multiverse

Contents

Name:
Date:

C

C# is a modern, object-oriented programming language that was developed by Microsoft in the early 2000s. It was designed to be a powerful and efficient language that could be used for a wide range of applications, from desktop software to web development to game programming.

C# is often compared to JavaScript because both languages are commonly used for web development, but they have different histories, purposes, and strengths. JavaScript was developed in the mid-1990s as a client-side scripting language for web browsers, while C# was developed as a general-purpose language for Microsoft's .NET platform.

JavaScript and C# have some similarities in terms of syntax and language features, but they are also distinct in many ways. Here are some of the similarities between the two languages:

- C# and JavaScript are both C-style languages, which means they use similar syntax for things like control structures (`if`/`else` statements, `for`/`while` loops) and function definitions.
- Both languages support object-oriented programming, with features like classes, inheritance, and encapsulation.
- Both languages use curly braces to delimit code blocks and semicolons to end statements.
- C# and JavaScript both have support for lambda expressions, which are a concise way of defining anonymous functions.
- Both languages have built-in support for arrays and other collection types, and both have a similar syntax for accessing array elements and iterating over collections.

However, there are also significant differences between the two languages, especially in terms of their execution environments, runtime behavior, and available libraries and frameworks. For example, JavaScript is typically executed in a web browser, while C# is used in a variety of environments including desktop applications, web services, and game development.

One of the main strengths of C# is its type safety and strong typing system, which helps prevent common programming errors and improves code quality. C# also has a robust standard library, including support for multi-threading, networking, and database access.

Learning Plan Tasks

Name:
Date:

#checkoutTheDocs

- **Microsoft**: Official C# Documentation
- **Microsoft**: Getting Started with C#
- **Udamy**: Understanding C#
- **C# Station**: C# Tutorial
- **C# Yellow Book by Rob Miles**
- **C# Discord Community**
- **C# Corner Discord Community**

1. Getting Started with C

This section covers the following topics:

- Introduction to C# and .NET Framework
- Writing Your First C# Program
- Basic Syntax and Data Types
- Variables, Constants, and Data Types in C#

By completing these tasks you should have a good understanding of the basics of C# programming, including its syntax and data types, how to write a simple C# program, and how to use variables and constants in your code.

Tasks

1. Introduction to C# and .NET Framework
   - Research what C# is and its history
   - Understand the .NET framework and how it relates to C#
   - Learn about the advantages of using C# and the .NET framework for programming
2. Writing Your First C# Program
   - Set up your development environment (Visual Studio, VS Code, or other)
   - Create a new C# project with a `.cs` file extension.
   - Write a simple "Hello, world!" program in C#
   - Build and run your program to verify that it works correctly
3. Basic Syntax and Data Types
   - Print `"Hello World"` to the console using C#
   - Learn the basic syntax of C# programming language
   - Understand C# data types, including integers, floating-point numbers, characters, booleans, and strings.
   - Practice creating and using arithmetic operators and expressions
4. Variables, Constants, and Data Types in C#
   - Explore how to declare and initialize variables in C#
   - Learn the difference between variables and constants in C#
   - Understand how to declare and initialize variables and constants
   - Explore the different C# data types in variable declarations and their uses

- Research C# single and multidimensional array data structures including useful methods.
- Practice using variables and constants to create variables of different data types in your C# programs

5. Build Something

    - **Name Badge Generator**: Create a name prompt program that asks users for their name, job, and contact information. Once the program has this information, output a name badge with this information.
        - Explore how to accept user input through use of the `ReadLine()` method.

2. Control Structures and Functions in C

This section covers the following topics:

- Conditional Statements (`if-else`, `switch`)
- Loops (`for`, `while`, `do-while`)
- Functions in C# (creating and calling)
- Passing Arguments to Functions
- Returning Values from Functions

By completing these tasks you should have a good understanding of how to use conditional statements and loops to control program flow, how to define and call functions in C#, and how to pass arguments and return values from functions.

Tasks

1. Conditional Statements (`if-else`, `switch`)

    - Learn about the different types of conditional statements in C# (`if-else`, `switch`)
    - Understand how to use logical operators (`AND`, `OR`, `NOT`) in C# conditional statements
    - Practice writing conditional statements to control program flow based on conditions. For example, create a simple program that outputs different responses based on the user input.

2. Loops (`for`, `while`, `do-while`)

    - Learn about the different types of loops in C# (`for`, `while`, `do-while`)
    - Understand how to use `break` and `continue` statements in loops
    - Practice writing loops to repeat code based on certain conditions. For example, create a program that prints out all even numbers between 0 and a value specified by a user.

