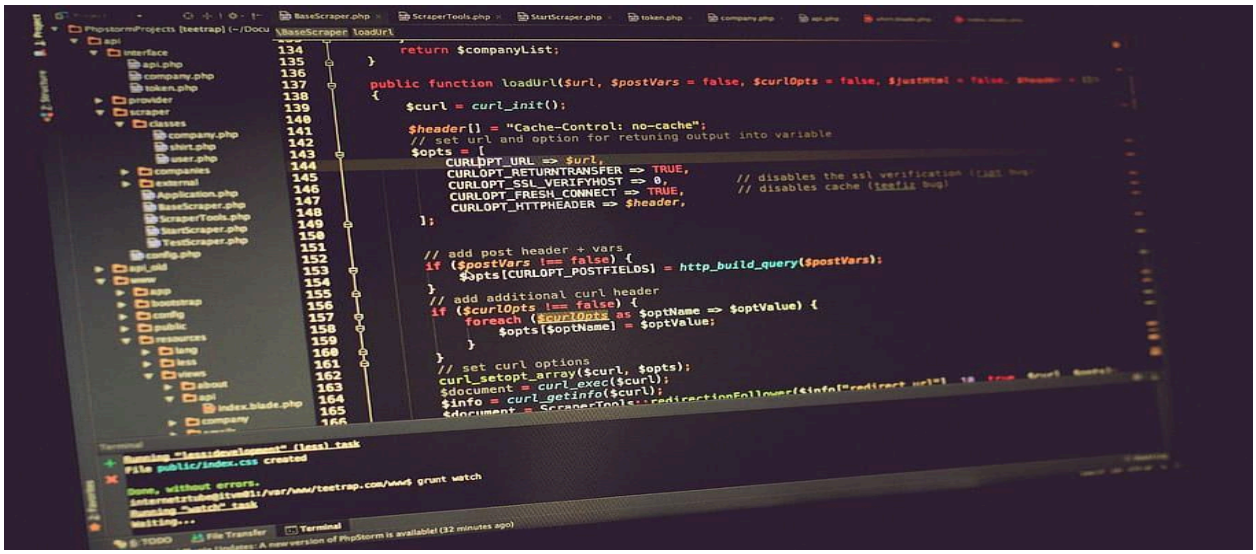
Tutorial: Introduction to the C# Programming Language



Introduction

C# (pronounced “C-Sharp”) is a modern object-oriented programming language developed by Microsoft. C# is widely used for creating desktop applications, web applications, video games, mobile apps, cloud services, and enterprise software. It works closely with the .NET platform and is commonly used in software development and cybersecurity environments.

This tutorial introduces the basics of C# programming, including syntax, variables, input/output, decision-making, loops, methods, and object-oriented programming concepts.

What You Need Before You Start

Required Software

To write and run C# programs, install:

- Visual Studio
- .NET SDK

Recommended System Requirements

- Windows, macOS, or Linux
- Internet connection
- Basic computer skills

Step 1: Understanding C# Basics

C# programs are built using:

- Classes
- Methods
- Variables
- Statements
- Objects

Structure of a Simple C# Program

```
using System;

class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello World!");
    }
```

```
}
```

Explanation

Code	Purpose
using System;	Imports system functions
class Program	Defines a class
Main()	Starting point of the program
Console.WriteLine()	Displays output

Output

Hello World!

Step 2: Variables and Data Types

Variables store information in a program.

Common Data Types

Data Type	Example
int	Whole numbers
double	Decimal numbers
char	Single character
string	Text
bool	True or false

Example

```
using System;
```

```
class Program
{
   static void Main()
   {
      string name = "Steven";
      int age = 30;
      double grade = 95.5;

      Console.WriteLine(name);
      Console.WriteLine(age);
      Console.WriteLine(grade);
   }
}
```

Step 3: User Input

Programs can accept input from users.

