An
Industrial Training Report
On

"Object Oriented Programming & Data Structure "

Submitted in partial fulfilment for the award of degree
Of
S. Y. B. Tech.
In

INFORMATION TECHNOLOGY

Submitted By

Krishna Ranjit Wable(21141216)

Under The Guidance of
Prof. R.S.Mawale

Government College of Engineering, Karad
(An Autonomous Institute of Government of Maharashtra)

Academic Year 2022-2023

Government College of Engineering, Karad
(An Autonomous Institute of Government of Maharashtra)

Department of Information Technology

Certificate

This is to certify that the Industrial Training entitled "Object Oriented Programming & Data Structure " has been carried out by **Krishna Ranjit Wable(21141216)** of Second Year B. Tech I.T. Class under the guidance of **Prof. R.S.Mawale** during the academic year 2022-23 (Sem – III)

Prof. R.S.Mawale
Guide Head

Dr. S. J. Wagh
Head
Information Technology
Department

ACKNOWLEDGEMENT

No work can be completed without other's help or contribution. The preparation of presentation of this humble work encompasses the immense and unlimited help and sound though of innumerable people.

It is our privilege to express our gratitude towards our industry mentor ***Mr. Sandesh Mahamure*** and academic guide ***Prof. R.S.Mawale*** for their valuable guidance, encouragement, inspiration and whole-hearted cooperation throughout the project work. We thank him for being a motivation through all our highs and importantly, our lows.

We deeply express our sincere thanks to our Head of Department **Dr. S. J. Wagh** for encouraging and allowing us to present the skills gained and work done during training period on " Title of training " and providing us with the necessary facilities to enable us to fulfill our training requirements as best as possible. We take this opportunity to thank all faculty members and staff of Department of Information Technology, who have directly or indirectly helped our project.

We pay our respects to honourable Principal **Dr. A. T. Pise** for their encouragement. Our thanks and appreciations also go to our family and friends, who have been a source of encouragement and inspiration throughout the duration of the industrial training.

KRISHNA R WABLE
B.TECH.S.Y.IT.
21141216
2024-2025

Weekly Overview of Internship Activities

Sr. No.	Date	Week	Topic Learned
1.	28/10/2022 To 30/10/2022	1st	Program execution in System,Type of language,Programing Paradigm,Datatypes, Variables,oprators,Typecasting ,Enum and Typedef,Dyanamic Allocation, Loops,Array,Linear Search,Binary Search
2.	31/10/2022 To 6/11/2022	2nd	2D Array,Pointer,Pointer Airthmatic,References,String, Function,Function overloading,Template,Default Arguments,Passing Methods,Return by Address and reference,Global,Local and static variable and its scope,OOP's basic concept,Class and objects,Pointers to objects,Type of Function,Scope Resolution,Inline Function,This pointer,Oprator Overloading,Friend class and Objects,Contractorsfunction,Inheritance and its types,IsA and HasA relationship,Function Overriding,Polymorphism And its types,Static member in class and static member function,Inner class,Template
3.	7/11/2022 To 13/11/2022	3rd	Constant Qualifiers,Preprocessor Directives,Destructors,Iostream,File handaling,STL,Vector,Map,Asympathic Notation,Recurance relation,Stack And its application,Queue and Its Application,Circular Queue,Priority Queue,Linked List And Its type,Basic of Tree
4.	14/11/2022 To 18/11/2022	4th	Binary tree and its traversal,BST,AVL Tree,Threaded Binary Tree,Minimum Spaining tree,B and B+ Tree,Heap TreeGraph And its Type,BFS,DFS,Hashing, Multilevel Indexing, All Sorting Techniques and its Algoritham
5.	19/11/2022	5th	Project Planning and Design
6.	20/11/2022	6th	Project Implementation and Testing

INFORMATION ABOUT COMPANY

IFS India Mercantile Pvt. Ltd. Pune provides the best Object Oriented Programming and Data Structures and Algorithm Training in both online and offline mode. IFS India Mercantile Pvt. Ltd. Pune labs are equipped with latest software so that students can get 100% practical training. They specialize in creating and improving products and services according to the current market needs. They also assist clients with market exploration and discovery for business development. Alongside this they guide clients in the implementation of marketing changes to make a discernible business difference. Furthermore, they keep them updated on the latest marketing trends.

Company Name	IFS India Mercantile Pvt. Ltd. Pune
LoC	Pune
Directors	Santoshasai Palakurthy, Rajeswari Kondeti
Involved In	Data Science Projects, Software's and Computer Related Activities, (Maintenance of websites of other firms/ Creation of multimedia presentation for other firms, etc.)
Class of Company	Private
Date of Incorporation	31st March, 2009
Other Information	Its Authorized Share Capital is Rs. 100,000 and its Paid Up Capital is Rs. 100,000
Contact Details	Email ID: admin@ifsindia.org / hr@ifsindia.org

TABLE CONTENT

INDUSTRIAL EXPERIENCE

As a part of online training conducted by 'IFS India Mercantile Pvt. Ltd. Pune' it was on "Data Structure and Object Oriented Programming". I got a list of tasks to accomplish it every day. Initially we were been trained on the basic concepts of C++ before getting into "Object Oriented Programming" concept, and then we progressed with "Data Structure". Every day we were asked to perform operations on different datasets on the basis of what we have learnt. Once we are done with the task given the solution was discussed and made sure that it is conveyed in the easiest way that is possible.

