Unit 4: Programming

Level: **3**

Unit type: **Internal**

Guided learning hours: **90**

Unit in brief

Learners study the underpinning concepts and implications of programming languages to design, develop and test computer programs.

Unit introduction

Organisations and individuals increasingly depend on the functions and services offered by computing devices such as smartphones, tablets, laptops and personal desktop computers. You make use of computing programs when using an operating system or application programs such as word processing and spreadsheets. Understanding the concepts of high-quality software application design and development is key to ensuring that products are effective. As a programmer, you will need to understand the characteristics of different programming languages in order to select and apply appropriate methodologies to meet a client's needs.

Many organisations and businesses rely on computer programs to help deliver products and services. Organisations and businesses (often known as 'clients') work closely with programmers to help design and build computer programs that fulfil their requirements. To complete the assessment task within this unit, you will need to draw on your learning from across your programme of study and apply programming skills to provide a solution for a new IT-related problem.

You will learn about computational thinking skills and the principles of designing and developing computer programs. You will apply computational thinking skills to design, develop, test, refine and review computer programs for a given range of purposes. By developing your analytical, problem-solving and programming skills, this unit will help you to progress to higher education or to employment as a software developer.

Learning aims

In this unit you will:

A Examine the computational thinking skills and principles of computer programming

B Design a software solution to meet client requirements

C Develop a software solution to meet client requirements.

Summary of unit

Learning aim	Key content areas	Recommended assessment approach
A Examine the computational thinking skills and principles of computer programming	**A1** Computational thinking skills **A2** Uses of software applications **A3** Features and characteristics of programming languages **A4** Constructs and techniques and their implementation in different languages **A5** Principles of logic applied to program design **A6** Quality of software applications	A report evaluating computational thinking skills and how the principles of software design and computer programming are applied to create effective, high-quality software applications.
B Design a software solution to meet client requirements	**B1** Software development life cycle **B2** Software solutions design	A project brief identifying the scope of the problem and user/client requirements. Design documentation for the suggested solution. User feedback and design refinement documentation. Development and support documentation, including development and testing logs, meeting notes and a report that evaluates the outcomes and development of the project.
C Develop a software solution to meet client requirements	**C1** Software solutions development **C2** Testing software solutions **C3** Improvement, refinement and optimisation of software applications **C4** Review of software solutions **C5** Skills, knowledge and behaviours	

Content

Learning aim A: Examine the computational thinking skills and principles of computer programming

A1 Computational thinking skills

Application of computational thinking skills involved in analysing problems and processes, in order to identify solutions that can be developed into software applications.

- Decomposition:
  - identifying and describing problems and processes
  - breaking down problems and processes into distinct steps
  - describing problems and processes as a set of structured steps
  - communicating the key features of problems and processes to others as relevant.
- Pattern recognition:
  - identifying common elements or features in problems or systems
  - identifying and interpreting common differences between processes or problems
  - identifying individual elements within problems
  - describing patterns that have been identified
  - making predictions based on identified patterns.
- Pattern generalisation and abstraction:
  - identifying information required to solve an identified problem
  - filtering out information required to solve an identified problem.
- Representing parts of a problem or system in general terms by identifying:
  - variables
  - constants
  - key processes
  - repeated processes
  - inputs
  - outputs.

A2 Uses of software applications

The uses and implications of software applications in solving problems and fulfilling needs, including:

- gaming and entertainment
- productivity
- information storage and management
- repetitive tasks or dangerous tasks
- social media
- search engines.

A3 Features and characteristics of programming languages

- The uses and applications of different types of high and low-level programming languages, developed to assist in the solution of particular problems, such as:
  - procedural, e.g. C, Perl®, Python™
  - object-orientated, e.g. C++, C#®, Java®
  - event-driven, e.g. Visual Basic®
  - machine, e.g. Assembler
  - mark-up, e.g. HTML.

- Factors to compare and contrast in programming languages, including:
  - hardware and software needed for running and developing a program
  - special devices required
  - performance
  - preferred application areas
  - development time
  - ease of development.

A4 Constructs and techniques and their implementation in different languages

- Programming languages, constructs and techniques, including:
  - command words
  - constants and variables, local and global variables
  - data types – character, string, integer, real, Boolean
  - statements – assignment, input and output, sequence, iteration, selection
  - logical operations.
- Other constructs, such as:
  - subroutines, functions and procedures
  - string handling, including examining single characters and substrings
  - arrays – two-dimensional and three-dimensional, splitting and joining
  - file handling – open, read, write, close, database
  - data structures
  - event handling.
- Documentation of code.

