-SQA-SCOTTISH QUALIFICATIONS AUTHORITY

NATIONAL CERTIFICATE MODULE: UNIT SPECIFICATION

GENERAL INFORMATION

-Module Number-	**8112122**	**-Session- 1992-93**
-Superclass-	**CB**	
-Title-	**INTRODUCTION TO COMPUTER PROGRAMMING: PRACTITIONERS**	

-DESCRIPTION-

GENERAL COMPETENCE FOR UNIT: Producing simple computer programs in a high-level procedural language.

OUTCOMES

1. solve computer-related problems using recognised techniques;
2. use a development environment to manage program production;
3. produce tested programming language code.

CREDIT VALUE: 1 NC Credit

ACCESS STATEMENT: There is no formal entry requirement for this module. However, a general familiarity with computers would be an advantage. This may be evidenced by possession of National Certificate Module 8111001 Information Technology 1 or equivalent experience.

For further information contact: Committee and Administration Unit, SQA, Hanover House, 24 Douglas Street, Glasgow G2 7NQ.

This specification is distributed free to all approved centres. Additional copies may be purchased from SQA (Sales and Despatch section) at a cost of £1.50 (minimum order £5).

NATIONAL CERTIFICATE MODULE: UNIT SPECIFICATION

STATEMENT OF STANDARDS

UNIT NUMBER: 8112122

UNIT TITLE: INTRODUCTION TO COMPUTER PROGRAMMING: PRACTITIONERS

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

OUTCOME

1. SOLVE COMPUTER-RELATED PROBLEMS USING RECOGNISED TECHNIQUES

PERFORMANCE CRITERIA

(a) Decomposition of problem is functionally correct.
(b) Application of problem solving techniques is appropriate.
(c) Selection of constructs minimises the complexity and length of the solution.
(d) Testing is carried out to verify the correctness of the derived algorithm.

RANGE STATEMENT

Problem: familiar, non-complex.

Problem Solving techniques: stepwise refinement; pseudocode.

Constructs: sequence; selection; iteration.

Selection Constructs: one-way; two-way; multi-way.

Iteration Constructs: condition-controlled; count-controlled.

Testing: desk checking.

EVIDENCE REQUIREMENTS

The solution detailed in PC (a)-(c) for a simple problem requiring sub-division into a least two sub-problems across all appropriate classes in the Range Statement.

Written evidence that then candidate has successfully tested the solution as detailed in PC(d) across all appropriate classes in the Range Statement.

OUTCOME

2. USE A DEVELOPMENT ENVIRONMENT TO MANAGE PROGRAM PRODUCTION

PERFORMANCE CRITERIA

(a) Selection of commands or options is correct.
(b) Use of development environment is efficient.
(c) Execution of commands or options is consistent with supplied documentation.
(d) Error recovery is accomplished without loss of source code or object code.
(e) Documentation is used efficiently and effectively to resolve errors or uncertainties.

RANGE STATEMENT

Development environment: text editor; language translator.

Program production: creating source code; editing source code; saving source code; loading source code; printing source code; translating source code to object code; executing object code.

EVIDENCE REQUIREMENTS

Performance evidence that the candidate can use a development environment to the standards specified by performance criteria (a) to (e) across all classes in the range statement.

OUTCOME

3. PRODUCE TESTED PROGRAMMING LANGUAGE CODE TO SOLVE DEFINED PROBLEM

PERFORMANCE CRITERIA

(a) Code is consistent with the derived algorithm.
(b) Execution of code is correct.
(c) Code is efficient.
(d) Selection of data types is appropriate to problem's information requirements.
(e) Use of program modules is effective in decomposing solution.
(f) Program modules are functionally cohesive.
(g) Use of internal documentation clarifies the operation of the program.

RANGE STATEMENT

Programming language: procedural language.

Control constructs: sequence, selection; iteration.

Data types: integer; real; string; character.

Arithmetic operations: addition;, subtraction; multiplication; division.

Logical operations: AND; OR; NOT.

Input: keyboard.

Output: screen; printer.

Arrays: 1-dimensional;

Files: sequential.

Problem: familiar; non-complex.

EVIDENCE REQUIREMENTS

A program listing consistent with a derived algorithm.
Oral or written evidence of the candidate's understanding of control constructs, data types and arithmetic and logical operations

----

ASSESSMENT RECORDS

In order to achieve this unit, candidates are required to present sufficient evidence that they have met all the performance criteria for each outcome within the range specified. Details of these requirements are given for each outcome. The assessment instruments used should follow the general guidance offered by the SQA assessment model and an integrative approach to assessment is encouraged. (See references at the end of support notes).

Accurate records should be made of assessment instruments used showing how evidence is generated for each outcome and giving marking schemes and/or checklists, etc. Records of candidates' achievements should be kept. These records will be available for external verification.

SPECIAL NEEDS

In certain cases, modified outcomes and range statements can be proposed for certification. See references at end of Support Notes.

© Copyright SQA 1992

Please note that this publication may be reproduced in whole or in part for educational purposes provided that:

(i) no profit is derived from the reproduction;
(ii) if reproduced in part, the source is acknowledged.

