Gyanmanjari Innovative University

Syllabus
Gyanmanjari Diploma Engineering College
Semester-1 (Diploma)

Subject: Practical Oriented Programing in C - DET2CE11201

Type of course: Professional Core

Prerequisite: Programming fundamental, logic and problem-solving skill, mathematical logic.

Rationale:

The core component which drives these tasks is a piece of code for the machine, known as a program. It is essential for the students to learn basic concepts and methodology to develop computer programs. This first and introductory level Computer Programming Course is intended to develop logical thinking skills and programs using a popular structured programming language 'C'. The programming skills thus acquired can be used for developing programs for the scientific, research and business purposes.

Teaching and Examination Scheme:

Teaching Scheme			Credits	Examination Marks		Total Marks
CI	T	P	C	SEE	CCE	
4	0	2	5	100	50	150

Legends: CI-Class Room Instructions; T – Tutorial; P - Practical; C – Credit; SEE - Semester End Evaluation; CCE-Continuous and Comprehensive Evaluation.

Course Content:

<table>
<tr><th>Sr. No</th><th>Course Content</th><th>Hrs.</th><th>% Weightage</th></tr>
<tr><td>1</td><td>
Introduction to C Programming and Basic Constructs

<u>Theory Topics:</u>

Overview of C Programming Language, History and features of C, Structure of a C program, Writing and executing the first C program. C Tokens, Keywords andIdentifiers, Constants, Variables, Data Types and type modifiers, Types of Operators, Input and output functions: printf(), scanf().

<u>Practical:</u>

1. Write a simple C program that displays "Hello, World!" and execute it.
2. Create a C program that demonstrates the use of keywords, identifiers, constants, and variables.
3. Write a C program to perform basic arithmetic operations using different data types and type modifiers.
4. Develop a C program that uses input and output functions (printf(), scanf()) to take user input and display the result.
5. Write a C program to experiment with different data types and type modifiers, showing how each affects variable size and memory usage.

Examination Style:

<table>
<tr><th>Sr. No.</th><th>Evaluation Methods</th><th>SEE</th><th>CCE</th></tr>
<tr><td>1</td><td>Debugging Task:
Correction of logical and syntactical errors in given programs.</td><td>20</td><td>-</td></tr>
<tr><td>2</td><td>Code Optimization:
Optimization of given programs by improving logic, performance, and code structure.</td><td>-</td><td>10</td></tr>
<tr><td colspan="2">Total</td><td>20</td><td>10</td></tr>
</table>
Examination Style:

1. Debugging Task (10 Marks):
• Students will be given four programs, each carrying 05 marks, containing logical or syntactical errors. The
</td><td>18</td><td>20%</td></tr>
</table>

	programs may be practical-based, unsolved, or competitive coding type and will be provided by the faculty. Students will be required to identify errors, correct the code, execute the program, and generate the desired output. This task is designed to assess students' debugging ability, code understanding, logical thinking, problem-solving skills, error identification, and successful execution of the corrected program. **2. Code Optimization (10 Marks)**: • This activity carries 10 marks and focuses on improving the logic, performance, readability, and structure of given programs. Students will be provided with two programs, each carrying 05 marks, and will be required to optimize the code by reducing unnecessary statements, improving time and/or space complexity, minimizing repeated logic, and reducing unnecessary lines of code without changing the actual functionality or output. The questions will be practical in nature and will be framed by the faculty to assess students' ability to write clean, efficient, concise, and optimized code.		
2	**Control Structures, Arrays and Strings in C** ***Theory Topics:*** Conditional Statements - if, if-else, nested if, switch-case. Loops - for, while, do-while, Use of break and continue. Logical and Relational Operators. Arrays - Single-dimensional arrays, multi-dimensional arrays, Strings - Introduction to strings, String handling functions (strlen, strcpy, strcat, etc). ***Practical:*** **6.** Write a C program using if, if-else, and nested if statements to check whether a number is positive, negative, or zero. **7.** Create a C program that uses a switch-case statement to determine the day of the week based on user input (1 to 7). **8.** Develop a C program that uses a for loop to print the first 10 natural numbers. **9.** Create a C program that uses break and continue in a loop to print all numbers from 1 to 20, skipping multiples of 5. **10.** Write a C program to declare and initialize a single-dimensional array, and then print its elements using a loop.	18	20%

11. Write a program to reverse a string and check if it is a palindrome.
12. Develop a C program to input a string from the user and display its length using the strlen() function.
13. Write a C program that uses the strcpy() function to copy one string to another and display the result.
14. Implement a C program that concatenates two strings using the strcat() function and prints the resulting string.

