Darrang College (Autonomous), Tezpur-784001

Syllabus for FYUGP Computer Science (Minor)

Approved by :

Board of Studies meeting held on 26-12-2025 &

&

Academic Council vide Resolution no. 2, dated 29-12-2025

SYLLABUS

Computer Science (Minor)

(FYUGP)

Department of Computer Science

Darrang College (Autonomous)

PROGRAMME STRUCTURE

Semester	Course Name	Credit	L+T+P	Course Type
1	**Introduction to C- Programming**	**4**	**3+0+1**	**Core**
2	**Computer Organization**	**4**	**4+0+0**	**Core**
3	**Object Oriented Programming through C++**	**4**	**3+0+1**	**Core**
4	**Database Management System**	**4**	**3+0+1**	**Core**
5	**Programming in Python**	**4**	**3+0+1**	**Core**
6	**Computer Networks**	**4**	**4+0+0**	**Core**

Introduction to C-Programming (CSC-MN-01014)

1. Learning Outcomes: At the end of the course, students will be able to:
   (a) Understand the basics of C programming like data types and operators
   (b) Understand and write program in C to implement conditions, loops, functions
   (c) Work on arrays, strings and basic file operations

2. Prerequisites: NIL

3. Semester: 1

4. Course type: Compulsory

5. Theory credit: 3

6. Practical credit: 1

7. Number of required hours:
   (a) Theory: 45 hours (45 classes)
   (b) Practical: 30 hours (15 classes)

Detailed Syllabus (Theory)

Unit 1: Basics of C Programming (10 Lectures)

Introduction to programming languages. Comparative study of different types of Programming languages. Translator and types. Structure of a C program. Introduction to Header files. Main function and a simple program execution. Compiling and executing a program. C tokens – keywords, identifiers, constants, operators. Statements and expressions in C. Basic data types in C - integers, float, double, character, void. Size and range of data types. Variables. Storage Class. Constants – integer constant, real constant, character constant, string constant. Declaration and initialization of variables and constants. Assigning values to variables. Operators in C – binary and unary operators. Arithmetic, assignment, logical, comparison, bitwise and conditional operators. Precedence and Associativity of operators. Input and output statements – getchar(), getch(), getche(), gets(), putchar(), puts(), scanf(), printf(), format specifiers. Typecasting.

Unit 2: Control Structures in C (9 Lectures)

Need and use of control structure. Conditional statements – if, else, switch case, nested if-else. Loops – while loop, for loop, do-while loop. Using loop for counting iterations. Using while loop for indefinite iterations. Nested loops. Use of Break and continue statements.

Unit 3: Arrays and Strings (8 Lectures)

Introduction to Arrays. Types of Arrays. Declaration and initialization of arrays. Processing/Accessing array elements. Multidimensional arrays. Introduction to Strings. Declaration and initialization of strings. String input and output in C.

Unit 4: Pointers and Functions (9 Lectures)

Introduction to Pointers. Pointer declaration and initialization. Pointers and addresses. Pointers and Arrays. Basic concept of dynamic memory allocation, malloc(), calloc(). Introduction to user defined functions. Function declaration and definition. Return types of function. Function arguments. Function calling – call by value vs call by reference. Passing an array as argument to a function. Recursive Function.

Unit 5: Introduction to Structures and Unions (4 Lectures)

Basic concept of Structures and Unions in C. Structure declaration and initialization. Union declaration and initialization. Difference between structures and unions.

Unit 6: File Processing and Preprocessor Directives (5 Lectures)

Basic concept of file handling. Comparative study of Different types of files and their uses. Opening and closing file. Reading contents from file, writing to files. Random access to files. File pointers. Error handling in file operations. Preprocessor directives in C - #define, #ifdef, #include, #ifndef, and #endif directives. Using preprocessor directives to define constants and macros.

