### Course Introduction

#### QUIZ: Basics of C# Language

* **True or False:** In C#, the statements end with a semi-colon ;
  + True [**answer**]
  + False
* Which of the following symbols are used to define the scope in C#?
  + Parentheses ( )
  + Curly braces { } [**answer**]
  + Square brackets [ ]
  + Forward slashes //
* What is the main purpose of using comments?
  + To maintain multiple versions of source code.
  + To make the source code easier for a programmer to understand. [**answer**]
  + To convey your emotions while writing the code.

#### DOWNLOAD: Complete Source Code

Course Bonuses!

You can download the entire source code, which covers the following from the below-mentioned GitHub Repo:

* Source Codes used in all the demos in this course.
* Assignments (with ready-made projects for you to code right away).
* Assignment Solutions.

**GitHub Repo:**

* <https://github.com/CodeWithPraveen/CSharpBasics.git>
* Also available at: https://github.com/PacktPublishing/C-sharp-Basics-for-Absolute-Beginners-in-C-Sharp-and-dot-NET

Ways to use this Repo:

1. **Clone**: You can use your favorite editor, such as Visual Studio Code, Visual Studio, etc, to clone the repo locally.
2. **Download Zip file**: You can download the zip file of the entire source code and view them locally on your machine.

### Understanding C# Variables

#### Introduction

In this section, “Understanding C# Variables,” you will learn

* + What are Variables?
  + How to work with Text such as characters and strings
  + How to work with Numbers such as whole and decimal numbers.
  + How to work with scenarios involving true or false states.
  + How to work with data collections of the same type in C#.
  + We will demo each type's usage, quizzes, and assignments.

Let us get started!

#### QUIZ: C# Text

* **True or False:** The character variable takes 1 byte of memory.
  + True
  + False [**answer**]
* Which variable type will you use to store a country name?
  + Character
  + String [**answer**]

#### QUIZ: C# Numbers

* **True or False:** uint variable type can store only positive numbers.
  + True [**answer**]
  + False
* Which option below is the right representation for 1,000,000?
  + int i = 1,000,000;
  + int i = 1\_000\_000; [**answer**]
* **True or False:** We must suffix ‘F’ with values for float variables.
  + True [**answer**]
  + False

#### QUIZ: C# Boolean

* **True or False:** Boolean variables can take only true or false as values.
  + True [**answer**]
  + False
* Which of the following Person’s detail can be represented as a boolean variable?
  + City name.
  + Male/Female. [**answer**]
  + Phone number.

#### QUIZ: C# Arrays

* **True or False:** The array index always starts with 0.
  + True [**answer**]
  + False
* Which among the following is a wrong way to declare an integer array of 5.
  + int[] intArray = new int[] { 1, 2, 3, 4, 5};
  + int[] intArray = new int[5];
  + int[5] intArray; [**answer**]

#### Conclusion

In this section, “Understanding C# Variables,” you learned:

* A variable is an identifier whose value can be changed during program execution & is used to store values.
* We can use Character (char) and String (string) variables to work with Text in C#.
* We can use Integer (int, short, long, uint) and Floating (float, double, decimal) types to work with numbers in C#.
* The Boolean (bool) type can work with true or false use cases.
* Finally, Arrays can store a group of variables of the same type.
* Syntaxes and examples for each of the operator types.
* Quizzes to check your understanding of using C# Variables.
* Assignments to apply your new learnings on C# Variables.

In the next section, “Understanding C# Operators, you will learn what operators are and the different types available in C#.

See you in another video!

### Understanding C# Operators

#### Introduction

In this section, “Understanding C# Operators,” you will learn

* What are C# Operators?
* Different types of C# Operators and their uses.
* Quizzes to check your understanding of using various C# Operators.
* Activities to apply your learnings to practice.

Let us get started!

#### QUIZ: C# Unary Operators

* What is the value of the variable *i* in the following code segment?

int a = 100;

int i = ++a;

* + 100
  + 101 [**answer**]
  + Compilation error
* **True or False**: If the operator is shown after the variable, it’s known as a postfix operator.
  + True [**answer**]
  + False

#### QUIZ: C# Mathematical Operators

* What is the value of the variable *result* in the following code segment?

int a = 20;

int b = 3;

int result = a % b;

* + 6
  + 2 [**answer**]
  + 20
* What is the value of the variable *result* in the following altered code segment?

int a = 21;

int b = 3;

int result = a / b;

* + 7 [**answer**]
  + 0
  + 24
  + 18

#### QUIZ: C# Assignment Operators

* What is the value of the variable *a* in the following code segment?

int a = 140;

int b = 50;

a += b;

* + 50
  + 140
  + 280
  + 190 [**answer**]
* What is the value of the variable *result* in the following code segment?

