8 Unit 8: Introduction to Programming
[AS level, double award, optional, internally assessed]

8.1 ABOUT THIS UNIT

This AS level unit is an optional part of the double award only and is internally assessed

This unit introduces you to a variety of programming languages. It will help you to:

- understand that there are different types of programming language, each having its own features and purposes;
- recognise the differences between programming languages and the similarities between these languages;
- understand the way in which programs are structured using different programming languages.

In this unit you will:

- develop simple programs using a programming language of your choice (these small programs could be joined together as a working solution to meet user requirements for Unit 17: *Program design, production and testing*);
- investigate the structure of programs written in a variety of languages and identify commonalities and differences between them.

This unit will be assessed on your portfolio work only. The mark on that assessment will be your mark for the unit. You will produce evidence for **two** different programming languages:

- annotated program listings for a number of small, simple working programs, *written by you*;
- an annotated program listing for a working program, given to you, and written in a different language to that used in the previous task;
- a report describing your performance in writing the working programme.

WHAT YOU NEED TO LEARN

You need to learn about:

- programming languages;
- program structure.

8.2.1 Programming Languages

You need to understand why there is a need for programming languages to be used.

You need to know about the different *levels* of languages (low level, intermediate level, imperative high level, declarative high level). You also need to know about the different *types of software* for which each level of language is appropriate (low level for operating systems and hardware specific applications, intermediate level for operating systems and network operating systems, high level for a wide variety of non-machine specific applications).

You need to know that there are different languages for different purposes:

- different low level languages for each type of processor;
- languages for embedded systems, e.g. Ada;
- general purpose intermediate level languages, e.g. C;
- languages for knowledge based systems, e.g. Prolog, Lisp;
- object-oriented languages, e.g. C++, SmallTalk, Java, VB.NET;
- languages for mathematical and scientific applications, e.g. Fortran;
- visual languages for event-driven user interfaces, e.g. Visual BASIC, Delphi, Visual C++;
- web scripting languages, e.g. HTML, and web application languages, e.g. JavaScript;
- database query languages, e.g. SQL;
- languages for writing application macros, e.g. VBA;
- languages developed for learning, e.g. Pascal, BASIC.

You need to understand the difference between *imperative* and *declarative* languages and need to know the features of each type of language, i.e. that *imperative* languages are procedural and contain data declarations, function declarations and program constructs, and that *declarative* languages are non-procedural and contain facts and rules.

You also need to know that some features are common to most programming languages, e.g. most are able to deal with constant and variable data and subroutines, and that some languages, though they may be different in structure and features, have a common purpose.

8.2.2 Program Structure

You need to know how programs written in different languages are structured. You need to identify features such as how constants and variables are identified and how subroutines are declared and called.

You need to identify and use the following program constructs:

- sequence;
- selection;
- repetition (count-controlled, test on entry, test on exit).

You need to identify and use the following to store and manipulate data:

- data types – number (integer, floating point), character, Boolean;
- data structures – string, array, record, file;
- operators – arithmetic, relational, logical;
- data manipulation processes – concatenation of strings, file handling, input, output.

You need to identify and use the following to produce a modular program:

- subroutines;
- local and global variables.

You need to develop simple programs using a chosen programming language and understand and comment on simple programs written in at least **one** other programming language.

You need to demonstrate an understanding of the need for good program design techniques to facilitate the ongoing maintenance of programs, e.g. the use of comments, meaningful data names, indentation and modularity.

8.3 ASSESSMENT EVIDENCE GRID

Please see over.

Unit 8: Introduction to programming

What you need to do:

Your evidence needs to include, for **two** different programming languages:

a: [AO1/2/3] annotated program listings for a number of small, simple working programs, *written by you* [**24**];

b: [AO1/2/3] an annotated program listing for a working program, given to you, and written in a different language to that used in **Task a** [**19**];

c: [AO4] a report describing your performance in writing the working program and annotating the given program [**7**].