Examination Style:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Program Development and Execution:** Development and execution of module-based C programs with correct logic and output.	20	-
2	**Lab Practice and Output Verification:** Continuous lab-based practice with program execution, output checking, and concept understanding.	-	10
	Total	20	10

Examination Style:

1. **Program Development and Execution Task (20 Marks):**
   - This task carries 20 marks and focuses on the practical implementation of C programs based on the topics covered in Module 2. Students will be required to understand the given problem statement, develop suitable logic, write the program, execute it, and produce the correct output. The questions will be framed and provided by the faculty to assess students' practical understanding, programming logic, syntax usage, code readability, execution ability, and problem-solving skills.

2. **Lab Practice and Output Verification (10 Marks):**
   This activity carries 10 marks and focuses on continuous lab-based programming practice during the module. Students will perform assigned practical exercises related to Module 2 topics and maintain proper program execution records with input, output, and observations. The activity will assess students' regular participation,

<table>
<tr><td></td><td>practical implementation, output verification. understanding of logic, and ability to explain the working of executed programs.</td><td></td><td></td></tr>
<tr><td>3</td><td>Functions, Library Functions and Recursion in C

<u>Theory Topics:</u>

Introduction to Functions - Syntax and declaration, call by value vs call by reference, Standard Library Functions, Math functions (sqrt, pow, etc.), Recursion.

<u>Practical:</u>

15. Write a function to find the factorial of a number using recursion.
16. Implement a program to calculate the greatest common divisor (GCD) of two numbers.
17. Demonstrate the use of library functions like pow() and sqrt().
18. Create a C program to show the difference between call by value and call by reference using a function that swaps two variables.
19. Implement a C program that uses recursion to calculate the factorial of a number.

Examination Style:
<table>
<tr><th>Sr. No.</th><th>Evaluation Methods</th><th>SEE</th><th>CCE</th></tr>
<tr><td>1</td><td>Code Optimization:
Optimization of recursion-based programs with improved logic, performance, and code structure.</td><td>15</td><td>-</td></tr>
<tr><td>2</td><td>Difference and Comparison Questions:
Conceptual comparison of recursion-related topics and programming techniques.</td><td>05</td><td>-</td></tr>
<tr><td>3</td><td>Active Learning Activity (ALA): Recursion Manifesto:
Preparation of a recursion policy document focusing on ethical and efficient coding practices.</td><td>-</td><td>10</td></tr>
<tr><td colspan="2">Total</td><td>20</td><td>10</td></tr>
</table></td><td>18</td><td>20%</td></tr>
</table>

	Examination Style: 1. **Code Optimization (15 Marks):** • This section consists of three questions, each carrying 05 marks, focusing on recursion-based code optimization. Students will be expected to improve the given C programs by improving logic, reducing unnecessary statements, minimizing repeated code, improving readability, and reducing time and/or space complexity without changing the actual functionality or output. The questions will be framed by the faculty to assess students' ability to write clean, efficient, concise, and optimized recursive programs. 2. **Difference and Comparison Questions (05 Marks):** • The difference and comparison question will be asked from the topics covered in Module 3. Students will be required to clearly differentiate or compare two related concepts, terms, or techniques related to recursion and C programming. This task is designed to assess conceptual clarity, understanding of recursion-based logic, and the ability to highlight precise distinctions between related programming concepts. 3. **Active Learning Activity (ALA): Recursion Manifesto: Ethical and Efficient Coding (10 Marks):** • Students will work collaboratively to create a comprehensive recursion policy for a hypothetical software development organization. The policy should cover best practices, acceptable use, efficiency considerations, debugging strategies, and optimization techniques in recursion for C programming. Students must prepare the document with proper structure, examples, and conclusion. The final document must be submitted on the GMIU Web Portal.		
4	***Theory Topics:*** Pointers and Memory Management -Introduction to Pointers: Declaration and initialization, Pointer arithmetic. Dynamic Memory Allocation – malloc(), calloc(), free(). ***Practical:*** **20.** Write a program to swap two numbers using pointers. **21.** Write a C program to declare and initialize a pointer, then use it to access and modify the value of a variable.	18	20%

22. Develop a C program that dynamically allocates memory using malloc() for an array, then frees the memory using free().
23. Write a C program that uses calloc() to allocate memory for an array of integers and initialize the memory to zero.
24. Implement a C program that demonstrates the use of dynamic memory allocation and deallocation by creating and manipulating a dynamically allocated 2D array.

