

Certificate in IT Syllabus

2. Software Development

Rationale:

Programming, using many styles and languages, provides solutions to a wide variety of scientific, engineering and business problems. Programming is a core skill that will be used throughout a computer practitioner's career. It is a skill acquired largely by practice and experience. Learning how to program requires a disciplined and structured approach in order to encourage good practice and to assist in the development of easily maintained systems. This module introduces candidates to the fundamental concepts of programming with the emphasis being laid on the whole of the software development process.

Aims:

- To stress the importance of good design, documentation and usability
- To emphasise skills in problem solving and algorithm specification rather than just writing syntactically correct code
- To introduce a systematic approach to algorithm development which will assist in subsequent programming and system design modules
- To introduce candidates to the environment in which software is developed and to the tools that assist in this process

Objectives:

- Distinguish between systems software and application software
- Understand the phases of software development
- Be able to develop and understand algorithms
- Be able to develop code from algorithms in a visual or 3rd generation high level programming language
- Be able to follow 3rd generation high level code and apply modifications to it
- Develop competence in the techniques of systematic problem analysis, program construction and documentation
- Gain an understanding of the basic concepts of good user-interface design
- Be able to test and document programs
- Gain an understanding of the principles of multiple module program construction
- Understand the need for compilers, interpreters, code generators
- Develop a knowledge and understanding of a range of fundamental algorithms

Prior Knowledge Expected:

None

Content:

2a FUNDAMENTAL CONCEPTS OF THE PROGRAMMING PROCESS

Concept of an algorithm

Development and semi-formal specification of algorithms, based on a simplified computer model

Development of code from an algorithm

Understanding of sequential and parallel processing

2b PHASE-SPECIFIC ISSUES OF SOFTWARE DEVELOPMENT

Development tools such as code generators, design modelling or test generators

Development techniques such as modular programming, defensive programming or recursion

Approaches to software build, such as evolutionary prototyping or 4GL development

Objectives and principles of testing and test-case specification

Testing and debugging strategies including dry-running, white-box and black-box

Styles of software documentation, such as for users or support personnel

Content of software documentation such as GUI descriptions or maintenance details

2c INTRODUCTION TO PROGRAMMING CONCEPTS

Types: numeric and non-numeric, elementary and derived, subtypes, and expressions such as assignments, input/output

Control structures: selection and iteration

Subprograms: procedures and functions

Data structures: Arrays (1- and 2-dimensions), linked lists using pointers; implementation of queues, stacks and lists. Concept of data abstraction

Sorting and searching algorithms: comparative effectiveness with respect to computation and storage of scanning versus indexing methods

2d FILES: SEQUENTIAL, INDEX-SEQUENTIAL AND RANDOM ACCESS

Comparative effectiveness of storage and retrieval for applications such as batch processing or on-line query or both

2e INTRODUCTION TO CONCEPT OF USER-INTERFACE DESIGN

User requirements and characteristics of user interfaces; principles and techniques of dialogue control, navigation and selection

2f ROLE AND NEED FOR SYSTEM SOFTWARE

System software and its relation to application software

2g CASE STUDIES IN PROBLEM SOLVING/ALGORITHM ANALYSIS

Primary Texts:

Lesley Anne Robertson, Simple Program Design: A Step-By-Step Approach, Thomson Learning Australia (4th Ed), 2003, ISBN: 0170107043

Not specific to any programming language or indeed to any single design method. Quite simple approach. Uses pseudocode, flowcharts and Nassi-Schneiderman charts (not widely used)

Bell, D. Software Engineering for Students: A Programming Approach, Addison Wesley (4th Ed), 2005, ISBN: 0321261275

Recommended for reading more about software engineering and structured design concepts in programming

Indicative Programming Texts:

Pascal:

Findlay, W., & Watt, D., PASCAL, An Introduction to Methodical Programming, Taylor & Francis (3rd Ed), ISBN: 0273021885

Although old, reliable and at the right level SECTION 4

Java Texts:

Goodrich, Michael, T, Tamassia, R., Data Structures & Algorithms in Java, John Wiley and Sons, 2005, ISBN: 0471738840

Horstmann, C.S., Computer Concepts with Java Essentials, Wiley (3rd Ed), 2003, ISBN: 0471379808

Deitel, Harvey, Java How to Program, Prentice Hall (6th Ed), 2005, ISBN: 0131290142

C/C++ Texts:

Barclay, K., and Gordon, B.J., Problem Solving and Programming, Pearson, 1994, ISBN: 013126673X

Mostly about the C++ language rather than problem solving but about at the right level as a first year book

Savitch, W., Problem solving with C++, Addison-Wesley (5th Ed), 2005, ISBN: 0321269756

Uses object-oriented ideas with the software life cycle. Good chapter on recursion, otherwise probably too advanced for first year programming students

Gotfried, B.S., Schaum's Outline of Programming with C, McGraw-Hill, 1996, ISBN: 0070240353

Visual Basic Text:

McMillan, M., Data Structure & Algorithms Using VisualBasic.NET, Cambridge University Press, 2005, ISBN: 0521547652

Schneider, D.I., An Introduction to Programming Using Visual Basic.net, Prentice Hall (4th Ed), 2003, ISBN: 0130306576

Other Reading:

Other textbooks that describe introductory programming will be appropriate. The computer trade press and the computing/IT supplements of newspapers will help to give candidates both an understanding of the scope of the discipline and also introduce new developments in the field.