National Unit Specification

General information

Unit title: Computer Programming (SCQF level 5)

Unit code: HY2C 45

Superclass: CB

Publication date: May 2018

Source: Scottish Qualifications Authority

Version: 02

Unit purpose

The purpose of this unit is to provide programming skills and knowledge of the principles of computer programming.

This is a **non-specialist** unit, suitable for a **wide range of learners**. It is designed for learners who require an introduction to coding for vocational purposes, and for learners who wish to appreciate programming for academic or personal reasons. It is particularly suitable for learners with an interest in STEM. No previous programming experience is required.

Learners will gain a range of practical skills and acquire relevant underpinning knowledge. They will learn how to write code in a contemporary high-level language and appreciate programming concepts and techniques, and develop their computational thinking skills.

On completion of this unit, learners will know how to write programs to solve real-world problems. Learners will be able to apply programming concepts by implementing them in a programming environment.

Learners may progress to HY2C 46 *Computer Programming* at SCQF level 6.

Outcomes

On successful completion of the unit, the learner will be able to:

1 Write algorithms to solve routine problems.
2 Explain programming concepts.
3 Write a computer program.

National Unit Specification: General information (cont)

Unit title: Computer Programming (SCQF level 5)

Credit points and level

1 National Unit credit(s) at SCQF level 5: (6 SCQF credit points at SCQF level 5).

Recommended entry to the unit

Entry is at the discretion of the centre and no previous knowledge or programming experience is required, but it is recommended that learners possess a basic level of ICT skills in using a computer within a modern operating system.

Core Skills

Achievement of this Unit gives automatic certification of the following Core Skills component:

Complete Core Skill	None
Core Skill component	Critical Thinking at SCQF level 5 Planning and Organising at SCQF level 5

There are also opportunities to develop aspects of Core Skills which are highlighted in the Support Notes of this Unit specification.

Context for delivery

If this unit is delivered as part of a group award, it is recommended that it should be taught and assessed within the subject area of the group award to which it contributes.

Equality and inclusion

This unit specification has been designed to ensure that there are no unnecessary barriers to learning or assessment. The individual needs of learners should be taken into account when planning learning experiences, selecting assessment methods or considering alternative evidence.

Further advice can be found on our website **www.sqa.org.uk/assessmentarrangements**.

National Unit Specification: Statement of standards

Unit title: Computer Programming (SCQF level 5)

Acceptable performance in this unit will be the satisfactory achievement of the standards set out in this part of the unit specification. All sections of the statement of standards are mandatory and cannot be altered without reference to SQA.

Where evidence for outcomes is assessed on a sample basis, the whole of the content listed in the performance criteria section must be taught and available for assessment. Learners should not know in advance the items on which they will be assessed and different items should be sampled on each assessment occasion.

Outcome 1

Write algorithms to solve routine problems.

Performance criteria

(a) Describe common methods of writing algorithms
(b) Create algorithms by breaking down routine problems into logical steps using stepwise refinement
(c) Represent sequence, selection, iteration and subroutines using algorithms
(d) Refine an algorithm to improve its efficiency

Outcome 2

Explain programming concepts.

Performance criteria

(a) Explain the concept of syntax and semantics
(b) Explain the concepts of data types and data structures, including naming conventions
(c) Explain the concepts of instructions, sequence, selection, iteration and subroutines
(d) Explain the correct construction of expressions, including operators, assignment statements and the order of evaluation
(e) Explain the correct use of arithmetic, relational and logical operators
(f) Explain the concepts of program testing and debugging

Outcome 3

Write a computer program.

Performance criteria

(a) Implement data structures in code, adhering to naming conventions
(b) Implement an algorithm in code
(c) Implement expressions in code
(d) Implement subroutines in code
(e) Combine instructions, sequences, selections and iterations to create a complete, working program, which accurately implements an algorithm
(f) Debug code to eliminate errors

National Unit Specification: Statement of standards (cont)

Unit title: Computer Programming (SCQF level 5)

Evidence requirements for this unit

Learners will need to provide evidence to demonstrate the performance criteria across all outcomes.