List of Practical

1. Write a program to take input of two numbers and print their sum, product, difference.
2. Write a program to find the smallest or greatest of three numbers given as input.
3. Write a program to print the sum and product of digits of an integer.
4. Write a program to check whether a input number is palindrome or not.
5. Write a program to take a number representing a month and print the name of the month using switch case.
6. Write a program that calculates the grade of a student based on their marks in a subject using nested if-else statements. Also print the range of marks for each grade using switch case.
7. Write a program to take a number as input and print all the even numbers up to that number using while and for loop.
8. Write a program to ask the user for an input to stop a loop or continue repeating after printing the iteration count using a do-while loop.
9. Write a program to find the maximum, minimum, sum and average of n numbers without using array.

10. Write a program that takes two integers as input and finds their greatest common divisor (GCD) and LCM
11. Write a program that calculates the sum of the first n terms of the Fibonacci sequence, where n is entered by the user, using a for-loop.
12. Write a program that takes an integer as input and checks if it is a prime number.
13. Write a program to create an array with inputs from the user and print the same.
14. Write a program to read an integer and display the binary/octal equivalent of the number.
15. Write a program to take a matrix from the user and print the transpose of the same.
16. Write a program to find the sum, product of 2 matrices.
17. Write a program to take a string of length more than 10 and find the number of vowels in the string. Also print the position of the vowels in the string.
18. Write a program using pointers to copy a string to another string variable without using library function.
19. Write a program to check whether an input string is palindrome or not.
20. Write a program that swaps two numbers using pointers.
21. Write a program to calculate Factorial of a number (i) using recursion, (ii) using iteration
22. Write a program to find sum of n elements entered by the user. To write this program, allocate memory dynamically using malloc() / calloc() functions or new operator.
23. Write a function to accept two arrays as argument and returns their sum as an array.
24. Write a program to implement struct in C. Create a structure of Student with RNo, Name and other credentials with proper datatype and print the same.
25. Write a program to implement union in C. Create a structure of Person with Pid, Name and other credentials with proper datatype and print the same.
26. Write a C program that opens a file for reading and displays the contents of the file in binary mode and text mode.
27. Write a C program that opens a file for reading and displays the contents of the file line by line on the screen.
28. Write a C program that opens a file in append mode and allows the user to add text to the end of the file.

Computer Organization (CSC-MN-02014)

1. Learning Outcomes: At the end of the course, students will be able to:
   (a) Learn about the structure, function and characteristics of computer systems.
   (b) Understand the design of the various functional units and components of computers.
   (c) Identify the elements of modern instructions sets and their impact on processor design.
   (d) Learn about the function of each element of a memory hierarchy.
   (e) Learn about identify and compare different methods for computer I/O.

2. Prerequisites: NIL
3. Semester: 2
4. Course type: Compulsory
5. Theory credit: 4
6. Practical credit: 0
7. Number of required hours: Theory: 60 hours (60 classes)

Detailed Syllabus (Theory)

UNIT 1: Data Representation (10 Lectures)

Number Systems and Conversion, Complements, Fixed Point Representation, Floating Point Representation, Error Detection Codes, Computer Arithmetic - Addition, Subtraction, Other Binary codes: BCD, Excess-3, Gray code

UNIT 2: Introduction and Digital Logic (10 Lectures)

Definitions of Computer Organization and Architecture, History of computer architecture, Basic functional blocks of a computer, Logic Gates, Boolean Algebra, Map Simplifications, Combinational Circuits, Flip-Flops, Sequential Circuits, Decoders, Multiplexers, Registers and Counters

UNIT 3: Register Transfer and Microoperations (7 Lectures)

Introduction to Register Transfer Language, register transfer, Bus and Memory transfers, Arithmetic micro-operation- Binary adder, Binary adder-subtractor, Binary incrementer, Arithmetic circuit, Logic micro-operation, Shift micro-operation, Arithmetic logic shift unit.

UNIT 4: Basic Computer Architecture and Design (10 Lectures)

Instruction Codes: Stored Program Organization, Data path in a CPU, Computer Instructions, Instruction Cycle, Memory-Reference Instructions, I/O Interrupt: Types of Interrupts, Interrupt Cycle, Control Unit: Operations of a control unit, Hardwired control unit, Micro-programmed control unit.