Example

```
using System;

class Program
{
   static void Main()
   {
      Console.Write("Enter your name: ");
      string name = Console.ReadLine();

      Console.WriteLine("Hello " + name);
   }
}
```

Output Example

```
Enter your name: Steven
Hello Steven
```

Step 4: Decision-Making Statements

C# uses conditional statements to make decisions.

If Statement

```
using System;

class Program
{
    static void Main()
    {
        int score = 85;

        if(score >= 70)
        {
            Console.WriteLine("You passed!");
        }
    }
}
```

Step 5: If-Else Statements

```
using System;

class Program
{
    static void Main()
    {
        int age = 16;

        if(age >= 18)
        {
            Console.WriteLine("Adult");
        }
        else
        {
            Console.WriteLine("Minor");
        }
    }
```

```
}
```

Step 6: Loops

Loops repeat instructions multiple times.

For Loop Example

```
using System;

class Program
{
   static void Main()
   {
      for(int i = 1; i <= 5; i++)
      {
         Console.WriteLine(i);
      }
   }
}
```

Output

```
1
2
3
4
5
```

Step 7: Methods

Methods organize code into reusable sections.

Example

```
using System;
```

```
class Program
{
   static void Greeting()
   {
      Console.WriteLine("Welcome to C#");
   }

   static void Main()
   {
      Greeting();
   }
}
```

Step 8: Arrays

Arrays store multiple values.

Example

```
using System;

class Program
{
   static void Main()
   {
      string[] colors = {"Red", "Blue", "Green"};

      foreach(string color in colors)
      {
         Console.WriteLine(color);
      }
   }
}
```

Step 9: Introduction to Object-Oriented Programming (OOP)

C# is an object-oriented programming language.

Main OOP Concepts

- Classes
- Objects
- Encapsulation
- Inheritance
- Polymorphism

Simple Class Example

```
using System;

class Car
{
   public string model = "Tesla";
}

class Program
{
   static void Main()
   {
      Car myCar = new Car();
      Console.WriteLine(myCar.model);
   }
}
```

Step 10: Debugging and Testing

Debugging helps programmers find and fix errors.

Common Errors

Error Type	Description
Syntax Error	Incorrect code format
Logic Error	Program gives wrong output

Runtime Error	Program crashes during execution

Debugging Tips

- Read error messages carefully
- Test one section at a time
- Use breakpoints in Visual Studio
- Print variable values for testing

Step 11: Building a Simple Calculator Program

Example

```
using System;

class Program
{
   static void Main()
   {
      Console.Write("Enter first number: ");
      int num1 = Convert.ToInt32(Console.ReadLine());

      Console.Write("Enter second number: ");
      int num2 = Convert.ToInt32(Console.ReadLine());

      int sum = num1 + num2;

      Console.WriteLine("Sum = " + sum);
   }
}
```

Real-World Applications of C#

Industry	Usage

Game Development	Unity game engine
Web Development	ASP.NET applications
Cybersecurity	Security tools and automation
Business	Enterprise software
Mobile Apps	Cross-platform applications

Best Practices for Beginners

- Practice coding daily
- Use meaningful variable names
- Comment your code
- Test programs frequently
- Break large problems into smaller steps
- Learn debugging techniques
- Build small projects regularly

Common Beginner Projects

- Calculator
- To-Do List
- Student Grade Tracker
- Password Generator
- Number Guessing Game
- Inventory System

Advantages of C#

Advantage	Description
Easy to Learn	Beginner-friendly syntax

Object-Oriented	Organized programming structure
Versatile	Used for many applications
Secure	Strong security features
Large Community	Extensive tutorials and support

Conclusion

C# is a powerful and flexible programming language widely used in software development, web applications, gaming, cybersecurity, and business environments. Learning C# helps students build computational thinking, problem-solving, and software development skills that are valuable in technology careers. By practicing the fundamentals regularly and building small projects, beginners can develop confidence and advance toward more complex programming concepts.