**COIMBATORE INSTITUTE OF TECHNOLOGY**

****

**MINI PROJECT PRESENTATION**

**TOPIC :** WORD AUTO COMPLETE

**TEAM MEMBERS :**

Arvinth Kumar.S (1831006)

Boobalaragavan.P(1831009)

Muthu Rahul(1831031)

Prakash.N(1831033)

Ruthsan.R(1831045)

WORD AUTO COMPLETE

**INTRODUCTION :-**

* The Word trie is an infix of the word “ RETRIEVAL “.
* Trie is nothing but a tree and each node in it contains the numbers

of pointers equal to the number of characters.

* The Tries can insert and find Strings in O(L) time , which is must faster than Hash tables and Binary Search Trees.

here L = length of a single word.

* A Binary Search Tree Which Stores each Word as a node Requires O(log(n)) time to search.

here n = no of nodes in BST.

**PROBLEM STATEMENT:-**

* To search for the names of the book stored in a trie structure.
* The book is then searched to display its details.
* Binary Search Tree data structure uses more time to search and for manipulation operations when compared to Trie data structure.
* So, we have used Trie data structure to store and retrieve Book names

**CONCEPT USED:-**

* Trie Data Structure
* Exceptional Handling
* Date Function
* Dat Files

**ALGORITHM ANALYSIS :-**

/ Trie node  
struct TrieNode  
{  
     struct TrieNode \*children[ALPHABET\_SIZE];

     // isEndOfWord is true if the node  
     // represents end of a word  
     bool isEndOfWord;  
};

Inserting a key into Trie is a simple approach. Every character of the input key is inserted as an individual Trie node. Note that the children is an array of pointers (or references) to next level trie nodes. The key character acts as an index into the array children. If the input key is new or an extension of the existing key, we need to construct non-existing nodes of the key, and mark end of the word for the last node. If the input key is a prefix of the existing key in Trie, we simply mark the last node of the key as the end of a word. The key length determines Trie depth.

**SOURCE CODE:-**

#include<bits/stdc++.h>

#include<fstream>

#include<iomanip>

#include<iostream>

#include<process.h>

#include<string.h>

#include<stdlib.h>

#include<conio.h>

using namespace std;

#define ALPHABET\_SIZE (26)

#define CHAR\_TO\_INDEX(c) ((int)c - (int)'a');

struct TrieNode

{

struct TrieNode \*children[ALPHABET\_SIZE];

bool isWordEnd;

};

struct TrieNode \*getNode(void)

{

struct TrieNode \*pNode = new TrieNode;

pNode->isWordEnd = false;

for (int i = 0; i < ALPHABET\_SIZE; i++)

pNode->children[i] = NULL;

return pNode;

}

class book

{

char bno[6];

char aname[20];

public:

char bname[50];

char bookname[50];

void create\_book()

{

cout<<"\nNEW BOOK ENTRY...\n";

cout<<"\nEnter The book no.";

cin>>bno;

cout<<"\n\nEnter The Name of The Book ";

cin>>bname;

strcpy(bookname,bname);

cout<<"\n\nEnter The Author's Name ";

cin>>aname;

cout<<"\n\n\nBook Created..";

}

void show\_book()

{

cout<<"\nBook no. : "<<bno;

cout<<"\nBook Name : ";

cout<<bname;

cout<<"Author Name : ";

cout<<aname;

}

void modify\_book()

{

cout<<"\nBook no. : "<<bno;

cout<<"\nModify Book Name : ";

cin>>bname;

cout<<"\nModify Author's Name of Book : ";

cin>>aname;

}

char\* retbno()

{

return bno;

}

void report()

{cout<<bno<<setw(30)<<bname<<setw(30)<<aname<<endl;}

}; //class ends here

class student

{

char admno[6];

char name[20];

char stbno[6];

int token;

public:

void create\_student()

{

system("cls");

cout<<"\nNEW STUDENT ENTRY...\n";

cout<<"\nEnter The admission no.";

cin>>admno;

cout<<"\n\nEnter The Name of The Student ";

cin>>name;

token=0;

stbno[0]='/0';

cout<<"\n\nStudent Record Created..";

}

void show\_student()

{

cout<<"\nAdmission no. : "<<admno;

cout<<"\nStudent Name : ";

cout<<name;

cout<<"\nNo of Book issued : "<<token;

if(token==1)

cout<<"\nBook No "<<stbno;

