

Diploma in Business Information Systems – Part 2

Principles of Programming

Introduction

This unit introduces the learner to the fundamental concepts of computer programming, and of modern program development within a typical software development environment..

Aims

1. Learn the knowledge and skills required to write simple routines and programs.
2. Be able to design and implement effective spreadsheet macros and small utilities.
3. Provide a secure foundation upon which more advanced concepts can be built, such as object-orientation and rapid prototyping.
4. Create the practical framework to enable students to achieve useful programming skills in an object orientated programming language such as Visual Basic.

Programme Content and Learning Objectives

On completion of the programme, the student should be able to:

1. Edit, compile and run programs using a modern software development environment (e.g., Visual Basic).
2. Explain the theoretical and practical differences between compiled languages and interpreted languages.
3. Make appropriate use of the primitive data types Integer, Long, Real, String, Boolean and Date/Time (or their equivalents in the chosen language of study).
4. Create variables, use variables in assignment statements and write code to carry out simple arithmetic operations.
5. Use the main control constructs of selection and iteration.
6. Analyse and construct simple logical expressions using the logical operators AND, OR and NOT.
7. Carry out simple processing of arrays.
8. Use built in functions of the chosen language of study.
9. Write and use programmer-defined functions and procedures, including the use of parameters.

Syllabus Content

Using a Modern Software Development Environment

Starting a new program; writing and editing program code; saving program files to disc; backing up program source files; loading program files back into the development environment; compiling programs; executing programs; interpreting error messages; recognising and correcting errors of syntax and errors of logic.

Compilers and Interpreters

Understanding the difference between compilers and interpreters, both in terms of their operation and in their effect on application development, deployment and maintenance.

Data Types

Integers, Long integers, Reals, Strings, Booleans, Date/Time; recognising the need for different data types; choosing the most appropriate data type for a given item of data.

Statements, Expressions and Variables

Declaring variables with specific data types; writing simple assignment statements; constructing simple arithmetic expressions. Distinguishing between expressions and statements.

Control Constructs

If-Then-Else statements; For-Next statements; test-before and test-after loops.

Logic

Truth tables; definitions of AND, OR, NOT; logical equivalence; Boolean expressions; relevance to programming.

Arrays

Declaring arrays; simple array processing – filling arrays with data; calculation of sum, arithmetic mean; finding largest/smallest element; linear search.

Built-in Functions

String related functions (e.g., length of a string, string slicing, case conversion); arithmetic functions (e.g., square root, absolute, sign, mod); type conversion functions (e.g., from Integer to Character and vice versa).

Programmer-defined Functions

Definition of simple functions; parameters and types; return types; defining return values; invocation of programmer defined functions.

Procedures

Definition of simple procedures; parameters and types; calling procedures.

Method of Assessment

By written examination. The pass mark is 40%. Time allowed 3 hours.

The question paper will contain:

Seven questions from which four must be answered. All questions carry 25 marks.

Although the exam is theoretical (candidates will not be asked to write program code) it is unlikely that a candidate will achieve a good grade unless they have completed and understood the practical parts of the module.

Reading List:

Essential Reading

I would strongly recommend the use of Visual Basic for this module, although the final decision must be left to the tutor, who will be aware of the constraints imposed by the particular learning environment in which the module is to be taught. There is no generic textbook that can be described as "Mandatory" for this module, but if the chosen language is Visual Basic, then Chapters 1 to 7 of the following book should prove very useful.

Learning to Program with Visual Basics	Patrick G McKeown	Wiley 1999
--	-------------------	------------

If the tutor decides to use a different language then (s)he must supply an alternative text..

Computer & Software Access

Required

Access to a computer with a suitable modern language and development environment is essential for this unit.

Recommended

Microsoft Visual Basic Version 6.

Guidance Notes for Tutors

It is recommended that the syllabus be followed mostly in the order given below, so that candidates can build on existing knowledge as they progress through the topics.

Using a Modern Software Development Environment

This section should furnish candidates with a set of good "housekeeping" skills. For example, starting each new project in a new folder; using the OS to copy the folder (not the individual files) for backup or transport purposes. It is important to acquire good habits and working procedures from the start, in order to avoid demotivating experiences such as seeming to lose one's work ("I saved my project but now I can't find it"). Also many candidates will want to work on their projects away from the classroom (e.g., at home) so they will need to transport their projects between sites on floppy disc or other removable media. As a project in a modern development environment such as Visual Basic can consist of a number of files, candidates will, in the early parts of the course, need to learn a few simple rules, which ensure that they can transport their work successfully. They should eventually learn more precisely the contents and significance of the various files, but this information is probably best introduced gradually, as their growing understanding equips them better to understand those details.

Although the details of particular kinds of syntax errors are best explained as they arise, as are logical errors, the distinction between syntax errors and logic errors should be introduced near the start. Some of the more common errors that they are likely to make in their first practicals – and the error messages that accompany them – should be explained.

