**Assignment number:3**

**Subject: ADVANCED DATA STRUCTURES LAB**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**PROBLEM STATEMENT:**

A Dictionary stores keywords & its meanings. Provide facility for adding new keywords,

deleting keywords, updating values of any entry. Provide facility to display whole data sorted

in ascending/

Descending order. Also find how many maximum comparisons may require for

finding any keyword. Use Binary Search Tree for implementation

**CODE:**

#include<iostream>

#include<string.h>

using namespace std;

class node

{

char data[10];

char mean[100];

node \*left,\*right;

friend class bst;

};

class bst

{

node \*new1,\*temp,\*t,\*root;

char mea[100];

public:

node \* getroot();

int create();

int display(node \*);

int print();

int insert();

int call();

void updation(node \*,char\*);

node \* deletion(node \*,char \*);

int delet();

};

node \* bst::getroot()

{

return root;

}

int bst::create()

{

int flag=1;

char ans;

do

{

new1=new node;

new1->left=NULL;

new1->right=NULL;

cout<<"\nenter data\n";

cin>>new1->data;

cout<<"\nenter the meaning of the data\n";

ws(cin);

cin.getline(new1->mean,100);

if(flag==1)

{

root=new1;

//temp=new1;

root->right=root->left=NULL;

flag=0;

}

else

{

temp=root;

while(1)

{

if(strcmp(temp->data,new1->data)==0 )

{

cout<<"\nduplicate entry not allowed\n";

break;

}

if(strcmp(temp->data,new1->data)>0 )

{

//cout<<temp->data;

if(temp->left==NULL)

{

temp->left=new1;

temp=new1;

temp->left=temp->right=NULL;

break;

}

else

{

temp=temp->left;

}

}

if(strcmp(temp->data,new1->data)<0)

{

if(temp->right==NULL)

{

temp->right=new1;

temp=new1;

temp->left=temp->right=NULL;

break;

}

else

{

temp=temp->right;

}

}

}

}

cout<<"\ndo you want to enter a new word? ";

cin>>ans;

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

}

int bst::display(node \*root)

{

if(root!=NULL)

{

display(root->left);

cout<<root->data<<"\t\t"<<root->mean<<endl;

display(root->right);

}

}

int bst::print()

{

cout<<"\nthe data in ascending order \n";

display(root);

}

int bst::insert()

{

new1=new node;

new1->left=NULL;

new1->right=NULL;

cout<<"\nenter new data to be inserted\n";

cin>>new1->data;

cout<<"\nenter new meaning to be inserted\n";

ws(cin);

cin.getline(new1->mean,100);

temp=root;

while(1)

{

if(strcmp(new1->data,temp->data)<0 && temp->left==NULL)

{

temp->left=new1;

new1->left=new1->right=NULL;

break;

}

else if(strcmp(new1->data,temp->data)<0 && temp->left!=NULL)

{

temp=temp->left;

}

else if(strcmp(new1->data,temp->data)>0 && temp->right==NULL)

{

temp->right=new1;

new1->left=new1->right=NULL;

break;

}

else if(strcmp(new1->data,temp->data)>0 && temp->right!=NULL)

{

temp=temp->right;

}

else

{

cout<<"\nduplicate entry not allowed\n";

break;

}

}

}

int bst::call()

{

cout<<"\nenter the meaning to be updated\n";

ws(cin);

cin.getline(mea,100);

}

void bst::updation(node \*root,char nme[100])

{

if(root!=NULL)

{

if(strcmp(root->data,nme)==0)

{

call();

strcpy(root->mean,mea);

}

updation(root->left,nme);

updation(root->right,nme);

}

}

int bst::delet()

{

char del[100];

cout<<"\nenter the name which has to be deleted\n";

ws(cin);

cin.getline(del,100);

deletion(root,del);

}

node \* bst::deletion(node \*t,char \*del)

{

if(t==NULL)

{

cout<<"\nelement not found in the dictionary\n";

return (t);

}

if(strcmp(t->data,del)>0)

{

t->left=deletion(t->left,del);

return (t);

}

if(strcmp(t->data,del)<0)

{

t->right=deletion(t->right,del);

return (t);

}

if(t->left==NULL && t->right==NULL)

{

temp=t;

free(temp);

return NULL;

}

if(t->left==NULL)

{

temp=t;

t=t->right;

free(temp);

return (t);

}

if(t->right==NULL)

{

temp=t;

t=t->left;

free(temp);

return (t);

}

}

int main()

{

bst obj;

int ans;

char ch;

obj.create();

do

{

cout<<"\n1) PRINT\n2)INSERT NEW MEMBER\n3)UPDATE MEANING \n4)DELETE DATA\n5)exit\n";

cin>>ans;

switch(ans)

{

case 1:obj.print(); break;

case 2:obj.insert(); break;

case 3:

{

char nme[100];

char mn[100];

cout<<"\nenter the name whose meaning you want to update\n";

ws(cin);

cin.getline(nme,100);

obj.updation(obj.getroot(),nme);break;

}

case 4:obj.delet();obj.print();break;

case 5:break;

}

cout<<"\ndo you want to continue?\n";

cin>>ch;

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

return 0;

}

**OUTPUT:**

enter data

a

enter the meaning of the data

apple

do you want to enter a new word? y

enter data

b

enter the meaning of the data

ball

do you want to enter a new word? y

enter data

c

enter the meaning of the data

cat

do you want to enter a new word? y

enter data

d

enter the meaning of the data

dog

do you want to enter a new word? y

enter data

1

enter the meaning of the data

number

do you want to enter a new word? y

enter data

2

enter the meaning of the data

number

do you want to enter a new word? y

enter data

3

enter the meaning of the data

number

do you want to enter a new word? n

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

1

the data in ascending order

1 number

2 number

3 number

a apple

b ball

c cat

d dog

do you want to continue?

y

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

2

enter new data to be inserted

0

enter new meaning to be inserted

new number

do you want to continue?

y

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

1

the data in ascending order

0 new number

1 number

2 number

3 number

a apple

b ball

c cat

d dog

do you want to continue?

y

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

3

enter the name whose meaning you want to update

a

enter the meaning to be updated

aeroplane

do you want to continue?

y

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

4

enter the name which has to be deleted

c

the data in ascending order

0 new number

1 number

2 number

3 number

a aeroplane

b ball

d dog

do you want to continue?

y

1) PRINT

2)INSERT NEW MEMBER

3)UPDATE MEANING

4)DELETE DATA

5)exit

5

do you want to continue?

n