}

void modify\_student()

{

cout<<"\nAdmission no. : "<<admno;

cout<<"\nModify Student Name : ";

cin>>name;

}

char\* retadmno()

{

return admno;

}

char\* retstbno()

{

return stbno;

}

int rettoken()

{

return token;

}

void addtoken()

{token=1;}

void resettoken()

{token=0;}

void getstbno(char t[])

{

strcpy(stbno,t);

}

void report()

{cout<<"\t"<<admno<<setw(20)<<name<<setw(10)<<token<<endl;}

}; //class ends here

fstream fp,fp1;

book bk;

student st;

void write\_book()

{

char ch;

fp.open("book.dat",ios::out|ios::app);

fp1.open("bookname.dat",ios::out|ios::app);

do

{

system("cls");

bk.create\_book();

fp.write((char\*)&bk,sizeof(book));

fp1.write((char\*)&bk.bookname,sizeof(bk.bookname));

fp1<<"\n";

cout<<"\n\nDo you want to add more record..(y/n?)";

cin>>ch;

}while(ch=='y'||ch=='Y');

fp.close();

fp1.close();

}

void write\_student()

{

char ch;

fp.open("student.dat",ios::out|ios::app);

do

{

st.create\_student();

fp.write((char\*)&st,sizeof(student));

cout<<"\n\ndo you want to add more record..(y/n?)";

cin>>ch;

}while(ch=='y'||ch=='Y');

fp.close();

}

void display\_spb(char n[])

{

cout<<"\nBOOK DETAILS\n";

int flag=0;

fp.open("book.dat",ios::in);

while(fp.read((char\*)&bk,sizeof(book)))

{

if(strcmp(bk.retbno(),n)==0)

{

bk.show\_book();

flag=1;

}

}

fp.close();

if(flag==0)

cout<<"\n\nBook does not exist";

getch();

}

void display\_sps(char n[])

{

cout<<"\nSTUDENT DETAILS\n";

int flag=0;

fp.open("student.dat",ios::in);

while(fp.read((char\*)&st,sizeof(student)))

{

if((strcmpi(st.retadmno(),n)==0))

{

st.show\_student();

flag=1;

}

}

fp.close();

if(flag==0)

cout<<"\n\nStudent does not exist";

getch();

}

void delete\_student()

{

char n[6];

int flag=0;

system("cls");

cout<<"\n\n\n\tDELETE STUDENT...";

cout<<"\n\nEnter The admission no. of the Student You Want To Delete : ";

cin>>n;

fp.open("student.dat",ios::in|ios::out);

fstream fp2;

fp2.open("Temp.dat",ios::out);

fp.seekg(0,ios::beg);

while(fp.read((char\*)&st,sizeof(student)))

{

if(strcmpi(st.retadmno(),n)!=0)

fp2.write((char\*)&st,sizeof(student));

else

flag=1;

}

fp2.close();

fp.close();

remove("student.dat");

rename("Temp.dat","student.dat");

if(flag==1)

cout<<"\n\n\tRecord Deleted ..";

else

cout<<"\n\nRecord not found";

getch();

}

void delete\_book()

{

char n[6];

system("cls");

cout<<"\n\n\n\tDELETE BOOK ...";

cout<<"\n\nEnter The Book no. of the Book You Want To Delete : ";

cin>>n;

fp.open("book.dat",ios::in|ios::out);

fstream fp2;

fp2.open("Temp.dat",ios::out);

fp.seekg(0,ios::beg);

while(fp.read((char\*)&bk,sizeof(book)))

{

if(strcmpi(bk.retbno(),n)!=0)

{

fp2.write((char\*)&bk,sizeof(book));

}

}

fp2.close();

fp.close();

remove("book.dat");

rename("Temp.dat","book.dat");

cout<<"\n\n\tRecord Deleted ..";

getch();

}

void display\_alls()

{

system("cls");

fp.open("student.dat",ios::in);

if(!fp)

{

cout<<"ERROR!!! FILE COULD NOT BE OPENED ";

getch();

return;

}

cout<<"\n\n\t\tSTUDENT LIST\n\n";

cout<<"==================================================================\n";

cout<<"Admission No."<<setw(10)<<" Name"<<setw(20)<<"Book Issued\n";

cout<<"==================================================================\n";

while(fp.read((char\*)&st,sizeof(student)))

