**Course Title: Programming Fundamentals and 'C' Programming**
**Course No.: CSIT.115**
**Nature of the Course: Theory+Lab**
**Level: B.Sc. CSIT**
**Year: First**
**Semester: First**

**Credit: 3**
**Number of hours per week: 3**
**Total hours: 48**

## 1. Course Introduction

The course intends to enable the students to be acquainted with the basic concepts of programming methodology, 'C' Programming language.

## 2. Objectives

At the end of this course the students should be able:

- To develop a programming logic.
- To teach basic principles of programming.
- To develop skills for writing programs using 'C'.

## 3. Specific Objectives and Contents

| Specific Objectives | Contents |
|---|---|
| • Define algorithm, use of algorithms<br>• Describe different notations of algorithms<br>• State standard notations and common functions<br>• Classify different Pseudo-code Conventions<br>• Develop fundamental algorithms<br>• Write different algorithms for different problems<br>• Differentiate different programming approaches and their benefits.<br>• Understand the basic structure of C program<br>• Understand different types of data types and qualifiers in terms of memory requirement and range.<br>• Write various programs using different data types, qualifiers. | **Unit I: Introduction To Algorithms and C (8 Hrs)**<br>**Fundamentals of algorithms:** Notion of an algorithm. Pseudo-code conventions like assignment statements and basic control structures.<br>**Algorithmic problems** : Develop fundamental algorithms for (i)Exchange the values of two variables with and without temporary variable, (ii) Counting positive numbers from a set of integers, (iii) Summation of set of numbers, (iv) Reversing the digits of an integer, (v) Find smallest positive divisor of an integer other then 1, (vi) Find G.C.D. and L.C.M. of two as well as three positive integers (vii) Generating prime numbers.<br>**Different approaches in programming:** Procedural approach, Object Oriented approach, Event Driven approach.<br>**Structure of C:** Header and body, Use of comments, Compilation of program.<br>**Data Concepts:** Variables, Constants, data types like: int, float char, double and void.<br>Qualifiers: Short and ling size qualifiers, signed and unsigned qualifiers. Declaring variables. Scope of the variables according to block. Hierarchy of data types. |
| | |
| • Write various 'C' programs to perform various types of operations on the data values which are to be processed.<br>• Input various types of data and obtain the output in a desired form<br>• Alter the sequence of the execution of the program<br>• Set up loops to repeat a set of | **Unit II : Basic of C (4 Hrs)**<br>**Types of operators:** Arithmetic, Relational, Logical, Compound Assignment, Increment and decrement, Conditional or ternary, Bitwise and Comma operators, Precedence and order of evaluation. Statements and Expressions.<br>**Type Conversions :** Automatic and Explicit type conversion<br>**Data Input and Output function :** Formatted I/O: printf(), scanf(), Character I/O format : getch(), gerche(), getchar(), getc(), gets(), |

| | |
|---|---|
| statements, desired number of times<br>• transfer control to different statements in the program | putchar(), putc(), puts()<br>**Iterations:** Control statements for decision making: (i) Branching: if statement, else.. If statement, switch statement (ii) Looping: while loop, do… while, for loop. (iii) Jump statements: break, continue and goto. |
| | |
| • Understand what arrays are<br>• What is the need for arrays<br>• How arrays can be used in C Language<br>• Declare and use one dimensional and two dimensional arrays<br>• Understand the need for character and string variables<br>• Declare and use character and string variables<br>• Use functions to handle character and string data<br>• Understand the Purpose of Sorting<br>• Understand the different methods of Sorting.<br>• Identify the advantages of different algorithms of Sorting<br>• Be able to write programs in C to implement the algorithms for Sorting<br>• Explain what is meant by Efficiency of an algorithm<br>• Compare algorithms for Efficiency | **Unit III : Arrays, Strings and Sorting Techniques (8 Hrs)**<br>**Arrays :** (One and multidimensional), declaring array variables, initialization of arrays, accessing array elements.<br>**Strings:** Declaring and initializing String variables. Character and string handling functions.<br>**Sorting Algorithms :** Bubble, Selection, Insertion and Merge sort, Efficiency of algorithms, Implement using C. |
| | |
| • Understand what Functions are and why are they needed.<br>• Be able to define a Function in terms of its arguments and return values<br>• Understand when and how to use Functions<br>• Understand what are Macros and why they are needed<br>• Explain how Macros are different from functions?<br>• Understand what is Recursion?<br>• Explain the Advantages of Recursion<br>• Write programs for some standard situations for recursive functions such as Fibonacci Sequence and Towers of Hanoi<br>• Be able to understand situations where recursion is needed<br>• Understand the concept of a storage class<br>• Understand the different storage classes<br>• Understand the concept of scope, | **Unit IV: Functions, Storage Classes and Recursion (8 Hrs)**<br>**Functions:** Global and local variables, Function definition, return statement, Calling a function by value, Macros in C, Different between functions and macros.<br>**Storage classes :** Automatic variables, External variables, Static variables, Register variables.<br>**Recursion:** Definition, Recursion function algorithms for factorial, Fibonacci sequence, Tower of Hanoi. Implement using C |