Data Structure and Object Oriented Programming:

Data Structure: A data structure is a group of data elements that provides the easiest way to store and perform different actions on the data of the computer. A data structure is a particular way of organizing data in a computer so that it can be used effectively. The idea is to reduce the space and time complexities of different tasks. Data Structure Concepts: Linear Data Structure (Array, Queue, Stack, Linked List) and Non-Linear Data Structure (Trees and Graphs).

Object Oriented Programming:

As the name suggests, Object-Oriented Programming or OOPs refers to languages that use objects in programming. Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc in programming. The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function. Object Oriented Programming Concepts: Class, Objects, Data Abstraction , Encapsulation, Inheritance, Polymorphism, etc.

INDEX

CHAPTER 1

Introduction

1. Program Execution and Data Types:

1.1 Program Execution in System:

Program is a set of instructions that you give to a computer so that it will do a particular task. The program is executed in the following manner, firstly the program is saved in it's extension form. Then it is been preprocessed or complied by the compiler and the code is converted into object code. Then the libray files and linkers combinedly converts object code to executable code (Fig. 1).

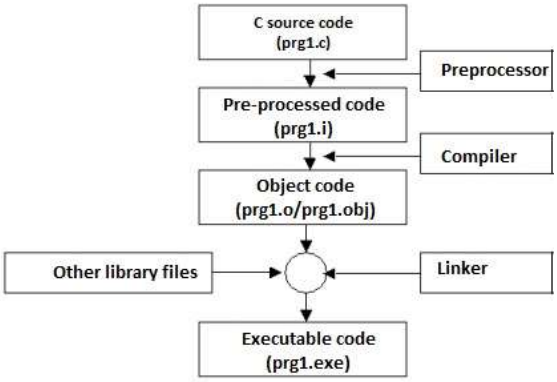


Fig 1: Program Execution and Data Types

1.2 Types Of Languages:

a. Low Level: Faster for processing but development time is high. e.g. Machine Language, Assembely Language.

b. Mid Level: Moderate speed for execution and takes moderate time for development. e.g. c,c++ (this are close to hardware).

c. High Level: Low speed of execution but high time for development. e.g. python

1.2.1 Languages baesd upon compiler, interpter and hybrid:

a. Compiler based: C , C++

b. Interpeter based: Python, JavaScript.

c. Hybrid based: Java

1.3 Programming Paradigm:

a. **Monolithic Programming**: If, we write an entire program in a single function that is in main function then, you call it as a monolithic type of programming.
b. **Modular / Procedural Programming**: If the program is divided into number of functional parts, then we use to call it as modular programming.
c. **Object Oriented Programming**: Object-oriented programming (OOP) is a computer programming model that organizes software design around data, or objects, rather than functions and logic. An object can be defined as a data field that has unique attributes and behaviour.
d. **Aspect Oriented / Component Assembly Programming**: In computing, aspect-oriented programming (AOP) is a programming paradigm that aims to increase modularity by allowing the separation of cross-cutting concerns.

1.4 Data types:

A **data type**, in programming, is a classification that specifies which type of value a variable has and what type of mathematical, relational or logical operations can be applied to it without causing an error (Fig.2).

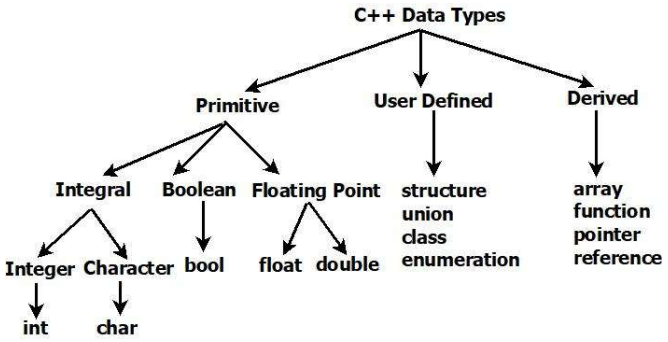


Fig.2:Data Types

1.4.1 Typecasting:

Typecasting is the process in which the compiler automatically converts one data type in a program to another one.

1.4.2 Enum and Typedef:

Enumerated type (enumeration) is a user-defined data type which can be assigned some limited values. These values are defined by the programmer at the time of declaring the enumerated type.

