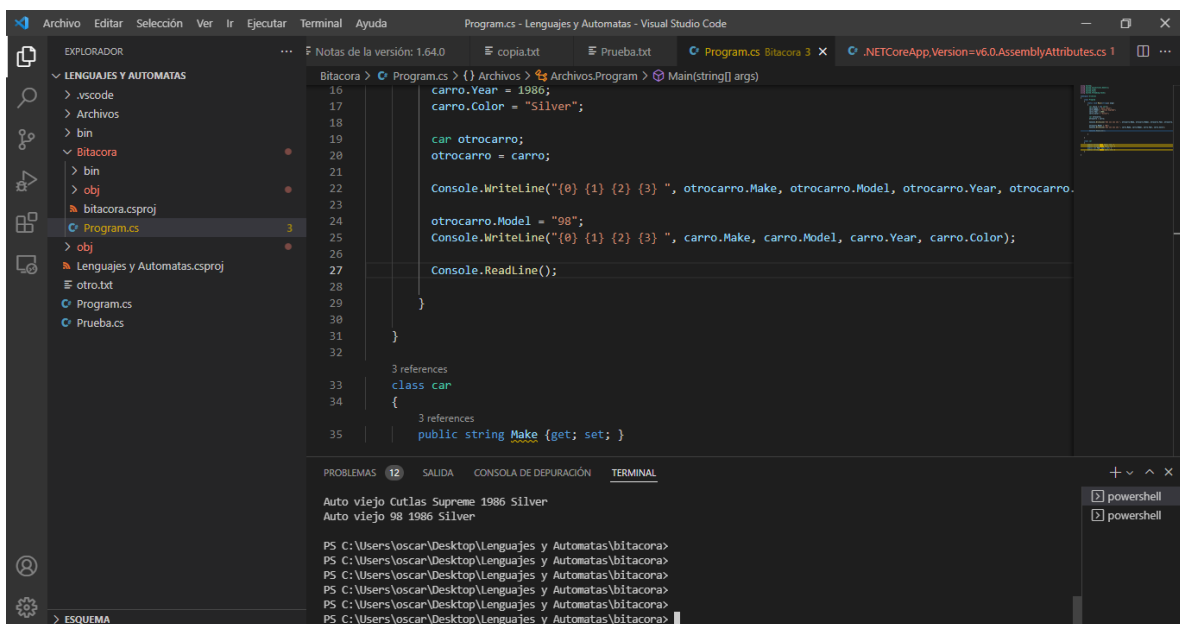
Más sobre clases



The screenshot shows the Visual Studio Code interface with a C# project named "Lenguajes y Automatas". The Explorer pane on the left shows the project structure, including a folder named "Bitacora" and a file named "Program.cs". The main editor displays the code for "Program.cs", which defines a "car" class and creates an instance of it. The code is as follows:

```
13 car carro = new car();
14 carro.Make = "Auto viejo";
15 carro.Model = "Cutlas Supreme";
16 carro.Year = 1986;
17 carro.Color = "Silver";
18
19 car otrocarro;
20 otrocarro = carro;
21
22 Console.WriteLine("{0} {1} {2} {3} ", otrocarro.Make, otrocarro.Model, otrocarro.Year, otrocarro.
23
24 otrocarro.Model = "98";
25 Console.WriteLine("{0} {1} {2} {3} ", carro.Make, carro.Model, carro.Year, carro.Color);
26
27 Console.ReadLine();
28
29 }
30
31 }
32
33 3 references
34 class car
35 {
```

The Output pane at the bottom shows the execution results, including the text "Auto viejo Cutlas Supreme 1986 Silver" and "\$100.00". The Terminal pane shows the command prompt output, including the command "PS C:\Users\oscar\Desktop\Lenguajes y Automatas\bitacora>" and the output "Auto viejo Cutlas Supreme 1986 Silver".