| | |
|---|---|
| visibility and longevity of a variable<br>• Understand which storage class should be used under what circumstances<br>• Learn the advantages and disadvantages of each storage class | |
| | |
| • Understand what are structures and why they are needed<br>• Be able to define a structure<br>• Be able to read and assign values to elements in a structure<br>• Be able to understand the relationship between arrays and structures<br>• Be able to define structures within structures<br>• Be able to understand the relationship between structures and functions<br>• Be able to understand what are unions<br>• Write programs involving the use of structures | **Unit V: Structure and Union (4 Hrs)**<br>**Structure:** Declaration of structure, reading and assignment of structure variables, Array of structures, arrays within structures, within structures, structures and functions.<br>**Unions :** Defining and working with union |
| | |
| • Understand the pointers<br>• Write dynamic programs<br>• Understand strength of pointers<br>• Store data in files<br>• Read data from files<br>• Understand File Handling Functions | **Unit VI: Pointers and File Handling (6 Hrs) Pointer:** Fundamentals, Pointer variables, Referencing and dereferencing, Pointer Arithmetic, Chain of pointers, Pointers and Arrays, Pointers and Strings, Array of Pointers, Pointers as function arguments, Functions returning pointers, Pointer to function, Pointer to structure, Pointers within structure.<br><br>**File Handling:** Different types of files like text and binary, Different types of functions fopen(), fclose(), fputc(), fscanf(), fprintf(), getw(), putw(), fread(), fwrite(), fseek()<br>**Dynamic Memory Allocation:** malloc(), calloc(), realloc(), free() and size of operator. |
| | |
| • Define a Linear Link List and list its features.<br>• Understand the advantages & shortcomings of link list over an array.<br>• Differentiate between Link List & Array.<br>• Write & Explain the basic operations of Linear Link List.<br>• Understand how to implement a link list.<br>• Write a program in C to implement linear link list. | **Unit VII : Link Lists (4 Hrs)**<br>**Linear Link lists:** Representation of link list in memory, Algorithms for traversing a link list, searching a particular node in link list, insertion into link list (insertion at the beginning of a node, insertion after a given node) deletion from a link list. Implement using C. |
| | |

| | |
|---|---|
| • Define a stack and its features.<br>• Write Algorithms for the basic operations of Stack.<br>• Understand the difference between Stack & Array.<br>• Understand how an Array is used to implement a Stack.<br>• Write a program in C to implement Stack. | **Unit VIII: Stacks (3 Hrs)**<br>**Stacks:** Definition, Array representation of stacks, Algorithms for basic operators to add and delete an element from the stack, Implement using C. |
| | |
| • Define a queue and state its features.<br>• State the applications that use queues.<br>• State the basic operations of a queue.<br>• Differentiate between straight queue and circular queue.<br>• Implement queues using arrays and linked lists. | **Unit VIII: Queues (3 Hrs)**<br>**Queues:** Representation of queue, Algorithm for insertion and deletion of an element in a queue, Implement using C. |