2. Statement:

2.1 Conditional Statement:-

1) **if else**: Decision making in programming is similar to decision making in real life. In decision making, a piece of code is executed when the given condition is fulfilled. Sometimes these are also termed as the Control flow statements.
2) **switch case:** Switch case statement evaluates a given expression and based on the evaluated value (matching a certain condition); it executes the statements associated with it. Basically, it is used to perform different actions based on different conditions (cases). Switch case statements follow a selection-control mechanism and allow a value to change control of execution.

2.2 Looping Statements:

1) **while:** While Loop is used in situations where we do not know the exact number of iterations of the loop before-hand. The loop execution is terminated on the basis of the test condition.
2) **do while**: Like while the do-while loop execution is also terminated on the basis of a test condition.
3) **for**: A for loop is a repetition control structure that allows us to write a loop that is executed a specific number of times. The loop enables us to perform n number of steps together in one line.
4) **for each:** 'for each' loop is used to iterate over the elements of a containers (array, vectors etc.) quickly without performing initialization, testing and increment/decrement

3. Array and Search:

3.1 Array:

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together.

- The size of the array should be mentioned while declaring it.
- Array elements are always counted from zero (0) onward.
- Array elements can be accessed using the position of the element in the array.
- The array can have one or more dimensions.

3.2 Linear Search:

Linear search (Searching algorithm) which is used to find whether a given number is present in an array and if it is present then at what location it occurs.

3.3 Binary Search:

Binary Search is a method to find the required element in a sorted array by repeatedly halving the array and searching in the half. This method is done by starting with the whole array. Then it is halved.

4. Function

A function is a set of statements that take inputs, do some specific computation, and produce output. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can call the function.

4.2.1 Default argument:

A default argument is a value provided in a function declaration that is automatically assigned by the compiler if the calling function doesn't provide a value for the argument. In case any value is passed, the default value is overridden.

4.2.2 Recursive Function:

A function that calls itself is known as a recursive function

5. Pointer and Reference

5.1 Pointer:

Pointers are used to store the address of variables or a memory location. This variable can be of any data type i.e. int, char, function, array, or any other pointer. The size of the pointer depends on the architecture (Fig. 5).

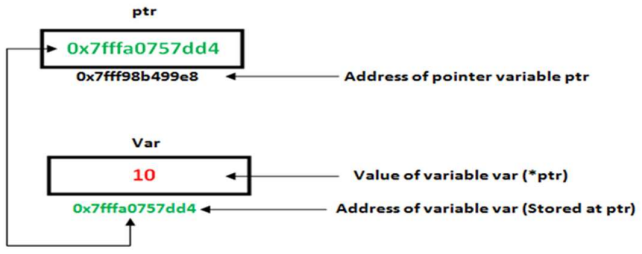


Fig.3:Pointers

5.2 References:

When a variable is declared as a reference, it becomes an alternative name for an existing variable. A variable can be declared as a reference by putting '&' in the declaration. It does not take any memory space.

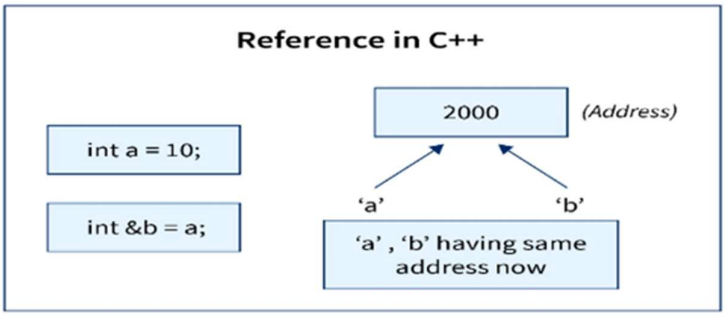


Fig. 4:Reference

5.3 Passing Methods:

1) **Passing by value**: In this value is passed to the function, i.e. formal argument contains values.
2) **Passing by reference**: In this we pass the reference to the function, i.e. formal an argument contains reference address of actual argument.
3) **Passing by address**: In this address is passed to the function, i.e. formal an argument contains pointers which points to actual argument.

5.4 Returning Type:

1) **Returning by value**: This function returns the value.
2) **Returning by reference**: This function returns the reference of the specified argument.
3) **Returning by address**: This function returns the address.

CHAPTER 2

Object Oriented Programming

1. Basic structure of Object Oriented Programming (OOPs):

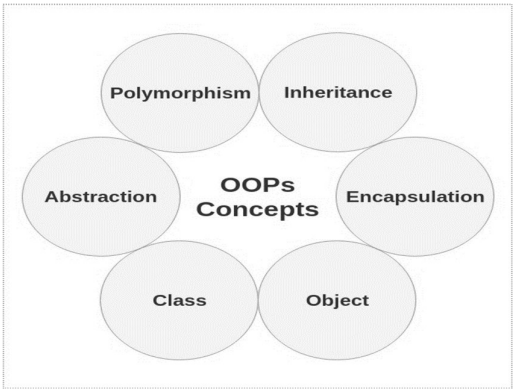


Fig. 5:Object Oriented Programming

1.1 Abstraction:

Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

1.2 Encapsulation:

Encapsulation is defined as wrapping up of data and information under a single unit. In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.