{

st.report();

}

fp.close();

getch();

}

void display\_allb()

{

system("cls");

fp.open("book.dat",ios::in);

if(!fp)

{

cout<<"ERROR!!! FILE COULD NOT BE OPEN ";

// getch();

return;

}

cout<<"\n\n\t\tBook LIST\n\n";

cout<<"=========================================================================\n";

cout<<"Book Number"<<setw(20)<<"Book Name"<<setw(25)<<"Author\n";

cout<<"=========================================================================\n";

while(fp.read((char\*)&bk,sizeof(book)))

{

bk.report();

}

fp.close();

getch();

}

void book\_issue()

{

char sn[6],bn[6];

int found=0,flag=0;

system("cls");

cout<<"\n\nBOOK ISSUE ...";

cout<<"\n\n\tEnter The student's admission no.";

cin>>sn;

fp.open("student.dat",ios::in|ios::out);

fp1.open("book.dat",ios::in|ios::out);

while(fp.read((char\*)&st,sizeof(student)) && found==0)

{

if(strcmpi(st.retadmno(),sn)==0)

{

found=1;

if(st.rettoken()==0)

{

cout<<"\n\n\tEnter the book no. ";

cin>>bn;

while(fp1.read((char\*)&bk,sizeof(book))&& flag==0)

{

if(strcmpi(bk.retbno(),bn)==0)

{

bk.show\_book();

flag=1;

st.addtoken();

st.getstbno(bk.retbno());

int pos=-1\*sizeof(st);

fp.seekp(pos,ios::cur);

fp.write((char\*)&st,sizeof(student));

cout<<"\n\n\t Book issued successfully\n\nPlease Note: Write the current date in backside of your book and submit within 15 days fine Rs. 1 for each day after 15 days period";

}

}

if(flag==0)

cout<<"Book no does not exist";

}

else

cout<<"You have not returned the last book ";

}

}

if(found==0)

cout<<"Student record not exist...";

getch();

fp.close();

fp1.close();

}

void insert(struct TrieNode \*root, const string key)

{

struct TrieNode \*pCrawl = root;

for (int level = 0; level < key.length(); level++)

{

int index = CHAR\_TO\_INDEX(key[level]);

if (!pCrawl->children[index])

pCrawl->children[index] = getNode();

pCrawl = pCrawl->children[index];

}

pCrawl->isWordEnd = true;

}

bool search(struct TrieNode \*root, const string key)

{

int length = key.length();

struct TrieNode \*pCrawl = root;

for (int level = 0; level < length; level++)

{

int index = CHAR\_TO\_INDEX(key[level]);

if (!pCrawl->children[index])

return false;

pCrawl = pCrawl->children[index];

}

return (pCrawl != NULL && pCrawl->isWordEnd);

}

bool isLastNode(struct TrieNode\* root)

{

for (int i = 0; i < ALPHABET\_SIZE; i++)

if (root->children[i])

return 0;

return 1;

}

void suggestionsRec(struct TrieNode\* root, string currPrefix)

{

if (root->isWordEnd)

cout << currPrefix<<endl;

if (isLastNode(root))

return;

for (int i = 0; i < ALPHABET\_SIZE; i++)

{

if (root->children[i])

{

currPrefix.push\_back(97 + i);

suggestionsRec(root->children[i], currPrefix);

currPrefix.pop\_back(); //The missing Statement

}

}

}

int printAutoSuggestions(TrieNode\* root, const string query)

{

struct TrieNode\* pCrawl = root;

int level;

int n = query.length();

for (level = 0; level < n; level++)

{

int index = CHAR\_TO\_INDEX(query[level]);

if (!pCrawl->children[index])

return 0;

pCrawl = pCrawl->children[index];

}

bool isWord = (pCrawl->isWordEnd == true);

bool isLast = isLastNode(pCrawl);

if (isWord && isLast)

{

cout << query << endl;

return -1;

}

if (!isLast)

{

string prefix = query;

suggestionsRec(pCrawl, prefix);

return 1;

}

}

void searchtrie()

{

struct TrieNode\* root = getNode();

fstream file1;

string t, q, filename;

string word;//[50];

filename = "bookname1.dat";

file1.open(filename.c\_str());

while (file1.good())

{

getline(file1,word,'\n');

cout << word << endl;

insert(root, word);