UNIT 5: Central Processing Unit (10 Lectures)

Computer registers, Types of register- general purpose registers, special purpose registers, index registers, General register organization, Stack organization, Computer instructions: Operands, Instruction format- Three-address instructions, Two-address instructions, One-address instructions, Zero- address instructions, Addressing modes, Data Transfer and Manipulation: Data transfer instructions, Data manipulation instructions, Arithmetic instructions, Logical and Bit manipulation instructions, Shift instructions, Program Control: Status bit conditions, Conditional branch instructions, Subroutine call and return, CISC and RISC architectures.

UNIT 5: Memory Organization (7 Lectures)

Semiconductor memories, Memory cells - SRAM and DRAM, Concept of hierarchical memory organization, Cache memory unit - Concept of cache memory, Mapping methods, Organization of a cache memory unit, Cache replacement policies, Write policy, Concept of virtual memory.

UNIT 6: I/O Organization (6 Lectures)

Access of I/O devices, I/O ports, I/O Interface, Modes of Transfer - Program controlled I/O, Interrupt driven I/O, DMA controlled I/O, Priority Interrupts, Handling interrupts.

OBJECT ORIENTED PROGRAMMING THROUGH C++ (CSC-MN-03014)

Learning Outcomes: At the end of the course, students will be able to:

(a) Understand the fundamental principles of Object-Oriented Programming.

(b) Develop the ability to design, implement, test, and debug C++ programs using OOP

(c) Enhance problem-solving and software development skills by applying object-oriented design techniques

2. Prerequisites: NIL

3. Semester: 3

4. Course type: Compulsory

5. Theory credit: 3

6. Practical credit: 1

7. Number of required hours:

(a) Theory: 45 hours (45 classes)

(b) Practical: 30 hours (30 classes)

Detailed Syllabus (Theory)

UNIT 1: Object Oriented Programming Cconcepts (8 Lectures)

Principles of OOP, OOP Paradigm, Basic Concepts of OOP, comparison of procedural programming and OOP, advantages of OOP, Application of OOP.

UNIT 2: Introduction to C++ (9 Lectures)

What is C++, Tokens, C++ data types, Operators in C++, Class, objects, Access specifiers: public, private, protected, Functions - Function Prototyping, Call by reference, Return by reference, Inline function, Function Overloading, Defining Member Functions, Static data members and functions, Array of objects, Friend functions, Constructor and Destructors - Constructors, Parameterized Constructors, Multiple Constructors in a class, Copy constructor, Destructors.

UNIT 3: Inheritance and Polymorphism (9 Lectures)

Inheritance: Types of Inheritance – Single, Multilevel, Multiple, Hierarchal, Hybrid, Defining Derived Classes, Virtual Base Classes, Constructors in derived classes, Pointers: Addresses and pointers, Pointer to object, Compile time and Runtime Polymorphism, this pointer, Pointers to derived classes, Virtual functions, Pure virtual functions.

UNIT 4: Operator overloading and Templates (9 Lectures)

Defining Operator Overloading, Operator Overloading: Overloading unary and binary operators, Overloading with Friend functions, Type conversion, Necessity of friend function,

Templates- Generic functions, generic class, template functions, Function templates, Class templates, Simple applications of templates.

UNIT 5: I/O Operations, Files & Exception Handling (10 Lectures)

Definition of Streams, Stream class hierarchy, Unformatted I/O Operations, Formatted I/O operations, Classes for File Streams, Opening and Closing a File : open() and close() functions, Manipulators of File Pointers : seekg(), seekp(), tellg(), tellp() functions, Sequential Input and output Operations : put (), get(), write(), read() functions, Error handling File Operations : eof(), fail(), bad(), good(), Exception Handling: Introduction, Exception Handling Mechanism, Concept of throw & catch with example Case studies in object-oriented application design.

List of Suggested Practical

1. Write a C++ program to create a class Student with data members: name, roll, and marks. Include a input function to read details and a display function to print details. Create two student objects and display their details.

2.Create a class Rectangle with two data members: length and breadth. Include functions: getData() to input values and area() to return area (length × breadth). In main(), create two Rectangle objects, input values, and display the area of each rectangle.

3.Define a class Number with one data member n. Include a function read() to input the number and a function check() to print whether the number is even or odd. Create an object of the class in main() and test the program.

