SUBJECT TITLE : PROGRAMMING METHODOLOGY
(Common for all Engineering / Technology Programmes)
SUBJECT CODE :
PERIODS/WEEK : 3+2
PERIODS /SEMESTER : 90
CREDITS : 4

Rationale :-

The subject deals with the analysis of problems, design of solution and implementation as programs. Rather than training the students on syntax of any particular programming language, a study of programming should focus on problem-solving - in a rigorous, systematic fashion - through algorithms. Thus, the primary focus is on the design of algorithms for solutions to numeric and non-numeric problems. Study of programming should enable students to program *into* a language rather than *in* a language. A study in this perspective offers the flexibility of choosing any language to code the designed algorithm.

Note:-

This course is designed to have theory examination only. The practical part included in the syllabus is expected to provide basic programming experience on programming methodology. Reasonably enough weightage for internal assessment shall be given on practical part (including practical test and lab record).

Course Objectives:-

On completion of this course, the student should be able to:

- Design a step-by-step solution to realistic problems
- Develop flow chart / algorithm for the solution
- Identify the common constructs shared by all programming languages
- Enable the student to write programs for an algorithm in his/her language of choice.
- The case studies gives the ability to compare the constructs in various programming languages.

TIME SCHEDULE

Module	Topic	Hours
1	Introduction to Problem solving & Programming Theory Practical	 12 8
2	Control Structures Theory Practical	 15 10
3	Arrays Theory Practical	 15 10
4	Module & Sub Programs Theory Practical	 12 8
Total		**90**

OBJECTIVES

1 MODULE - I (Introduction to Problem Solving & Programming)

1.1 Introduction to Programming

1.1.1 Understand what problems are, and in a general way, how they are solved.

1.1.2 List the steps in the general problem solving strategy.

1.1.3 Familiarise the steps in general problem solving strategy through illustration, problem solving, discussion etc.

1.1.4 Study the different types of problems - heuristic and algorithmic

1.1.5 Understand heuristic and algorithmic solutions to problems through illustration of examples, discussion and describe the difference between heuristic and algorithmic solutions.

1.1.6 Define Computer Program.

1.1.7 Understand what programs are, and in a general way, how they are solved and apply the problem-solving strategy in the computer programming problems through illustrations, problem solving, discussions etc

1.2 Basic Programming Concept

1.2.1 Understand basic components of a computer program - input, processing, and output

1.2.2 Familiarise in writing generic program statements for performing input, processing and output operations.

1.3 Data Types

1.3.1 Familiarise different data types - numeric, character, string, logical etc.

1.3.2 Understand and identify the use constants and variables, rules for naming and using variables in a program, differentiate between constants and variables through illustration, problem solving, discussion and prepare notes on them .

1.3.3 Understand what the string and character data types are and how characters and strings are represented in a computer.

1.3.4 Understand what an Integer data type is and how signed and unsigned number are represented in the computer.

1.3.5 Understand what Floating Point data types are and how it is represented in the computer and how they differ from the Integer data type.

1.3.6 Understand logical data type and identify the use of logical data type in a computer program.

1.3.7 Differentiate between character, numeric, and logical data types and prepare a comparison table.

1.3.8 Understand the use basic arithmetic, relational, logical operations in a program .

1.3.9 Identify and use built in functions.

1.3.10 Understand how computers use hierarchy of operations.

1.3.11 Setup and evaluate expressions and equations using variables, constants, operators, and hierarchy of operations.

1.4 List the steps in problem development cycle.

1.4.1 Study the general program development process of problem analysis, program design, coding and documenting, and testing.

1.4.2 Familiarise the program design phase and the principles of top down modular program design.

1.4.3 Understand the importance of designing the program logic with pseudo code and flow chart to design a program.

1.4.4 Familiarise pseudo code, flow chart to design program logic through illustration, problem solving, discussion etc.

1.4.5 Describe the advantages and disadvantages of using pseudo code and flow chart to design a program.

1.4.6 List the tools required for developing a computer program.

1.4.7 Understand the role of programming languages, translators, editors, and debugger in writing programs.

1.4.8 Study different type of programming languages, translators, editors, and debugger through discussion and prepare notes on them and prepare comparison table for each.

