55337A: Introduction to Programming

Course Details

Course Code: 55337A

Duration: 5 days

Notes:

- This course syllabus should be used to determine whether the course is appropriate for the students, based on their current skills and technical training needs.
- Course content, prices, and availability are subject to change without notice.
- Terms and Conditions apply

Elements of this syllabus are subject to change.

About this course

In this 5-day course, students will learn the basics of computer programming through the use of Microsoft Visual Studio 2022 and the Visual C# and Visual Basic programming languages. The course assumes no prior programming experience and introduces the concepts needed to progress to the intermediate courses on programming, Programming in C#.

The focus will be on core programming concepts such as computer storage, data types, decision structures, and repetition by using loops. The course also covers an introduction to object-oriented programming covering classes, encapsulation, inheritance, and polymorphism. Coverage is also included around exception handling, application security, performance, and memory management.

55337AC is the equivalent of the retired MOC Course 10975AC - Introduction to Programming.

Audience Profile

This course is intended for anyone who is new to software development and wants, or needs, 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. They might want to gain an understanding of the core programming fundamentals before moving on to more advanced courses such as Programming in C#.

At Course Completion

- 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.
- Explain pseudocode and its role in programming.
- Explain the basic computer data structures such as arrays, lists, stacks, and queues.
- Implement object-oriented programming concepts.
- Create and use classes in a computer program.
- Implement encapsulation, inheritance, and polymorphism.
- Describe the base class library (BCL) in the .NET Framework.
- Explain the application security concepts.
- Implement simple I/O in a computer program.
- Identify application errors and explain how to debug an application and handle errors.
- Identify the performance considerations for applications.

Academy IT Pty Ltd
Harmer House
Level 2, 5 Leigh Street
ADELAIDE 5000

Email: sales@academyit.com.au
Web: www.academyit.com.au

Phone: 08 7324 9800
Brian: 0400 112 083

Prerequisites

Before attending this course, students should have:

- Ability to use computers to start programs, open and save files, navigate application menus and interfaces
- Ability to understand logical concepts such as comparisons
- Understand number theory
- Ability to create, understand, and follow structured directions or step-by-step procedures
- Ability to understand and apply abstract concepts to concrete examples.

ACADEMY IT

Table of Contents

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

After completing this module, students will be able to:

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

After completing this module, students will be able to:

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