Examination Style:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Code Optimization:** Optimization of pointer and memory-based C programs with improved logic and performance.	15	-
2	**Viva/Oral Examination:** Oral assessment of pointer concepts, memory handling, logic explanation, and code reasoning.	05	-
3	**Active Learning Activity (ALA): The Memory Map:** Exploration of pointers, memory allocation, and memory management in C programming.	-	10
	Total	20	10

Examination Style:

1. **Code Optimization (15 Marks):**
   This section carries 15 marks and focuses on the optimization of pointer and memory-based C programs. Students will be provided with programs requiring improvement in logic, readability, memory usage, and performance. They will be expected to reduce unnecessary statements, remove repeated logic, apply proper pointer handling, improve dynamic memory usage, and minimize time and/or space complexity without changing the actual functionality or output. The questions will be framed by the faculty to assess students' ability to write clean, efficient, concise, and memory-aware C programs.
2. **Viva/Oral Examination Style (05 Marks):**
   - The viva/oral examination carries 05 marks and will be conducted individually to evaluate students' understanding of practical concepts related to pointers

	and memory management. Questions may include pointer declaration, pointer arithmetic, pointer with arrays and functions, dynamic memory allocation, memory deallocation, dangling pointers, memory leaks, logic explanation, code reasoning, and real-time application relevance. This task is designed to assess conceptual clarity, communication skills, logical explanation, and depth of understanding. **3. ALA: The Memory Map: Understanding Pointers in C (10 Marks):** • In this activity, students will explore different aspects of pointers and memory management in C programming. They will study pointer usage, memory allocation techniques, dynamic memory handling, memory optimization, and debugging strategies. Students will prepare a structured report with suitable examples, memory representation, observations, and conclusions. The final report must be submitted on the GMIU Web Portal.		
5	***Theory Topics:*** Structures and File Handling - Introduction to Structures: Definition, declaration, and usage. File Handling -File operations: fopen(), fclose(), fprintf(), fscanf(), fread(), fwrite().Reading from and writing to files. ***Practical:*** **25.** Write a C program to define a structure to store student information (name, age, and grade) and display it using a pointer. **26.** Create a C program to read data into a structure from the user and display it using a function. **27.** Develop a C program that uses fopen() to open a file in write mode, fprintf() to write data to the file, and fclose() to close the file. **28.** Write a C program to read data from a file using fscanf() and display the contents on the screen. **29.** Implement a C program that uses fread() and fwrite() for binary file handling by reading and writing data from a binary file.	18	20%

Examination Style:

Sr. No.	Evaluation Methods	SEE	CCE
1	**Program Tracing and Output Analysis:** Analysis of given C programs to trace logic, file operations, and expected output.	10	-
2	**Real-Time Problem-Solving:** Development of C programs for practical real-time coding scenarios.	10	-
3	**Active Learning Activity: StructBytes:** Development of C applications using structures and file handling for real-world data.	-	10
	Total	20	10

Examination Style:

1. **Program Tracing and Output Analysis (10 Marks):**
   - This task carries 10 marks and focuses on analysing given C programs related to structures and file handling. Students will be required to trace the flow of execution, understand the use of structure variables, file modes, read/write operations, and record processing logic. They must determine the expected output or explain the result of the given program. This task is designed to assess students' logical understanding, program reading ability, output prediction, and conceptual clarity in structured data handling and file operations.

2. **Real-Time Problem-Solving Task (10 Marks):**
   - This task carries 10 marks and focuses on practical coding through real-time problem-solving scenarios. Students will be required to understand the given problem statement, design suitable logic, write the C program, execute it, and produce the required output. The question will be aligned with the prescribed practical scope of the module and may include structured records, file operations, data storage, data retrieval, searching, updating, or processing of records. This task is designed to assess logical thinking, coding ability, practical application, and problem-solving skills.

