Unit 1: Programming

Unit code	**D/615/1618**
Unit type	**Core**
Unit level	**4**
Credit value	**15**

Introduction

Programming involves describing processes and procedures which are derived from algorithms. The ability to program is what sets apart a developer and an end user. Typically the role of the developer is to instruct a device (such as a computer) to carry out instructions; the instructions are known as source code and is written in a language that is converted into something the device can understand. The device executes the instructions it is given.

Algorithms help to describe the solution to a problem or task; by identifying the data and the process needed to represent the problem or task *and* the set of steps needed to produce the desired result.

Programming languages typically provide the representation of both the data and the process; they provide control constructs and data types (which can be numbers, words, and objects, and be constant or variable).

The control constructs are used to represent the steps of an algorithm in a convenient yet unambiguous fashion. Algorithms require constructs that can perform sequential processing, selection for decision-making, and iteration for repetitive control. Any programming language that provides these basic features can be used for algorithm representation.

This unit introduces students to the core concepts of programming with an introduction to algorithms and the characteristics of programming paradigms.

Among the topics included in this unit are: introduction to algorithms, procedural, object-orientated & event-driven programming, security considerations, the integrated development environment and the debugging process.

On successful completion of this unit students will be able to design and implement algorithms in a chosen language within a suitable Integrated Development Environment (IDE). This IDE will be used to develop and help track any issues with the code.

As a result they will develop skills such as communication literacy, critical thinking, analysis, reasoning and interpretation which are crucial for gaining employment and developing academic competence.

Learning Outcomes

By the end of this unit students will be able to:

LO1. Define basic algorithms to carry out an operation and outline the process of programming an application.

LO2. Explain the characteristics of procedural, object-orientated and event-driven programming.

LO3. Implement basic algorithms in code using an IDE.

LO4. Determine the debugging process and explain the importance of a coding standard.

Essential Content

LO1 **Define basic algorithms to carry out an operation and outline the process of programming an application**

Algorithm definition:

Writing algorithms to carry out an operation, e.g. Bubble sort.

The relationship between algorithms and code.

The generation process of code; the roles of the pre-processor, compiler and linker, interpreter.

LO2 **Explain the characteristics of procedural, object-orientated and event-driven programming**

Characteristics of code:

Definitions of: data types (the role of constants/variables), methods (including input/output), control structures, iteration, scope, parameter passing, classes, inheritance and events.

Key components of an IDE with a brief explanation each component.

Use of addition of advanced text editors to view code, such as Notepad++, Atom, Sublime text, etc

LO3 **Implement basic algorithms in code using an IDE**

Implementation:

Developing simple applications which implements basic algorithms covered in LO1, using the features of a suitable language and IDE. Consider possible security concerns and how these could be solved.

LO4 **Determine the debugging process and explain the importance of a coding standard**

Review and reflection:

Documentation of the debugging process in the IDE, with reference to watch lists, breakpoints and tracing.

How the debugging process can be used to help developers fix vulnerabilities, defects and bugs in their code.

What a coding standard is and its benefits when writing code.

Learning Outcomes and Assessment Criteria

Pass	Merit	Distinction
LO1 Define basic algorithms to carry out an operation and outline the process of programming an application		
P1 Provide a definition of what an algorithm is and outline the process in building an application.	**M1** Determine the steps taken from writing code to execution.	**D1** Evaluate the implementation of an algorithm in a suitable language and the relationship between the written algorithm and the code variant.
LO2 Explain the characteristics of procedural, object-orientated and event-driven programming		
P2 Give explanations of what procedural, object-orientated and event-driven paradigms are; their characteristics and the relationship between them.	**M2** Compare and contrast the procedural, object orientated and event driven paradigms used in given source code of an application	**D2** Critically evaluate the source code of an application which implements the procedural, object-orientated and event driven paradigms, in terms of the code structure and characteristics.
LO3 Implement basic algorithms in code using an IDE		
P3 Write a program that implements an algorithm using an IDE.	**M3** Use the IDE to manage the development process of the program.	**D3** Evaluate the use of an IDE for development of applications contrasted with not using an IDE.
LO4 Determine the debugging process and explain the importance of a coding standard		
P4 Explain the debugging process and explain the debugging facilities available in the IDE. **P5** Outline the coding standard you have used in your code.	**M4** Evaluate how the debugging process can be used to help develop more secure, robust applications.	**D4** Critically evaluate why a coding standard is necessary in a team as well as for the individual.

Recommended Resources

This unit does not specify which programme language should be used to deliver this content – this decision can be made by the tutor.

Examples of languages that are used in industry are C#, Python, Ruby, Java, but any language which will allow the student to achieve the Learning Outcomes is acceptable.

Textbooks

AHO, A. V. et al. (1987) *Data Structures and Algorithms*. 1st Ed. Addison-Wesley.

HUNT, A. et al. (2000) *The Pragmatic Programmer: From Journeyman to Master.* 1st Ed. Addison-Wesley.

MCCONNELL, S. (2004) *Code Complete: A Practical Handbook of Software Construction*. 2nd Ed. Microsoft Press.

Links

This unit links to the following related units:

Unit 19: Data Structures & Algorithms

Unit 20: Advanced Programming

Unit 28: Prototyping