4. Write a C++ program to create a class Box with data members: length, breadth, and height. The class should include a default constructor (initialize all values to 1), A parameterized constructor (initialize with user-defined values), A function volume() to calculate and return the volume, A destructor that displays a message when an object is destroyed. In main(), create one object using the default constructor, one object using the parameterized constructor. Display the volume of both boxes.

5. Create a class Counter with a static int count, a constructor that increments and displays the count and a destructor that decrements and displays "Object destroyed". In main(), create multiple objects inside a block to show how constructors and destructors are called automatically.

6. Write a C++ program to demonstrate the use of pointers by declaring an integer variable, storing its address in a pointer, and displaying both the value of the variable and its address using the pointer.

7. Write a C++ program to create a function int findMax(int a, int b, int c) that returns the largest of three numbers. In the main() function ask the user to enter three integers, Call the function to find the maximum and Display the result.

8. Write a C++ program to demonstrate function overloading by creating multiple area() functions to calculate the area of a circle, a rectangle, and a triangle based on different parameter lists.

9. Write a C++ program to demonstrate the difference between call by value and call by reference by creating two swap functions—one using value parameters and one using reference parameters—and show how the original values change or remain unchanged after each function call.

10. Write a C++ program to demonstrate single inheritance by creating a base class Person with data members for name and age, and a derived class Student that adds roll number and marks; take input and display all details using objects of the derived class.

11. Write a C++ program to demonstrate multilevel inheritance by creating a base class Vehicle, a derived class Car, and another class ElectricCar derived from Car, and display the details of an electric car object using data from all three classes.

12. Write a C++ program to overload the + operator to add two complex numbers by creating a class Complex with real and imaginary parts, and display the result using an object returned by the overloaded operator.

13. Write a C++ program to demonstrate function overloading by creating multiple area() functions to calculate the area of a circle, a rectangle, and a triangle based on different parameter lists.

14. Write a C++ program to demonstrate both formatted and unformatted I/O operations by using cin.get(), getline(), cout.put(), and cout.write() to read and display different types of input from the user.

15. Write a C++ program to create a text file and write user-entered data into it using ofstream, open(), write(), and put() functions.

16. Write a C++ program to copy the contents of one file into another file using read() and write() functions.

17. Write a C++ program to demonstrate exception handling by creating a function that divides two numbers. If the denominator is zero, throw an exception. In the main() function, use try, catch, and throw to handle the division operation and display an appropriate error message when division by zero occurs.

DATABASE MANAGEMENT SYSTEM (CSC-MN-04014)

1. Learning Outcomes: At the end of the course, students will be able to:
   (a) Understand the fundamental concepts of relational database management systems (RDBMS) and their applications.
   (b) Demonstrate proficiency in designing entity-relationship (ER) models to represent real-world scenarios.
   (c) Utilize SQL to manipulate and query relational databases effectively.
   (d) Develop practical skills through hands-on exercises and projects using a popular RDBMS software.
2. Prerequisites: NIL
3. Semester: 4
4. Course type: Compulsory
5. Theory credit: 3
6. Practical credit: 1
7. Number of required hours:
   (a) Theory: 45 hours (45 classes) (b) Practical: 30 hours (30 classes)

Detailed Syllabus (Theory)

UNIT 1: Introduction and DBMS concepts (8 Lectures)

Importance of database, difference between database approach and File oriented approach, meaning of database system, Database users, Database management systems (DBMS) Meaning, advantages of using a DBMS, Types of DBMS - Hierarchical, network, relational, object-oriented, object-relational, Database system concepts and architecture - Data Models, schemas, and instances, Three schema architecture, Data Independence.

UNIT 2: ER model and RDBMS (9 Lectures)

Overview, ER-Model, Components of ER model - Entities and attributes, Entity types, entity sets, keys and value set, symbols of ER-Diagram Constraints, ER-Diagrams, Relationship: one-to-one, one-to-many, many-to-many, examples of ER-Diagram, Enhanced Entity-Relationship (EER): Super classes, subclasses and inheritance, Specialization and Generalization and their constraints. Introduction to RDBMS Terminologies: characteristics of relation, Domain constraints, Entity, attributes and tuples, relational data integrity, Relational Schemas, Keys (Super key, candidate key, primary key, foreign key), finding keys.

