COMHAIRLE NÁISIÚNTA NA gCÁILÍOCHTAÍ GAIRMOIDEACHAIS

NATIONAL COUNCIL FOR VOCATIONAL AWARDS

Draft Module Descriptor

Computer Programming

Level 2 C20013

September 1995

1 Title **Computer Programming**

2 Code **C20013**

3 Level **2**

4 Value **1**

5 Purpose

This module has been designed to introduce the learner to the concepts of programming and the techniques involved in constructing small programs**.** The module as described is language independent and includes only concepts which are fundamental to an *imperative style* of programming.

6 Preferred Entry Level

Leaving Certificate, or National Vocational Certificate Level 1 or equivalent

7 Special Requirements

None

8 General Aims

This module aims to enable the learner to:

8.1 understand the concepts involved in programming

8.2 be familiar with industry standard programming practices

8.3 learn the principles of software design

8.4 acquire skills to construct reliable software

8.5 test programs effectively

8.6 develop good work practices in the use and care of computing equipment.

9 Units

Unit 1 **Programming Constructs**

Unit 2 **Array Processing**

Unit 3 **Character and String Processing**

Unit 4 **Procedures and Functions**

10 Specific Learning Outcomes

Unit 1 Programming Constructs

The learner should be able to :

10.1.1 define a program

10.1.2 define a programming language

10.1.3 identify the different generations of programming languages

10.1.4 describe the relative advantages and disadvantages of each generation

10.1.5 distinguish between *system software* and *application software*

10.1.6 list examples of *system software*

10.1.7 list examples of *application software*

10.1.8 list the uses of an *editor*

10.1.9 use an *editor* to write simple program text

10.1.10 use standard editor facilities to include:

- find and replace
- block copy
- block insertion
- block deletion

10.1.11 lay out program text legibly

10.1.12 indent program text efficiently

10.1.13 document the program code

10.1.14 distinguish between a *compiler* and an *interpreter*

10.1.15 use a *compiler* to create executable code

10.1.16 execute a program and enter requested data

10.1.17 understand and use the following programming constructs:

- input / output
- cursor and screen handling (position cursor, clear screen, reverse video, ...)
- assignment statement

10.1.18 explain what a *variable* is

10.1.19 distinguish between different simple data types such as *integer, real, character and boolean*

10.1.20 explain the syntax and semantics of the *conditional statement*

10.1.21 solve problems using an *if .. statement*

10.1.22 explain the syntax and semantics of an *iteration(loop) statement*

10.1.23 solve problems which require a loop construct as a solution

10.1.24 list the stages in constructing a loop

- initialise values of variables
- place guard on loop
- develop body of loop
- progress towards termination

10.1.25 write code to read data and process it

10.1.26 explain the role of a *sentinel* (i.e. a value appended to a list to denote the end of the list)

10.1.27 devise an outline schema for processing lists

e.g.

```
read(x)
while x <> sentinel do
begin
{process x }
read(x)
end
```

10.1.28 explain the technique: *top-down development*

10.1.29 use the *top-down* strategy to devise a program to solve a simple problem

10.1.30 explain the need for data validation
(e.g. to check if a month number entered is in the range: 1-12)

10.1.31 define the boolean operators **and**, **or**, **not**

10.1.32 construct compound boolean expressions

10.1.33 evaluate the truth value of compound boolean expressions

10.1.34 solve problems using boolean expressions

10.1.35 design data to test all the programming statements

10.1.36 test written programs with relevant data to check that the outputs are correct.

Unit 2 Array Processing

The learner should be able to:

10.2.1 explain why the data structure “array” (table) is necessary

10.2.2 define a linear (1-D) array

10.2.3 distinguish between the value of an element in an array and its corresponding index value

10.2.4 use arrays of different data types

10.2.5 construct loops to process the elements in an array

10.2.6 write a program to search (linearly) for an element in an array

10.2.7 solve problems whose solution requires the use of an array.

Unit 3 Character and String Processing

The learner should be able to:

10.3.1 explain the A.S.C.I.I. table

10.3.2 explain and list examples of control characters

10.3.3 justify the statement: the A.S.C.I.I. table is an ordinal set of values

10.3.4 explain the role of the extended A.S.C.I.I. set

10.3.5 write simple programs to process the character set (e.g. solve problems such as read a character and print one of "*upper case letter*", "*lower case letter*" or "*not alphabetic character*")

10.3.6 use the extended character set to draw graphical shapes

10.3.7 define a string

10.3.8 list the relational operators for strings

10.3.9 define the length of a string.

