12.Compute Divergence of a vector field.

13.Compute Curl of a vector field.

DSC 04: Object Oriented Programming with C++

Course Objective

This course is designed to introduce programming concepts using C++ to students. The course aims to develop structured as well as object-oriented programming skills using C++ programming language. The course also aims to achieve competence amongst its students to develop correct and efficient C++ programs to solve problems spanning multiple domains.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Write simple programs using built-in data types of C++.
2. Implement arrays and user defined functions in C++.
3. Write programs using dynamic memory allocation, handling external files, interrupts and exceptions.
4. Solve problems spanning multiple domains using suitable programming constructs in C++.
5. Solve problems spanning multiple domains using the concepts of object oriented programming in C++.

Syllabus

Unit 1 Introduction to C++: Overview of Procedural and Object-Oriented Programming, Using main() function, Header Files, Compiling and Executing Simple Programs in C++.

Unit 2 Programming Fundamentals: Data types, Variables, Operators, Expressions, Arrays, Keywords, Decision making constructs, Iteration, Type Casting, Input-output statements, Functions, Command Line Arguments/Parameters

Unit 3 Object Oriented Programming: Concepts of Abstraction, Encapsulation. Creating Classes and objects, Modifiers and Access Control, Constructors, Destructors, Implementation of Inheritance and Polymorphism, Template functions and classes

Unit 4 Pointers and References: Static and dynamic memory allocation, Pointer and Reference Variables, Implementing Runtime polymorphism using pointers and references

Unit 5 Exception and File Handling: Using try, catch, throw, throws and finally; Nested try, creating user defined exceptions, File I/O Basics, File Operations

References

1. Stephen Prata *C++ Primer Plus*, 6th Edition, Pearson India, 2015
2. E Balaguruswamy *Object Oriented Programming with C++*, 8th edition, McGraw-Hill Education, 2020
3. D.S. Malik, *C++ Programming: From Problem Analysis to Program Design*, 6th edition, Cengage Learning, 2013

Additional References

(i) Herbert Schildt, C++: *The Complete Reference*, 4th Edition, McGraw Hill, 2003

(ii) A. B. Forouzan, Richard F. Gilberg, *Computer Science: A Structured Approach using C++,* 2nd edition, Cengage Learning, 2010

Suggested Practical list

1. Write a program to compute the sum of the first n terms of the following series:

   $$S = 1 - \frac{1}{2^2} + \frac{1}{3^3} - \frac{1}{4^4} + \frac{1}{5^5} - \ldots.$$

   The number of terms n is to be taken from the user through the command line. If the command line argument is not found then prompt the user to enter the value of n.

2. Write a program to remove the duplicates from an array.
3. Write a program that prints a table indicating the number of occurrences of each alphabet in the text entered as command line arguments.
4. Write a menu driven program to perform string manipulation (without using inbuilt string functions):
   a. Show address of each character in string
   b. Concatenate two strings.
   c. Compare two strings
   d. Calculate length of the string (use pointers)
   e. Convert all lowercase characters to uppercase
   f. Reverse the string
   g. Insert a string in another string at a user specified position
5. Write a program to merge two ordered arrays to get a single ordered array.

6. Write a program to search a given element in a set of N numbers using Binary search (i) with recursion (ii) without recursion.
7. Write a program to calculate GCD of two numbers (i) with recursion (ii) without recursion.
8. Create a Matrix class. Write a menu-driven program to perform following Matrix operations (exceptions should be thrown by the functions if matrices passed to them are incompatible and handled by the main() function):
    a. Sum
    b. Product
    c. Transpose
9. Define a class Person having name as a data member. Inherit two classes Student and Employee from Person. Student has additional attributes as course, marks and year and Employee has department and salary. Write display() method in all the three classes to display the corresponding attributes. Provide the necessary methods to show runtime polymorphism.
10. Create a Triangle class. Add exception handling statements to ensure the following conditions: all sides are greater than 0 and sum of any two sides are greater than the third side. The class should also have overloaded functions for calculating the area of a right angled triangle as well as using Heron's formula to calculate the area of any type of triangle.
11. Create a class Student containing fields for Roll No., Name, Class, Year and Total Marks. Write a program to store 5 objects of Student class in a file. Retrieve these records from the file and display them.
12. Copy the contents of one text file to another file, after removing all whitespaces.

DSC 05: Discrete Mathematical Structures

Course Objective

This course is designed as a foundational course to make students learn about the mathematical constructs that are used in Computer Science such as Boolean algebra, sets, relations, functions, principles of counting, and recurrences. In this course, the knowledge of mathematical notation, ideas and concepts learnt at the pre-college levels is extended to orient the students towards mathematical thinking required in Computer Science.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Relate mathematical concepts and terminology to examples in the domain of Computer Science.
2. Model real world problems using various mathematical constructs.
3. Use different proofing techniques; construct simple mathematical proofs using logical arguments.
4. Formulate mathematical claims and construct counterexamples.