int result = 10;

result \*= 3;

* + 30 [**answer**]
  + 10
  + 13
  + 3
* What is the value of the variable *b* in the following code segment?

int b = 15;

b -= 5;

b \*= 5;

* + 10
  + 25
  + 50 [**answer**]
  + 75

#### QUIZ: C# Comparison Operators

* **True or False:** The operators == and != are opposite to each other in terms of functionality.
  + True [**answer**]
  + False
* What is the value of the variable *result* in the following code segment?

bool result = 5.3 > 3.4;

* + True [**answer**]
  + False
  + Compilation error
* What is the value of the variable *result* in the following code segment?

int a = 9;

int b = 7+2;

bool result = (a != b);

* + True
  + False [**answer**]
  + Compilation error

#### QUIZ: C# Conditional Operators

|  |  |
| --- | --- |
| Lecture Type | Quiz |
| Description | It’s time to check your understanding so far!  This lecture is part of the course “C# Basics: Learn C# Programming with .NET Core”, a guide to learning the fundamentals of C# programming for beginners for starting a C# programming career. |

* Which one among the below is **one** of the Conditional operators in C#?
  + AND
  + && [**answer**]
  + OR
  + \*
* What is the value of the variable *result* in the following code segment?

bool a = true;

bool b = false;

bool result = a && b;

* + True
  + False [**answer**]

#### Conclusion

In this section, “Understanding C# Operators,” you learned:

* + Operators are symbols that inform the compiler to **perform certain operations** on one or two operands.
  + Types of operators:
    - Unary (++, --)
    - Mathematical (+, -, \*, /, %)
    - Assignment (=, +=, -=, \*=, /=)
    - Comparison (==, !=, >, <)
    - Conditional (&&, ||)
    - Bitwise (&, |, ^)
  + **Syntaxes and examples** for each of the operator types.
  + **Quizzes** to check your understanding of using C# Operators.
  + **Assignments** to apply your new learnings on C# Operators.

In the next section, “Understanding C# Selection Statements, you will learn about program flow and how to select a block of code based on certain conditions.

See you in another video!

### Understanding C# Selection Statements

#### Introduction

In this section, “Understanding C# Selection Statements,” you will learn

* Overview of program flow.
* Types of C# Selection statements.
* Demo on the usage of C# Selection statements.
* Quizzes and activities to check your understanding and apply the new learnings.

Let’s get started!

#### QUIZ: C# If Statements

* Are the below statements valid?

string if = “hello, world”;

Console.WriteLine(if);

* + Yes
  + No [**answer**]
* Which among the following is **incorrect**?
  + The *if* statement can contain more than 1 conditional check.
  + The body of the If condition can contain only a single statement. [**answer**]
  + The *else* statement is optional while using an *if* statement.
* When does the *else* part gets executed in the below example assuming *isAllowed* variable is of type bool?

if (isAllowed == true)

{

Console.WriteLine(“Success. It is allowed to execute”);

}

else

{

Console.WriteLine(“Fail. Not allowed to execute”);

}

* + isAllowed value is false. [**answer**]
  + isAllowed value is true.
  + isAllowed is either true or false.
* **True or False**: Nested *if-else* refers to having more than one condition in a single *if* statement.
  + True
  + False [**answer**]

#### QUIZ: C# Switch Statements

* **True or False**: We can’t use case or switch as variable names as they are considered keywords.
  + True [**answer**]
  + False
* Is the below statement correct?

If you have nested if-else statements against a variable with many possibilities, it's **better** to use the ‘switch’ statement.

* + True [**answer**]
  + False
* What is the purpose of the ‘default’ case in the ‘switch’ conditional logic?
  + It gets executed first before other case statements.

Ideally, only one case is executed as per the matching case.

* + It gets executed when no other case statements match the variable value. [**answer**]
  + It always gets executed after executing the matching case statement.

#### ACTIVITY: Coding Assignment 1

**Goal:**

Check your understanding of C# Selection Statements by writing a C# program.

**Duration**:

5-10 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a C# code snippet to read an integer and print it if divisible by 3.

**Additional Resources:**

You can use int.Parse(string) function to convert a string to an integer.

**Evaluate:**

* Execute your program and check the output mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### ACTIVITY: Coding Assignment 2

**Goal:**

Check your understanding of C# Selection Statements by writing a C# program.

**Duration:**

5-10 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Assume a variable phonePrice of type integer holding the price of a phone in dollars.

Write a sequence of C# selection statements to print “budget” or “mid-range” or “high end” based on the price range of the phone - less than 300, between 300 and 800, and more than 800 - respectively.