A5 Principles of logic applied to program design

Principles, including:

- iteration – repetition of a computational procedure applied to the result of a previous application
- mathematical logic – inference, consistency, completeness, verification by truth tables
- propositional dynamic logic to demonstrate the function of algorithms
- use of sets, e.g. properties and interrelationships of sets of data, search/filter sets of data.

A6 Quality of software applications

How the design and implementation of a software application affects quality, including:

- efficiency/performance, e.g. the system resources consumed by the program, CPU cycles, processor time, memory space, accessing storage media
- maintainability, e.g. ease with which a program can be modified by its present or future developer in order to carry out corrective, perfective or adaptive maintenance
- portability, e.g. range of computer hardware, operating systems and platforms on which the source code can be run/compiled/interpreted
- reliability, e.g. accuracy and the consistency of its outputs
- robustness, e.g. quality of coding and testing to ensure that extreme and erroneous data can be processed without causing the program to crash
- usability, e.g. ease with which an end user can use the program.

Learning aim B: Design a software solution to meet client requirements

B1 Software development life cycle

Application of the software development life-cycle stages, including:

- assessment of the requirements for an identified problem
- design specification, e.g. scope, inputs/outputs, user interface, timescales
- develop code
- implementation
- test, e.g. white box and black box testing, refinement, optimisation
- maintenance, e.g. corrective, adaptive and increased functionality.

B2 Software solutions design

- Problem definition statements, to include: intended users, full summary of the problem to be solved, constraints, benefits, nature of interactivity, complexity of problem.
- Purpose and any other requirements as defined in a client brief.
- Features of software:
  - description of main program tasks, input and output formats
  - diagrammatic illustrations, to include screen layouts, user interfaces, navigation
  - algorithms and processing stages, to include flowcharts, pseudocode and events
  - data structures
  - data storage
  - control structures
  - data validation
  - error handling and reporting.
- Choice of language.
- List of pre-defined programs and/or code snippets.
- List of ready-made and/or original assets such as a digital animation, digital graphic, digital audio and video.
- Feedback from others to help refine alternative design ideas/prototypes and make decisions.
- Test plan with test data to include typical, extreme and erroneous data.
- Technical and design constraints, e.g. connectivity, memory storage, programming languages.

Learning aim C: Develop a software solution to meet client requirements

C1 Software solutions development

The process of software development, including:

- the development environment to produce code
- the development and refinement of software programs using a suitable programming language
- library routines, standard code and user generated subroutines used to add to the efficiency of a program.

C2 Testing software solutions

Testing of the programs, including:

- test plan
- test data – typical, extreme and erroneous data
- selection and use of appropriate types of testing to test part or all of a program, e.g. functional testing, stability, compatibility.

C3 Improvement, refinement and optimisation of software applications

Methods of improving, refining and optimising, e.g.:

- annotated code to allow effective repair/debugging of the program and maintainability
- program compilation for a designated platform or environment
- review – quality of a program in terms of reliability, usability, efficiency/performance, maintainability, portability
- eliciting feedback from users
- making use of the outcomes of testing and feedback
- documenting changes to the design and solution.

C4 Review of software solutions

Evaluation of software solutions, including:

- suitability for audience and purpose
- ease of use
- quality of the software solution, e.g. reliability, usability, efficiency/performance, maintainability, portability
- constraints of the programming language
- other constraints, e.g. time, programmer knowledge, rules of languages vary with implementation
- strengths and weaknesses of the software solutions
- improvements that can be made
- optimising software solutions, e.g. improving robustness, improving efficiency of the code, adding additional functionality.