Syllabus

Unit 1 Sets, Functions, Sequences and Summations, Relations: Sets: Set Operations, Computer Representation of Sets, Countable and Uncountable Sets, Principle of Inclusion and Exclusion, Multisets; Functions: One-to-one and Onto Functions, Inverse Functions and Compositions of Functions, Graphs of Functions; Sequences and Summations: Sequences, Special Integer Sequences, Summations; Relations: Properties of Binary Relations, Equivalence relations and Partitions, Partial Ordering Relations and Lattices.

Unit 2 Logic and Proofs: Propositional Logic, Propositional Equivalences, Use of first-order logic to express natural language predicates, Quantifiers, Nested Quantifiers, Rules of Inference, Introduction to Proofs, Proof Methods and Strategies, Mathematical Induction.

Unit 3 Number Theory: Division and Integers, Primes and Greatest Common Divisors, Representation of Integers, Algorithms for Integer Operations, Modular Exponentiation, Applications of Number Theory.

Unit 4 Combinatorics/Counting: The Pigeonhole Principle, Permutations and Combinations, Binomial Coefficients, Generalized Permutations and Combinations, Generating Permutations and Combinations.

Unit 5 Graphs and Trees: Graphs: Basic Terminology, Multigraphs and Weighted Graphs, Paths and Circuits, Eulerian Paths and Circuits, Hamiltonian paths and Circuits, Shortest Paths, Spanning Trees, Graph Isomorphism, Planar Graphs; Trees: Trees, Rooted Trees, Path Lengths in Rooted Trees.

Unit 6 Recurrence: Recurrence Relations, Generating Functions, Linear Recurrence Relations with Constant Coefficients and their solution.

References

1. Liu, C.L., Mohapatra, D.P. *Elements of Discrete Mathematics: A Computer Oriented Approach*, 4th edition, Tata McGraw Hill, 2017.
2. Rosen, K.H.. *Discrete Mathematics and Its Applications*, 8th edition, Mc Graw Hill, 2018.

Additional References

(i) Cormen, T.H., Leiserson, C.E., Rivest, R. L., Stein C. *Introduction to Algorithms*, 4th edition, Prentice Hall of India. 2022.
(ii) Trembley, J.P., Manohar, R. *Discrete Mathematical Structures with Application to Computer Science*, Tata McGraw Hill, 1997.
(iii) Albertson, M. O. and Hutchinson, J. P. *Discrete Mathematics with Algorithms*, John Wiley and Sons, 1988.

Suggested Practical list

1. Create a class SET. Create member functions to perform the following SET operations:
    1) ismember: check whether an element belongs to the set or not and return value as true/false.
    2) powerset: list all the elements of the power set of a set .
    3) subset: Check whether one set is a subset of the other or not.
    4) union and Intersection of two Sets.
    5) complement: Assume Universal Set as per the input elements from the user.
    6) set Difference and Symmetric Difference between two sets.
    7) cartesian Product of Sets.

    Write a menu driven program to perform the above functions on an instance of the SET class.
2. Create a class RELATION, use Matrix notation to represent a relation. Include member functions to check if the relation is Reflexive, Symmetric, Anti-symmetric, Transitive. Using these functions check whether the given relation is: Equivalence or Partial Order relation or None
3. Write a Program that generates all the permutations of a given set of digits, with or without repetition.
4. For any number n, write a program to list all the solutions of the equation $x_1 + x_2 + x_3 + ...+ x_n = C$, where C is a constant (C<=10) and $x_1, x_2, x_3, ..., x_n$ are nonnegative integers, using brute force strategy.

5. Write a Program to evaluate a polynomial function. (For example store $f(x) = 4n^2 + 2n + 9$ in an array and for a given value of n, say n = 5, compute the value of f(n)).
6. Write a Program to check if a given graph is a complete graph. Represent the graph using the Adjacency Matrix representation.
7. Write a Program to check if a given graph is a complete graph. Represent the graph using the Adjacency List representation.
8. Write a Program to accept a directed graph G and compute the in-degree and out-degree of each vertex.

DSC 06: Probability for Computing

This course introduces the students to the fundamental concepts and topics of probability and statistics, whose knowledge is important in other computer science courses. The course aims to build the foundation for some of the core courses in later semesters.

Course Learning Outcomes

After successful completion of this course, the student will be able to:

1. Use probability theory to evaluate the probability of real-world events.
2. Describe discrete and continuous probability distribution functions and generate random numbers from the given distributions.
3. Find the distance between two probability distributions
4. Define and quantify the information contained in the data.
5. Perform data analysis in a probabilistic framework.
6. Visualize and model the given problem using mathematical concepts covered in the course.

Syllabus

Unit 1 Basic Probability: Introduction to the notion of probability, Random experiment, Sample space and Events, Probability defined on events, Algebra of events. Conditional probabilities, independent events, Bayes' theorem.

Unit 2 Random Variables: Introduction to Random Variables, Probability mass/density functions, Cumulative distribution functions. Discrete Random Variables (Bernoulli, Binomial, Poisson, Multinomial and Geometric). Continuous Random Variables (Uniform, Exponential and Normal). Expectation of a Random Variable, Expectation of Function of a Random Variable and Variance. Markov inequality, Chebyshev's inequality, Central Limit Theorem, Weak and Strong Laws of Large Numbers.

