**TRIES**

TO INSERT, DELETE, SEARCH AND DISPLAY WORDS USING TRIES

horizontal line

# 

# 

## INTRODUCTION

Trie is a data structure which is used to store collection of strings and makes searching of strings more easy. The term trie came from the word re**trie**val because trie data structure makes retrieval of stings from a collection of strings easy. Hence, they are widely used in many string search applications like auto-complete, text search etc.

## OBJECTIVE:

The goal of this project is to implement trie data structure which supports Insertion, Deletion and Search operations on words using arrays.

## IMPLEMENTATION:

Each node in the trie is a structure containing an array of structure pointers of size 26 (called **trienode)** representing 26 alphabets and an integer (called **endofword)** determining if it’s the end of the word.

Insertion is done by checking if there is a pointer pointing to the next alphabet corresponding to the alphabet in the word at every stage. If the pointer is pointing to NULL, then a new trienode is created and the process repeats. Once the entire word is inserted, then endofword for the last trienode that was used is made one.

Searching is also done along the same lines except there is no insertion happening but just traversing to find the word. We have to note that the word is only found if endofword of the last trienode compared is one.

Deletion is implemented with help of a stack. It is divided into two phases. In the first phase we have to check if the word is present to delete it. So this is done by comparing letter by letter and pushing it to a stack if its present until the end of word is reached. We then move on to the second phase where we delete each letter by popping from the stack.

## CODE:

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

struct stack

{

struct trienode \*m;

int index;

};

struct trienode

{

struct trienode\* child[26];

int endofword;

};

struct trienode \*getnode();

void insert(struct trienode \*, char \*);

void display(struct trienode\*);

void delete\_trie(struct trienode\*,char\*);

int search(struct trienode \*,char \*);

int check(struct trienode \*);

void push(struct trienode\*,int);

struct stack pop();

char word[100];

int length;

int top;

struct stack s[100];

int main()

{

struct trienode \*root;

char key[100];

int ch,k;

root=getnode();

while(1)

{

printf("\n1. Insert");

printf("\n2. Delete");

printf("\n3. Search");

printf("\n4. Display");

printf("\n5. Exit\n\n");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("Enter word\n");

scanf("%s",key);

insert(root,key);

break;

case 2: printf("Enter the word for deletion\n");

scanf("%s",key);

top=0;

delete\_trie(root,key);

break;

case 3: printf("Enter the word to search\n");

scanf("%s",key);

k=search(root,key);

if(k)

printf("\nWord found\n");

else

printf("\nNot found\n");

break;

case 4:length=0;

display(root);

printf("\n");

break;

case 5: exit(1);

}

}

}

struct trienode \*getnode()

{

int i;

struct trienode \*temp;

temp=(struct trienode\*)malloc(sizeof(struct trienode));

for(i=0;i<26;i++)

temp->child[i]=NULL;

temp->endofword=0;

return temp;

}

void insert(struct trienode \*root, char \*key)

{

struct trienode \*curr;

int i,index;

curr=root;

for(i=0;key[i]!='\0';i++)

{

index=key[i]-'a';

if(curr->child[index]==NULL)

curr->child[index]=getnode();

curr=curr->child[index];

}

curr->endofword=1;

}

void display(struct trienode \*curr)

{

int i,j;

for(i=0;i<26;i++)

{

if(curr->child[i]!=NULL)

{

word[length++]=i+97;

if(curr->child[i]->endofword==1)//if end of word

{

printf("\n");//print word

for(j=0;j<length;j++)

printf("%c",word[j]);

}

display(curr->child[i]);

}

}

length--;

return ;

}

int search(struct trienode \* root,char \*key)

{

int i,index;

struct trienode \*curr;

curr=root;

for(i=0;key[i]!='\0';i++)

{

index=key[i]-'a';

if(curr->child[index]==NULL)

return 0;

curr=curr->child[index];

}

if(curr->endofword==1)

return 1;

return 0;

}

void delete\_trie(struct trienode \*root,char \*key)

{

int i,j,index,k;

struct trienode\* curr,\*temp;

struct stack x;

curr=root;

for(i=0;key[i]!='\0';i++)

{

index=key[i]-'a';

if(curr->child[index]==NULL)

{

printf("Key not found\n");

return;

}

push(curr,index);

curr=curr->child[index];

}

curr->endofword=0;

push(curr,-1);

while(1)

{

x=pop();

if(x.index!=-1)

x.m->child[x.index]=NULL;

if(x.m==root)

break;

k=check(x.m);

if((k>=1)||(x.m->endofword==1))

break;

else

free(x.m);

}

return;

}

int check(struct trienode \*x)

{

int i;

int count=0;

for(i=0;i<26;i++)

{

if(x->child[i]!=NULL)

count++;

}

return count;

}

void push(struct trienode \*x,int k)

{

++top;

s[top].m=x;

s[top].index=k;

}

struct stack pop()

{

struct stack temp;

temp=s[top--];

return temp;

}