How you will be assessed:

Task	Assessment Objective	Mark Band 1	Mark Band 2	Mark Band 3	Mark Awarded
a	AO1	You use ICT tools to produce simple working programs with annotated program listings; **[0 1 2 3]**	you use some techniques – at least **one** example each of program constructs, and data manipulation, meaningful data names, correct indentation; **[4 5 6]**	you use a wide range of techniques – constructs, data types, manipulation and modularity. **[7 8]**	/24
	AO2	You demonstrate an understanding of components and functions of programming languages by annotating your program listings to show where you have used data types and/or input/output; **[0 1 2 3]**	you demonstrate an understanding of components and functions of programming languages by annotating your program listings to show where you have used data types, input/output and data manipulation; **[4 5]**	you demonstrate an understanding of components and functions of programming languages by annotating your program listings, fully and clearly, to show where you have used data types, data manipulation and modularity. **[6 7]**	
	AO3	You apply your knowledge of ICT tools and techniques to produce a working program to meet the given task; **[0 1 2 3]**	you produce an effective solution by using appropriate program constructs, data types and data manipulation; **[4 5 6]**	you produce an efficient solution with subroutines used in the program. **[7 8 9]**	

Unit 8: Introduction to programming (continued)					
Task	**Assessment Objective**	**Mark Band 1**	**Mark Band 2**	**Mark Band 3**	**Mark Awarded**
b	**AO1**	You use ICT tools to annotate the given program listing; **[0 1 2]**	you identify some techniques – at least **one** example each of program constructs and data manipulation; **[3 4 5]**	you identify a wide range of techniques – constructs, data types, manipulation and modularity. **[6 7]**	**/19**
	AO2	You demonstrate an understanding of components and functions of programming languages by annotating the program listing to show where data types and/or input/output have been used; **[0 1 2]**	you demonstrate an understanding of components and functions of programming languages by annotating the program listing to show where data types, input/output and data manipulation have been used; **[3 4]**	you demonstrate an understanding of components and functions of programming languages by annotating the program listing, fully and clearly, to show where data types, data manipulation and modularity have been used. **[5 6]**	
	AO3	You apply your knowledge of ICT tools and techniques by correctly annotating the given program listing; **[0 1 2]**	you apply your knowledge of ICT tools and techniques by identifying and annotating program constructs, data types and data manipulation; **[3 4]**	you apply your knowledge of ICT tools and techniques by identifying and annotating program constructs, data types and data manipulation subroutines in the given program. **[5 6]**	
c	**AO4**	You comment on the effectiveness of solutions by describing why the languages used for both programs are suited to the given tasks; you comment on your actions and role in solving the problem; **[0 1 2]**	you identify strengths and weaknesses in the initial solution and refine it in relation to the user's needs by suggesting **one** improvement to each of the programs; you include an analysis of your experiences in order to improve your own performance; **[3 4]**	you identify strengths and weaknesses in the initial solution and refine it in relation to the user's needs by suggesting **one** improvement to each of the programs and giving a valid reason for this suggestion; you include an analysis on your experiences, suggesting how you might approach a similar task in the future. **[5 6 7]**	**/7**
				Total mark awarded:	**/50**

8.4 GUIDANCE FOR TEACHERS

8.4.1 Guidance on Delivery

This unit is intended to give candidates an overall introduction to programming using a variety of programming languages, tools and techniques. Candidates need to, at the end of the unit, write simple programs, such as calculating the average of a given set of numbers stored in a file, performing a selection of tasks chosen from a menu, carrying out a selection of basic transactions, etc.

It is recommended that candidates spend around **50** hours studying this unit, with more time devoted to language **one**, in which a program must be produced, than language **two**, which candidates need to understand but not necessarily use.

8.4.2 Guidance on Assessment

It needs to be stressed that you determine only the *mark* for a candidate's portfolio evidence and not the *grade* which will be determined by OCR.

Regular, early and constructive feedback to candidates on their performance is essential and crucial. Help with planning and structuring their portfolio work in a logical manner throughout the course will lead to better understanding of their work and is likely to achieve higher marks.

Giving candidates deadlines for the completion of various sections of their work, and encouraging them to adhere to them, is also essential if candidates are not going to rush to complete and possibly finish up with marks below their potential.

You need to mark each portfolio according to the assessment objectives and content requirements in the *Assessment Evidence Grid* (Section 8.3).

The information on this *grid* will eventually be transferred onto a *Unit Recording Sheet* to be attached to the front of each candidate's piece of work at the point when the work is submitted for moderation. A *Coursework Administration Pack* will be supplied, containing all relevant *Unit Recording Sheets*. Where marking for this unit has been carried out by more than **one** teacher in a centre, there must be a process of internal standardisation carried out to ensure that there is a consistent application of the criteria as laid down in the *Assessment Evidence Grids*.

