**Practical GroupD\_19**

**Aim:**

A Dictionary stores keywords and 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 Height balance tree and find the complexity for finding a keyword

**Code:**

#include"iostream"

#include<string.h>

using namespace std;

//////////////////////////////////////////////////////////////

// //

// Name : Vivek S. Sharma //

// Title : BST //

// //

// //

//////////////////////////////////////////////////////////////

typedef struct node

{

char k[20];

char m[20];

class node \*left;

class node \* right;

}node;

class dict

{

public:

node \*root;

void create();

void disp(node \*);

void insert(node \* root,node \*temp);

int search(node \*,char []);

int update(node \*,char []);

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

node \* min(node \*);

};

void dict :: create()

{

class node \*temp;

int ch;

do

{

temp = new node;

cout<<"\nEnter Keyword:";

cin>>temp->k;

cout<<"\nEnter Meaning:";

cin>>temp->m;

temp->left = NULL;

temp->right = NULL;

if(root == NULL)

{

root = temp;

}

else

{

insert(root, temp);

}

cout<<"\nDo u want to add more (y=1/n=0):";

cin>>ch;

}

while(ch == 1);

}

void dict :: insert(node \* root,node \*temp)

{

if(strcmp (temp->k, root->k) < 0 )

{

if(root->left == NULL)

root->left = temp;

else

insert(root->left,temp);

}

else

{ if(root->right == NULL)

root->right = temp;

else

insert(root->right,temp);

}

}

void dict:: disp(node \* root)

{

if( root != NULL)

{

disp(root->left);

cout<<"\n Key Word :"<<root->k;

cout<<"\t Meaning :"<<root->m;

disp(root->right);

}

}

int dict :: search(node \* root,char k[20])

{

int c=0;

while(root != NULL)

{

c++;

if(strcmp (k,root->k) == 0)

{

cout<<"\nNo of Comparisons:"<<c;

return 1;

}

if(strcmp (k, root->k) < 0)

root = root->left;

if(strcmp (k, root->k) > 0)

root = root->right;

}

return -1;

}

int dict :: update(node \* root,char k[20])

{

while(root != NULL)

{

if(strcmp (k,root->k) == 0)

{

cout<<"\nEnter New Meaning ofKeyword"<<root->k;

cin>>root->m;

return 1;

}

if(strcmp (k, root->k) < 0)

root = root->left;

if(strcmp (k, root->k) > 0)

root = root->right;

}

return -1;

}

node\* dict :: del(node \* root,char k[20])

{

node \*temp;

if(root == NULL)

{

cout<<"\nElement No Found";

return root;

}

if (strcmp(k,root->k) < 0)

{

root->left = del(root->left, k);

return root;

}

if (strcmp(k,root->k) > 0)

{

root->right = del(root->right, k);

return root;

}

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

{

temp = root;

delete temp;

return NULL;

}

if(root->right==NULL)

{

temp = root;

root = root->left;

delete temp;

return root;

}

else if(root->left==NULL)

{

temp = root;

root = root->right;

delete temp;

return root;

}

temp = min(root->right);

strcpy(root->k,temp->k);

root->right = del(root->right, temp->k);

return root;

}

node \* dict :: min(node \*q)

{

while(q->left != NULL)

{

q = q->left;

}

return q;

}

int main()

{

int ch;

dict d;

d.root = NULL;

do

{

cout<<"\nMenu\n1.Create\n2.Disp\n3.Search\n4.Update\n5.Delete\nEnter Ur CH:";

cin>>ch;

switch(ch)

{

case 1: d.create();

break;

case 2: if(d.root == NULL)

{

cout<<"\nNo any Keyword";

}

else

{

d.disp(d.root);

}

break;

case 3: if(d.root == NULL)

{

cout<<"\nDictionary is Empty. First add keywords then try again ";

}

else

{

cout<<"\nEnter Keyword which u want to search:";

char k[20];

cin>>k;

if( d.search(d.root,k) == 1)

cout<<"\nKeyword Found";

else

cout<<"\nKeyword Not Found";

}

break;

case 4:

if(d.root == NULL)

{

cout<<"\nDictionary is Empty. First add keywords then try again ";

}

else

{

cout<<"\nEnter Keyword which meaning want to update:";

char k[20];

cin>>k;

if(d.update(d.root,k) == 1)

cout<<"\nMeaning Updated";

else

cout<<"\nMeaning Not Found";

}

break;

case 5:

if(d.root == NULL)

{

cout<<"\nDictionary is Empty. First add keywords then try again ";

}

else

{

cout<<"\nEnter Keyword which u want to delete:";

char k[20];

cin>>k;

if(d.root == NULL)

{

cout<<"\nNo any Keyword";

}

else

{

d.root = d.del(d.root,k);

}

}

}

}

while(ch<=5);

return 0;

}