SARDAR RAJA COLLEGE OF ENGINEERING

RAJA NAGAR, ALANGULAM

Department of Electronics & Communication Engineering

Subject Name: OBJECT ORIENTED PROGRAMMING AND DATA STRUCTURES

Subject Code : EC6301

Year/Branch : II/ECE

Semester : III

Prepared By,

A.Kanagaraj,

Asst.prof/CSE.

SYLLABUS

EC6301 OBJECT ORIENTED PROGRAMMING AND DATA STRUCTURES L T P C
3 0 0 3

OBJECTIVES:

- To comprehend the fundamentals of object oriented programming, particularly in C++.
- To use object oriented programming to implement data structures.
- To introduce linear, non-linear data structures and their applications.

UNIT I DATA ABSTRACTION & OVERLOADING 9

Overview of C++ – Structures – Class Scope and Accessing Class Members – Reference Variables – Initialization – Constructors – Destructors – Member Functions and Classes – Friend Function – Dynamic Memory Allocation – Static Class Members – Container Classes and Integrators – Proxy Classes – Overloading: Function overloading and Operator Overloading.

UNIT II INHERITANCE & POLYMORPHISM 9

Base Classes and Derived Classes – Protected Members – Casting Class pointers and Member Functions – Overriding – Public, Protected and Private Inheritance – Constructors and Destructors in derived Classes – Implicit Derived – Class Object To Base – Class Object Conversion – Composition Vs. Inheritance – Virtual functions – This Pointer – Abstract Base Classes and Concrete Classes – Virtual Destructors – Dynamic Binding.

UNIT III LINEAR DATA STRUCTURES 10

Abstract Data Types (ADTs) – List ADT – array-based implementation – linked list implementation — singly linked lists –Polynomial Manipulation - Stack ADT – Queue ADT - Evaluating arithmetic expressions

UNIT IV NON-LINEAR DATA STRUCTURES 9

Trees – Binary Trees – Binary tree representation and traversals – Application of trees: Set representation and Union-Find operations – Graph and its representations – Graph Traversals – Representation of Graphs – Breadth-first search – Depth-first search - Connected components.

UNIT V SORTING and SEARCHING 8

Sorting algorithms: Insertion sort - Quick sort - Merge sort - Searching: Linear search –Binary Search

TOTAL: 45 PERIODS

OUTCOMES:

Upon completion of the course, students will be able to:

- Explain the concepts of Object oriented programming.
- Write simple applications using C++.
- Discuss the different methods of organizing large amount of data.

TEXT BOOKS:

1. Deitel and Deitel, “C++, How To Program”, Fifth Edition, Pearson Education, 2005.
2. Mark Allen Weiss, “Data Structures and Algorithm Analysis in C++”, Third Edition, Addison-Wesley, 2007.

REFERENCES:

1. Bhushan Trivedi, “Programming with ANSI C++, A Step-By-Step approach”, Oxford University Press, 2010.
2. Goodrich, Michael T., Roberto Tamassia, David Mount, “Data Structures and Algorithms in C++”, 7th Edition, Wiley. 2004.
3. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest and Clifford Stein, "Introduction to Algorithms", Second Edition, Mc Graw Hill, 2002.
4. Bjarne Stroustrup, “The C++ Programming Language”, 3rd Edition, Pearson Education, 2007.
5. Ellis Horowitz, Sartaj Sahni and Dinesh Mehta, “Fundamentals of Data Structures in C++”, Galgotia Publications, 2007.

Aim:

The goals of object-oriented programming and data structures are:

- Increased understanding.
- Ease of maintenance.
- Ease of evolution.
- Use different data structures

Description:

Object-oriented programming is an approach to designing modular, reusable software systems. Although discussions of object-oriented technology often get mired in the details of one language vs. the other, the real key to the object-oriented approach is that it is a modeling approach first Although often hyped as a revolutionary way to develop software by zealous proponents, the object-oriented approach is in reality a logical extension of good design practices that go back to the very beginning of computer programming. Object-orientation is simply the logical extension of older techniques such as structured programming and abstract data types. An object is an abstract data type with the addition of polymorphism and inheritance. Object orientation eases maintenance by the use of encapsulation and information hiding.

Object-orientation takes this to the next step. It essentially merges abstract data types with structured programming and divides systems into modular objects which own their own data and are responsible for their own behaviour. This feature is known as encapsulation. With encapsulation, not only can the "square root" and "launch missiles" functions not interfere with each other, but also the data for the two are divided up so that changes to one object cannot affect the other. Note that all this relies on the various languages being used appropriately, which, of course, is never certain. Object-orientation is not a software silver bullet, and it is not magic that makes all development problems go away.

Data structures are generally based on the ability of a computer to fetch and store data at any place in its memory, specified by an address a bit string that can be itself stored in memory and manipulated by the program. Thus the record and array data structures are based on computing the addresses of data items with arithmetic operations; while the linked data structures are based on storing addresses of data items within the structure itself. Many data structures use both principles, sometimes combined in non-trivial ways (as in XOR linking).