	3. ALA: StructBytes: File Handling for Real-World Applications (10 Marks): • In this practical activity, students will work with real-world data to develop C applications using structures and file handling. They will focus on data storage, retrieval, updating, and manipulation through structured records and file operations. Students will implement programs that read, write, search, update, and process structured data efficiently using C. The final project must be submitted on the GMIU Web Portal.		

Suggested Specification table with Marks (Theory):100

Distribution of Theory Marks (Revised Bloom's Taxonomy)						
Level	**Remembrance (R)**	**Understanding (U)**	**Application (A)**	**Analyze (N)**	**Evaluate (E)**	**Create (C)**
Weightage %	10%	15%	30%	15%	10%	20%

Note: This specification table shall be treated as a general guideline for students and teachers. The actual distribution of marks in the question paper may vary slightly from above table.

Course Outcome:

After learning the course the students should be able to:	
CO1	Develop logical thinking using C programming.
CO2	Analyze program flow using control statements.
CO3	Apply standard library functions in programs.
CO4	Use pointers for memory access and efficient coding.
CO5	Develop programs using file handling techniques.

Instructional Method:

The course delivery method will depend upon the requirement of content and need of students. The teacher in addition to conventional teaching method by black board, may also use any of tools such as demonstration, role play, Quiz, brainstorming, MOOCs etc.

From the content 10% topics are suggested for flipped mode instruction.

Students will use supplementary resources such as online videos, NPTEL/SWAYAM videos, e-courses, Virtual Laboratory.

The internal evaluation will be done on the basis of Active Learning Assignment.

Practical/Viva examination will be conducted at the end of semester for evaluation of performance of students in laboratory.

Reference Books:

[1] Programming inAnsi-C, by Balagurusamy Tata McGraw-Hill Publishing.
[2] Programming in C, by Kochan Stephen GPublisher:Pearson.
[3] Programming in C, by Kamthane Ashok Publisher:Pearson.
[4] Programming in C, by Gotfried Boyron Publisher:Pearson.
[5] Computer Programming in C, by V. Rajaraman, PHI Learning.

Suggested Assessment Guidelines:

Module 1: Fundamentals of C Programming and Basic Logic Development

Criteria	Description	Marks
Program Understanding	Understand the given problem, input, process, and expected output.	5
Syntax and Structure	Use correct C syntax, data types, operators, and program structure.	5
Debugging	Identify and correct logical or syntactical errors in basic C programs.	5
Output Explanation	Execute the program and explain the generated output.	5
Total		**20**

Module 2: Prompt Engineering and AI-Assisted Content Creation

Criteria	Description	Marks
Problem Analysi	Analyse the problem and prepare suitable programming logic.	5
Control Statement Usage	Apply decision-making and looping statements correctly.	5
Program Execution	Write, compile, and execute the C program successfully.	5
Logic Explanation	Explain the conditions, iterations, and output clearly.	5
Total		**20**

Module 3: Functions, Recursion and Code Optimization

Criteria	Description	Marks
Function and Recursion Logic	Apply functions or recursion to solve programming problems.	5
Code Optimization	Improve code logic, readability, and efficiency.	5
Library Function Usage	Use suitable standard library functions where required.	5
Conceptual Comparison	Compare related programming concepts clearly.	5
Total		**20**

Module 4: Pointers and Dynamic Memory Management

Criteria	Description	Marks
Pointer Usage	Apply pointer declaration, dereferencing, and pointer logic.	5
Dynamic Memory Handling	Use memory allocation and deallocation functions correctly.	5
Pointer-Based Optimization	Optimize pointer-based programs for better efficiency.	5
Viva/Oral Explanation	Explain pointer concepts, memory usage, and program logic.	5
Total		**20**

Module 5: Structures, Unions and File Handling

Criteria	Description	Marks
Structure Handling	Use structures or unions to manage related data records.	5
File Operations	Perform file operations such as read, write, append, and close.	5
Program Tracing	Trace program flow and identify expected output.	5
Real-Time Problem Solving	Develop a C program for a practical file-handling scenario	5
Total		**20**