Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Technology

(An Autonomous Institute affiliated to Savitribai Phule Pune University)

Structure & Syllabus

Multi-Disciplinary Minor Courses

Offered by

Computer Engineering

With Effect from Academic Year 2025-26

Prepared by: - Board of Studies in Computer Engineering

Approved by: - Academic Board, Vishwakarma Institute of Technology, Pune

Chairman – BOS **Chairman – Academic Board**

Vision of the Institution

"To be globally acclaimed Institute in Technical Education and Research for holistic Socio-economic development".

Mission of the Institution

- To ensure that 100% students are employable and employed in Industry, Higher Studies, become Entrepreneurs, Civil / Defense Services / Govt. Jobs and other areas like Sports and Theatre.
- To strengthen Academic Practices in terms of Curriculum, Pedagogy, Assessment and Faculty Competence.
- Promote Research Culture among Students and Faculty through Projects and Consultancy.
- To make students Socially Responsible Citizen.

Vision of the Department

"To be a leader in the world of computing education practising creativity and innovation".

Mission of the Department

- To ensure students' employability by developing aptitude, computing, soft, and entrepreneurial skills
- To enhance academic excellence through effective curriculum blended learning and comprehensive assessment with active participation of industry
- To cultivate research culture resulting in knowledge-base, quality publications, innovative products and patents
- To develop ethical consciousness among students for social and professional maturity to become responsible citizens

Knowledge and Attitude Profile (WK)

WK	WK Statements
WK1	A systematic, theory-based understanding of the natural sciences applicable to the discipline and awareness of relevant social sciences.
WK2	Conceptually-based mathematics, numerical analysis, data analysis, statistics and formal aspects of computer and information science to support detailed analysis and modelling applicable to the discipline.
WK3	A systematic, theory-based formulation of engineering fundamentals required in the engineering discipline.
WK4	Engineering specialist knowledge that provides theoretical frameworks and bodies of knowledge for the accepted practice areas in the engineering discipline; much is at the forefront of the discipline.
WK5	Knowledge, including efficient resource use, environmental impacts, whole-life cost, reuse of resources, net zero carbon, and similar concepts, that supports engineering design and operations in a practice area.
WK6	Knowledge of engineering practice (technology) in the practice areas in the engineering discipline.
WK7	Knowledge of the role of engineering in society and identified issues in engineering practice in the discipline, such as the professional responsibility of an engineer to public safety and sustainable development.
WK8	Engagement with selected knowledge in the current research literature of the discipline, awareness of the power of critical thinking and creative approaches to evaluate emerging issues.
WK9	Ethics, inclusive behavior and conduct. Knowledge of professional ethics, responsibilities, and norms of engineering practice. Awareness of the need for diversity by reason of ethnicity, gender, age, physical ability etc. with mutual understanding and respect, and of inclusive attitudes.

List of Programme Outcomes [PO]

PO	PO Statements
PO1	**Engineering Knowledge:** Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified inWK1 to WK4 respectively to develop to the solution of complex engineering problems.
PO2	**Problem Analysis:** Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable development. (WK1 to WK4)
PO3	**Design/Development of Solutions:** Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required. (WK5)
PO4	**Conduct Investigations of Complex Problems:** Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions. (WK8).
PO5	**Engineering Tool Usage:** Create, select and apply appropriate techniques, resources and modern engineering & IT tools, including prediction and modelling recognizing their limitations to solve complex engineering problems. (WK2 and WK6)

PO6	**The Engineer and The World:** Analyze and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment. (WK1, WK5, and WK7).
PO7	**Ethics:** Apply ethical principles and commit to professional ethics, human values, diversity and inclusion; adhere to national & international laws. (WK9)
PO8	**Individual and Collaborative Team work:** Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.
PO9	**Communication:** Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences
PO10	**Project Management and Finance:** Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.
PO11	**Life-Long Learning:** Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies and iii) critical thinking in the broadest context of technological change.