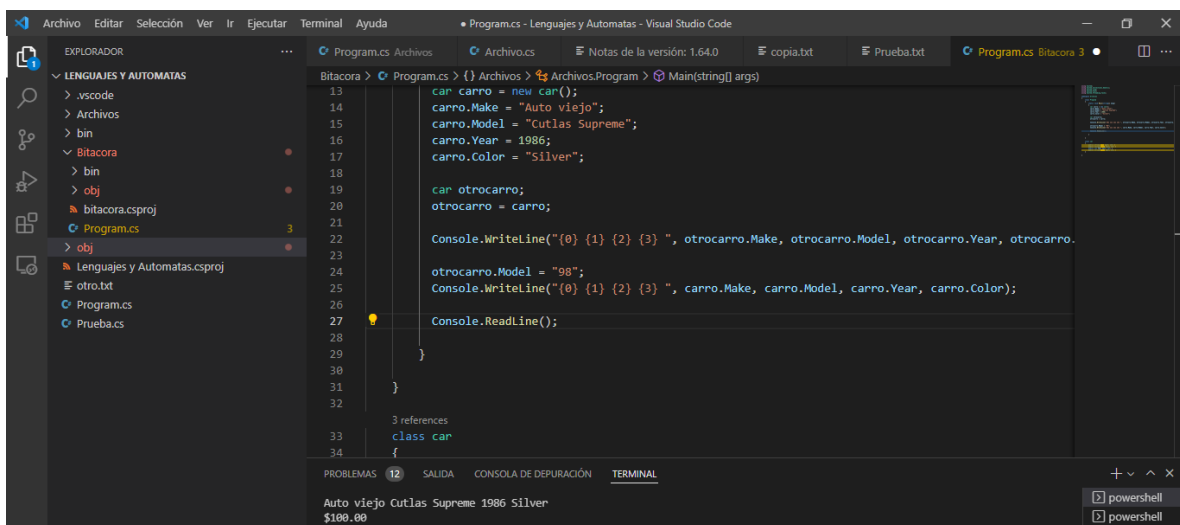
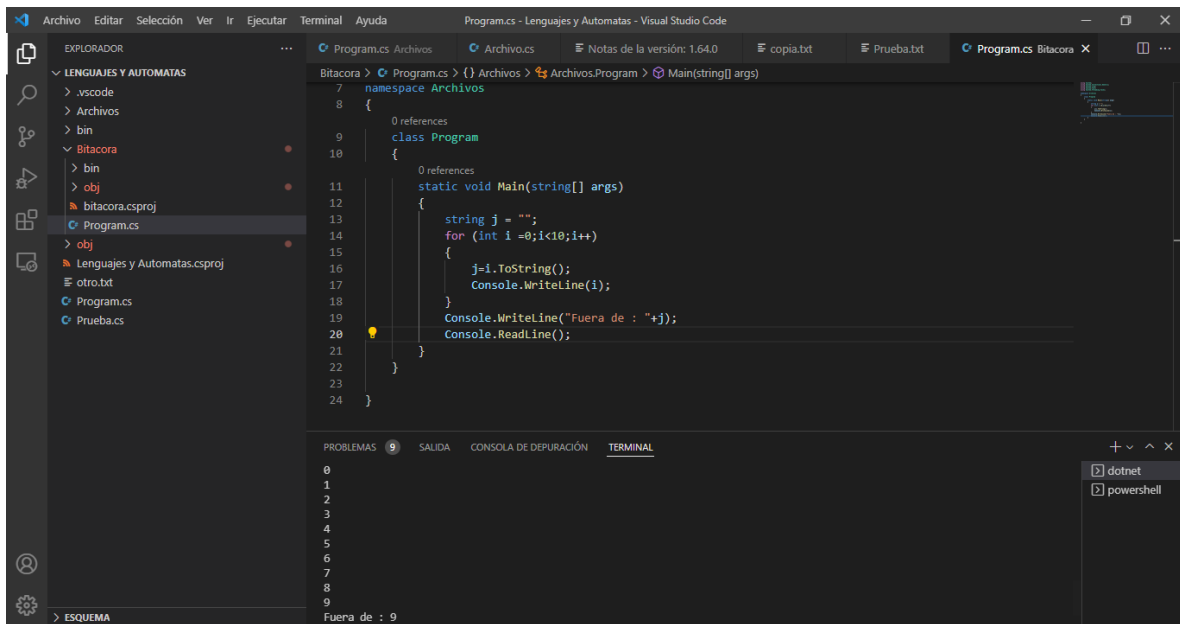


The screenshot shows the Visual Studio Code interface with a C# project named "Lenguajes y Automatas". The Explorer pane on the left shows the project structure, including a folder named "Bitacora" and a file named "Program.cs". The main editor displays the code for "Program.cs", which defines a "car" class and creates an instance of it. The code is as follows:

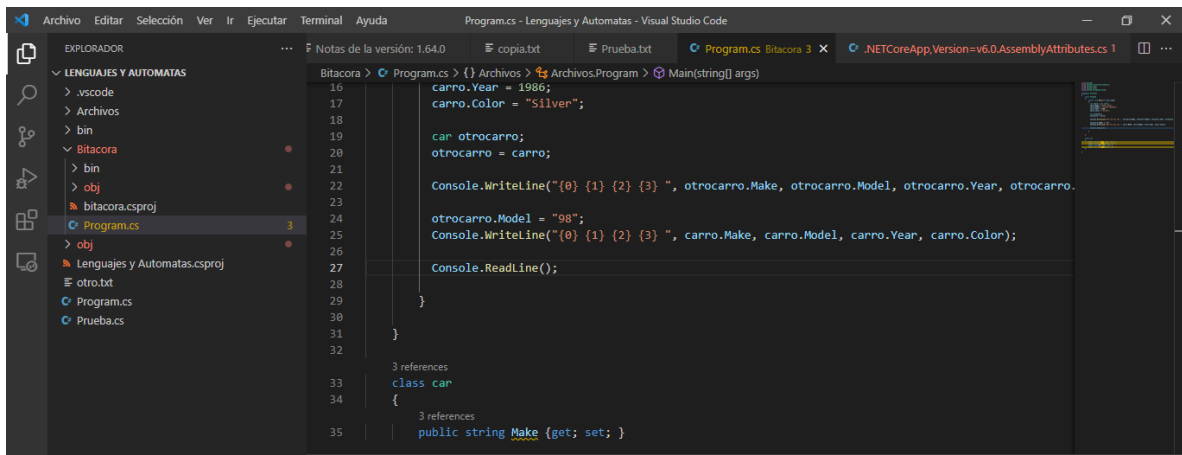
```
16 carro.Year = 1986;
17 carro.Color = "Silver";
18
19 car otrocarro;
20 otrocarro = carro;
21
22 Console.WriteLine("{0} {1} {2} {3} ", otrocarro.Make, otrocarro.Model, otrocarro.Year, otrocarro.
23
24 otrocarro.Model = "98";
25 Console.WriteLine("{0} {1} {2} {3} ", carro.Make, carro.Model, carro.Year, carro.Color);
26
27 Console.ReadLine();
28
29 }
30
31 }
32
33 3 references
34 class car
35 {
36     3 references
37     public string Make {get; set; }
```

The Output pane at the bottom shows the execution results, including the text "Auto viejo Cutlas Supreme 1986 Silver" and "Auto viejo 98 1986 Silver". The Terminal pane shows the command prompt output, including the command "PS C:\Users\oscar\Desktop\Lenguajes y Automatas\bitacora>" and the output "Auto viejo Cutlas Supreme 1986 Silver".