NATIONAL CERTIFICATE MODULE: UNIT SPECIFICATION

SUPPORT NOTES

UNIT NUMBER	8112122
UNIT TITLE	INTRODUCTION TO COMPUTER PROGRAMMING: PRACTITIONERS

SUPPORT NOTES: This part of the unit specification is offered as guidance. None of the sections of the support notes is mandatory.

NOTIONAL DESIGN LENGTH: SQA allocates a notional design length to a unit on the basis of time estimated for achievement of the stated standards by a candidate whose starting point is as described in the access statement. The notional design length for this unit is 40 hours. The use of notional design length for programme design and timetabling is advisory only.

PURPOSE This module is designed to introduce the student to a computer programming language in a problem-solving context.

The module is suitable for a range of applications and can be tailored to suit the needs of the student's occupation or special interest. It is particularly suitable for mainstream computing students.

SQA publishes summaries of NC units for easy reference, publicity purposes, centre handbooks, etc. The summary statement for this unit is as follows:

This module will introduce you to the techniques involved in writing simple computer programs using a high-level procedural language. You will learn how to break down a problem into a services of simpler problems, expressed the solution to these in a programming language and test your solution for correctness.

CONTENT/CONTEXT Corresponding to Outcomes 1-3:

1. Use of stepwise refinement to break down a simple problem into a series of smaller problems which can be solved by means of a computer program.

   Initial problem should be non-complex, familiar to the candidate and sufficient to exercise all constructs.

   Non-rigorous testing of algorithm; checking that a algorithm deals adequately with all sets of conditions.

   Informal approach to program complexity, eg: program is not unnecessarily long or involved. Emphasis on simple and obvious code rather than minimum program size.

2. Use of program editor, compiler or interpreter and such operating system commands as are necessary to manage source and object code files on the system being used.

   Development environment may be integrated or discrete.

   Documentation can include manuals provided by software supplier.

3. Any procedural programming language can be used. Suitable languages include (but are not confined to): Pascal, C, Modula-2, Ada, BASIC , COMAL, FORTRAN, COBOL, PL/I.

   Use of control constructs, data types, arithmetic and logical operations, and input/output as supported by the programming language chosen.

   Simple use of arrays (1-dimensional only) and files (for storage and retrieval of data).

APPROACHES TO GENERATING EVIDENCE A candidate-centred, resource-based learning approach is recommended. During the work of the module candidates should have several opportunities to develop their practical skills and should be assessed at appropriate points. Terminology should be presented in context throughout the module.

Where the candidate is unsuccessful in achieving an outcome, provision should be made for remediation and reassessment.

ASSESSMENT PROCEDURES Centres may use the Instruments of Assessment which are considered by the tutors/trainers to be most appropriate. Examples of instruments of assessment which could be used are as follows:

1-3 One major programming exercise incorporating most of the required language elements. Additional minor exercises should be used to demonstrate language features not covered by the major exercise.

   Below are examples of suitable exercises:

1. **MAJOR EXERCISE** - covers files, arrays, count-controlled and condition controlled loops, multi-way decisions, real, integer and string data, addition, multiplication, division (real), keyboard input, screen and printer output.

   Write a program that reads the names of an unspecified number of students from a file and stores them in an array. Display the name of each student in turn on the screen and ask the user to enter the mark gained in a Computing exam. These marks should also be stored in an array. Write a new file storing the name and mark for each student. Produce a printout showing the name, mark and grade for each student, where the grades are allocated as follows: Distinction: >= 80%, Pass >= 50%, Fail <50%. Calculate the average mark for the class and print this at the foot of the printout.

2. **MINOR EXERCISE** - covers one way and two way conditions, logical operations, keyboard input, screen output.

Applicants for a post as a security guard must be at least 175cm tall if they are male and 165 cm tall if they are female. They must possess at least 5 Standard Grade passes or 10 SQA modules and must not have a criminal record. Write a program which allows the user to enter a candidate's details from the keyboard and displays a message on the screen indicating whether or not the candidate is eligible for the post.

3. **MINOR EXERCISE**- addition, subtraction, multiplication, division (integer), keyboard input, screen output.

   A painter is required to paint a room with one door and one window. Write a program which asks the user to enter the length, breadth and height of the room and the breadth and height of the door and window. All measurements should be entered in centimetres. Calculate the area to be painted. Assuming that one can of paint can cover 5 square metres, calculate and display the number of one litre tins of paint required.

PROGRESSION This module contributes towards general Scottish Vocational Qualifications and occupational Scottish Vocational Qualifications in Information Technology.

RECOGNITION Many SQA NC units are recognised for entry/recruitment purposes. For up-to-date information see the SQA guide 'Recognised and Recommended Groupings'.

REFERENCES

1. Guidelines for Module Writers.
2. SQA's National Standards for Assessment and Verification.
3. For a fuller discussion on assessment issues, please refer to SQA's Guide to Assessment.
4. Procedures for special needs statements are set out in SQA's guide 'Students with Special Needs'.

© Copyright SQA 1992

Please note that this publication may be reproduced in whole or in part for educational purposes provided that:

(i) no profit is derived from the reproduction;
(ii) if reproduced in part, the source is acknowledged.