List of Programme Specific Outcomes [PSO]

List of PSO Statements

PSO1	Demonstrate proficiency in programming with a sound understanding of fundamental computing principles
PSO2	Conceive well-structured design and proficient implementation to address real world challenges using software paradigms, algorithms and technologies
PSO3	Acquire and showcase expertise in emerging fields within computer science, engineering, and technology

Program Educational Objectives (PEOs)

- Demonstrate application of sound engineering foundations to be a committed technology workforce
- Apply mathematical and computing theory knowledge base to provide realistic computer engineering solutions
- Exhibit problem-solving skills and engineering practices to address problems faced by the industry with innovative methods, tools, and techniques
- Develop professional and ethical practices adopting effective guidelines to acquire desired soft skills in the societal and global context
- Aim for continuing education and entrepreneurship in emerging areas of computing

Course Name Nomenclature as per NEP (For FY and SY)

BSC: Basic Science Course	MDOE: Multi Disciplinary Open Elective
ESC: Engineering Science Course	CC: Co-curricular Course
PCC: Program Core Course	HSSM: Humanities Social Science and Management
PEC: Program Elective Course	IKS: Indian Knowledge System
ELC: Experiential Learning Course	FP: Field Project
MD: Multi Disciplinary	INT: Internship

Nomenclature for Teaching and Examination Assessment Scheme AY 2024-25

Sr No.	Category	Head of Teaching/ Assessment	Abbreviation used
1	Teaching	Theory	Th
2	Teaching	Laboratory	Lab
3	Teaching	Tutorial	Tut
4	Teaching	Open Elective	OE
5	Teaching	Multi Disciplinary	MD
6	Teaching	Computer Science	CS
7	Assessment	Laboratory Continuous Assessment	CA
8	Assessment	Mid Semester Assessment	MSA
9	Assessment	End Semester Assessment	ESE
10	Assessment	Home Assignment	HA
11	Assessment	Course Project	CP
12	Assessment	Group Discussion	GD
13	Assessment	PowerPoint Presentation	PPT
14	Assessment	Class Test –1	CT1
15	Assessment	Class Test –2	CT2
16	Assessment	Mid Semester Examination	MSE
17	Assessment	End Semester Examination	ESE
18	Assessment	Written Examination	WRT
19	Assessment	Multiple Choice Questions	MCQ
20	Assessment	Laboratory	LAB

Title: Course Structure **FF No. 653**

Multi-Disciplinary Minor Courses

Offered By: Computer Engineering **A.Y.:** 2025-26

Course No.	Course Code	Course Name	Teaching Scheme (Hrs/Week)			Examination Scheme											Credits
			Th	Lab	Tut	CA	Test -1	MSA	Test -2	ESA						Total	
						LAB (%)	CT1 (%)	MSE (%)	CT2 (%)	HA (%)	LAB (%)	CP (%)	PPT /GD (%)	CVV (%)	ESE (%)		
		MDM: Discrete Structures and Graph Theory	2	0	1												3
		MDM: Object Oriented Programming	2	0	1												3
		MDM: Data Structures and Algorithms	2	0	1												3

Multi-Disciplinary Minor Courses

Offered by

Computer Engineering

AY 2025-26

FF No. : 654

Multidisciplinary Minor

MDMXXX: Discrete Structure and Graph Theory

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week

Course Prerequisites: Basic understanding of school mathematics

Course Objectives:

1. To study basic discrete structures (such as functions, relations, sets, graphs, and trees)
2. To express mathematical properties via the formal language of propositional and predicate logic
3. To develop recurrence relations for a wide variety of combinatorial problems
4. To study advanced combinatorial techniques
5. To study elementary topics in number theory
6. To study elementary concepts in graph theory

Course Relevance: This is a foundational course for Computer Science and Engineering. The discrete structures play an essential role while modeling problems in computer science and engineering. The reasoning with different discrete structures is useful in understanding the underlying computer science problem more concretely. The course also builds problem solving ability.