1.4.9 Familiarise different high level languages and their features.

1.4.10 Understand Coding and Documenting a program.

1.4.11 Familiarise coding and internal documentation through illustration, problem solving, code designing, lab work etc.

1.4.12 Familiarise the concept of debugging and state need of debugging.

1.4.13 Understand the types of error in coding the program and steps eliminate them through illustration, discussion, lab work etc.

1.4.14 Briefly discuss the testing and documentation of commercial program through illustration and discussion and prepare notes on them.

1.5 Define structured programming.

1.5.1 Familiarise programming conventions and principles of good programming style through illustration, discussion, designing programs.

1.5.2 Design and Implement simple problems involving input, process and output through illustration, problem solving, code designing, discussion, lab work etc.

1.5.3 Use simple built in functions in the program.

1.6 Case Study:- Students may be asked to do following through discussion, team work, assignment, presentation etc on following topics.

1.6.1 Identify and study data types and operators in various programming languages.

1.6.2 Identify and study various translators, debuggers, editors, IDE's used by various programming languages.

2 MODULE - II (Control Structures)

2.1 An Introduction to Selection Structures

2.1.1 Familiarise the different selection structures for making decisions through discussion, illustration, problem solving, output prediction etc.

2.1.2 Understand single (if - then), dual (if - then - else), multiple (if ladder, nested if) alternatives using flow chart and algorithm through problem solving, code designing, lab work etc.

2.1.3 Setup and evaluate expressions and equations using relational and logical operators, and hierarchy of operations.

2.1.4 Identify and apply relational, logical operators and Boolean data type in selection process.

2.1.5 Study the usage of the ASCII coding scheme for associating a number with a character.

2.1.6 Study the usage of the relational operators with arbitrary character strings.

2.1.7 Understand multiple alternatives -if ladder, nested if , CASE like statements using flow chart and algorithm through problem solving, code designing, lab work etc.

2.1.8 Study the usage of different types of selection structures to solve different programming problems.

2.1.9 Understand the use of defensive programming to avoid program crashes, such as division by zero and taking the square root of negative number through observation, illustration, problem solving etc.

2.1.10 Familiarise menu driven programs and its application through observation, discussion etc and develop algorithms.

2.1.11 Discuss the application of selection structures and develop algorithms.

2.2 An introduction to Repetition(iteration) Structures

2.2.1 Familiarise the different repetition structures through discussion, illustration, problem solving, output prediction etc.

2.2.2 Understand pre-test and post test loops using flow chart and algorithm through problem solving, code designing, lab work etc and compare pre-test and post-test loops.

2.2.3 Understand the usage of relational and logical operators in loop conditions.
2.2.4 Study infinite loops and loops that never get executed.
2.2.5 Study to create test conditions to avoid infinite loops and loops that never get executed.
2.2.6 Understand the construction counter controlled loops and study the usage of counter-controlled loops to increment or decrement the counter by any integer value through illustration, problem solving etc .
2.2.7 Familiarise construction 'for' loops through problem solving.
2.2.8 Study the construction of sentinel-controlled loops and data validation techniques.
2.2.9 Discuss the application of repetition structures and develop algorithms.

2.3 More on loops and decisions

2.3.1 Understand the use of loops with selection structures - break, continue through illustration, discussion, problem solving etc and develop programs.
2.3.2 Understand the use of nested loops through illustration of example, problem solving and develop programs.
2.3.3 Combine loops with Case statements to solve problems.
2.3.4 Case Study: Students may be asked to do following through discussion, team work, assignment, presentation etc on following topics:-

1. Study of selection structures in various languages.
2. Study of repetition structures in various languages.

3 MODULE - III (Arrays)

3.1 One Dimensional Arrays

3.1.1 Understand the need and use of arrays through problem solving, illustration, discussion etc and prepare notes on concepts on arrays, memory allocation for arrays and their advantages. Familiarise the declaration and use one dimensional arrays through illustration, problem solving etc.
3.1.2 Familiarise various array operations through illustration, discussion, problem solving and develop programs.
3.1.3 Familiarise the manipulation of parallel arrays.
3.1.4 Use linear search technique to search an array for a specified element.
3.1.5 Use bubble sort/sink sort technique to sort an array into specified order.
3.1.6 Familiarise the character strings as arrays and develop program involving string operations.