Unit 3 Joint Distributions: Jointly distributed Random Variables, Joint distribution functions, Independent Random Variables, Covariance of Random Variables, Correlation Coefficients, Conditional Expectation.

Unit 4 Markov Chain and Information Theory: Introduction to Stochastic Processes, Chapman–Kolmogorov equations, Classification of states, Limiting and Stationary Probabilities. Random Number Generation, Pseudo Random Numbers, Inverse Transformation Method, Rejection Method, Uncertainty, Information and Entropy, Mutual Information, KL Divergence.

References

1. Ross **S**heldon M. *Introduction to Probability Models*, 12th Edition, Elsevier, 2019.
2. Trivedi, K.S. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*, 2nd edition, Wiley, 2015.
3. Deisenroth, Marc Peter, Faisal A. Aldo and Ong Cheng Soon, *Mathematics for Machine Learning*, 1st edition, Cambridge University Press, 2020.
4. Ian F. Blake, *An Introduction to Applied Probability*, John Wiley.

Additional References

(i) Johnson James L. *Probability and Statistics for Computer Science*, 6th edition, Wiley, 2004.

(ii) Forsyth David *Probability and Statistics for Computer Science*, 1st edition, Springer, 2019.

(iii) Freund J.E. *Mathematical Statistics with Applications*, 8th edition, Pearson Education, 2013.

(iv) Devore Jay L. *Probability and Statistics for Engineering and the Sciences*, 9th edition, Cengage Learning, 2020

Suggested Practical List

The goal of this lab is to develop data interpretation skills. Following exercises are designed to enable students to understand data characteristics either by visualization or by interpreting computed measures. All the exercises are to be completed using MS Excel functions and graphs. At the end of each exercise, the student should be able to draw a conclusion and state in a concise manner.Teachers are expected to guide students to obtain real data available through the internet for the following exercises.

1. Plotting and fitting of Binomial distribution and graphical representation of probabilities.

2. Plotting and fitting of Multinomial distribution and graphical representation of probabilities.

3.Plotting and fitting of Poisson distribution and graphical representation of probabilities.

4.Plotting and fitting of Geometric distribution and graphical representation of probabilities.

5.Plotting and fitting of Uniform distribution and graphical representation of probabilities.

6.Plotting and fitting of Exponential distribution and graphical representation of probabilities.

7.Plotting and fitting of Normal distribution and graphical representation of probabilities.

8.Calculation of cumulative distribution functions for Exponential and Normal distribution.

9.Given data from two distributions, find the distance between the distributions.

10.Application problems based on the Binomial distribution.

11.Application problems based on the Poisson distribution.

12.Application problems based on the Normal distribution.

13.Presentation of bivariate data through scatter-plot diagrams and calculations of covariance.

14.Calculation of Karl Pearson’s correlation coefficients.

15.To find the correlation coefficient for a bivariate frequency distribution.

16.Generating Random numbers from discrete (Bernoulli, Binomial, Poisson) distributions.

17.Generating Random numbers from continuous (Uniform, Normal) distributions.

18.Find the entropy from the given data set.

DSC 07: Data Structures

Course Objective

The course aims at developing the ability to use basic data structures like arrays, stacks, queues, lists, trees to solve problems. C++ is chosen as the language to understand implementation of these data structures.

Course Learning Outcomes

On successful completion of the course, students will be able to:

1. Compare two functions for their rates of growth.
2. Understand abstract specification of data-structures and their implementation.
3. Compute time and space complexity of operations on a data-structure.
4. Identify the appropriate data structure(s) for a given application and understand the trade-offs involved in terms of time and space complexity.
5. Apply recursive techniques to solve problems.

Syllabus

Unit 1 Growth of Functions, Recurrence Relations: Functions used in analysis, asymptotic notations, asymptotic analysis, solving recurrences using recursion trees, Master Theorem.

Unit 2 Arrays, Linked Lists, Stacks, Queues, Deques: Arrays: array operations, applications, sorting, two-dimensional arrays, dynamic allocation of arrays; Linked Lists: singly linked lists, doubly linked lists, circularly linked lists, Stacks: stack as an ADT, implementing stacks using arrays, implementing stacks using linked lists, applications of stacks; Queues: queue as an ADT, implementing queues using arrays, implementing queues using linked lists, double-ended queue as an ADT, implementing double-ended queues using linked lists. Time complexity analysis.

Unit 3 Recursion: Recursive functions, linear recursion, binary recursion.

Unit 4 Trees, Binary Trees: Trees: definition and properties, tree traversal algorithms and their time complexity analysis; binary trees: definition and properties, traversal of binary trees and their time complexity analysis.

Unit 5 Binary Search Trees, Balanced Search Trees: Binary Search Trees: insert, delete, search operations, time complexity analysis of these operations; Balanced Search Trees: insert, search operations, time complexity analysis of these operations. Time complexity analysis.

Unit 6 Binary Heap, Priority Queue: Binary Heaps: heaps, heap operations, implementing heaps; Priority Queues: priority queue as an ADT. Time complexity analysis.