2. Class and Constructor:

2.1 Class:

A class is the building block that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.When we write function inside class we call it as **Method**.When a class is defined, only the specification for the object is defined; no memory or storage is allocated. To use the data and access functions defined in the class, you need to create **objects**.

2.2 Constructor:

Constructor is a special method that is invoked automatically at the time of object creation. It is used to initialize the data members of new objects generally. The constructor has the same name as the class or structure. It constructs the values i.e. provides data for the object which is why it is known as **constructors**.

2.1 Default constructor:

Default constructor is the constructor which doesn't take any argument. It has no parameters. It is also called a zero-argument constructor.

1) **Non-parameterised constructor:**
   Non-Parametric Methods use the flexible number of parameters to build the model. Non-Parametric Methods requires much more data than Parametric Methods.
2) **Parameterised constructor:**
   It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created.
3) **Copy constructor:**
   A copy constructor is a member function that initializes an object using another object of the same class
4) **Deep Copy Constructor:**
   In Deep copy, an object is created by copying data of all variables, and it also allocates similar memory resources with the same value to the object

3. Function in class:

- **Function overloading:**
  Writing a function within a same function, with same name but different number of parameters.
- **Function Overriding:**
  It is the redefinition of base class function in its derived class with same signature i.e. return type and parameters.
- **Function template:**
  The duplication of function by changing its data type is known as templating.
- **Inline Function:**

Inline function is a function that is expanded in line when it is called. When the inline function is called whole code of the inline function gets inserted or substituted at the point of inline function call. This substitution is performed by the compiler at compile time. Inline function may increase efficiency if it is small.

4. Inheritance:

The capability of a class to derive properties and characteristics from another class is called Inheritance (Fig.7).

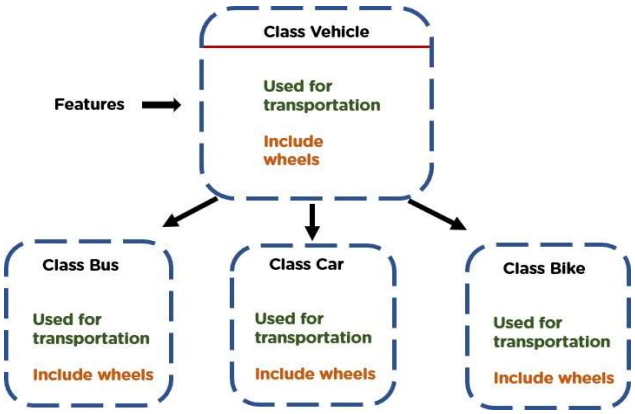


Fig.6:Inheritance

4.1 Modes of Inheritance:

There are 3 modes of inheritance:

1. Public Mode: If we derive a subclass from a public base class. Then the public member of the base class will become public in the derived class and protected members of the base class will become protected in the derived class.

2. Protected Mode: If we derive a subclass from a Protected base class. Then both public members and protected members of the base class will become protected in the derived class.

3. Private Mode: If we derive a subclass from a Private base class. Then both public members and protected members of the base class will become Private in the derived class.

4.2 Types of Inheritance:

Types of inheritance are:

1) **Single Inheritance:**

   In single inheritance, a class is allowed to inherit from only one class. i.e. one subclass is inherited by one base class only.

2) **Multiple Inheritance:**

   Multiple Inheritance is a feature of C++ where a class can inherit from more than one class i.e. one subclass is inherited from more than one base class.

3) **Multilevel Inheritance:**

   In this type of inheritance, a derived class is created from another derived class.

4) **Hierarchical Inheritance:**

   In this type of inheritance, more than one subclass is inherited from a single base class i.e. more than one derived class is created from a single base class.

5) **Hybrid (Virtual) Inheritance:**

   Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

4.3 IsA and HasA:

IsA is used for inheriting properties only, while HasA inherits using object of other class.

5. Polymorphism:

The word polymorphism means having many forms.

5.1 Types of polymorphism:

1) **Compile Time Polymorphism:**

   This type of polymorphism is achieved by function overloading or operator overloading.

   Run time is the time period where the executable code is running.

   Errors can be detected only after the execution of the program.

   Errors that occur during the execution of a program are called run-time errors.

   Run time errors aren't detected by the compiler.

2) **Run Time Polymorphism:**

This type of polymorphism is achieved by Function Overriding. Late binding and dynamic polymorphism are other names for runtime polymorphism. The function call is resolved at runtime in runtime polymorphism. In contrast, with compile time polymorphism, the compiler determines which function call to bind to the object after deducing it at runtime.

6. More about class and Exception handling:

6.1 Friend Function:

Friend Function Like friend class, a friend function can be given a special grant to access private and protected members. A friend function can be a member of another class or a global function.