3.2 Multidimensional Arrays

3.2.1 Familiarise the declaration and use two dimensional arrays through illustration, problem solving etc and develop programs.
3.2.2 Understand the idea of multidimensional arrays used in many scientific applications.

3.3 Case Study:- Students may be asked to do following through discussion, team work, assignment, presentation etc on following topics.

3.3.1 Definition & indexing of arrays in various languages

4 MODULE - IV (Modules and Subprograms)

4.1 Sub Programs

4.1.1 Understand and Identify the need of sub programs.
4.1.2 Formulate the concept of subprograms through problem solving, discussion etc and prepare a table differentiate procedures and functions.
4.1.3 Identify, analyse and design the problems requiring sub programs.
4.1.4 Understand the use of data flow diagram to indicate the data being transmitted among the programs modules.
4.1.5 Identify the need of arguments and their types through problem solving, illustration, discussion etc and write subprograms using arguments.
4.1.6 Understand the different methods of passing arguments through illustration, discussion etc and prepare notes on differentiating them.
4.1.7 Understand the scope of variables through problem solving and prepare a comparison table.

4.1.8 Familiarise functions commonly built into a programming language and develop programs.

4.1.9 Develop your own functions.

4.1.10 Use recursive functions to solve certain programming problems and illustrate the working of recursive functions.

4.2 Sequential Files

4.2.1 Identify the different types of data files.

4.2.2 Identify records and fields within a data files.

4.2.3 Familiarise to create, write data to, and read data from a sequential file.

4.2.4 Familiarise to delete, modify and insert records in a sequential file.

4.3 Case Study:- Students may be asked to do following through discussion, team work, assignment, presentation etc on following topics.

4.3.1 Study of subprograms (definition structure, parameter passing) in various programming languages.

CONTENT DETAILS

MODULE - I (Introduction to Problem Solving & Programming)

Introduction to programming - General Problem solving-Algorithmic & Heuristic, Steps of problem solving, Problem solving using computer – programming; Basic programming concept – Input, Processing Data, Output Data; Data types - Variables and constants - Integer, Float, Character, Boolean, String etc.- Format of variable names; Type specific operations - arithmetic, logical, relational etc , Built-in functions for operation, Operator Precedence & Associativity, Expressions

The Program Development - Development Cycle; Program Design - Modular Programming, pseudo code, flowchart; Coding, Documenting & Testing - Programming Languages – Types, Examples, Translators – Assemblers, Interpreters & Compilers, Editors, Debuggers; Coding & Documenting - Testing & Errors; Commercial Programs – Testing & Documenting, Testing phases, External Documentation; Structured Programming, Flow chart - Creating flowchart, Control Structures, Programming Conventions.

Case Study :-

Data types and operations in various programming language

Various Assemblers, Interpreters & Compilers

MODULE - II (Control Structures)

Introduction to Selection Structures - Single ,dual, multiple alternative structures, Constructing flow charts with selection structure, ASCII Code, Revisit Logical & Relational Operators, Hierarchy of operators, Using **IF** structure - Simple IF, IF-Else, Nested IF & IF-ladder, Using **Case** - like structures, Applications of Selection Structures, Problem solving with selection structures – examples.

Case Study : Study of selection structures in various languages.

Introduction to Repetition(iteration) Structures - Basics of Loop - Types of loops, Pre-test and post-test loops - **while, repeat-until,** Constructing flowcharts with Loop structure, Counter controlled looping, **FOR** loop structure, Sentinel Controlled looping, Applications of repetition structures, Problem solving with repetition structures – examples,

Case Study :-

Study of repetition structures in various languages.