C5 Skills, knowledge and behaviours

- Planning and recording, including the setting of relevant targets with timescales, how and when feedback from others will be gathered.
- Reviewing and responding to outcomes, including the use of feedback from others, e.g. IT professionals and users who can provide feedback on the quality of the program and its suitability when assessed against the original requirements.
- Demonstrating own behaviours and their impact on outcomes, to include professionalism, etiquette, supportive of others, timely and appropriate leadership, accountability and individual responsibility.
- Evaluating outcomes to help inform high-quality justified recommendations and decisions.
- Evaluating targets to obtain insights into own performance.
- Media and communication skills, including:
  - the ability to convey intended meaning, e.g. written (email, design documentation, recording documentation, reports, visual aids for presentation use); verbal communication requirements (one to one and group informal and formal situations)
  - use of tone and language for verbal and written communications to convey intended meaning and make a positive and constructive impact on audience, e.g. positive and engaging tone, technical/ vocational language suitable for intended audience, and avoidance of jargon
  - responding constructively to the contributions of others, e.g. supportive, managing contributions so all have the opportunity to contribute, responding to objections, managing expectations and resolving conflict.

Assessment criteria

<table>
<tr><th>Pass</th><th>Merit</th><th>Distinction</th></tr>
<tr><td colspan="2">Learning aim A: Examine the computational thinking skills and principles of computer programming</td><td rowspan="2">A.D1 Evaluate how computational thinking skills can impact software design and the quality of the software applications produced.</td></tr>
<tr><td>A.P1 Explain how computational thinking skills are applied in finding solutions that can be interpreted into software applications.
A.P2 Explain how principles of computer programming are applied in different languages to produce software applications.
A.P3 Explain how the principles of software design are used to produce high-quality software applications that meet the needs of users.</td><td>A.M1 Analyse how computational thinking skills can impact software design and the quality of the software applications produced.</td></tr>
<tr><td colspan="2">Learning aim B: Design a software solution to meet client requirements</td><td rowspan="3">BC.D2 Evaluate the design and optimised computer program against client requirements.
BC.D3 Demonstrate individual responsibility, creativity and effective self-management in the design, development and review of the computer program.</td></tr>
<tr><td>B.P4 Produce a design for a computer program to meet client requirements.
B.P5 Review the design with others to identify and inform improvements to the proposed solution.</td><td>B.M2 Justify design decisions, showing how the design will result in an effective solution.</td></tr>
<tr><td colspan="2">Learning aim C: Develop a software solution to meet client requirements</td></tr>
<tr><td>C.P6 Produce a computer program that meets client requirements.
C.P7 Review the extent to which the computer program meets client requirements.</td><td>C.M3 Optimise the computer program to meet client requirements.</td><td></td></tr>
</table>

Essential information for assignments

The recommended structure of assessment is shown in the unit summary along with suitable forms of evidence. *Section 6* gives information on setting assignments and there is further information on our website.

There is a maximum number of two summative assignments for this unit. The relationship of the learning aims and criteria is:

Learning aim: A (A.P1, A.P2, A.P3, A.M1, A.D1)

Learning aims: B and C (B.P4, B.P5, C.P6, C.P7, B.M2, C.M3, BC.D2, BC.D3)

Further information for teachers and assessors

Resource requirements

For this unit, learners must have access to a range of programming languages, IDEs (integrated development environment) and diagramming tools to allow them to use a variety of tools and techniques (given in the unit content) to design and develop computer programs.

Learners will need access to examples of programs and code bases written in a range of languages for a number of different purposes. While access to the code base of many proprietary applications is restricted, there are many open-source alternatives that can be used.

Essential information for assessment decisions

Learning aim A

Evidence for this assignment will be in the form of a written response that investigates computational thinking skills and the principles and purpose of different programming languages. The report will make use of specific examples of code implementation (and the chosen paradigm) to explore how the example code has been implemented to meet specific needs.

The code base used by learners in their investigation must be of sufficient complexity to allow analysis of the implementation of a range of programming constructs, including standard and language-specific techniques, logical structures and mathematical principles.

For distinction standard, learners will provide an evaluation of how computational thinking skills are used to find solutions to problems and how this can impact software design and the applications developed. They will provide a clear and balanced evaluation of the use of different programming languages (in identified programs) to solve different, specific problems. Learners will provide a detailed analysis of the programming principles used in the identified programmes. They will evaluate the success of their implementation in terms of the quality of code produced, and in a wider context where applicable. Quality will be considered in terms of the degree to which user requirements are met, the robustness of the code, its maintainability, efficiency, portability and ease of use.

Learners will provide an evaluation of the identified programming languages. They will consider the principles they have analysed and explain why specific programming languages are used and what advantages they may offer to the programmer and the end user.

Learners must articulate their arguments and views fluently and concisely, providing an evaluation that makes reasoned and valid judgements. The evidence will demonstrate high-quality written/ oral communication through the use of accurate and fluent technical vocabulary to support a well-structured and considered response that clearly connects chains of reasoning.