**Additional Resources:**

None

**Evaluate:**

* Execute your program and check the output mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### ACTIVITY: Coding Assignment 3

**Goal:**

Check your understanding of C# Selection Statements by writing a C# program.

**Duration:**

10-20 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a C# code snippet to check whether an entered password is valid or not as per the below rules:

- Password length between 8 and 14 (both inclusive)

- At least one upper-case letter

- At least one lowercase letter

- At least one numeric letter.

E.g.,

Password123 --> Valid

hello9090 --> Invalid

**Additional Resources:**

I’ve intentionally provided this assignment which involves searching in Google or StackOverflow.com for certain unknowns, such as how to find the presence of upper case letters, etc. This is to inform you of the coding environment and process followed by general C# developers.

**Evaluate:**

* Execute your program and check the output mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### ACTIVITY: Coding Assignment 4

**Goal:**

Check your understanding of C# Selection Statements by writing a C# program.

**Duration:**

10-20 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a simple C# calculator program to do the essential mathematical operations (+, -, \*, /) by reading the type of operation and two values. Use switch statements to check the type of operation entered and print the result to the console.

**Additional Resources:**

You can assume the inputs are valid. Also, assume the numbers entered are positive integer numbers.

**Evaluate:**

* Execute your program and check the output as the problem statement above explains.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### Conclusion

In this section, “Understanding C# Selection Statements,” you learned:

* Program Flow refers to the **sequence of execution** of instructions in a program.
* The ‘if’ statement determines which block to select based on a boolean condition.
* **Two types of C# Selection statements**:
  + if statements.
  + switch statements.
* You validated your learning through quizzes and many assignments.

In the next section, “Understanding C# Iterative Statements,” we will discuss how to execute a block of code repeatedly or navigate through them one at a time.

See you in another video!

### Understanding C# Iterative Statements

#### Introduction

In this section, “Understanding C# Iterative Statements,” you will learn:

* What is meant by iteration, and why are they required?
* Different types of C# Iterative statements.
* Demo on the usage of C# Iterative statements.
* Quizzes and activities to check your understanding and apply the new learnings.

Let’s get started!

#### QUIZ: C# Iterative Statements

* **True or False:** Loop code blocks are always executed only once.
  + True
  + False [**answer**]
* Which among the below options isn’t a looping construct?
  + If Else [**answer**]
  + While
  + Do While
* Which loop can you use to execute the code block at least once?
  + While loop.
  + Do While loop. [**answer**]
  + For loop.

#### ACTIVITY: Coding Assignment 1

**Goal:**

Check your understanding of C# Iterative Statements by writing a C# program.

**Duration:**

5-10 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a C# code snippet to read a positive number from the console and print the total number of digits in it.

For eg., 135 --> 3, 90 --> 2, 98788 --> 5

**Additional Resources:**

You can use int.Parse(string) function to convert a string to an integer.

**Evaluate:**

* Execute your program and check the output as mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### ACTIVITY: Coding Assignment 2

**Goal:**

Check your understanding of C# Iterative Statements by writing a C# program.

**Duration:**

5-10 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a C# code snippet to read a positive integer from the console and print its factorial value.

Factorial of a number refers to the product of all numbers below it.

For eg.,

factorial(3) = 3 \* 2 \* 1 = 6

factorial(5) = 5 \* 4 \* 3 \* 2 \* 1 = 120

**Additional Resources:**

None

**Evaluate:**

* Execute your program and check the output as mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### ACTIVITY: Coding Assignment 3

**Goal:**

Check your understanding of C# Iterative Statements by writing a C# program.

**Duration:**

10-15 mins

**Software needed:**

Visual Studio Code or Visual Studio or Visual Studio for Mac or any other IDE which supports C#.

**Problem Statement:**

Write a C# code snippet to print only the numbers from an array containing both integer and string values.

For eg.,

{ "2", "Red", "67" } --> 2, 67

{ "Hello", "45", "12", "Rose" } --> 45, 12

**Additional Resources:**

None

**Evaluate:**

* Execute your program and check the output as mentioned in the problem statement above.
* You may cross-refer my solution for this assignment provided in the downloaded source code.

#### Conclusion

In this section, “Understanding C# Iterative Statements,” you learned:

* Iteration statements execute a block of statements until a condition is met or for each item in a collection.
* Types of C# Iterative Statements:
  + While statement
  + Do While statement
  + For statement
  + Foreach statement
* You validated your learning through quizzes and many assignments.

In the next section, “Understanding Debugging in C# .NET Environment,” we will discuss the approach for debugging your C# program.

See you in another video!

### Understanding Debugging in C#

#### Introduction

In this section, “Understanding Debugging in C#,” you will learn:

* + Debugging in C#
  + Quizzes and activities to check your understanding and apply the new learnings.

Let’s get started!