3. Functions in C# (creating and calling)

    - Learn how to define and call functions in C#
    - Understand the difference between methods and functions in C#
    - Practice writing functions that perform simple tasks like a function that prints out a message when called.

4. Passing Arguments to Functions

- Understand how to pass arguments to functions in C#
- Learn about the different types of parameters in C# functions
- Practice passing arguments to functions to perform complex tasks like a function that returns a letter grade when a score is provided.

5. Returning Values from Functions
   - Learn how to return values from functions in C#
   - Understand the difference between void and non-void functions in C#
   - Practice using functions to perform complex tasks and return values to the main program.

6. Build Something
   - **Option #1 - Rock, Paper, Scissors**: You now know enough to rebuild your Rock, Paper, Scissors project with a few new additions. Create a program that does the following:
     – Asks the user for an input of rock, paper, or scissors.
     – Randomly selects rock, paper, or scissors for the computer.
     – Determines the winner of the game based on the random values that were selected.
     – Tracks the player and computers scores.
     – Allows the player to play until they opt to end the game.
   - **Option #2 - Command Line Calculator**: Create a C# console application that performs various operations on a given number based on user input.The program should display a menu to the user with the different options like:
     – Check if a number is positive or negative.
     – Determine if a number is prime.
     – Calculate the factorial of a number.
     – Exit the program

3. Object-Oriented Programming in C

This section covers the following topics:

- Introduction to Object-Oriented Programming (OOP) Concepts
- Creating Classes and Objects in C#
- Defining and Accessing Class Members (fields, properties, methods)
- Inheritance and Polymorphism in C#
- Interfaces and Abstract Classes in C#

By completing these tasks you should have a good understanding of OOP concepts, how to create classes and objects in C#, how to define and access class members, and how to use inheritance, polymorphism, interfaces, and abstract classes in C#.

Tasks

1. Introduction to Object-Oriented Programming (OOP) Concepts
   - Learn about the key concepts of OOP, including encapsulation, inheritance, and polymorphism
   - Understand the benefits of using OOP in software development

- Practice identifying objects, classes, and properties in real-world scenarios

2. Creating Classes and Objects in C#
   - Learn how to create classes in C#
   - Understand how to instantiate objects from classes
   - Practice creating and using objects to perform simple tasks.

3. Defining and Accessing Class Members (fields, properties, methods)
   - Learn how to define fields, properties, and methods in C# classes
   - Understand the differences between fields and properties in C#
   - Practice accessing class members from objects to perform complex tasks
     - E.g. Create a `Person` class with a `name`, `age`, and `hometown` fields. The class should also have a `bio` method that prints on the persons information.

4. Inheritance and Polymorphism in C#
   - Learn about inheritance and polymorphism in C# programming
   - Understand how to create derived classes that inherit from base classes
   - Create a program that uses polymorphism and inheritance to create objects that can be used in a variety of ways
     - E.g. Create a shape calculator that has a parent `Shape` class and child subclasses like `Circle`, `Triangle`, `Square`, etc. Calculate the area of different shapes such as rectangles, circles, and triangles each with a `area()` method.

5. Interfaces and Abstract Classes in C#
   - Learn about interfaces and abstract classes in C# programming
   - Understand the differences between interfaces and abstract classes
   - Practice creating interfaces and abstract classes to define common behavior for classes. For example, recreate the program you created in the last section, but incorporate interfaces and/or abstract classes.

6. Build Something
   - **Option #1 - Scooter Application**: Recreate the Scooter Application from Bootcamp using C#.
   - **Option #2 - Inventory Management System**: Create an online store inventory that allows users to add, update, and view products in an store's inventory. Users should also be able to perform operations such as checking stock availability, calculating total inventory value, and generating inventory reports.

4. TDD, Exception Handling, and File I/O in C

This section covers the following topics:

- Test Driven Development in C#
- Exception Handling in C#
- Handling Exceptions with `try-catch` Blocks
- Reading and Writing Files in C#
- Streams and Byte Arrays in C#

Name:
Date:

By completing these tasks you should have a good understanding of how to create unit tests in C#, handle exceptions in C# programs, how to read and write files using the File and StreamReader/StreamWriter classes, and how to use streams and byte arrays to handle data efficiently in C# programs. Good luck with your learning!