## Evaluation System:

| Undergraduate Programs | | | | |
|---|---|---|---|---|
| **External Evaluation** | **Marks** | **Internal Evaluation** | **Weightage** | **Marks** |
| End semester examination | 60 | Assignments | 10% | 40 |
| (Details are given in the separate table at the end) | | Quizzes | 10% | |
| | | Attendance | 10% | |
| | | Presentation | 10% | |
| | | Term papers | 10% | |
| | | Mid-Term exam | 40% | |
| | | Group work | 10% | |
| Total External | 60 | Total Internal | 100% | 40 |
| Full Marks 60+40 = 100 | | | | |

**External evaluation:**

**End semester examination:** It is a written examination at the end of the semester. The questions will be asked covering all the units of the course. The question model, full marks, time and others will be as per the following grid.

Full Marks: 100, Pass Marks: 50, Time: 3 Hrs

| Nature of question | Total questions to be asked | Total questions to be answered | Total marks | Weightage | External exam marks |
|---|---|---|---|---|---|
| Group A:<br>multiple choice* | 20 | 20 | 20×1 = 20 | 20% | 12 |
| Group B:<br>Short answer type questions | 11 questions | 8 | 8×5 = 40 | 40% | 24 |
| Group C:<br>Long answer type question/case studies | 6 questions | 4 | 4×10 =40 | 40% | 24 |
| | | | 100 | 100% | 60 |

*Scoring scheme will not follow negative marking.

Each student must secure at least 50% marks in internal evaluation in order to appear in the end semester examination. Failing to get such score will be given NOT QUILIFIED (NQ) and the student will not be eligible to appear in the end semester examinations.

**Practical examination:** Practical examination will be taken at the end of the semester. Students must demonstrate the knowledge of the subject matter.

## Internal evaluation

**Assignment:** Each student must submit the assignment individually. The stipulated time for submission of the assignment will be seriously taken.

**Quizzes:** Unannounced and announced quizzes/tests will be taken by the respective subject teachers. Such quizzes/tests will be conducted twice per semester. The students will be evaluated accordingly.

**Attendance in class:** Students should regularly attend and participate in class discussion. Eighty percent class attendance is mandatory for the students to enable them to appear in the end semester examination. Below 80% attendance in the class will signify NOT QUALIFIED (NQ) to attend the end semester examination.

**Presentation:** Students will be divided into groups and each group will be provided with a topic for presentation. It will be evaluated individually as well as group-wise. Individual students have to make presentations on the given topics.

**Term paper**: Term paper must be prepared by using computer in a standard format of technical writing and must contain the required number of pages. It should be prepared and submitted individually. The stipulated time for submission of the paper will be seriously taken as one of the major criteria of the evaluation.

**Mid-term examination:** It is a written examination and the questions will be asked covering all the topics in the session of the course.

**Discussion and participation**: Students will be evaluated on the basis of their active participation in the classroom discussions.

**Instructional Techniques:** All topics are discussed with emphasis on real-world application. List of instructional techniques is as follows:

- Lecture and Discussion
- Group work and Individual work
- Self study
- Assignments
- Presentation by Students
- Term Paper writing
- Quizzes
- Guest Lecture

Students are advised to attend all the classes and complete all the assignments within the specified time period. If a student does not attend the class(es), it is his/her sole responsibility to cover the topic(s) taught during that period. If a student fails to attend a formal exam/quiz/test, there won't be any provision for re-exam. Unless and until the student clears one semester he/she will not be allowed to study in the following semesters.

