Advanced Data Structures Assignment

Name : Shrirang R Mhalgi

Roll number : 222006

Class : SE B

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;

int flag=1,i=0;

char key[10];

class node

{

char keyw[10];

char m[100];

node \*right;

node \*left;

friend class dicn;

};

class dicn

{

node \*root,\*temp,\*curr,\*New,\*t;

public:

int create();

void display(node \*);

node \* search(node \*);

void update(node \*);

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

node \* ret\_r(){

return root;

}

node \* findmax(node \* temp){

while(temp->right!=NULL)

{

temp=temp->right;

}

return temp;

}

};

int dicn :: create(){

cout<<endl;

New= new node;

cout<<"Enter keyword : ";

ws(cin);

cin.getline(New->keyw,10);

cout<<"Enter meaning : ";

cin.getline(New->m,100);

New->right=NULL;

New->left=NULL;

if(flag==1)

{

root=New;

temp=root;

flag=0;

}

else

{

temp=root;

while(1)

{

if(strcmp(temp->keyw,New->keyw)>0)

{

if(temp->left==NULL)

{

temp->left=New;

temp=New;

break;

}

else

{

temp=temp->left;

}

}

if(strcmp(temp->keyw,New->keyw)<0)

{

if(temp->right==NULL)

{

temp->right=New;

temp=New;

break;

}

else

{

temp=temp->right;

}

}

}

}

}

void dicn :: display(node \*root){

if(root!=NULL)

{

display(root->left);

cout<<root->keyw<<"\t\t"<<root->m<<endl;

display(root->right);

}

}

node \* dicn :: search(node \*root){

if(root!=NULL)

{

if(strcmp(key,root->keyw)== 0)

return root;

else if(strcmp(key,root->keyw)>0)

search(root->right);

else if(strcmp(key,root->keyw)<0)

search(root->left);

else return NULL;

}

else return NULL;

}

void dicn :: update(node \*temp){

//cout<<temp->keyw;

cout<<"\nUpdate meaning : ";

//ws(cin);

cin.getline(temp->m,100);

}

node \* dicn :: del(node \*temp, char key[]){

node \*p;

char temp1[10];

if(temp==NULL)

return root;

if(strcmp(key,temp->keyw)<0)

temp->left=del(temp->left,key);

else if(strcmp(key,temp->keyw)>0)

temp->right=del(temp->right,key);

else

{

if(temp->left==NULL)

{

p=temp->right;

delete temp;

return p;

}

else if(temp->right==NULL)

{

p=temp->left;

delete temp;

return p;

}

else

{

p=findmax(temp->left);

for(i=0;i<10;i++)

temp->keyw[i]=p->keyw[i];

for(i=0;i<100;i++)

temp->m[i]=p->m[i];

temp->left=del(temp->left,p->keyw);

}

}

return temp;

}

int main(){

dicn ob;

int ans;

do{

cout<<"\nEnter your choice\n";

cout<<"1: Creation of dictionary\n";

cout<<"2: Display in ascending order\n";

cout<<"3: Update a keyword\n";

cout<<"4: Add new keyword\n";

cout<<"5: Deletion\n";

cout<<"0: Exit\n";

cin>>ans;

switch(ans){

case 1: int c;

cout<<"\nEnter number of words to be stored : ";

cin>>c;

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

ob.create();

break;

case 2: cout<<"\n \*\*\*\* DICTIONARY \*\*\*\*\n";

ob.display(ob.ret\_r());

break;

case 3: cout<<"\nEnter keyword which is to be updated\n";

ws(cin);

cin.getline(key,10);

ob.update(ob.search(ob.ret\_r()));

break;

case 4: ob.create();

break;

case 5: cout<<"\nEnter keyword which is to be deleted\n";

ws(cin);

cin.getline(key,10);

ob.del(ob.ret\_r(),key);

break;

}

}while(ans!=0);

}

**Output:**

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

1

Enter number of words to be stored : 3

Enter keyword : apple

Enter meaning : fruit

Enter keyword : sun

Enter meaning : star

Enter keyword : horse

Enter meaning : animal

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

2

\*\*\*\* DICTIONARY \*\*\*\*

apple fruit

horse animal

sun star

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

4

Enter keyword : doll

Enter meaning : toy

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

2

\*\*\*\* DICTIONARY \*\*\*\*

apple fruit

doll toy

horse animal

sun star

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

3

Enter keyword which is to be updated

sun

Update meaning : big star

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

2

\*\*\*\* DICTIONARY \*\*\*\*

apple fruit

doll toy

horse animal

sun big star

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

5

Enter keyword which is to be deleted

horse

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

2

\*\*\*\* DICTIONARY \*\*\*\*

apple fruit

doll toy

sun big star

Enter your choice

1: Creation of dictionary

2: Display in ascending order

3: Update a keyword

4: Add new keyword

5: Deletion

0: Exit

0