Evidence is required to demonstrate that learners have achieved all outcomes and performance criteria. Assessors should use their professional judgement, subject knowledge and experience, and understanding of their learners to determine the most appropriate ways to generate evidence and the conditions and contexts in which they are used.

The evidence requirements for this unit will consist of **two** types of evidence: **knowledge** evidence and **product** evidence.

The **knowledge evidence** relates to Outcomes 1 and 2. It will comprise explicit knowledge (such as knowledge of syntax and semantics) and underpinning knowledge (such as knowledge of data structures). Evidence for all performance criteria must be produced, except when sampling is used. The evidence may be written or oral or a combination of these. Evidence may be captured, stored and presented in a range of media (including audio and video) and formats (analogue and digital).

The knowledge evidence should demonstrate an understanding of the following concepts:

- Common methods of writing algorithms (such as flowcharts, pseudocode, etc)
- Syntax and semantics
- Data types, data structures and naming conventions
- Instructions, sequence, selection, iteration and subroutines
- Correct construction of expressions, including operators, assignment statements and the order of evaluation
- Correct use of arithmetic, relational and logical operators
- Program testing and debugging

Sampling of knowledge is permissible in certain contexts, such as when traditional testing is used to generate the evidence. When sampling is used, the sampling frame must be broad enough to ensure that every outcome (but not necessarily every performance criterion) is included. In this circumstance, the test must be carried out under controlled, supervised and timed conditions, without access to reference materials. Where traditional testing is used, this must be carried out in a single assessment occasion ('sitting') and an appropriate pass mark should be set. Where re-assessment is required, it should contain a significantly different sample from that previously used.

The **product evidence** will relate to Outcomes 1 and 3, and take the form of **at least two** algorithms and **at least one** program. It will demonstrate the learner's computational thinking skills by creating algorithms to solve routine problems, and will demonstrate the learner's understanding of key programming concepts and coding skills by creating at least one program. This evidence may be produced over the life of the unit, under loosely controlled conditions (including access to reference materials).

National Unit Specification: Statement of standards (cont)

Unit title: Computer Programming (SCQF level 5)

The **product** evidence is required to demonstrate that the learner can:

- create **at least two** algorithms for at least two different types of routine problem
- refine **at least one** algorithm to make it more efficient
- create one large program or two short programs based on algorithm(s)
- use variables and naming conventions correctly
- use **at least three** different data types
- use **at least one** data structure (array or equivalent)
- implement **at least two** different arithmetic operators
- implement **at least two** assignment operators
- implement **at least two** relational operators
- implement **at least two** logical operators
- implement selection, including conditional statements
- implement sequence and iteration with at least two different types of loops
- use subroutines (which could be, for example, functions, procedures, methods, behaviours or actions, depending on the programming language)

The problems to be solved may be routine but must be non-trivial. Their solution should require relatively lengthy algorithms and significant (if non-complex) coding.

The **product** evidence may be generated in supervised or unsupervised conditions with access to learning materials (open-book). When evidence is produced in uncontrolled or loosely controlled conditions, it must be authenticated. The *Guide to Assessment* provides further advice on methods of authentication. It is anticipated that the evidence will be submitted electronically.

The SCQF level (Level 5) of this unit provides additional context on the nature of the required evidence and the associated standards. The SCQF level descriptors should be used (explicitly or implicitly) when making judgements about the evidence.

The support notes provide specific examples of instruments of assessment that could be used to generate the required evidence (see *Guidelines on Approaches to Assessment*).

National Unit Support Notes

Unit title: Computer Programming (SCQF level 5)

Unit support notes are offered as guidance and are not mandatory.

While the exact time allocated to this unit is at the discretion of the centre, the notional design length is 40 hours.

Guidance on the content and context for this unit

The purpose of this unit is to introduce learners to problem solving using algorithms (computational thinking), as well as the key common concepts underpinning all programming languages. Learners will be given the opportunity to develop skills, knowledge and understanding of computational thinking, programming concepts, and implement those programming concepts and techniques, while taking a problem from inception to solution.