Each row in the grid reflects the development of an assessment objective from a task or sub-task in the banner (there may be one or more assessment objectives to any particular task/sub-task).

The maximum mark for each *strand* of work (each row) is shown in the far right-hand column of the grid and this maximum mark is further broken down into a number of mark bands across each row with a range of descriptors.

You use your professional judgement to determine which descriptor in a strand (row) best suits the candidate's work and from the range of marks available within that particular mark band, you circle the mark that best fits the work. You then record this mark in the column headed *Mark*.

You should use the full range of marks available. You must award *full* marks in any strand for work which *fully* meets the criteria. This is work which is the best one could expect from candidates working at AS level.

Only **one** mark per strand/row will be entered. The final mark for the candidate is out of a total of **50** and is found by totalling the marks for each strand of work.

The further guidance below clarifies the criteria in the *Assessment Evidence Grid* and will help you to determine the appropriate mark to be awarded for each strand of work.

Amplification of Criteria			
Task	**AO**	**Mark Band**	**Characteristics of the work one may expect to see at this mark band can be summarised as follows:**
a	**AO1**	**1**	Candidates produce a simple annotated program with **one** form and little use of repetition, e.g. they may use a number of variables when **one** could be used repeatedly; (up to **three** marks, depending on the quality of the program);
		2	candidates use some repetition or selection in their program;
		3	candidates make appropriate use of modules in their program.
	AO2	**1**	Candidates identify the data types used and/or the input and output;
		2	candidates annotate data manipulation, e.g. input, output, file handling, string handling;
		3	candidates annotate subroutines with an indication of what they do, and annotate data manipulation, e.g. input, output, file handling, string handling.
	AO3	**1**	Candidates show evidence that the program works as requested; (up to **three** marks depending on the quality and ease of use of the program);
		2	candidates also use selection and repetition appropriately;
		3	candidates make appropriate use of modules.
b	**AO1**	**1**	Candidates annotate the program listing with appropriate comments; (up to **three** marks depending on the quality of the annotation);
		2	candidates annotate examples of program constructs and data manipulation;
		3	candidates produce extensive annotation, including data types and modules.
	AO2	**1**	Candidates identify and annotate the data types used and the input and output used;
		2	candidates annotate data manipulation, e.g. input, output, file handling, string handling;
		3	candidates annotate subroutines with an indication of what they do, and annotate data manipulation, e.g. input, output, file handling, string handling.

Task	AO	Mark Band	Characteristics of the work one may expect to see at this mark band can be summarised as follows:
b	AO3	1	candidates show, through their annotation, that they understand the functionality of the given program; (up to **two** marks, depending on the quality of the annotation);
		2	candidates identify and annotate where selection and repetition have been used;
		3	candidates annotate all subroutines to describe their functionality.
c	AO4	1	Candidates produce a brief description of why the languages used suit the program written by themselves and the **one** given to them, e.g. program is user-oriented and so a visual language is appropriate, functionality is more important than the user interface and so low-level or non-visual could be used, a more complex program with a high-level language will make it easier to program; candidates make some relevant comments on their approach to the task;
		2	candidates suggest improvements to their code and the given code, e.g. they could have used a different type of loop or a subroutine instead of code in the main program; candidates highlight their strengths and weaknesses in completing the task;
		3	candidates might submit **two** versions (*before* and *after* an improvement) but need to explain why the improvement was made; candidates analyse their strengths and weaknesses and suggest how they might approach a similar task in future; (a further mark is allocated for a high quality answer relating to either of the above).

8.4.3 Resources

Textbooks	French CS	*Computer Science*	Continuum
	Holmes A	*Learning to Use Visual Basic*	Heinemann
	Holzschlag ME	*Using HTML 4*	QUE
	Horton I	*Beginning Visual C++ 6*	WROX Press
	Lhotka R & Hollis B	*Fast Track Visual Basic.NET*	WROX Press
	Prinz P & Kinch-Prinz U	*A Complete Guide to Programming in C++*	Jones and Bartlett
	Summers P	*Visual Basic 6.0 for Windows*	WROX Press
	Wright P	*Beginning Visual Basic 6*	WROX Press
Websites	www.freenetpages.co.uk www.VBcode.com www.wtvl.net/mike/webjr/begcpp.htm		