UNIT 3: Normalization, Relational Algebra (9 Lectures)

Definition of Functional dependency, Interference Rules for Functional Dependencies, equivalence of sets of Functional Dependencies, Minimal sets of Functional Dependencies, Introduction to Normalization, Definitions of INF, 2NF, 3NF, BCNF, Relational algebra - Relational algebraic operations - (union, intersection, difference, project, Cartesian product, rename, select, division, join.), Examples of queries in Relational Algebra.

UNIT 4: Introduction to SQL (10 Lectures)

Characteristics of SQL, advantage of SQL, data types, literals, string, numeric, specifying constraints in SQL, Types of SQL commands: DDL, DML, DCL, SQL operations - arithmetic, comparison, logical and set operators, Operator precedence, Examples of basic Queries and sub queries, Aggregate functions - Applications, general rules, examples of Aggregate functions provided by SQL, Table - create, modify, alter, drop. Views and indexes, Insert, update and Delete operations. Joints, unions, Intersections, Minus. Cursors in SQL, Embedded SQL.

UNIT 5: Transaction Processing Concepts (9 Lectures)

Introduction to Transaction Processing, Read & Write Operations, Need for Concurrency Control, Transaction States, Commit Point of a Transaction, Transaction Properties, Schedules and Recoverability - Schedules, Characterizing Schedules, Serializability of Schedules, Testing for Conflict Serializability of a Schedule, Uses of Serializability, Concurrency Control Techniques - The Locking Protocol, Locks. Two Phase Locking (2PL), Deadlock and its Prevention.

Practical

1. Case Study of ER Diagram, Relational Model, Normalization
2. Practical Assignments: Use any appropriate RDBMS to be made available – preferably MySQL)
    a. Exercises using DDL Commands
    b. Exercises using DML Commands
    c. Querying using ANY, ALL, IN, Exists, Not Exists, Union, Intersection Constraints etc.
    d. Querying using Aggregate functions, group by, having and Creation and dropping of views.
3. Create a database and a table named STUDENT with fields (RollNo, Name, Course, Marks), insert at least five records, and display all records.

4.Write SQL queries to update a student's marks, delete a record, and retrieve students scoring more than 80 marks.

5.Create two tables DEPARTMENT and EMPLOYEE, insert records, and perform INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL JOIN.

6.Write queries to demonstrate GROUP BY and HAVING using an EMPLOYEE table (group by department and show average salary).

7.Create a table PRODUCT and perform operations on constraints: PRIMARY KEY, UNIQUE, NOT NULL, DEFAULT, and CHECK.

8.Demonstrate use of ORDER BY, DISTINCT, BETWEEN, IN, LIKE, and aggregation functions (SUM, MAX, MIN, AVG, COUNT).

9. Find SK, CK, no of SK and FK from given FDs.

PROGRAMMING IN PYTHON (CSC-MN-05014)

1. Learning Outcomes: After completing this course, students will know about fundamentals of Python Programming and Problem Solving.
2. Prerequisites: NIL
3. Semester: 5
4. Course type: Compulsory
5. Theory credit: 3
6. Practical credit: 1
7. Number of required hours:
    (a) Theory: 45 hours (45 classes)
    (b) Practical: 30 hours (30 classes)

Detailed Syllabus (Theory)

UNIT 1: Programming and Problem Solving (8 Lectures)

Concept of problem solving, Problem definition, Program design, Debugging, Types of errors in programming, Documentation. Flowchart, decision table, algorithms, Structured programming concepts, Programming methodologies viz. top-down and bottom-up programming.

UNIT 2: Introduction to Python (8 Lectures)

Structure of a Python Program, Elements of Python. Python Interpreter, Using Python as calculator, Python shell, Indentation, Atoms, Identifiers and keywords, Literals, Strings, Operators (Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment Operator, Ternary operator, Bit wise operator, Increment or Decrement operator)

UNIT 3: Creating Python Programs (8 Lectures)

Input and Output Statements, Control statements (Branching, Looping, Conditional Statement, exit function, Difference between break, continue and pass), Defining Functions, default arguments, Errors and Exceptions.