This unit is intended for learners studying any type of computing and is particularly suitable for learners who want to progress to, or who are on, Computer Science courses or related courses, such as Software Development or Games Development. It is suitable for learners who have little or no experience of programming, but will also be valuable for those with previous experience of programming.

The unit covers the following knowledge and skills: algorithms, flowcharts, variables, data types, arithmetic operators, assignment operators, relational operators, logical operators, sequence, selection, iteration, data structures (arrays), subroutines and naming conventions.

Although the focus is on acquiring programming skills that are transferrable and can apply to all programming languages, it is anticipated that learners will gain experience in one or two modern programming languages and Integrated Development Environments (IDE).

The type of problems set for learners and the development environment they will use are at the discretion of the centre and will vary depending on the resources available to the centre. If it is delivered as part of an award, the problems set and development environment may reflect the subject area of the award and the emphasis individual centres have placed on that award. The chosen language could be Python, Basic, JavaScript, Java, C#, C++, and the IDE could be Visual Studio. However, any other modern programming language and environment, which make use of the programming concepts that must be covered, would be acceptable.

It is recommended that centres produce a brief for the learners, stating the problems to solve and the development environment or environments available to them. It is up to centres as to how prescriptive and specific the brief should be in terms of the problems set and the development environments they can make use of. It is anticipated that most centres will recommend a single development environment for learners to use.

National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 5)

Please note that the following guidance, relating to specific outcomes, does not seek to explain each performance criterion, which is left to the professionalism of the teacher. It seeks to clarify the statement of standards where it is potentially ambiguous. It also focuses on non-apparent teaching and learning issues that may be over-looked, or not emphasised, during unit delivery. As such, it is not representative of the relative importance of each outcome or performance criterion.

Outcome 1

This outcome covers computational thinking by teaching learners how to create algorithms by breaking down routine problems into logical steps. These problems should be set at an appropriate level for students studying at SCQF level 5. Learners will produce flowcharts, which can be based on the algorithms they create. Learners will learn how to refine algorithms and/or flowcharts to make them more efficient.

Outcomes 2 and 3

Outcome 2 covers key programming concepts, including variables, naming conventions, data types, arithmetic operators, assignment operators, relational operators, logical operators, sequence, selection, iteration, data structures (arrays) and subroutines.

The same programming concepts are covered in Outcome 3, but whereas in Outcome 2 learners are expected to gain an understanding of these concepts, in Outcome 3 they must be able to implement them in short programs based on algorithms.

Guidance on approaches to delivery of this unit

It is anticipated that learners would complete the outcomes sequentially, so they would complete Outcome 1, before moving onto Outcome 2, before finally moving onto Outcome 3. However, that does not mean that they should only cover the knowledge and skills required for each individual outcome before attempting the summative assessment for each outcome. It is anticipated that the knowledge and skills required for Outcome 1 will be covered first and that summative assessment may occur at that point before moving onto Outcome 2. However, it is expected that learners will cover the knowledge and skills required for Outcomes 2 and 3, before attempting the summative assessments for both of those outcomes.

For **Outcome 1,** learners should learn how to create algorithms to solve routine problems. This requires looking at how problems can be broken down into logical steps, which is computational thinking. They should also look at how solutions to problems (aka, algorithms) can be displayed as flowcharts and how algorithms can be refined to be more efficient. The problems set do not need to be computing problems; in fact, initially, it would make sense to look at problems that they are familiar with, such as how to get from home to college, etc. Flowcharts will allow them to think more about selection, sequence, iteration (loops), and express those concepts.

For **Outcomes 2 and 3,** learners should learn about the programming concepts, techniques and skills mentioned in the content section. They should spend most of their time learning about each concept and then putting it into practice in the chosen programming language. Simple algorithms should be provided to them, which they should be tasked with turning into code.