}

string c,input;

cout<<"\nEnter the Book Name to Search : ";

cin>>input;

int comp = printAutoSuggestions(root, input);

if (comp == -1)

cout << "No Books found with this entered prefix\n";

else if (comp == 0)

cout << "No Books found with this entered prefix\n";

}

void book\_deposit()

{

char sn[6],bn[6];

int found=0,flag=0,day,fine;

system("cls");

cout<<"\n\nBOOK DEPOSIT ...";

cout<<"\n\n\tEnter The student's admission no.";

cin>>sn;

fp.open("student.dat",ios::in|ios::out);

fp1.open("book.dat",ios::in|ios::out);

while(fp.read((char\*)&st,sizeof(student)) && found==0)

{

if(strcmpi(st.retadmno(),sn)==0)

{

found=1;

if(st.rettoken()==1)

{

while(fp1.read((char\*)&bk,sizeof(book))&& flag==0)

{

if(strcmpi(bk.retbno(),st.retstbno())==0)

{

bk.show\_book();

flag=1;

cout<<"\n\nBook deposited in no. of days";

cin>>day;

if(day>15)

{

fine=(day-15)\*1;

cout<<"\n\nFine has to deposited Rs. "<<fine;

}

st.resettoken();

int pos=-1\*sizeof(st);

fp.seekp(pos,ios::cur);

fp.write((char\*)&st,sizeof(student));

cout<<"\n\n\t Book deposited successfully";

}

}

if(flag==0)

cout<<"Book no does not exist";

}

else

cout<<"No book is issued..please check!!";

}

}

if(found==0)

cout<<"Student record not exist...";

getch();

fp.close();

fp1.close();

}

void admin\_menu()

{

system("cls");

int ch2;

cout<<"\n\n\n\tLIBRARIAN MENU";

cout<<"\n\n\t1.CREATE STUDENT RECORD";

cout<<"\n\n\t2.DISPLAY ALL STUDENTS RECORD";

cout<<"\n\n\t3.DISPLAY SPECIFIC STUDENT RECORD ";

cout<<"\n\n\t4.DELETE STUDENT RECORD";

cout<<"\n\n\t5.CREATE BOOK ";

cout<<"\n\n\t6.DISPLAY ALL BOOKS ";

cout<<"\n\n\t7.DISPLAY SPECIFIC BOOK ";

cout<<"\n\n\t8.DELETE BOOK ";

cout<<"\n\n\t9.BACK TO MAIN MENU";

cout<<"\n\n\tPlease Enter Your Choice (1-9) ";

cin>>ch2;

switch(ch2)

{

case 1: system("cls");

write\_student();break;

case 2: display\_alls();break;

case 3:

char num[6];

system("cls");

cout<<"\n\n\tPlease Enter The Admission No. ";

cin>>num;

display\_sps(num);

break;

case 4: delete\_student();break;

case 5: system("cls");

write\_book();break;

case 6: display\_allb();break;

case 7: {

char num[6];

system("cls");

cout<<"\n\n\tPlease Enter The book No. ";

cin>>num;

display\_spb(num);

break;

}

case 8: delete\_book();break;

case 9: return;

default:cout<<"\a";

}

admin\_menu();

}

int main()

{

char ch;

do

{

system("cls");

cout<<"\n\n\n\tMAIN MENU";

cout<<"\n\n\t01. BOOK ISSUE";

cout<<"\n\n\t02. BOOK RETURN";

cout<<"\n\n\t03. SEARCH FOR BOOKS";

cout<<"\n\n\t04. LIBRARIAN";

cout<<"\n\n\t05. EXIT";

cout<<"\n\n\tPlease Select Your Option (1-5) ";

cin >> ch;

switch(ch)

{

case '3':searchtrie();

getch();

break;

case '1'://system("cls");

book\_issue();

break;

case '2'://system("cls");

book\_deposit();

break;

case '4':system("cls");

admin\_menu();

break;

case '5':exit(0);

default :cout<<"\a"; }

}while(ch!='5');

cout<<sizeof(TrieNode);

return 0;

}

**RESULT DISCUSSION :-**

THE Above Code to search a word using WORD AUTO COMPLETE using TRIE DATA STRUCTURE is Implemented in Library Management System is executed and it takes a very good time and space complexities. It satisfies the aim and provides Exact Output.

**OUTPUT :-**