Compilers and Interpreters

It is not expected that candidates will acquire an in-depth technical understanding of how compilers/interpreters work. They should however acquire a high level appreciation of the major difference in their approach to program translation and execution, and the consequences that this has on execution speed, development, deployment and maintenance/upgrades.

Data Types

The concept of data types should be introduced here. Candidates should appreciate the nature of the commonly available primitive data types. They should know the important differences between Integer and Long, for example, and between Integer types and real types. They should understand the limitations of size and accuracy of the different types, and how this is influenced by the number of bytes used to represent the data. However, it is not necessary to understand the details of their binary representation, or to be able to carry out binary arithmetic. The implications regarding what kind of data can be stored using particular data types and what kind of operations can be carried out on particular types of data, should be covered. Most importantly, candidates should be able to choose appropriate data types to represent a variety of data items, such as names, ages, mean values, very large distances, very small quantities and truth values.

Statements, Expressions and Variables

The concept of a variable as a store for a data value should be covered. Candidates should understand not only how to declare variables to be of a specific type, but also why it is important to do this for all variables. The assignment statement should be introduced here and the difference in the way that the computer interprets variable names on either side of the assignment operator should be explained. The common arithmetic operators (addition, subtraction, multiplication, division) are introduced, along with the concept of precedence and the use of parentheses. The fundamental distinction between a statement and an expression can be introduced here.

Control Constructs

This section should deal with the fundamental control constructs of sequence, selection and iteration, and candidates should learn how these constructs are used in the language of study. The importance of choosing the most appropriate construct for any given requirement should be stressed. In particular the use of the For-Next construct only when the number of iterations is known, or can be calculated, in advance, should be pointed out. Also the essential differences between while and until, and between pre-test and post-test, and when to use them, should be covered.

Logic

The purpose of this section is not to learn complex Boolean algebra, but to enable students to construct appropriate Boolean expressions in their selection and conditional iteration constructs. Purely intuitive understanding of the Boolean operators is not sufficient to write successful code. A basic understanding of Boolean concepts makes a huge difference to this aspect of programming. Hence the formal definition – using truth tables (a readily understood medium/tool) of AND, OR, NOT and XOR should be given, and the concept of logical equivalence should be covered. Candidates should also learn how to construct truth tables for Boolean expressions using up to three Boolean variables, and they should be able to use these tables to show that two expressions are (or are not) logically equivalent. Some examples should be given of logically equivalent expressions where one expression is clearly simpler than the other, and the use of this in programming should be made clear. For example, candidates should understand why the expressions `<found = true>` and `<found>` are logically equivalent.

Arrays

The aim is to give candidates a basic understanding of the concept of a collection of data items. Only simple one-dimensional arrays need be covered here, and only simple processing. Binary searches and sorting routines are not required. However the use of two "parallel" arrays should be covered. (For example matching student names with marks, and being able to look up the mark of a student, given a name.)

Built-in Functions

It is anticipated that many of these functions will be assimilated as they are used in appropriate contexts as parts of examples and exercises covering other areas of the syllabus. (For example, converting string representations of numbers into variables of numeric types.) However it may also be useful to issue an assignment in which candidates specifically research and summarise some the functions available in the language.

Programmer-defined Functions

This should be covered after candidates have become familiar with the use of built-in functions. It may be useful to present it as a mechanism for extending the language's repertoire of functions. Although languages vary in how strictly they enforce it, the notion that a function should return a result should be stressed, and candidates should be encouraged to always make their functions return a result – thus clearly distinguishing functions from procedures. Particular care should be taken to ensure that candidates fully understand the concept of parameters and how they make functions flexible and reusable. However the mechanisms of passing parameters by value and by reference can be glossed over at this stage, adopting the language's default mechanism in the examples. It is suggested that simple, one-line functions with just one parameter are used to start with, leading to functions no more complex than half a dozen lines, for example to calculate the sum of the contents of an array. Examples should be chosen to illustrate how a function can be invoked a number of times with different actual parameters.

Procedures

Again it may be useful to present procedures as extending the language's repertoire of available statements (such as Print and Input). See the notes regarding functions – many of the same considerations apply.

CHIEF EXAMINER'S COMMENTS

This module is aimed at business professionals who may need to know the fundamental concepts of programming. It is not intended to create expert programmers, but it is expected to provide a firm grounding for any candidates who decide to take application development further. Hence the emphasis throughout is on keeping the examples and exercises simple and ensuring that the concepts under study are firmly grasped, rather than getting too complex.

The theoretical aspect of the module **MUST** be backed up with plenty of practical exercises involving the candidate in actually writing small programs and executing them on a computer. Purely paper-based coding will not suffice. The examination will be such that a candidate who has understood the theory will gain a marginal pass. In order to attain a higher grade the candidate will have had to complete the practical exercises set by the tutor.