National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 5)

A suggested distribution of time, across the outcomes, is:

Outcome 1: 12 hours
Outcome 2: 14 hours
Outcome 3: 14 hours

Summative assessment may be carried out at any time. However, when testing is used (see evidence requirements) it is recommended that this is carried out towards the end of the unit (but with sufficient time for remediation and re-assessment). When continuous assessment is used (such as the use of a web log), this could commence early in the life of the unit and be carried out throughout the life of the unit.

There are opportunities to carry out formative assessment at various stages in the unit. For example, formative assessment could be carried out on the completion of each outcome to ensure that learners have grasped the knowledge contained within it. This would provide assessors with an opportunity to diagnose misconceptions, and intervene to remedy them before progressing to the next outcome.

Guidance on approaches to assessment of this unit

Evidence can be generated using several types of assessment. The following are suggestions only. There may be other methods that would be more suitable to learners. Assessment evidence is required at all stages and outcomes. It must be documented and recorded electronically or in written/printed form; however, centres are encouraged to look at alternate approaches making use of modern technology.

One approach to assessment is the use of **selected-response questions** and a **practical assignment** as the instruments of assessment. The selected response questions would generate the knowledge evidence. The practical assignment would generate the product evidence.

The selected-response questions could take the form of test that samples across Outcomes 1 and 2. For example, the test could comprise 25 multiple-choice questions, with five covering Outcome 1 and 20 covering Outcome 2. An appropriate pass mark would need to be set. The resulting evidence would be the learner's (marked) responses. If this form of assessment is used, it should be carried out under supervised, closed-book conditions. It can be carried out via a digital online assessment or a paper-based one (although a digital online test is preferred).

Alternatively, the evidence could be generated via the use of a smaller number of short-answer questions, which could be written or oral. These questions could cover multiple concepts in each question and, therefore, may not require as many questions. Evidence could also be generated by oral questioning requiring explanations of key programming concepts, as demonstrated in the program(s) created for Outcome 3. In this case, the learner would create the program(s) first and would then answer questions on the code to demonstrate a full understanding of the concepts. In this circumstance, all three outcomes would be assessed as one large assignment.

A final approach would be a combination of short answer questions and multiple-choice questions. The short answer questions do not need to be written or carried out online; they could be verbal questions and answers, which could be evidenced by a recording or a checklist.

National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 5)

If a traditional test is used to assess the learner's knowledge and understanding, this test should be timed and should be completed in a single assessment occasion ('sitting') and an appropriate pass mark being set. Where re-assessment is required, it should contain a different sample from that previously used. If a multiple-choice test is selected, then it is recommended that it should be completed within an hour and should have a pass mark of 60%.

The practical assignment could take the form of a task requiring learners to solve one, large, relatively routine problem. The task could require learners to create a flowchart to represent the solution to the problem (or, more likely, several flowcharts representing the solution to various sub-problems) and then write the corresponding code. The resulting evidence would be the flowchart(s) and source code. Suitable problems include:

- data entry and query application
- simple computer game
- spellcheck a text file
- encoding and decoding text

The practical assignment could take the form of the creation of two smaller algorithms and programs to solve two problems. In this case, suitable problems include:

- printing prime numbers
- sorting numbers
- guessing a secret number
- checking if a string is a valid password

Whatever problem is set, learners must evidence the minimum product evidence requirements.

The amount of control will vary from context to context. However, in every case, the conditions of assessment must be controlled to some extent. Where the amount of control is low, the amount of authentication should rise. It is not acceptable to produce evidence in lightly controlled conditions with little authentication.

Authentication may take various forms including, but not limited to, oral questioning and plagiarism checks. Some forms of evidence generation (such as video recordings) have intrinsic authentication and would require no further means of verification. Where evidence is not generated under closely controlled conditions (for example, out of class), then a statement of authenticity should be provided by the learner to verify the work as their own, and state any necessary sources and permissions.

Centres are reminded that prior verification of centre-devised assessments would help to ensure that the national standard is being met. Where learners experience a range of assessment methods, this helps them to develop different skills that should be transferable to work or further and higher education.

