**Class: B. Tech. 2nd Semester Branch: Computer science and Engineering**

**Course Title: Computer Programming Course Code: ESC-201**

**Course objectives:**

1. To learn fundamental programming concepts and methodologies which are essential to building C programs.
2. To learn basic C keywords and their usage in designing computer programmes.
3. To practice the fundamental programming Concepts in C, such as data Types, control statements, Arrays etc.
4. To introduce students to Pointers and Dynamic Memory Management.
5. To write reusable modules (collections of functions).

**Course Outcomes:**

1. Propose solutions for a given problem using algorithm and flowchart designs.
2. Infer the fundamental programming elements in C language and learn to apply basic control structures in C.
3. Visualize the capabilities of modular programming approach in C and demonstrate it in the real-world scenario.
4. Understand the basic principles of pointers and their association with various data structures.
5. Demonstrate the applications of structures and unions and File handling in C.
6. Showcase the attained knowledge by applying them to solve various real-world problems

**Unit I: Introduction to Programming.** Introduction to components of a computer system: Memory, processor, I/O Devices, storage devices, operating system, Concept of assembler, compiler, interpreter, loader and linker. Representation of Algorithm, Flowchart, Pseudo-code with examples, from algorithms to programs, Structure of C program, writing and executing the first C program, Syntax and logical errors in compilation, object and executable code. Components of C language. Standard I/O in C, Fundamental data types, Variables and memory locations, Storage classes.

**Unit II: Arithmetic expressions, Conditional Branching & Looping.** Operators and expression using numeric and relational operators, mixed operands, type conversion, logical operators, bit operations, assignment operator, operator precedence and associatively. Applying if and switch statements, nesting if and else, use of break and default with switch. Use of while, do while and for loops, multiple loop variables, use of break and continue statements.

**Unit III: Functions, Arrays & Basic Algorithms.** Functions: Introduction, types of functions, functions with array, passing parameters to functions, call by value, call by reference, recursive functions. Arrays: Array notation and representation, manipulating array elements, using multi-dimensional arrays. Character arrays and strings, working with string functions, passing arrays to functions. Basic Algorithms: Searching &Basic Sorting Algorithms (Bubble, Insertion and Selection), Finding roots of equations, Notion of order of complexity.

**Unit IV: Structure, Unions, Pointers and Files.** Structure, union, enumerated data types, Array of structures, passing structures to functions. Pointers: Introduction, declaration, and applications of pointers, Introduction to dynamic memory allocation (malloc, calloc, realloc, free), Use of pointers in self-referential structures, notion of linked list (no implementation). File handling: File I/O functions, Sequential and random Access files. Standard C preprocessors, defining and calling macros, command-line arguments. Multi file programming.

**BOOKS RECOMMENDED:**

1. Yashavant Kanetkar, “Let Us C”, BPB Publications, New Delhi, 13th Edition, 2012.

2. Oualline Steve, “Practical C Programming”, O’Reilly Media, 3rd Edition, 1997.

3. King K N, “C Programming: A Modern Approach”, Atlantic Publishers, 2nd Edition, 2015.

4. Kochan Stephen G, “Programming in C – A Complete Introduction to the C Programming Language”, Sam’s Publishers, 3rd Edition, 2004.

5. Linden Peter V, “Expert C Programming: Deep C Secrets”, Pearson India, 1st Edition, 1994.

The practicals are based on the topics covered under Theory Syllabus. The Students are required to perform at least 25 Programs.