Syllabus

Theory

Section 1: Topics/Contents

Unit 1: Logic and Proofs **(4 Hours)**

Propositional logic, propositional equivalences, truth tables, predicates and quantifiers, rules of inference, introduction to proofs: direct, contraposition, contradiction, counterexamples, principle of mathematical induction, strong induction. Proving correctness of programs.

Unit 2: Elementary Discrete Structures and Basic Counting **(5 Hours)**

Elementary set theory (sets, set builder notation, cardinality, subsets, some finite and infinite sets, operations on sets), relations (relations and their properties, representing relations, closure of relations, equivalence relations), functions, partial orders, basic counting principles, permutations, combinations, generalized permutations and combinations (with/without repetitions, distinguishable/indistinguishable objects), Binomial coefficients and identities.

Unit 3: Advanced Combinatorial Techniques (5 Hours)
Double counting, combinatorial proof technique, Pigeon-Hole Principle, generalized pigeon-hole principle, some applications from: Ramsey theorem, Mantel's theorem, Turan's theorem, Erdos-Szekeres theorem.
Inclusion Exclusion Principle: Counting with Venn Diagrams, counting Derangements, number of primes up to n, number of onto functions, Euler's phi function.

Section 2: Topics/Contents
Unit 4: Recurrence relations and Generating Functions (5 Hours)
Recurrence relations, modelling using recurrence relations, some examples from: Fibonacci numbers, Catlan numbers, Derangements, Tower of Hanoi, partitions, solution of linear recurrence relations with constant coefficients (homogenous and non-homogenous), generating functions and their application in counting.

Unit 5: Modular Arithmetic (4 Hours)
Divisibility and modular arithmetic, Division Algorithm, primes, greatest common divisor, Euclid's Algorithm, extended Euclid's algorithm, modular inversion, Fundamental Theorem of Arithmetic, Congruence's, Fermat's little theorem, Euler's phi function, Chinese remainder theorem.

Unit 6: Graph Theory (5 Hours)
Graphs, different representations, properties of incidence and adjacency matrices, directed/undirected graphs, connected components, degree of a vertex, paths, cycles in graph, Euler and Hamiltonian tours/graphs, Trees, bipartite graphs (graph with only odd cycles, 2-colorable graphs), Planar graphs, Theorem on bound on number of edges, Graph colorings, matching in bipartite graphs

Syllabus
Tutorials

List of Tutorials
1. Problem solving based on propositional logic
2. Problem solving based on basic set theory
3. Problem solving based on relations and functions
4. Problem solving based on basic counting principles
5. Problem solving based on properties of binomial coefficients
6. Problem solving based on permutations, combinations
7. Problem solving based on combinatorial proof technique
8. Problem solving based on double counting
9. Problem solving based on pigeon-hole principle

10. Problem solving based on inclusion exclusion principle
11. Problem solving based on modular arithmetic
12. Problem solving based on recurrence relations
13. Problem solving based on generating functions
14. Problem solving based on graphs and their properties

Course Outcomes

1. Reason mathematically about elementary discrete structures (such as functions, relations, sets, graphs, and trees) used in computer algorithms and systems
2. Express mathematical properties via the formal language of propositional and predicate logic
3. Develop recurrence relations for a wide variety of combinatorial problems
4. Demonstrate use of advanced combinatorial techniques
5. Describe elementary concepts in modular arithmetic and their applications
6. Exhibit understanding of basic graph theory and its applications

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	3	1		1							2	2		
CO2	3	1				1			2		2	2	1	
CO3	3	2	3								2	2		
CO4	3	3	3	3							2	2		
CO5	3	3	1			1	2		1		2	2	1	
CO6	3	3	3	2		1		1			2	2	1	
Average	**3**	**2.3**	**1.6**	**1**	**0**	**0.5**	**0.3**	**0.1**	**0.5**		**2**	**2**	**0.5**	**0**

Future Courses Mapping: Data structures, Design and analysis of algorithms, theory of computation, artificial intelligence, machine learning.

