

## Introduction to Programming (55337AC)

### Course outline

#### *Module 1: Introduction to Core Programming Concepts*

This module provides background and foundational information on how computers process information, discusses the different types of applications that a programmer might be creating, and then provides information on how code is compiled and interpreted by a computer.

#### *Lessons*

- Computer Data Storage and Processing
- Application Types
- Application Lifecycle
- Code Compilation

#### Lab 1: Thinking Like a Computer

- Describe computer data storage and processing concepts
- Describe application types
- Describe the lifecycle of an application
- Describe code compilation

After completing this module, students will be able to:

- Explain core programming fundamentals such as computer storage and processing.
- Explain computer number systems such as binary.
- Create and use variables and constants in programs.
- Explain how to create and use functions in a program.
- Create and use decisions structures in a computer program.
- Create and use repetition (loops) in a computer program.

#### *Module 2: Core Programming Language Concepts*

This module covers programming language syntax and the importance of using good syntax and following the syntax rules for the chosen language. This module also discusses the core data types and how to store these data types in computer memory by using variables and constants.

#### *Lessons*

- Syntax
- Data Types
- Variables and Constants
- 

#### Lab 1: Working with Data Types

- Define syntax
- Explain the different types of core data used in programs
- Declare and use variables and constants in a computer program

After completing this module, students will be able to:

- Define syntax
- Explain the different types of core data used in programs
- Declare and use variables and constants in a computer program

### ***Module 3: Program Flow***

This module covers how code is executed in a computer program, such as top to bottom, in structured programming and branching in code execution. The module teaches these concepts using functions, decision structures, and looping constructs.

#### ***Lessons***

- Introduction to Structured Programming Concepts
- Introduction to Branching
- Using Functions
- Using Decision Structures
- Introducing Repetition

#### Lab 1: Creating Functions, Decisions, and Looping

- Describe structured programming
- Create and use functions in your code
- Create and use decision structures
- Create and use looping structures

After completing this module, students will be able to:

- Describe structured programming
- Create and use functions in your code
- Create and use decision structures
- Create and use looping structures

### ***Module 4: Algorithms and Data Structures***

This module introduces the concept of an algorithm by outlining all the steps required including the decisions to be made as the routine progresses. The module also discusses how to translate these set of steps into pseudocode for evaluation of the algorithm that will be translated into actual code.

#### ***Lessons***

- Understand How to Write Pseudocode
- Algorithm Examples
- Introduction to Data Structures

#### Lab 1: Working with Algorithms and Data Structure

- Transfer problem statements into pseudo code
- Create algorithms
- Translate pseudo code into programming code
- Create simple algorithms in code
- Create data structures to store data

### ***Module 5: Error Handling and Debugging***

This module helps students understand that errors are a part of programming, and they must understand how to anticipate errors, handle those errors in code, and present a good user experience. This module introduces structured exception handling as a mechanism to deal with errors.

#### ***Lessons***

- Introduction to Program Errors
- Introduction to Structured Error Handling
- Introduction to Debugging

#### **Lab 1: Implementing Debugging and Error Handling**

- Implement structured exception handling
- Debug applications by using Visual Studio 2022

### ***Module 6: Introduction to Object-Oriented Programming***

This module covers an introduction to the concepts related to object-oriented programming (OOP). The content has been split across two modules with this module focusing on basic OOP concepts that will provide sufficient knowledge to understand complex data structures starting with structs and then moving on to classes. This module helps the students gain an understanding of how to encapsulate data and related functionality within a class.

#### ***Lessons***

- Introduction to Complex Structures
- Introduction to Structs
- Introduction to Classes
- Introducing Encapsulation

#### **Lab 1: Implementing Complex Data Structures**

- Create and use structure types
- Create and use basic class files
- Choose when to use a struct vs a class

After completing this module, students will be able to:

- Create and use structure types
- Create and use basic class files
- Choose when to use a struct vs a class

### ***Module 7: More Object-Oriented Programming***

This module teaches students about inheritance and polymorphism in classes and function overloading. Function overloading and polymorphism often go hand-in-hand, such as when you inherit from a class, or when you want to override or change the existing behaviour to suit the needs of your class.

The module also introduces the base class library of .NET so that students can start to think about the existence of functionality in other class files and how they can search the .NET libraries to find this functionality and take advantage of it.

## ***Lessons***

- Introduction to Inheritance
- Introduction to Polymorphism
- Introduction to .NET and the Base Class Library

### Lab 1: Implementing Inheritance

- Implementing Inheritance

### Lab 2: Implementing Polymorphism

- Use inheritance in OOP
- Implement polymorphism in your classes
- Describe how the base class library is constructed
- Find class information by using the Object Browser

After completing this module, students will be able to:

- Use inheritance in OOP
- Implement polymorphism in your classes
- Describe how the base class library is constructed
- Find class information by using the Object Browser

## ***Module 8: Introduction to Application Security***

This module helps students think about security in their applications. It introduces the concepts of authentication and authorization for users, and also introduces the concept of permissions for running code. It explains that operating systems might prevent certain aspects of the program from executing, such as saving a file to a directory to which the user running the app might not have permission to write. The module briefly covers code signing and why programmers might want to consider using it.

## ***Lessons***

- Authentication and Authorization
- Code Permissions on Computers
- Introducing Code Signing

### Lab 1: No Lab

- None

After completing this module, students will be able to:

- None

## ***Module 9: Core I/O Programming***

This module introduces some core input/output (I/O) concepts that programmers will use while creating applications. Starting with console I/O, this module introduces input and output to the Console window.

The module also talks about reading from and writing to the filesystem.

## ***Lessons***

- Using Console I/O

- Using File I/O

#### Lab 1: Core I/O Programming

- Read input from a console
- Output data to the console
- Read and write text files

After completing this module, students will be able to:

- Read input from a console
- Output data to the console
- Read and write text files

### ***Module 10: Application Performance and Memory Management***

This module enables students to understand that memory on a computer is a finite resource. It talks about how good application design and good coding discipline with memory management will help programmers learn to develop applications that are fast, responsive, and do not negatively impact other applications.

#### ***Lessons***

- Value Types vs Reference Types
- Converting Types
- The Garbage Collector

#### Lab 1: Using Value Types and Reference Types

- Implement value and reference types correctly in an application
- Convert between value types and reference types
- Use the garbage collector

After completing this module, students will be able to:

- Implement value and reference types correctly in an application
- Convert between value types and reference types
- Use the garbage collector