Tasks

1. Test Driven Development in C#
   - Research the NUnit, MSTest, and xUnit.net unit testing framework. Determine which is best based on your company tech stack.
   - Go back to the mini-project you made from the last section and create some tests for functions you you wrote.
   - Use a TDD approach to create some functions by writing the tests ***before*** creating the functions.
     - E.g. Write tests for a function to return the largest element of an array before writing the function itself.
2. Exception Handling in C#
   - Learn about exceptions and how they can be used to handle errors in C# programs
   - Understand the importance of exception handling for writing robust and reliable programs
   - Go back to a previous section's code and identify potential exceptions and define exception types.
3. Handling Exceptions with `try-catch` Blocks
   - Learn how to use `try-catch` blocks to handle exceptions in C#
   - Understand the difference between checked and unchecked exceptions in C#
   - Go back to a previous section's code and practice handling exceptions in C# programs using `try-catch` blocks
4. Reading and Writing Files in C#
   - Learn how to read and write files in C# using the File and `StreamReader`/`StreamWriter` classes
   - Understand how to handle exceptions that may occur when reading or writing files
   - Practice reading and writing files in C# programs to perform simple tasks
     - E.g. Write a C# program that reads a text file and counts the number of words in it. The program should handle file handling and exception handling to ensure that the file exists and can be accessed without errors.
5. Streams and Byte Arrays in C#
   - Learn about streams and byte arrays in C# programming
   - Understand how to use streams to read and write data from files and other sources
   - Practice using byte arrays to efficiently handle data in C# programs

6. Build Something
    - **Option #1 - Word Frequency Analyzer**: Create a word frequency analyzer that reads a text file, process its contents, and generate a report of the frequency of each word in the file. Be sure to utilize exception handling and write unit tests.
    - **Option #2 - Movie Recommendation System**: Create a movie recommendation system that will suggest movies to users based on their preferences and previously watched movies. The program should read a file containing a list of previously watched movies and their genres and recommends movies based on the values in the file. Be sure to utilize exception handling and write unit tests.

5. Advanced Topics in C

This section covers the following topics:

- Delegates and Events in C#
- Generics in C#
- LINQ (Language Integrated Query) in C#
- Asynchronous Programming in C#

By completing these tasks you should have a good understanding of delegates and events, generics, LINQ, and asynchronous programming in C# programming. You'll have a good foundation to continue your learning and tackle more complex programming tasks in the future.

Tasks

1. Delegates and Events in C#
    - Learn about delegates and events in C# programming
    - Understand how to use delegates to define and reference methods
    - Practice using events to handle user input and other program events
2. Generics in C#
    - Learn about generics in C# programming
    - Understand how to use generic types and methods to write reusable and flexible code
    - Practice using generic types and methods in C# programs to perform complex tasks
3. LINQ (Language Integrated Query) in C#
    - Learn about LINQ (Language Integrated Query) in C# programming
    - Understand how to use LINQ to query and manipulate data in C# programs
    - Practice using LINQ to perform complex data queries and manipulations in C# programs
4. Asynchronous Programming in C#
    - Learn about asynchronous programming in C# programming

  - Understand how to use the async and await keywords to write asynchronous code
  - Practice writing asynchronous code in C# programs to improve program performance

5. Build Something
   - **Option #1** - **Task Management Application**: The application will allow users to create, update, and track tasks, as well as perform various operations using LINQ, asynchronous programming, and exception handling techniques.
   - **Option #2** - **Movie Recommendation System**: Update the movie recommendation system from the previous section. The system should suggest movies to users based on their preferences. Use LINQ to query and filter a collection of movies based on various criteria such as genre, rating, or release year. This project will help you master LINQ and its capabilities for querying and manipulating data.
   - **Option #3** - **Event Management System**: Create an event management system that allows users to register for events and receive notifications. Use delegates and events to handle event registration and notification processes. This project will help you understand the concept of delegates and events and their practical application.

Build Something Using C#!

After completing your C# Learning Plan, pick one of the projects below and build it using your newly learned skills:

- **Address Book** - build an application that allows users to add, edit, and delete contacts in an address book. You could use file I/O to store the contacts, and exception handling to handle errors that may occur during contact management.
- **Calculator** - build a calculator application that allows users to perform basic arithmetic operations. You could use classes and objects to encapsulate the calculator's functionality, and control structures to handle user input.
- **Hangman Game** - build a console-based game of Hangman. You could use file I/O to read in a list of words, and classes and objects to encapsulate the game's functionality.
- **File Organizer** - build an application that allows users to organize files on their computer. You could use file I/O to read in a list of files, and classes and objects to encapsulate the file organization logic.