UNIT 4: User Defined Function (4 Lectures)

User defined function, the return statement, Recursive functions, Multiple assignment.

UNIT 5: Strings and Lists **(8 Lectures)**

String as a compound data type, Length, Traversal and the for loop, String slices, String comparison, A find function, Looping and counting, List values, accessing elements, List length, List membership, Lists and for loops, List operations, List deletion. Cloning lists, Nested lists

UNIT 6: Data Structures **(5 Lectures)**

Arrays, list, set, stacks and queues.

UNIT 7: Searching and Sorting **(4 Lectures)**

Linear and Binary Search, Bubble sort, Selection sort and Insertion sort.

List of Suggested Practical

1. Write a function that takes an integer input and calculates the factorial of that number.
2. Write a function that takes a string input and checks if it is a palindrome or not.
3. Write a list function to convert a string into a list, as in list ('abc') gives [a, b, c].
4. Write a program to generate Fibonacci series.
5. Write a program to check a number is Armstrong or not
6. Write a program to check whether the input number is even or odd.
7. Write a program to print all even number between a range (for example between 1 and 100).
8. Write a program to print all prime number between a range (for example between 1 and 100).
9. Write a program to compare three numbers and print the largest one.
10. Write a program to print factors of a given number.
11. Write a method to calculate GCD of two numbers.
12. Write a program to implement linear and binary search on lists.
13. Write a program to sort a list using insertion sort and bubble sort and selection sort.

COMPUTER NETWORKS (CSC-MN-06014)

1. Learning Outcomes: At the end of the course, students will be able to:

   (a) Explain the fundamental concepts of computer networks, including network architectures, protocols, OSI & TCP/IP models, and data transmission techniques.

   (b) configure, analyze, and troubleshoot basic networking devices and protocols, such as IP addressing, routing algorithms, switching, and error-control mechanisms.

   (c) evaluate different network technologies and communication methods to design secure, efficient, and reliable small-scale network solutions.

2. Prerequisites: NIL

3. Semester: 6

4. Course type: Compulsory

5. Theory credit: 4

6. Practical credit: 0

7. Number of required hours:

   (a) Theory: 60 hours (60 classes)

Detailed Syllabus (Theory)

UNIT 1: Basic Concept of Networking (10 Lectures)

Introduction, Components of network, Networking Advantages, Types of networks, Network devices: Bridges, Repeaters, Switches, Routers, Modem, Gateway, Hub, Wifi-routers, Client/Server Method of Connecting Computers, Physical structure and topology, categories of network, ISO OSI model, Data and signals: Analog and digital signal, transmission impairment, performance.

UNIT 2: Physical layer (10 Lectures)

Different transmission Media, Guided Media, Unguided Media, Digital and analogue transmission, Digital to analogue conversion - ASK, FSK, PSK, Analog to digital conversion - PCM, Multiplexing – Frequency Division Multiplexing, Wavelength Division Multiplexing, and Time Division Multiplexing, Switching - Circuit switch network, datagram network.

UNIT 3: Data Link Layer (15 Lectures)

Introduction to DLL, Workings of DLL - Error detection and correction, Linear Block code - Simple parity check code, hamming codes, Cyclic Redundancy Check, Flow and error control

- Stop and wait, Stop-and-Wait ARQ, Go Back-N ARQ and Selective repeat ARQ, PPP - framing, transition phase.

UNIT 4: Network and Transport Layer (15 Lectures)

Network Layer: Introduction to Network layer, IPv4 overview, IPv4 and IPv6 address, IPv4 vs IPv6, Network address translation, Network Management, Network Elements, ICMP, Routing algorithms - Flooding, Distance-vector routing, link state routing. Transport layer: function of Transport layer, process to process delivery, UDP operation, TCP - feature, segment, connection establishment and termination.

UNIT 5: Application layer and LAN's (10 Lectures)

Application layer: DNS, TELNET, E-mail, FTP, WWW (architecture), HTTP, RTP. LAN: Ethernet (Standard, Fast and Gigabit Ethernet), IEEE 802.11 (Architecture, MAC sublayer), Bluetooth (Architecture, layer), Subnetting.