6.2 Static Member of Class:

Static data members are class members that are declared using static keywords. A static member has certain special characteristics. These are only one copy of that member is created for the entire class and is shared by all the objects of that class, no matter how many objects are created, it is initialized before any object of this class is being created, even before main starts and it is visible only within the class, but its lifetime is the entire program.

6.3 Nested / Inner Class:

A nested class is a class which is declared in another enclosing class. A nested class is a member and as such has the same access rights as any other member. The members of an enclosing class have no special access to members of a nested class; the usual access rules shall be obeyed.

6.4 Exception Handling:

Exceptions are runtime anomalies or abnormal conditions that a program encounters during its execution.

Exception handling consists of three keywords: try, throw and catch.

The try statement allows you to define a block of code to be tested for errors while it is being executed. The throw keyword throws an exception when a problem is detected, which lets us create a custom error. The catch statement allows you to define a block of code to be executed if an error occurs in the try block.

7. Destructor :

Destructor is an instance member function which is invoked automatically whenever an object is going to be destroyed. Meaning, a destructor is the last function that is going to be called before an object is destroyed. Characteristics of destructor frees the memory, free the resource, used to delocalisation of resource.

7.1 Virtual Destructors:

A virtual destructor is used to free up the memory space allocated by the derived class object or instance while deleting instances of the derived class using a base class pointer object.

8. File Handling

File handling is used to store data permanently in a computer. Using file handling we can store our data in secondary memory (Hard disk).

In C++, files are mainly dealt by using three classes fstream, ifstream, ofstream available in fstream headerfile.

ofstream: Stream class to write on files

ifstream: Stream class to read from files

fstream: Stream class to both read and write from/to files.

9.Standard Template Library

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. Working knowledge of template classes is a prerequisite for working with STL.

STL has 4 components:

9.1 Alogorithm

The header algorithm defines a collection of functions specially designed to be used on a range of elements. They act on containers and provide means for various operations for the contents of the containers.

9.2 Containers

Containers or container classes store objects and data. There are in total seven standards “first-class” container classes and three container adaptor classes and only seven header files that provide access to these containers or container adaptors.

- Sequence Containers: implement data structures that can be accessed in a sequential manner.
  Ex.vector, list, deque, arrays, forward_list
- Container Adaptors: provide a different interface for sequential containers.
  Ex.queue ,priority_queue,stack
- Associative Containers: implement sorted data structures that can be quickly searched (O(log n) complexity).
  Ex.set,multiset, map, multimap
- Unordered Associative Containers: implement unordered data structures that can be quickly searched
  Ex.unordered_set ,unordered_multiset ,unordered_map ,unordered_multimap

9.3 Iterators

Iterators are pointer-like entities used to access the individual elements in a container.Iterators are moved sequentially from one element to another element. This process is known as iterating through a container.

9.4 Function

A Function object is a function wrapped in a class so that it looks like an object. A function object extends the characteristics of a regular function by using the feature of aN object oriented such as generic programming. Therefore, we can say that the function object is a smart pointer that has many advantages over the normal function.

CHAPTER 3

Data Structure

Data Structures are the programmatic way of storing data so that data can be used efficiently. Almost every enterprise application uses various types of data structures in one or the other way. This tutorial will give you a great understanding on Data Structures needed to understand the complexity of enterprise level applications and need of algorithms, and data structures.

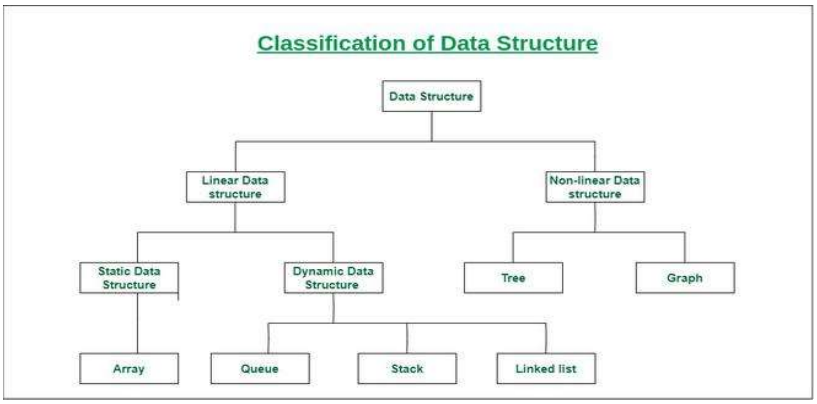


Fig.7:Classification Of Data Structure

1. Asymptotic notation :

1.1 Asymptotic Notation:

Asymptotic notations are mathematical tools to represent the time complexity of algorithms for asymptotic analysis. There are mainly three asymptotic notations:

1) **Big-O Notation (O-notation):**
   Big-O notation represents the upper bound of the running time of an algorithm. Therefore, it gives the worst-case complexity of an algorithm (Fig. 9).
2) **Omega Notation (Ω-notation):**
   Omega notation represents the lower bound of the running time of an algorithm. Thus, it provides the best case complexity of an algorithm (Fig. 10).
3) **Theta Notation (Θ-notation):**

Theta notation encloses the function from above and below. Since it represents the upper and the lower bound of the running time of an algorithm, it is used for analysing the average-case complexity of an algorithm (Fig. 11).

