55337AC Introduction to Programming

Overview

Course Duration: 4 Days

About This Course

Have you heard that computer coding is the 'cool' skill to have, but don't know where to start? Or have you been working with no-code solutions and feel as though it's time to level up? In this 4-day course, you'll be introduced to the fundamentals of computer programming. The course does not require prior programming experience and establishes the concepts needed to progress to intermediate courses on programming, such as M55339A – Programming in C#. The course is faithful to the spirit of the original M10975A course but has been completely updated and extended to give a solid grounding in computer programming using the latest tools, and a basis for developing as a professional. These principles can be applied to any object-oriented programming language, e.g., C++, Java, or VB, but in the course we will use the latest version of the C# language and .NET 6. Once you've learned to code in one language it will be easy to transfer those skills to other computer languages as needed. The labs use the free Microsoft Visual Studio 2022 Community edition as a development environment, which provides an excellent environment for learning to code. But it is not intended to be a tutorial about using Visual Studio, and the principles could be applied to other development tools.

The course focuses on core programming concepts such as storage, data types, flow control, and repetition by using looping constructs. The course also introduces object-oriented programming concepts like classes, encapsulation, inheritance, and polymorphism. There are also sections on exception handling, application security, performance, and memory management. A final section aims to prepare a new developer for the software development world by covering good programming practices and coding style and introducing different development approaches such as waterfall and agile.

The course is designed to be delivered in a 4-day format but can be delivered in 3 days for more advanced learners.

Audience Profile

This course is intended for anyone who is new to software development, or has previously used no-code solutions, and wants to gain an understanding of programming fundamentals and object-oriented programming concepts. They will typically be high school students, post-secondary school students, or career changers, with no prior programming experience.

Course Details

1. **Module 1: Introduction to Programming**

This module provides a foundational understanding of how computers process information, looks at different types of software applications, and discusses how code is compiled and run on computer hardware. It also discusses the software development lifecycle.

- Lesson 1: How Computers Work
- Lesson 2: Types of Software Application
- Lesson 3: The Software Development Lifecycle
- Lesson 4: Compiling Code
- Lab: Compiling and Running Code
- Module Review

Module 2: Programming Language Concepts

This module introduces programming language syntax and the syntax rules for C#. It also discusses core data types and how to work with these using variables and constants.

- Lesson 1: C# Syntax
- Lesson 2: Types of Data
- Lesson 3: Working with Variables and Constants
- Lab: Using Different Data Types
- Module Review

Module 3: Understanding Program Flow

This module looks at how code is executed in a computer program and introduces the thinking behind structured programming and the idea of branching in code execution. The module then expands on these concepts using functions, decision structures, and different looping constructs.

- Lesson 1: Fundamentals of Structured Programming
- Lesson 2: Decisions and Branching
- Lesson 3: Calling Functions
- Lesson 4: Decision Structures
- Lesson 5: Looping
- Lab: Decisions, Functions, and Looping
- Module Review

Module 4: Algorithms and Data

This module covers the concept of an algorithm by looking at different ways of expressing algorithms, especially when communicating understanding with stakeholders. The module then explores how to translate those ideas into working code. The module also discusses a number of different simple data structures and collections that can be used in developing algorithms.

- Lesson 1: Formulating Algorithms
- Lesson 2: Implementing Algorithms
- Lesson 3: Working with Data Structures
- Lab: Data Structures and Algorithms
- Module Review

Module 5: Bugs and Errors

This module is intended to give an understanding that errors are an inevitable part of software development. It introduces ways to anticipate and handle those errors in code, and provides a good user experience. The module also covers structured exception handling as a means of handling errors gracefully.

- Lesson 1: Program Bugs and Errors
- Lesson 2: Structured Exception Handling
- Lesson 3: Using Visual Studio Debugging
- Lab: Debugging and Exception Handling

Module 6: Inputs and Outputs

This module covers the fundamental input/output (I/O) concepts beginning with console I/O, and then moving on to File I/O using the various stream APIs and the File API, so that students understand how to persist data using the filesystem.

- Lesson 1: Console I/O
- Lesson 2: File I/O
- Lab: I/O Programming
- Module Review

Module 7: Structures, Objects, and Classes

In this module, the concepts related to object-oriented programming (OOP) are introduced for the first time. This module starts by introducing complex data structures using the struct keyword, before moving on to the basics of object-oriented design and classes. The module provides an understanding of encapsulation as one of the fundamental tenets of object-oriented programming, and the notion of private and public methods and member variables.

1. Lesson 1: Complex Data Structures
2. Lesson 2: Structs
3. Lesson 3: Classes
4. Lesson 4: Encapsulation
5. Lab: Using Complex Data Structures
6. Module Review

Module 8: Object-Oriented Programming

This module follows on from the previous one in teaching students about inheritance and polymorphism in classes. In this module, the students will understand the purpose of classes and how they differ from structs. Function overloading is also introduced as a type of polymorphism. As students learn how to override or change the existing behavior in derived classes, they'll gain an understanding of how proper use of OOP can lead to code that scales and is manageable and easy to maintain. The .NET libraries are also introduced so that students can learn about how they can search the .NET libraries to find and take advantage of existing functionality.

- Lesson 1: Inheritance
- Lesson 2: Polymorphism
- Lesson 3: The .NET Class Libraries
- Lab A: Using Inheritance
- Lab B: Using Polymorphism
- Module Review

Module 9: Application Security and Performance

This module starts with an introduction to application security covering the closely related topics of authentication and authorization. It then delves deeper into the topics of value types and reference types, the stack and the heap, and some of the rules around type conversion. This topic is then further discussed in terms of memory management by discussing how the garbage collector works, and some of the implications for writing well-behaved code.

- Lesson 1: AuthN and AuthZ
- Lesson 2: Value Types, Reference Types, and Type Conversion
- Lesson 3: Collecting the Garbage
- Lab: Comparing Value Types and Reference Types
- Module Review

Module 10: Programming with Style

This module talks about how good application design and good coding discipline will help programmers develop applications that are well structured and maintainable. It introduces the idea of design patterns. It then discusses how to write self-documenting code by good choice of variable names and method names, and coding style. The topic of refactoring and dealing with legacy codebases is then discussed, followed by an introduction to Agile software development practices and how this contrasts with traditional software development methodologies.

- Lesson 1: Design Patterns
- Lesson 2: Principles of Coding Style
- Lesson 3: The Art of Refactoring
- Lesson 4: Agile Practices
- Lab: Refactoring
- Module Review