## Recommended Books:


- Introduction to Algorithms (Second Edition): *Cormen*, Leiserson, Rivest, Stein, PHI (Chapter 1, 2, 3, 10).
- Data Structures (Schaum's outline series in computers): Seymour Lipschutz McGraw-Hill book Company (Chapter 2, 5, 6, 9)
- Programming in ANSI C (Third Edition) : E Balguruswamy TMH (Chapters 2 to 13)
- Fundamental Algorithms (Art of Computer Programming Vol. I: Knuth Narosa Publishing House.
- Mastering Algorithms with C, Kyle Loudon, Shroff Publishers
- Algorithms in C (Third Edition): Robert Sedgewick, Pearson Education Asia.
- Data Structures A Pseudo code Approach with C: Richard F. Gilberg, Behrouz A. Forouzan, Thomas.
- Let us C by Yashwant Kanetkar, BPB
- Programming in ANSI C by Ram Kumar, Rakesh Agrawal, TMH
- Programming with C (Second Edition): Byron S. Gottfried. (Adapted by Jitender Kumar Chhabra) Schaum's Outlines (TMH)
- Programming with C: K.R. Venugopal, Sudeep R. Prasad TMH Outline Series.
- Unix and C : M.D. Bhave and S. A. Pateker, Nandu Printer and publishers private limited.

**Course Title: Programming Fundamentals and 'C' Programming** **Credit: 1**
**Nature of the Course: Lab.** **Number of hours per week:**
**Level: CSIT.115** **(2 hrX3times or 3 hr x 2 times) 6**
**Year: First** **Total hours: 48**
**Semester: First**

hout the semester and Practical examination will be conducted at the end of academic year. The practical exam will be graded on the basis of the following marking scheme:

| | |
|---|---|
| In-Semester Evaluation (Lab Book or Journal) | 20 % |
| Final Exam Written | 60 % |
| Final Exam Oral | 20 % |

Following are the guideline for the lab work:

1. There should be a lab book for the practical work related to the subject
2. The lab book will contain details of all practical's to be conducted in the lab
3. Students should read the lab book before coming to the lab
4. Every practical should have:
    a. Title
    b. Objectives
    c. Description
    d. Examples
    e. Self Activities
        i. Objective questions
        ii. Sample programs to be typed and executed
    f. Task list to be decided by the lab in-charge.
    g. Outputs to be verified by the lab in-charge.
5. Each practical should be conducted in the following manner:
    a. Explanation by lab in-charge – 10 minutes
    b. Self activities by students
    c. Lab in-charge will allocate tasks to each student (selection from a list / modify given task / specify new task )
    d. At the end of the slot, the lab in-charge has to verify the outputs and give a remark (Complete / Incomplete / Needs Improvement)

**Assignment List for Lab Work**

All the students will have to complete the following set of programming. Lab in-charge may assign additional assignment depending upon the time available.

1. Assignment to demonstrate use of data types, simple operators (expressions)
2. Assignment to demonstrate decision making statements (if and if-else, nested structures)
3. Assignment to demonstrate decision making statements (switch case)
4. Assignment to demonstrate use of simple loops
5. Assignment to demonstrate use of nested loops
6. Assignment to demonstrate menu driven programs.
7. Assignment to demonstrate writing C programs in modular way (use of user defined functions)
8. Assignment to demonstrate recursive functions.
9. Assignment to demonstrate use of arrays (1-d arrays ) and functions
10. Assignment to demonstrate use of multidimensional array(2-d arrays ) and functions
11. Assignment to demonstrate use of pointers
12. Assignment to demonstrate concept of strings ( string & pointers)
13. Assignment to demonstrate array of strings.
14. Assignment to demonstrate use of bitwise operators.
15. Assignment to demonstrate structures (using array and functions )
16. Assignment to demonstrate structures and unions

17. Assignment to demonstrate command line arguments and pre-processor directives.
18. Assignment to demonstrate file handling (text files)
19. Assignment to demonstrate file handling (binary files and random access to files)
**20.** Assignment to demonstrate graphics using C

**Recommended Books**


- Deitel, C.: **How to Program**, 2/e (With CD), Pearson Education.
- Al Kelley, Ira Pohl: "**A Book on C**", Pearson Education.
- Brian W. Keringhan & Dennis M. Ritchie: "**The C programming Language**", PHI
- Bryons S. Gotterfried: "**Programming with C**," TMH
- Stephen G. Kochan: "**Programming in C**", CBS publishers & distributors.
- Yashavant Kanetkar: "**Let us C**", BPB Publications
- Herbert Schildt - **Complete C Reference**
- Forouzan and Gilberg: **Structured Programming approach using C,** Thomson learning publications