2.Array

An array is a collection of items stored at contiguous memory locations. The idea is to store multiple items of the same type together. This makes it easier to calculate the position of each element by simply adding an offset to a base value, i.e., the memory location of the first element of the array (generally denoted by the name of the array).

2.1 Application of array:

a. Stack b. Queue c. Circular Queue d. Matrix e.subarray f.searching and sorting

3. Stack:

It is a linear data structure that follows a particular order in which the operations are performed. This strategy states that the element that is inserted last will come out first.

Operation	Description
push()	Process of adding or storing an element at the top of stack
pop()	Process of removing or accessing an element from the top of stack
isempty()	Check if the stack is empty

Applications of Stacks

- A Stack can be used for evaluating expressions consisting of operands and operators.
- Stacks can be used for Backtracking, i.e., to check parenthesis matching in an expression.
- It can also be used to convert one form of expression to another form.

4. Queue:

A queue is defined as a linear data structure that is open at both ends and the operations are performed in First In First Out (FIFO) order.

We define a queue to be a list in which all additions to the list are made at one end, and all deletions from the list are made at the other end. The element which is first pushed into the order, the operation is first performed on that.

4.1Simple Queue

It is the most basic queue in which the insertion of an item is done at the front of the queue and deletion takes place at the end of the queue.

4.2 Circular Queue:

A Circular Queue is a special version of queue where the last element of the queue is connected to the first element of the queue forming a circle.

4.3 Priority Queue:

Priority queues are a type of container adapters, specifically designed such that the first element of the queue is either the greatest or the smallest of all elements in the queue and elements are in non-increasing order (hence we can see that each element of the queue has a priority (fixed order).

5. List:

Lists are sequence containers that allow non-contiguous memory allocation. As compared to vector, the list has slow traversal, but once a position has been found, insertion and deletion are quick.

5.1 Linear Linked list:

A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers.

5.2 Doubly Linked List:

A Doubly Linked List (DLL) contains an extra pointer, typically called the previous pointer, together with the next pointer and data which are there in the singly linked list.

5.3 Circular Linked List:

The circular linked list is a linked list where all nodes are connected to form a circle. In a circular linked list, the first node and the last node are connected to each other which form a circle.

Types of circular linked list-1.Singly circular linked list 2.Doubly circular linked list

5.4 Application of linked list:

- Implementation of stacks and queues

- Implementation of graphs: Adjacency list representation of graphs is the most popular which uses a linked list to store adjacent vertices.
- Dynamic memory allocation: We use a linked list of free blocks.
- Maintaining a directory of names
- Performing arithmetic operations on long integers
- Manipulation of polynomials by storing constants in the node of the linked list
- representing sparse matrices

6. Trees

A tree is a non-linear data structure that stimulates a tree structure with a root value and subset of children with parent nodes represented as a set of linked nodes.

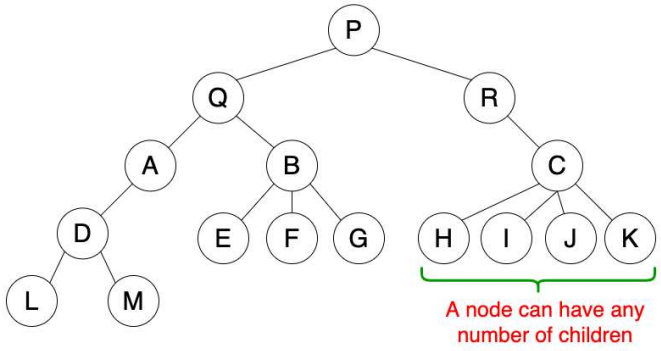


Fig.8:Tree

6.1 Types of trees:

a. General tree b. Binary tree c. Binary search tree d.AVL tree
e. Expression tree f. B tree g. B++ tree

6.2 Application of Trees:

- One reason to use trees might be because you want to store information that naturally forms a hierarchy.
- If we organize keys in form of a tree (with some ordering e.g., BST), we can search for a given key in moderate time (quicker than Linked List and slower than arrays).
- trees guarantee an upper bound of O(log n) for insertion/deletion.
- Binary Search Tree is a tree that allows fast search, insert, delete on a sorted data. It also allows finding closest item

- B-Tree and B+ Tree: They are used to implement indexing in databases.
- Syntax Tree: Scanning, parsing, generation of code and evaluation of arithmetic expressions in Compiler design.
- Spanning Trees and shortest path trees are used in routers and bridges respectively in computer networks.