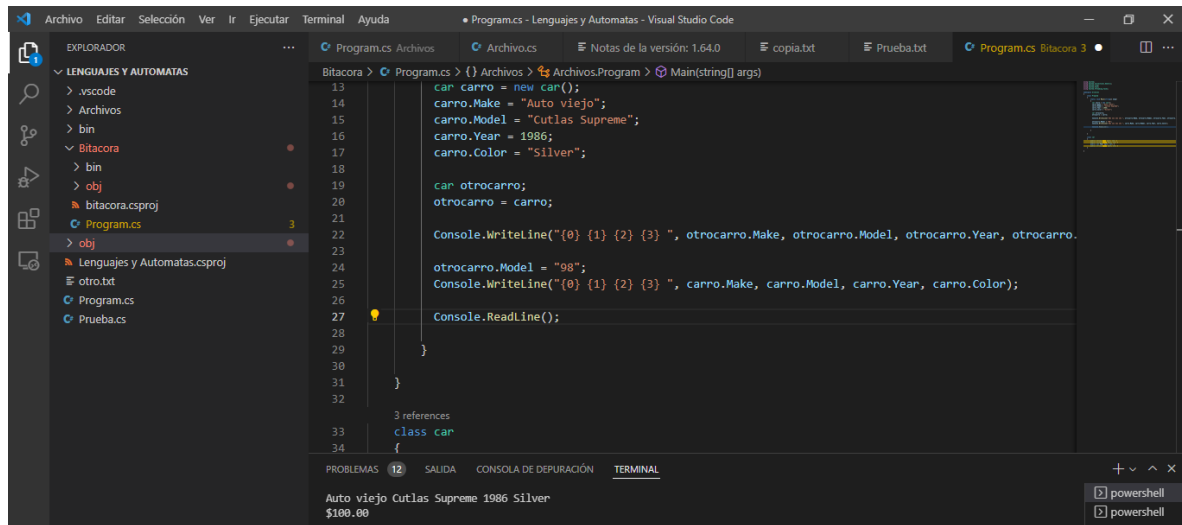
Entendiendo el scope



Creando y añadiendo referencias



Trabajando con Linq




Course introduction

Rob Trow

Developer University

https://dev.microsoft.com



00:08

EPISODE 0

Course Introduction

10 Jun 2019

Tools for every developer and every app

Visual Studio

Visual Studio Test Runner

Visual Studio Code

Now with Xamarin

00:54

EPISODE 1

Installing Visual Studio

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

10:40

EPISODE 2

Creating Your First C# Program

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

23:42

EPISODE 3

The if Decision Statement

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

26:13

EPISODE 4

Understanding Data Types and Variables

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

18:37

EPISODE 5

for Iteration Statement

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

17:48

EPISODE 6

Understanding Arrays

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

17:54

EPISODE 7

Defining and Calling Methods

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

10:08

EPISODE 8

Working with Dates and Times

10 Jun 2019

Visual Studio

Microsoft Visual Studio Community 2019

Visual Studio

Visual Studio

23:09

EPISODE 9

Working with Strings

10 Jun 2019

1 2 3 4

Equalizer (igualad)

© Tunes

Documentación de versiones anteriores

Blog

Contribuir

Privacidad & cookies

Términos de uso

Marcas comerciales

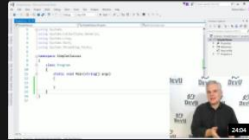
© Microsoft 2022

C# Fundamentals for Absolute Beginners

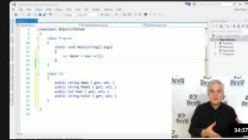
English

Learn C# programming from an expert in the industry. Get the tools, see how to write code, debug features, explore customizations, and more.
For newer videos head over to dot.net/videos

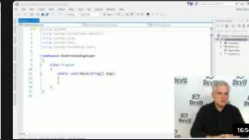
Episodes (25)



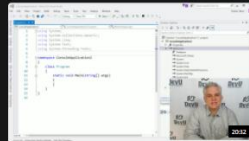
EP1150010
Understanding Classes
10 Jun 2019



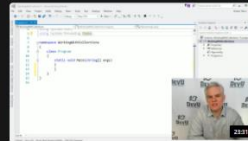
EP1150010
More About Classes and Methods
10 Jun 2019



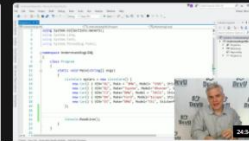
EP1150010
Understanding Scope and Accessibility Modifiers
10 Jun 2019



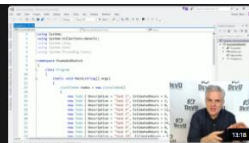
EP1150010
Creating and Adding References to Assemblies
10 Jun 2019



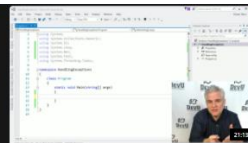
EP1150010
Working with Collections
10 Jun 2019



EP1150010
Working with LINQ
10 Jun 2019



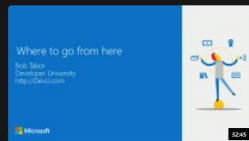
EP1150010
Enumerations and the Switch Decision Statement
10 Jun 2019



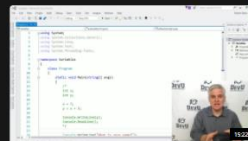
EP1150010
Gracefully Handling Exceptions
10 Jun 2019



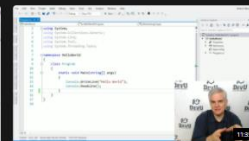
EP1150010
Understanding Events and Event-Driven Programming
10 Jun 2019



EP1150010
Where to Go From Here
10 Jun 2019



EP1150010
Operators, Expressions, and Statements
1 ago 2019



EP1150010
Working with Code Files, Projects, and Solutions
1 ago 2019

SII : Acceso Alumnos

C# Fundamentals for Absolute Beginners

Bitácora de curso de C#

Autómatas I 15:00 - 16:00

Nueva pestaña

docs.microsoft.com/es-es/shows/CSharp-Fundamentals-for-Absolute-Beginners/?page=3

Correo de Tecnológ...Metodos NumericosAutómatas I 15:00 - ...Cálculo Integral 20...Investigación de op...Tópicos lista de asi...SII : Acceso AlumnosDashboard

MVA

SERIES

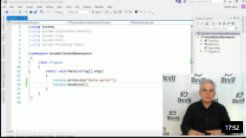
C# Fundamentals for Absolute Beginners

English

Learn C# programming from an expert in the industry. Get the tools, see how to write code, debug features, explore customizations, and more.

For newer videos head over to dot.net/videos

Episodes (25)



EPISODE 01

Understanding Namespaces and Working with the .NET Class Library

1 ago 2019

←123

Español (English)Tema

Documentación de versiones anterioresBlogContribuirPrivacidad & cookiesTérminos de usoMenús comerciales© Microsoft 2022

61%23°C Muy sole...ESP06:36 p. m.