Job Mapping:

Wherever one wants to model a computer science problem concretely the use of discrete structures is essential. Due to abstract nature of the course, the principles learnt have wide applicability. In any job which requires algorithmic thinking, programming, use of data structures, the knowledge of discrete structures is very helpful.

Text Books:

1. *"Discrete Mathematics and its applications" by Kenneth Rosen (William C Brown Publisher)*
2. *"Applied Combinatorics" by Alan Tucker (Wiley Publishing company)*
3. *"Combinatorics: Topics, techniques, algorithms" by Peter J. Cameron (Cambridge University Press)*
4. *Graph Theory by Reinhard Diestel (Springer Verlag Publishing Company)*
5. *Introduction to Graph Theory by Douglas B. West (Prentice-Hall publishers)*

FF No. : 654

Multidisciplinary Minor

Object Oriented Programming

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week

Course Prerequisites:

1. C programming language
2. Problem solving using programming language

Course Objectives:

1. To introduce the fundamentals of programming using C++ and compare procedural and object-oriented approaches.
2. To develop an understanding of key object-oriented concepts such as encapsulation, inheritance, abstraction, and polymorphism.
3. To enable the implementation of classes, constructors, destructors, and member functions in C++.
4. To explore advanced features of C++ including arrays, strings, pointers, friend functions, and dynamic memory management.
5. To build the ability to use operator overloading, inheritance types, templates, and exception handling for designing flexible systems.
6. To apply file handling techniques and utilize Standard Template Library (STL) components like vectors and iterators for efficient data management.

Course Relevance:

This course introduces fundamental concepts of Object-Oriented Programming (OOP) using the C++ language. It emphasizes problem-solving and practical coding skills using OOP. By the end of the course, students will be able to design and implement simple object-oriented applications.

Syllabus
Theory

Section 1: Topics/Contents

Unit 1: Introduction (4 Hours)

Introduction to Programming using C++; Procedural Vs Object Oriented Programming; Variables; Keywords; Data Types; Input / Output; Control Structures (if, switch, loops); Functions; Scope;

Unit 2: Object-Oriented Programming Fundamentals (4 Hours)

Basic building blocks: Encapsulation, Abstraction, Polymorphism, Inheritance; Principles & Advantages; Classes and Objects; Constructors and Destructors; Member Functions and Access Specifiers

Unit 3: C++ Arrays, Pointers, and Memory Management (6 Hours)

Arrays; Strings; Pointers Basics; Dynamic Memory Management (new, delete); Pointers to Objects; Static Members; Friend Functions

Section 2: Topics/Contents

Unit 4: Inheritance (4 Hours)

Operator Overloading; Inheritance – Basics; Types of Inheritance (Single, Multiple, Multilevel); Constructors in Inheritance

Unit 5: Polymorphism & Exception Handling (4 Hours)

Polymorphism and Virtual Functions; Abstract Classes; Templates – Function and Class Templates; Exception Handling Basics

Unit 6: File Handling & STL (6 Hours)

File Handling – Streams, File Modes; Reading/Writing to Files; Standard Template Library (STL) Overview; Vectors and Iterators

Syllabus
Tutorials

List of Tutorials

1. Write a program to input two integers and display their sum, difference, and product. Demonstrate the use of different data types and keywords
2. Accept a number and check whether it is positive, negative, or zero using if-else.
3. Display the name of the day using a switch statement.
4. Write a program to compute factorial using a user-defined function.
5. Demonstrate the difference between global and local variables.
6. Create a BankAccount class with private balance, public deposit/withdraw methods.