7. Binary Tree:

A binary tree is a tree data structure in which each parent node can have at most two children.

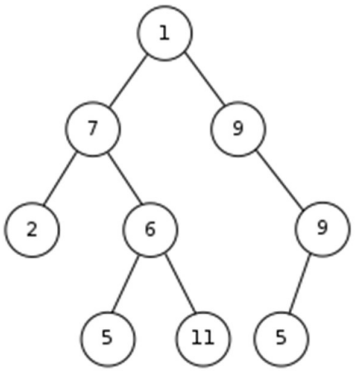


Fig.9:Binary search Tree

7.1 Types of binary tree:

1) **Full binary tree:**

   A full Binary tree is a special type of binary tree in which every parent node/internal node has either two or no children. It is also known as a proper binary tree.

2) **Perfect binary tree**:

   A perfect binary tree is a type of binary tree in which every internal node has exactly two child nodes and all the leaf nodes are at the same level.

3) **Complete binary tree:**

   A complete binary tree is a binary tree in which all the levels are completely filled except possibly the lowest one, which is filled from the left.

4) **Balanced binary tree:**

A balanced binary tree, also referred to as a height-balanced binary tree, is defined as a binary tree in which the height of the left and right sub-tree of any node differs by not more than 1.

7.2 Tree Traversal:

1) **Preorder Traversal:**

In the preorder traversal, the root node is processed first, followed by left subtree and then right subtree. The root gets processed before subtrees.

2) **Inorder Traversal:**

The inorder traversal processes the left subtree first, the root and finally right subtree.

3) **Postorder Traversal:**

In postorder the root is processed after the left and right subtree have been processed.

4) **Binary Search Tree:**

Binary search tree is a data structure that quickly allows us to maintain a sorted list of numbers.

8. Graph:

A Graph is a non-linear data structure consisting of vertices and edges. The vertices are sometimes also referred to as nodes and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph is composed of a set of vertices and a set of edges.

8.1 Types of graph:

8.1.1 Null Graph

A graph is known as a null graph if there are no edges in the graph.

8.1.2 Trivial Graph

Graph having only a single vertex, it is also the smallest graph possible.

8.1.3 Undirected Graph

A graph in which edges do not have any direction. That is the nodes are unordered pairs in the definition of every edge. \

8.1.4 Directed Graph

A graph in which edge has direction. That is the nodes are ordered pairs in the definition of every edge.

8.1.5 Weighted Graph

A graph in which the edges are already specified with suitable weight is known as a weighted graph. Weighted graphs can be further classified as directed weighted graphs and undirected weighted graphs.

8.2 BFS and DFS:

1) **Breadth First Search:**

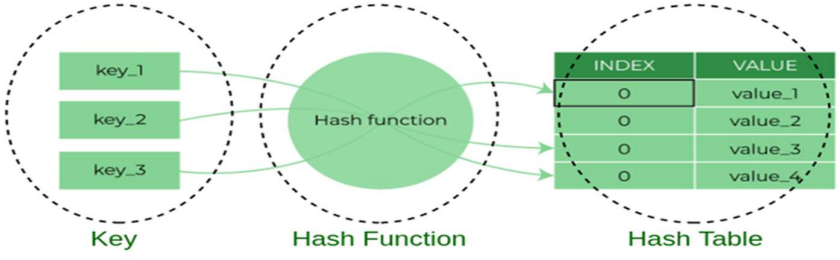
   Breadth-first search is a graph traversal algorithm that starts traversing the graph from the root node and explores all the neighbouring nodes. Then, it selects the nearest node and explores all the unexplored nodes. While using BFS for traversal, any node in the graph can be considered as the root node.

2) **Depth First Traversal:**

   Depth First Traversal for a graph is similar to Depth First Traversal of a tree. The only catch here is, that, unlike trees, graphs may contain cycles (a node may be visited twice). To avoid processing a node more than once, use a boolean visited array. A graph can have more than one DFS traversal.

9. Hashing:

Hashing is a technique or process of mapping keys, and values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.



Components of Hashing

Fig 10:Components of hashing

9.1 Types of Hashing:

A. Closed Hashing:

Open addressing. The types of closed hashing are:

a. Linear Probing

In linear probing, the hash table is searched sequentially that starts from the original location of the hash. If in case the location that we get is already occupied, then we check for the next location.

The function used for rehashing is as follows: rehash(key) = (n+1)%table-size.

b. Quadratic Probing

Quadratic probing is an open-addressing scheme where we look for the i2 'th slot in the i'th iteration if the given hash value x collides in the hash table.

If the slot hash(x) % S is full, then we try (hash(x) + 1*1) % S.

c. Double Probing

In this technique, the increments for the probing sequence are computed by using another hash function. We use another hash function hash2(x) and look for the i*hash2(x) slot in the i th rotation.

Double hashing can be done using : (hash1(key) + i * hash2(key)) % table-size

B. Open Hashing:

Close addressing.

