**Code for Program to simulate a dictionary using linked list in C Programming**

#include <stdio.h>

#include <conio.h>

#include <malloc.h>

#include <stdlib.h>

#include <string.h>

#include <ctype.h>

struct node

{

char data [ 20 ] ;

char m [ 5 ] [ 20 ] ;

int mcount ;

struct node \* link ;

} ;

struct node \* dic [ 26 ] ;

void add ( char \* ) ;

int search ( char \* ) ;

void show( ) ;

void deldic( ) ;

void main( )

{

char word [ 20 ] , ch ;

int i ;

clrscr( ) ;

while ( 1 )

{

clrscr( ) ;

printf ( "\n\t\tDictionary\n" ) ;

printf ( "\n\t\t1.Add Word.\n" ) ;

printf ( "\t\t2.Search Word.\n" ) ;

printf ( "\t\t3.Show Dictionary.\n" ) ;

printf ( "\t\t0.Exit." ) ;

printf ( "\n\n\t\tYour Choice ") ;

scanf ( "%d", &ch ) ;

switch ( ch )

{

case 1 :

printf ( "\nEnter any word : " ) ;

fflush ( stdin ) ;

gets ( word ) ;

add ( word ) ;

break ;

case 2 :

printf ( "\nEnter the word to search : " ) ;

fflush ( stdin ) ;

gets ( word ) ;

i = search ( word ) ;

if ( ! i )

printf ( "Word does not exists." ) ;

getch( ) ;

break ;

case 3 :

show( ) ;

getch( ) ;

break ;

case 0 :

deldic( ) ;

exit ( 0 ) ;

default :

printf ( "\nWrong Choice" ) ;

}

}

}

void add ( char \* str )

{

int i, j = toupper ( str [ 0 ] ) - 65 ;

struct node \* r, \* temp = dic [ j ], \* q ;

char mean [ 5 ] [ 20 ], ch = 'y' ;

i = search ( str ) ;

if ( i )

{

printf ( "\nWord already exists." ) ;

getch( ) ;

return ;

}

q = ( struct node \* ) malloc ( sizeof ( struct node ) ) ;

strcpy ( q -> data, str ) ;

q -> link = NULL ;

for ( i = 0 ; tolower ( ch ) == 'y' && i < 5 ; i++ )

{

fflush ( stdin ) ;

printf ( "\n\nEnter the meaning(s) : " ) ;

gets ( mean [ i ] ) ;

strcpy ( q -> m [ i ] , mean [ i ] ) ;

if ( i != 4 )

printf ( "\nAdd more meanings (y/n) " ) ;

else

printf ( "You cannot enter more than 5 meanings." ) ;

fflush ( stdin ) ;

ch = getche( ) ;

}

q -> mcount = i ;

if ( dic [ j ] == NULL || strcmp ( dic [ j ] -> data, str ) > 0 )

{

r = dic [ j ] ;

dic [ j ] = q ;

q -> link = r ;

return ;

}

else

{

while ( temp != NULL )

{

if ( ( strcmp ( temp -> data, str ) < 0 ) && ( ( strcmp ( temp -> link -> data, str ) > 0 ) ||

temp -> link == NULL ) )

{

q -> link = temp -> link ;

temp -> link = q ;

return ;

}

temp = temp -> link ;

}

}

}

int search ( char \*str )

{

struct node \*n ;

char temp1 [ 20 ] ;

char temp2 [ 20 ] ;

int i ;

n = dic [ toupper ( str [ 0 ] ) - 65 ] ;

strcpy ( temp2, str ) ;

strupr ( temp2 ) ;

while ( n != NULL )

{

strcpy ( temp1, n -> data ) ;

if ( strcmp ( strupr ( temp1 ), temp2 ) == 0 )

{

printf ( "\n%s\t\t%s", n -> data, n -> m [ 0 ] ) ;

for ( i = 1 ; i < n -> mcount ; i++ )

printf ( "\n\t\t%s", n -> m [ i ] ) ;

return 1 ;

}

n = n -> link ;

}

return 0 ;

}

void show( )

{

struct node \*n ;

int i, j ;

printf ( "Word\t\tMeaning\n" ) ;

for ( i = 0 ; i <= 30 ; i++ )

printf ( "-" ) ;

for ( i = 0 ; i <= 25 ; i++ )

{

n = dic [ i ] ;

while ( n != NULL )

{

printf ( "\n%s\t\t%s", n -> data, n -> m [ 0 ] ) ;

for ( j = 1 ; j < n -> mcount ; j++ )

printf ( "\n\t\t%s", n -> m [ j ] ) ;

n = n -> link ;

}

}

}

void deldic( )

{

struct node \*n, \*t ;

int i ;

for ( i = 0 ; i <= 25 ; i++ )

{

n = dic [ i ] ;

while ( n != NULL )

{

t = n -> link ;

free ( n ) ;

n = t ;

}

}

}