For merit standard, learners will analyse how computational thinking skills can impact software design, highlighting features of decomposition, pattern recognition and pattern generalisation and abstraction. Learners will show a clear understanding of how different programming languages are implemented to solve problems. They will provide a balanced and reasoned analysis of the strengths and weaknesses of the identified code in solving the problems and the quality of the implementations. They will analyse the strengths and weaknesses of the identified languages and how they affect the requirements of the user and the development of a program to meet defined needs. The evidence will be technically accurate and demonstrate good-quality written or oral communication.

For pass standard, learners will explain how computational thinking skills are used to find solutions to problems. They will explain the range of programming languages available, as given in the unit content. Learners will explain how each differs in terms of constructs, techniques, use and requirements. They must choose one example program that has been created to solve a particular problem/meet a specific need, and provide descriptions of how programming constructs and the principles of software design have been applied to develop a solution to meet the required needs of users. Learners will also consider how computational thinking skills may have been applied when exploring the principles of software design. They will explain how different software design methods can be used to produce effective applications. This can be achieved by using supporting examples. The evidence may have some inaccuracies and may include limited use of examples to illustrate the explanations.

Learning aims B and C

Learners must develop a program to solve a specific problem. The problem must be of significant complexity to allow learners to demonstrate the application of a range of appropriate problem solving and programming skills.

For distinction standard, learners will draw on and show synthesis of knowledge across the learning aims to produce a detailed evaluation of the planning, development and refinement of the solutions in line with client requirements. They must explain methodologies applied throughout the process and justify their use in ensuring the requirements of the client are met.

Learners must provide a thorough evaluation of the effectiveness of the final program, including a systematic evaluation of the techniques, principles and constructs applied in their program. They will provide well-considered, justifiable suggestions for future improvements to the program.

Evaluation of behaviours will consider learners' use of 'soft skills' in relation to the vocational context of the project, such as managing and liaising with other members of the team or clients and time management. Learners will evaluate their own behaviours throughout the project and the impact they have on the outcomes. Learners must refer to tangible evidence to support their evaluation such as meeting notes, correspondence and time plans.

For merit standard, learners will apply their knowledge through the selection and application of appropriate methodologies to plan, develop and test an effective, optimised computer program. Learners will use feedback from others to identify how their design could be improved and produce a solution design.

Learners must provide a clear, accurate and well-reasoned justification for the decisions made throughout the development of the program, linking decisions to their effectiveness in meeting user requirements. In doing this, learners will optimise the effectiveness and efficiency of their solution in line with the user requirements. They will take feedback from others into account and explain how they decided to accept or reject recommendations.

Learners must optimise their computer program by making use of testing and feedback throughout development to improve and refine their code to fully meet client requirements, for example improving data validation procedures, the efficiency of the code or the usability of the program.

For pass standard, learners will apply their understanding of the software development life cycle to design and develop a computer program to meet identified requirements. Learners must apply an understanding of client requirements and provide planning documentation that demonstrate the possible solutions to the identified problems. They will seek feedback on their design and use this feedback to improve the quality of their design solution for the problem.

Learners must produce evidence that the finished program has been tested using a number of different appropriate testing methods to ensure they are functional. They must produce solutions that meet the requirements of the client; however some small issues may persist.

Learners must provide a review of whether their work meets client requirements, considering both positive and negative aspects of the outcomes, although their review may be unbalanced and/or superficial. They will use relevant feedback, such as client feedback, to make suggestions regarding possible alternative solutions that could be implemented.

Links to other units

The assessment for this unit should draw on knowledge, understanding and skills developed from:

- Unit 1: Information Technology Systems
- Unit 2: Creating Systems to Manage Information
- Unit 3: Using Social Media in Business.

This unit would relate to teaching of:

- Unit 5: Data Modelling
- Unit 7: Mobile Apps Development
- Unit 8: Computer Games Development.

Employer involvement

This unit would benefit from employer involvement in the form of:

- guest speakers
- technical workshops involving staff from local organisations/businesses
- contribution of design/ideas to unit assignment/scenario/case study/project materials, including own organisation/business materials as exemplars where appropriate
- feedback from staff from local organisations/businesses on plans/designs/items developed
- opportunities for observation of organisational/business application during work experience
- support from local organisation/business staff as mentors.