10. Sorting

A Sorting Algorithm is used to rearrange a given array or list of elements according to a comparison operator on the elements. The comparison operator is used to decide the new order of elements in the respective data structure.

10.1 Bubble Sort:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

Follow the below steps for bobble sort:

1. Run a nested for loop to traverse the input array using two variables i and j, such that $0 \leq i < n-1$ and $0 \leq j < n-i-1$.

2. If arr[j] is greater than arr[j+1] then swap these adjacent elements, else move on.
3. Print the sorted array.

10.2 Insertion Sort:

Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands. The array is virtually split into a sorted and an unsorted part. Values from the unsorted part are picked and placed at the correct position in the sorted part.

To sort an array of size N in ascending order:

1. Iterate from arr[1] to arr[N] over the array.
2. Compare the current element (key) to its predecessor.
3. If the key element is smaller than its predecessor, compare it to the elements before. Move the greater elements one position up to make space for the swapped element.

10.3 Selection Sort:

The selection sort algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.

1. Follow the below steps for selection sort:
2. Initialize minimum value(min_idx) to location 0.
3. Traverse the array to find the minimum element in the array.
4. While traversing if any element smaller than min_idx is found then swap both the values.
5. Then, increment min_idx to point to the next element.
6. Repeat until the array is sorted.

10.4 Heap Sort:

First convert the array into heap data structure using heapify, than one by one delete the root node of the Max-heap and replace it with the last node in the heap and then heapify the root of the heap. Repeat this process until size of heap is greater than 1.

Follow the given steps for heap sort:

1. Build a max heap from the input data.

2. At this point, the maximum element is stored at the root of the heap. Replace it with the last item of the heap followed by reducing the size of the heap by 1. Finally, heapify the root of the tree.
3. Repeat step 2 while the size of the heap is greater than 1.

10.5 Merge Sort:

Merge sort is one of the most efficient sorting algorithms. It is based on the divide-and-conquer strategy. Merge sort continuously cuts down a list into multiple sublists until each has only one item, then merges those sublists into a sorted list.

10.6 Quick Sort:

Quicksort is a divide-and-conquer algorithm. It works by selecting a 'pivot' element from the array and partitioning the other elements into two sub-arrays, according to whether they are less than or greater than the pivot. For this reason, it is sometimes called partition-exchange sort.

Follow the given steps for Quick sort:

1. An array is divided into subarrays by selecting a pivot element (element selected from the array). While dividing the array, the pivot element should be positioned in such a way that elements less than pivot are kept on the left side and elements greater than pivot are on the right side of the pivot.
2. The left and right subarrays are also divided using the same approach. This process continues until each subarray contains a single element.
3. At this point, elements are already sorted. Finally, elements are combined to form a sorted array.

PROJECT

```
--------------------------------------------------------------------------------
                    WELCOME TO RED ROCK INN
--------------------------------------------------------------------------------

________________________________________________________________________________
                     WELCOME TO LOGIN PAGE
--------------------------------------------------------------------------------
        1.Manager Login
        2.Register
        3.Forgot password
        4.Close Application
                Enter your choice-
```

```
               You login is successfully
########################WELCOME-krish##############################
--------------------------------------------------------------------------------
                     WELCOME TO RED ROCK INN
--------------------------------------------------------------------------------
Choose the following Actions
1.Get inforamtion about empty Rooms
3.Get information about Remaining stock of drinks
4.To add new customer
5.Get information about customer in room
6.Place a order for a custumer
7.Search a custumer record
8.Delete a customer record
9.Exit from Application
```

```
Enter a customer name=nandan
Information of customer


Name of customer is:nandan
email of customer is:nandan@
Aadhar number of customer:123654789
Mobile number of customer is: 147852369
DOB of customer is: 19aug2002
Age of customer is: 20
```

```
 Current room Status
 Total Floors in a hotel:3
 Total rooms are:10 on each floor
 Choose the options to view information about each floor

 1.first floor
 2.Second floor
 3.Third floor1
 Rooms Booked on floor 1 are :
 5
 2
```

From my Industrial Training at ‘IFS India Mercantile Pvt. Ltd. Pune’, I was able to get a better understanding of how the software industry works and how effective it is. I enjoyed working on the data science projects. However, I still have a long way to go in understanding the more deeper concepts of the data science. From working on the technical aspects of a data science project to presenting my work in a way that everyone can understand, this has been a great experience!

Overall, I found the Data Structure and Object Oriented Programming internship experience to be positive, and I'm sure I will be able to use skills I learnt in my career later.

REFERENCE

[1] https://www.geeksforgeeks.org/c-plus-plus/?ref=shm

[2] https://www.javatpoint.com/cpp-tutorial

[3] https://www.programiz.com/dsa/quick-sort

[4] https://www.codingninjas.com/codestudio/library/infix-postfix-and-prefix-conversion