**DATA STRUCTURES AND ALGORITHMS**

**TOPIC: PHONE DIRECTORY USING LINKED LIST & BINARY SEARCH TRIE**

**ABSTRACT**

**SLOT-G2**

Phone directory also known as phone book/phone catalogue which stores the details of the customer. Telephone directory is one such thing which is used in our daily lives. The details include Name, D.O.B, phone number, address , pin code, state, operator, email id etc. We will try to make a program using C language for a telephone directory with certain features. Main feature of the telephone directory is to search for a number, enquire for balance available in the account, transferring balance etc.

Features of C programming that will be used to create the telephone directory are Data structures (linked list and binary search Trie).A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. A binary search Trie (BST), also known as an ordered binary Trie, is a node-based data structure in which each node has no more than two child nodes. The BST data structure is the basis for a number of highly efficient sorting and searching algorithms, and it can be used to construct more abstract data structures including sets, multi sets, and associative arrays.

Some advantages of making this program will be easy to use, reduced paper work. The program enhances the working strategies by supplanting the current manual framework with the PC based framework, so the response time is expected to be very less. We will compare the response time in both the cases.

The program will have a friendly user interface and will also provide accuracy. BST and linked list data structures is used to implement this phone directory.

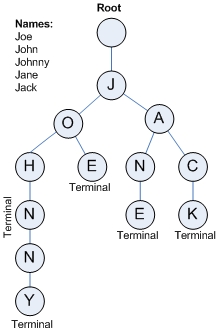
**Keywords**

**Introduction**

Trie is an ordered tree data structure that uses strings as keys. Unlike **Binary** **Trees**, Tries do not store keys associated with the node. The key is actually determined based on the position of the node on the tree. Any descendants of a node shares a common prefix of the key string associated with that node. Hence, trie is also called as Prefix Tree. The word "trie" comes from Re**trie**val, and it is pronounced as "try". To read more about Trie click here.

Since this data structure is a prefix tree, trie is commonly used in Dictionaries, Phone Directories and matching algorithms. Trie is best-suited for phone directory (any matching application for that matter) because it is very efficient in matching strings.

1. **BINARY TREE**



The code we have made are of BST and linked list . Number of lines in BST code are around 571 and linked list code is about 350 lines code.

Program execution time :

**CYCLOMATIC COMPLEXITY**

Cyclomatic complexness may be a software package metric (measurement), accustomed indicate the complexness of a program. it's a quantitative live of the quantity of linearly freelance ways through a program's ASCII text file. it absolutely was developed by Thomas J. McCabe, Sr. in 1976.  
  
Cyclomatic complexness is computed victimisation the management flow graph of the program: the nodes of the graph correspond to indivisible by teams of commands of a program, and a directed edge connects 2 nodes if the second command may well be dead directly once the primary command. Cyclomatic complexness may additionally be applied to individual functions, modules, strategies or categories at intervals a program.  
  
One testing strategy, referred to as basis path testing by McCabe United Nations agency 1st projected it, is to checkevery linearly freelance path through the program; during this case, the quantity of take a look at cases can equal the cyclomatic complexness of the program.

Halstead **quality** measures **square measure** **computer code** metrics introduced by Maurice Howard Halstead in 1977[1] as **a part of** his **writing** on establishing **AN** empirical science of **computer code** development. Halstead **created** the observation that metrics of the **computer code** **ought to** **replicate** the implementation or expression of algorithms **in numerous** languages, **however** be **freelance** of their execution on **a selected** platform. These metrics **square measure** **so** computed statically from the code.  
  
Halstead's goal was **to spot** measurable properties of **computer code**, **and therefore the** relations between them. **this is often** **kind of like** the identification of measurable properties of matter (like **the amount**, mass, and pressure of a gas) **and therefore the** relationships between them (analogous to the gas equation). **so** his metrics **are literally** not **simply** **quality** metrics.  
  
Coupling (computer programming)  
From Wikipedia, the free **cyclopedia**  
(Redirected from Coupling (computer science))  
In **computer code** engineering, coupling **is that the** degree of **mutuality** between **computer code** modules; a **live** of **however** closely connected **2** routines or modules are;[1] the strength of the relationships between modules.[2]  
  
Coupling **is typically** contrasted with cohesion. Low coupling **usually** correlates with high cohesion, and **contrariwise**. Low coupling **is commonly** **a symbol** of a well-structured **ADP system** and **an honest** **style**, and **once** combined with high cohesion, supports **the final** goals of high readability and maintainability.[citation needed]  
  
Types of coupling  
  
Conceptual model of coupling  
Coupling **are often** "low" (also "loose" and "weak") or "high" (also "tight" and "strong"). Some **sorts of** coupling, **so as** of highest to lowest coupling, **square measure** as follows:  
  
Procedural programming[edit]  
A module here refers to a **software system** of any kind, i.e. **a group** **of 1** or **additional** statements having **a reputation** and **ideally** its own set of variable names.  
  
Content coupling (high)  
Content coupling (also **called** Pathological coupling) **happens** **once** one module modifies or **depends** on **the interior** workings of another module (e.g., accessing **native** **information** of another module). **during this** **scenario**, a **modification** **within the** **means** the second module produces **information** (location, type, timing) **may additionallyneed** a **modification** **within the** dependent module.  
Common coupling  
Common coupling (also **called** **international** coupling) **happens** **once** **2** modules share **an equivalent** **internationalinformation** (e.g., **a worldwide** variable). **dynamical** the shared resource **would possibly** imply **dynamical** all the modules **victimisation** it.  
External coupling  
External coupling **happens** **once** **2** modules share **AN** **outwardly** **obligatory** **format**, communication protocol, or device interface. **this is often** **essentially** **associated with** the communication to external tools and devices.  
Control coupling  
Control coupling is one module **dominant** the flow of another, by passing it **info** on what **to try to to** (e.g., passing a what-to-do flag).  
Stamp coupling (Data-structured coupling)  
Stamp coupling **happens** **once** modules share a composite **arrangement** and use **solely** **elements** of it, **presumably** **completely different** **elements** (e.g., passing **an entire** record to a **perform** that **solely** **desires** one field of it).  
In this **scenario**, a modification **during a** field that a module **doesn't** **would like** **could** **cause** **dynamical** the **means** the module reads the record.  
Data coupling  
Data coupling **happens** **once** modules share **information** through, **as an example**, parameters. **every** **data point** is **AN** elementary piece, and these **square measure** **the sole** **information** shared (e.g., passing **AN** **number** to a **perform** that computes a **sq.** root).  
Message coupling (low)  
This is the loosest **variety of** coupling. It **are often** achieved by state decentralization (as in objects) and **part** communication **is completed** via parameters or message passing.  
No coupling  
Modules **don't** communicate **in the least** with **each other**.