7. Input marks of 5 students using an array and display average.
8. Manipulate strings using character arrays and string class.
9. Display address and value using pointers.
10. Track number of objects using static data members.
11. Create a friend function to access private data of two different classes.
12. Overload + operator to add two complex numbers.
13. Demonstrate Single Inheritance using a Person and Employee class.
14. Implement Multiple Inheritance using Teacher and Researcher classes.
15. Show constructor and destructor call order in multilevel inheritance.
16. Create a function template to swap values of any data type.
17. Create a class template for a simple calculator.
18. Write a program to write and read student data (roll, name, marks) to/from a file.
19. Use a vector to store 10 integers and display them using iterators.
20. Implement a mini student management system using classes, file handling, and STL.

Course Outcomes

Upon completion of this course, students will be able to,

1. distinguish object-oriented programming features from other programming paradigms.
2. Apply classes, objects, methods, and handle object creation, initialization, and destruction to model real-world problems.
3. Understand and implement various features like pointers, dynamic memory, arrays, and friend functions for efficient coding.
4. Identify relationships among objects and implement it using inheritance and polymorphism principles.
5. Demonstrate templates, and exception handling to build flexible and robust applications.
6. Use files for persistent data storage to model real world applications and apply generic programming concepts.

CO-PO Mapping

	Program Outcomes (PO)											PSO		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	3	2	2									3		
CO2	3	3	3	1	1	2	2	2			2	3		3
CO3	3	2	3	1	1		2					3		3
CO4	3	3	3	1							2	3	3	3
CO5	3	3	3	1	1						2	3		3
CO6	3	2	3									3		3
Average	**3**	**2.5**	**2.83**	**1**	**1**	**2**	**2**	**2**			**2**	**3**	**3**	**3**

Future Courses Mapping:
Advanced Data structure, Operating System, Advanced C++ / Java , Object-Oriented Design (OOD), Design Patterns, Embedded Systems (with C++), CUDA Cyber Security.

Job Mapping:
Software Developer / Engineer, Game Developer, System Programmer, Software Architect

Books and E-Resources

For Reference Print /E-Book -

1. The C++ Programming Language, Bjarne Stroustrup, 4th Edition, Addison-Wesley Pearson Education .
2. Herbert Schildt , "C++: The Complete Reference", 4th Edition.
3. E. Balaguruswamy, "Object Oriented Programming Using C++ ", Tata McGraw Hill.
4. An Introduction to Object Oriented Programming (3rd Ed), by Timothy A. Budd, published by Addison-Wesley,2002.

For MOOCs and other learning Resources

1. https://onlinecourses.nptel.ac.in/noc25_cs34/preview
2. https://onlinecourses.nptel.ac.in/noc20_cs07/preview
3. https://www.coursera.org/learn/packt-fundamentals-of-object-oriented-programming-c-b5fxn#modules
4. SOLID principles: implementation and examples in C++ | by Oleksandra Shershen | Medium

Syllabus Template **FF No. : 654**

Multidisciplinary Minor

MDMXXX: Data Structures and Algorithms

Credits: 3 **Teaching Scheme Theory: 2 Hours/Week**

Tutorial: 1 Hour/Week

Course Prerequisites: Introductory course on programming

Course Objectives:

1. To understand basic abstract data types
2. To implement basic data structures such as arrays, linked lists, stacks, queues, trees, graphs.
3. To implement basic searching and sorting algorithms
4. To implement simple graph algorithms
5. To understand how to solve problems using step by step approach with the help of fundamental data structures.
6. To understand role of data structures in algorithm design.

Course Relevance: The course has wide applicability. Whenever one wants to represent data, reason about the data, solve some computational problems, the concept of data structures and algorithms are relevant. In modern day, most of the industries need to work with a large amount of data and efficient data processing, thereby making the principles from data structures and algorithms very relevant.

Syllabus
Theory

Section 1: Topics/Contents

Unit 1: Introduction **(4 Hours)**

Introduction to algorithms, data structures, abstract data types, role of data structures in algorithm design, analysis of algorithms, notion of time complexity, asymptotic notations, figuring out time complexity of simple algorithms.