10.3.10 distinguish between the length of a string and its' dimension

10.3.11 write programs to process text data in the form of strings.

Unit 4 Procedures and Functions

The learner should be able to:

10.4.1 explain the need for *procedure*s

10.4.2 define a *procedure*

10.4.3 write down the standard syntax for a *procedure* definition in the chosen language

10.4.4 write simple *procedure*s without using parameters

10.4.5 write programs to test a given *procedure*

10.4.6 explain *scope* rules of variables

10.4.7 take a sample program and for each variable declared identify its *scope*

10.4.8 define a *function*

10.4.9 distinguish between user defined *function*s and standard *function*s such as *cos, sqrt*

10.4.10 write expressions which use standard *function*s

10.4.11 write user defined *function*s

10.4.12 test user defined *function*s

10.4.13 explain the difference between a *function* and a *procedure*.

11 Assessment

Summary	Portfolio of Coursework	50%
	Written Examination	50%

11.1 Technique Portfolio of Coursework

Mode School-based with external moderation by the NCVA.

Weighting 50%

Components The portfolio will consist of four assignments, equally weighted:

Assignment 1: based on Unit 1 and must involve the use of the *conditional* statement (12.5%)

Assignment 2: based on Unit 1 and must involve the use of a *loop* construct (12.5%)

Assignment 3: based on Unit 2 and must involve the use of on *array* to store data (12.5%)

Assignment 4: based on Unit 3 and Unit 4 and must involve the use of *strings* and *procedures* (12.5%).

Each of these assignments may be taken at the completion of the relevant unit.

11.2 Technique Written Examination

Mode School-based with external moderation by the NCVA

Weighting 50%

Duration 2 hours

Format 10 questions based on the four units of study.

The written examination is to consist of programming tasks and questions on the particular programming environment used.

The nature and range of questions is at the discretion of the tutor, subject to the approval of the NCVA.

12 Performance Criteria

12.1 Portfolio of Coursework

The performance criteria for each component of the portfolio are detailed on the accompanying Individual Candidate Marking Sheet C20013/MS1

12.2 Written Examination

A detailed marking scheme with the examination paper must be submitted to the NCVA for approval (see note on Approval of Details of School-based Assessment in the Guide to NCVA Level 2 Awards).

13 Grading

Pass	50 - 64%
Merit	65 - 79%
Distinction	80 - 100%

C20013/MS1

Individual Candidate Marking Sheet	**Computer Programming** **(C20013)** **Assignment no: _____**

Candidate: ______________________________ NCVA Examination No:____________

Assignment Brief: ____

Performance Criteria	**Maximum Mark**	**Candidate Mark**
Document program code (10.1.13)	15	
Screen layout (10.1.11)	15	
Code layout (10.1.12) (indentation etc.)	10	
Program correctness (Syntactically)	10	
Program correctness (Semantically)	30	
Test data	20	
Total	**100**	
Weighted total (= total x 0.125)	**12.5%**	

Tutor's Signature: ____________________________________ ***Date:*** ____________

External Examiner's Signature: _________________________ ***Date:*** ____________

C20013/MS2

Individual Candidate Marking Sheet		**Computer Programming** **(C20013)** **Written Examination**

Candidate: _______________________________ NCVA Examination No: ___________

Question	Maximum Mark	Candidate Mark
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		
TOTAL	**500**	
WEIGHTED TOTAL (=TOTAL X 0.1)	**50%**	

***Tutor's Signature:* _____________________________________ *Date:* _____________**

***External Examiner's Signature:* ________________________ *Date:* _____________**

	National Council for Vocational Awards **Rank Order Form** **(Candidate results to be entered in descending order of total marks)**	**Computer Programming** **(C20013)**

Sheet number _______ of _______ Centre:____Roll no:________________

RANK	Candidate Name	NCVA Examinatio n Number	Asg. 1	Asg. 2	Asg. 3	Asg. 4	Written Examination	Total Percentage Mark	Grade Pass=50% Merit=65% Dist.=80%	Moderated Mark/ Grade	NCVA use only
			(12.5%)	(12.5%)	(12.5%)	(12.5%)	(50%)	(100%)			

***Tutor's signature:*____ ***Date:*____________________

***Principal's signature:*____ ***Date:*____________________

***External Examiner's signature:*__________________________________ ***Date:*____________________