|  |  |
| --- | --- |
| **ESC201.1** | 1. Write a C program to check whether a number is even or odd using ternary operator. 2. Write a C program to perform the addition of two numbers without using + operator. 3. Write a C program to evaluate the arithmetic expression ((a + b / c \* d - e) \* (f - g)). Read the values a, b, c, d, e, f, g from the standard input device. 4. Write a C program to find the sum of individual digits of a 3-digit number. 5. Write a C program to read the values of x and y and print the results of the following expressions in one line: 6. (x + y) / (x - y) 7. ii. (x + y)(x - y) |
| **ESC201.2** | 1. Write a C program to find the sum of individual digits of a positive integer. 2. A Fibonacci sequence is defined as follows: the first and second terms in the sequence are 0 and 1. Subsequent terms are found by adding the preceding two terms in the sequence. Write a C program to generate the first n terms of the sequence. 3. Write a C program to generate all the prime numbers between 1 and n, where n is a value supplied by the user. 4. A character is entered through keyboard. Write a C program to determine whether the character entered is a capital letter, a small case letter, a digit or a special symbol using if-else and switch case. The following table shows the range of ASCII values for various characters. Characters ASCII values A – Z 65 – 90 a – z 97 – 122 0 – 9 48 – 57 Special symbols 0 – 47, 58 – 64, 91 – 96, 123 – 127 5. If cost price and selling price of an item is input through the keyboard, write a program to determine whether the seller has made profit or incurred loss. Write a C program to determine how much profit or loss incurred in percentage. |
| **ESC201.3** | 1. Write a C program, which takes two integer operands and one operator from the user, performs the operation and then prints the result. (Consider the operators +, -, \*, /, % and use switch statement). 2. b. Write a C program to calculate the following sum: sum = 1 – x 2 /2! + x4 /4! – x 6 /6! +x8 /8! – x 10/10! 3. Write a C program to find the roots of a quadratic equation. 4. Write a C program to check whether a given 3 digit number is Armstrong number or not. 5. Write a C program to print the numbers in triangular form   1  1 2  1 2 3  1 2 3 4   1. Write a C program to find the second largest integer in a list of integers. 2. Write a C program to perform the following:   Addition of two matrices   1. Multiplication of two matrices. 2. Write a C program to count and display positive, negative, odd and even numbers in an array. 3. Write a C program to merge two sorted arrays into another array in a sorted order. 4. Write a C program to find the frequency of a particular number in a list of integers. 5. Write a C program that uses functions to perform the following operations: 6. To insert a sub string into a given main string from a given position. 7. To delete n characters from a given position in a given string. b. Write a C program to determine if the given string is a palindrome or not. 8. Write a C program to find a string within a sentence and replace it with another string. 9. Write a C program that reads a line of text and counts all occurrence of a particular word. 10. Write a C program that displays the position or index in the string S where the string T begins, or 1 if S 11. Write C programs that use both recursive and non-recursive functions 12. To find the factorial of a given integer. 13. To find the greatest common divisor of two given integers. 14. Write C programs that use both recursive and non-recursive functions 15. To print Fibonacci series. 16. To solve towers of Hanoi problem. 17. Write a C program to print the transpose of a given matrix using function. 18. Write a C program that uses a function to reverse a given string |
| **ESC201.4** | 1. Write a C program to concatenate two strings using pointers. 2. Write a C program to find the length of string using pointers. 3. Write a C program to compare two strings using pointers. 4. Write a C program to copy a string from source to destination using pointers. 5. Write a C program to reverse a string using pointers 6. Write a C program that uses functions to perform the following operations: 7. Reading a complex number 8. Writing a complex number 9. Addition and subtraction of two complex numbers 10. Multiplication of two complex numbers. Note: represent complex number using a structure. 11. Write a C program to compute the monthly pay of 100 employees using each employee’s name, basic pay. The DA is computed as 52% of the basic pay. Gross-salary (basic pay + DA). Print the employees name and gross salary. 12. Create a Book structure containing book\_id, title, author name and price. Write a C program to pass a structure as a function argument and print the book details. 13. Create a union containing 6 strings: name, home address, hostel address, city, state and zip. Write a C program to display your present address. 14. Write a C program to define a structure named DOB, which contains name, day, month and year. Using the concept of nested structures display your name and date of birth. |

**Web References:**

1. [www.sanfoundry.com/c-programming-examples](http://www.sanfoundry.com/c-programming-examples)

2. [www.geeksforgeeks.org/c](http://www.geeksforgeeks.org/c)

3. [www.cprogramming.com/tutorial/c](http://www.cprogramming.com/tutorial/c)

4. [www.cs.princeton.edu](http://www.cs.princeton.edu)

Lab:

#include <stdio.h>

// a. Check whether a number is even or odd using a ternary operator

void checkEvenOdd() {

int num;

printf("\n[a] Check Even or Odd\n");

printf("Enter a number: ");

scanf("%d", &num);

(num % 2 == 0) ? printf("Even\n") : printf("Odd\n");

}

// b. Perform addition of two numbers without using the + operator

void addWithoutPlus() {

int a, b;

printf("\n[b] Addition without using + operator\n");

printf("Enter two numbers: ");

scanf("%d%d", &a, &b);

while (b != 0) {

int carry = a & b;

a = a ^ b;

b = carry << 1;

}

printf("Sum = %d\n", a);

}

// c. Evaluate the arithmetic expression ((a + b / c \* d - e) \* (f - g))

void evaluateExpression() {

float a, b, c, d, e, f, g, result;

printf("\n[c] Evaluate Arithmetic Expression\n");

printf("Enter values for a, b, c, d, e, f, g: ");

scanf("%f %f %f %f %f %f %f", &a, &b, &c, &d, &e, &f, &g);

result = ((a + (b / c) \* d - e) \* (f - g));

printf("Result = %.2f\n", result);

}

// d. Find the sum of individual digits of a 3-digit number

void sumOfDigits() {

int num, sum = 0;

printf("\n[d] Sum of Individual Digits\n");

printf("Enter a 3-digit number: ");

scanf("%d", &num);

sum += num % 10; // Get the last digit

num /= 10; // Remove the last digit

sum += num % 10; // Get the middle digit

num /= 10; // Remove the middle digit

sum += num; // Add the first digit

printf("Sum of digits = %d\n", sum);

}

// e. Evaluate expressions (x + y) / (x - y) and (x + y)(x - y)

void evaluateExpressions() {

float x, y;

printf("\n[e] Evaluate Expressions\n");

printf("Enter values for x and y: ");

scanf("%f %f", &x, &y);

if (x - y != 0) { // Check to avoid division by zero

float result1 = (x + y) / (x - y);

float result2 = (x + y) \* (x - y);

printf("Results: (x + y) / (x - y) = %.2f, (x +