More on loops and decisions

Combining selection & loop structure, Early exit -- eg. Break, skipping instructions -- eg. Continue, Nested loops, Problem solving involving the above concepts

MODULE - III (Arrays)

Arrays in everyday world, One dimensional Arrays, Array Basics, Working with multiple arrays, Properties and Advantages of arrays, Searching and Sorting, Linear Searching, Bubble / Sink Sort, Strings as Array of characters, String operations, Two Dimensional Arrays, Introduction to two dimensional arrays, Using Nested loops to work with two dimensional arrays, Multi Dimensional Arrays, Problem solving with One, Two, Multi-Dimensional Arrays

Case Study:- Definition & indexing of arrays in various languages

MODULE - IV (Modules and Subprograms)

Need of Sub Programs - Problems requiring modules, Problem analysis, Design, and Data flow diagrams, Arguments and Parameters, Passing data between modules, Assigning types to parameters, Value & Reference parameters, Scope of Variables - Global, Local; Functions - Revisit built-in functions, User defined functions; Recursion - Recursive Process, Simple recursion examples, Problem solving involving

subprograms
Case Study : Study of subprograms (definition structure, parameter passing) in various programming languages.
Introduction to sequential files - File Basics, Creating, Reading & Writing into files.

GUIDELINES FOR THE LABORATORY PRACTICES

Each module of the above course requires two or more practice sessions to be done. ***Experiments similar to the following sample practice experiments may be conducted in C/PYTHON***. PYTHON is a "higher-level" language which is easy to learn because of its simple syntax. PYTHON / C may be used for the practices.

Also, a course assignment (preferably group work) is recommended for the above course. Students may be asked to identify a reasonably large real-time problem (numeric or non-numeric), give step-by-step solution, flow chart (separate for each module, if any), and algorithm. Finally they may also be asked to implement it in any programming language.

Recommended Software's
Operating System: GNU/LINUX (Ubuntu, Debian, Redhat), gcc, python, geany (GUI)

LIST OF SAMPLE EXPERIMENTS FOR PRACTICE (Only indicative)

- Familiarization of compiled and interpreted programs.
  - Demonstrate sample programs for the same
- Familiarization of data types, operators, input & output
  - Program to read & print numerical, character data.
  - Program to read temperature in Celsius scale and print its Fahrenheit equivalent whose equation is F = 95C + 32.
- Problems involving selection process - if , if-else, nested if , elseif ladder, case
  - Program to evaluate quadratic expression.
  - Program to calculate salary from the following data

Basic Pay(BP)	DA	HRA	PF
BP< 5000	30% of BP	500	150
5000BP<10000	25% of BP, Minimum 1800	10%	5%
BP10000	20% of BP, Minimum 3000	15%, Maximum 10000	5%, Maximum 5000

  - Gross Salary = BP+DA+HRA
  - Deduction = PF
  - Net Salary= Gross Salary - Deductions
- Problems involving Repetition(looping) - while, do-while, for
  - Programs to find reverse of a number.
  - Programs to find factorial of a number.
  - Programs to check whether the given number is prime or not.
  - Program to find sum of a series
- Programs requiring one dimensional arrays
  - Implementing programs to do string operations.
  - Implementation of linear search.
  - Implementation of bubble sort.
- Programs involving two dimensional arrays
  - Programs involving matrix operations.

- Modular programming using subprograms
  - check whether the given number is prime or not.
  - Depreciation of an item based on the cost of *item* & *lifetime* is calculated as depreciation = cost of item lifetime and the book value of each year is calculates as book value = cost of item - depreciation. Implement program to calculate the depreciation table as using subprogram to calculate depreciation for each year.

Year	Depreciation	Book Value
0	0	250.00
1	50.00	200.00
2	40.00	160.00
3	32.00	128.00
4	25.60	102.40
5	20.48	81.92

  - Sum of *n* numbers using recursion.
  - Generate Fibonacci series using recursion.

Program to read and write into a file.

Text Book :-

1. Venit, S & Drake E., *Prelude to Programming : Concepts & Design*, 4th Ed. , Addison-Wesley (Pearson)

References :-

2. Sprankle, Maureen., *Problem Solving and Programming Concepts*, 7th Ed., Pearson.
3. Juliff, Peter, *Program Design,* 4th Ed., Prentice-Hall India
4. Tremblay, J & Bunt, R B., *Introduction to Computer Science : An Algorithmic Approach,* 2nd Ed., Tata-McGraw Hill
5. Balaguruswamy, E., *Programming in ANSI C*, 4th Ed., Tata-McGraw Hill
6. http://docs.python.org/tutorial/
7. http://radiantbytes.com/books/pdfs/pylatex.pdf