Formative assessment could be used to assess learners' knowledge at various stages throughout the life of the unit. An ideal time to gauge their knowledge would be at the end of each outcome. This assessment could be delivered through an item bank of selected response questions, providing diagnostic feedback to learners.

National Unit Support Notes (cont)

Unit title: Computer Programming (SCQF level 5)

Opportunities for e-assessment

E-assessment may be appropriate for some assessments in this unit. By e-assessment we mean assessment which is supported by Information and Communication Technology (ICT), such as e-testing or the use of e-portfolios or social software.

Centres which wish to use e-assessment must ensure that the national standard is applied to all learner evidence and that conditions of assessment as specified in the evidence requirements are met, regardless of the mode of gathering evidence. The most up-to-date guidance on the use of e-assessment to support SQA's qualifications is available at **www.sqa.org.uk/e-assessment**.

Opportunities for developing Core and other essential skills

This unit will provide opportunities for learners to develop Core Skills in *Problem Solving* and *Information and Communication Technology* at SCQF Level 5.

The Processing Information component of *Information and Communication Technology* could be developed in Outcome 3, where the learner has to create a computer program.

All three components of *Problem Solving* (Critical Thinking, Planning and Organising, Reviewing and Evaluating) could be developed in Outcome 1, where the learner has to create and refine algorithms to solve problems. Aspects of *Problem Solving* could be developed further in Outcome 3, where the learner has to implement an algorithm into code and then debug it. It is highly likely that once an algorithm is translated into code, problems will appear requiring refinements to the algorithm/approach.

This Unit has the Critical Thinking and Planning and Organising components of Problem Solving embedded in it. This means that when learners achieve the Unit, their Core Skills profile will also be updated to show they have achieved Critical Thinking at SCQF level 5 and Critical Thinking at SCQF level 5.

History of changes to unit

Version	Description of change	Date
02	Core Skills Components Critical Thinking and Planning and Organising at SCQF level 5 embedded.	31/05/18

General information for learners

Unit title: Computer Programming (SCQF level 5)

This section will help you decide whether this is the unit for you by explaining what the unit is about, what you should know or be able to do before you start, what you will need to do during the unit and opportunities for further learning and employment.

The purpose of this unit is to provide you with programming skills and knowledge of the principles of computer programming.

This is a **non-specialist** unit, suitable for you if require an introduction to coding for vocational purposes, and if you wish to appreciate programming for academic or personal reasons. It is particularly suitable for you if you have an interest in STEM. No previous programming experience is required, but it is particular suitable for you if you have achieved HY2C 44 *Computer Programming* at SCQF level 4.

You will gain a range of practical skills and acquire relevant underpinning knowledge. You will learn how to write code in a contemporary high-level language and appreciate programming concepts and techniques, as well as develop your computational thinking skills.

On completion of this unit, you will know how to write programs to solve real-world problems. You will be able to apply programming concepts by implementing them in a programming environment.

You may progress to HY2C 46 *Computer Programming* at SCQF level 6.

Outcomes

On successful completion of the unit, you will be able to:

1 Write algorithms to solve routine problems.
2 Explain programming concepts.
3 Write a computer program.

In Outcome 1, you will cover computational thinking by learning how to create algorithms by breaking down routine problems into logical steps. You will produce flowcharts, which can be based on the algorithms you create. You will learn how to refine algorithms and/or flowcharts to make them more efficient.

In Outcome 2, you will learn about key programming concepts, including variables, naming conventions, data types, arithmetic operators, assignment operators, relational operators, logical operators, sequence, selection, iteration, data structures (arrays) and subroutines.

In Outcome 3, you will learn how to implement those programming concepts in code based on algorithms.

The language may be Python, Basic, JavaScript, Java, C#, C++, and the IDE could be Visual Studio, but any other modern programming language and environment, which make use of the required programming concepts, would be acceptable.

This Unit has the Critical Thinking and Planning and Organising components of Problem Solving embedded in it. This means that when you achieve the Unit, your Core Skills profile will also be updated to show you have achieved Critical Thinking at SCQF level 5 and Critical Thinking at SCQF level 5.