Unit 2: Basic Searching and sorting algorithms **(5 Hours)**

Array representation, 1-D, 2-D arrays, basic searching algorithms (linear search, binary search, Fibonacci search) implementation and analysis. Basic sorting algorithms (bubble sort, insertion sort, selection sort, quick sort, merge sort) implementation and analysis. Some simple problem solving based on arrays (finding max, array reversal, finding frequencies of elements, finding majority element)

Unit 3: Linked list **(5 Hours)**
Dynamic memory allocation, singly linked list, doubly linked list, circular linked list and generalized linked list, applications of linked lists. Problem solving based on linked lists (reversing singly linked list, Josephus problem using circular linked list, polynomial multiplication using linked lists)

Section 2: Topics/Contents
Unit 4: Stacks and queues **(5 Hours)**
Stack representation and Implementation using arrays and linked lists. Applications of stack in recursion, expression conversions and evaluations.
Queue representation and implementation using array and linked lists, Types of queues. Applications of queues in job scheduling.
Problem solving based on stacks and queues (balanced parenthesis strings, tower of Hanoi, priority queue, implement stack using queues, implement queue using two stacks)

Unit 5: Trees **(4 Hours)**
Tree representation using array and linked lists. Tree traversals: recursive and non-recursive, operations on binary trees, binary search trees (BST).

Unit 6: Graphs **(5 Hours)**
Graph data structure, representation using adjacency matrix and adjacency lists.
Graph traversals, Breadth First Search (BFS), Depth First Search (DFS) and its applications. Finding shortest path in a graph: Dijkstra's algorithm. Minimum Spanning Tree: Prim's and Kruskal's algorithm

Syllabus
Tutorials

List of Tutorials

1. Problem solving based on arrays
2. Finding majority element in the array
3. Implementing Josephus problem
4. Expression evaluation
5. Checking if string is balanced parentheses
6. Implementing tower of Hanoi game using stacks
7. Singly linked list reversal
8. Problem solving based on trees (finding height, mirror image, displaying leaf nodes, level-wise traversal)

9. Some applications of DFS and BFS. Finding connected components in graph, finding a cycle in a graph
10. Implementing Dijkstra's shortest path algorithm
11. Efficient implementation of Prim's and Kruskal's algorithm

Course Outcomes

1. To implement simple searching and sorting algorithms
2. To implement simple data structures like linked lists, stacks, queues, trees
3. To understand graph traversals and their applications
4. To understand simple algorithms for shortest paths in graph, finding minimum spanning trees
5. To solve simple problems using suitable data structures
6. To understand role of data structures in algorithm design.

CO-PO Mapping

	Program Outcomes (PO)											**PSO**		
CO/PO	**PO1**	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PSO1**	**PSO2**	**PSO3**
CO1	2	2	3								2	3		2
CO2	2	3	2								2	3		2
CO3	3	3	3			2	2				2	3	2	2
CO4	3	3	3	3		2	2				2	3	2	2
CO5	3	3	2								2	3		2
CO6	2	3	3								2	3		2
Average	2.5	3.0	2.66	3.0		2.0	2.0				2.0	3.0	2.0	2.0

Future Courses Mapping:

Advanced data structures, Design and analysis of algorithms

Job Mapping:

Data Structures and Algorithms is must necessary part of any core programming job. Without Data structures and Algorithms, it is not possible to be good in Competitive coding. All Industries always look for a strong knowledge in Data structures and Algorithms. Without learning this course, one can't imagine a job in computer/IT related industries and research.

Text Books:

1. E. Horwitz , S. Sahani, Anderson-Freed, " Fundamentals of Data Structures in C", Second Edition, Universities Press.

2. Y. Langsam, M.J. Augenstein, A.M.Tenenbaum, "Data structures using C and C++", Pearson Education, Second Edition.

3.1.J. Tremblay, P. Soresan, “An Introduction to data Structures with applications”, TMHPublication, 2nd Edition.
4. R. G. Dromey, “How to solve it by computer”, Prentice Hall