The implementation of a data structure usually requires writing a set of procedures that create and manipulate instances of that structure. The efficiency of a data structure cannot be analysed separately from those operations. This observation motivates the theoretical concept of an abstract data type, a data structure that is defined indirectly by the operations that may be performed on it, and the mathematical properties of those operations (including their space and time cost).

.

MICRO LESSON PLAN

WEEKS	HOURS	LECTURE TOPICS	TEXT BOOKS
UNIT-I DATA ABSTRACTION & OVERLOADING			
I	1	Overview of C++ , Structures (AV class)	T1
	2	Class Scope and Accessing Class Members	T1
	3	Reference Variables, Initialization, Constructor	T1
II	4	Destructors, Member Functions and Classes	T1
	5	Friend Function, Dynamic Memory Allocation (AV class)	T1
	6	Static Class Members	T1
	7	Container Classes and Integrators (AV class)	T1
	8	Proxy Classes	T1
III	9	Overloading: Function overloading and Operator Overloading.	T1
UNIT-II INHERITANCE & POLYMORPHISM			
III	10	Base Classes and Derived Classes, Protected Members (AV class)	T1
	11	Casting Class pointers and Member Functions	T1
	12	Overriding, Public, Protected and Private Inheritance	T1
	13	Constructors and Destructors in derived Classes (AV class)	T1
IV	14	Implicit Derived, Class Object To Base	T1
	15	Class Object Conversion, Composition Vs. Inheritance	T1
	16	Virtual functions, This Pointer	T1
	17	Abstract Base Classes and Concrete Classes	T1
	18	Virtual Destructors, Dynamic Binding (AV class)	T1
UNIT-III LINEAR DATA STRUCTURES			
V	19	Abstract Data Types (ADTs) (AV class)	T2
	20	List ADT	T2
	21	array-based implementation	T2
	22	linked list implementation	T2
	23	singly linked lists	T2
VI	24	Polynomial Manipulation	T2
	25,26	Stack ADT (AV class)	T2
	27	Queue ADT (AV class)	T2
	28	Evaluating arithmetic expressions	T2
UNIT-IV NON-LINEAR DATA STRUCTURES			
VII	29	Trees, Binary Trees (AV class)	T2
	30	Binary tree representation and traversals	T2
	31	Application of trees: Set representation and Union, Find operations	T2
VIII	32	Graph and its representations	T2
	33	Graph Traversals (AV class)	T2
	34	Representation of Graphs	T2
	35	Breadth-first search (AV class)	T2
IX	36	Depth-first search	T2
	37	Connected components	T2
UNIT-V SORTING and SEARCHING			
IX	38	Sorting algorithms: Insertion sort (AV class)	T2
X	39,40	Quick sort	T2
	41,42	Merge sort (AV class)	T2
XI	43	Searching: Linear search	T2
	44,45	Binary Search (AV class)	T2

Prepared By :A.Kanagaraj AP/CSE

EC6312 OOPS AND DATA STRUCTURES LABORATORY **L T P C**

0 0 3 2

OBJECTIVES:

The student should be made to:

- Learn C++ programming language.
- Be exposed to the different data structures
- Be familiar with applications using different data structures

LIST OF EXPERIMENTS:

1. Basic Programs for C++ Concepts
2. Array implementation of List Abstract Data Type (ADT)
3. Linked list implementation of List ADT
4. Cursor implementation of List ADT
5. Stack ADT - Array and linked list implementations
6. The next two exercises are to be done by implementing the following source files
   i. Program source files for Stack Application 1
   ii. Array implementation of Stack ADT
   iii. Linked list implementation of Stack ADT
   iv. Program source files for Stack Application 2
   v. An appropriate header file for the Stack ADT should be included in (i) and (iv)
7. Implement any Stack Application using array implementation of Stack ADT (by implementing files (i) and (ii) given above) and then using linked list
8. Implementation of Stack ADT (by using files (i) and implementing file (iii))
9. Implement another Stack Application using array and linked list implementations of Stack ADT (by implementing files (iv) and using file (ii), and then by using files (iv) and (iii))
11. Queue ADT – Array and linked list implementations
12. Search Tree ADT - Binary Search Tree
13. Implement an interesting application as separate source files and using any of the searchable ADT files developed earlier. Replace the ADT file alone with other appropriate ADT files. Compare the performance.
14. Quick Sort

TOTAL: 45 PERIODS

REFERENCE:

spoken-tutorial.org.

OUTCOMES:

At the end of the course, the student should be able to:

- Design and implement C++ programs for manipulating stacks, queues, linked lists, trees, and graphs.
- Apply good programming design methods for program development.
- Apply the different data structures for implementing solutions to practical problems.

LAB EQUIPMENT FOR A BATCH OF 30 STUDENTS:

Standalone desktops with C++ Compiler - 30 Nos.

(or)

Server with C++ compiler supporting 30 terminals or more.