//**C PROGRAM TO ADD DELETE NODES FROM ADJACENCY LIST**

|  |  |  |
| --- | --- | --- |
| #include<stdio.h> struct edge; struct node {  struct node \*next; char name; struct edge \*adjacency; }\*start = NULL; struct edge { char dest; struct edge \*link; }; struct node \*find( char item ); main() { int choice; char node, origin, destin; while ( 1 ) { printf( "1.Insert a node\n" ); printf( "2.Insert an edge\n" ); printf( "3.Delete a node\n" ); printf( "4.Delete an edge\n" ); printf( "5.Display\n" ); printf( "6.Exit\n" ); printf( "Enter your choice : " ); scanf( "%d", &choice ); switch ( choice ) { case 1: printf( "Enter a node to be inserted : " ); fflush( stdin ); scanf( "%c", &node ); insert\_node( node ); break; case 2: printf( "Enter an edge to be inserted : " ); fflush( stdin ); scanf( "%c %c", &origin, &destin ); insert\_edge( origin, destin ); break; case 3: printf( "Enter a node to delete : " ); fflush( stdin ); scanf( "%c", &node ); /\*This fn deletes the node from header node list\*/ delete\_node( node ); /\* This fn deletes all edges coming to this node \*/ delnode\_edge( node ); break; case 4: printf( "Enter an edge to be delete : " ); fflush( stdin ); scanf( "%c %c", &origin, &destin ); del\_edge( origin, destin ); break;  case 5: display(); break; case 6: exit(); default: printf( "Wrong choice\n" ); break; } /\*End of switch\*/ } /\*End of while\*/  } /\*End of main()\*/  insert\_node( char node\_name ){ struct node \* tmp, \*ptr;  tmp = malloc( sizeof( struct node ) ); tmp->name = node\_name; tmp->next = NULL; tmp->adjacency= NULL;  if ( start == NULL ) { start = tmp; return ; }  ptr = start;  while ( ptr->next != NULL ) ptr = ptr->next;  ptr->next = tmp; } /\*End of insert\_node()\*/  delete\_node( char u ) {  struct node \* tmp, \*q; if ( start->name == u ) { tmp = start; start = start->next; /\* first element deleted \*/ free( tmp ); return ; } q = start; while ( q->next->next != NULL ) { if ( q->next->name == u ) /\* element deleted in between \*/ { tmp = q->next; q->next = tmp->next; free( tmp ); return ; } q = q->next; } /\*End of while\*/  if ( q->next->name == u ) /\* last element deleted \*/ { tmp = q->next; free( tmp ); q->next = NULL; } } /\*End of delete\_node()\*/ delnode\_edge( char u ) {  struct node \* ptr;  struct edge \*q, \*start\_edge, \*tmp; ptr = start;  while ( ptr != NULL ) { /\* ptr->adjacency points to first node of edge linked list \*/  if ( ptr->adjacency->dest == u ) { tmp = ptr->adjacency; ptr->adjacency = ptr->adjacency->link; /\* first element deleted \*/ free( tmp ); continue; /\* continue searching in another edge lists \*/ }  q = ptr->adjacency;  while ( q->link->link != NULL ) { if ( q->link->dest == u )/\* element deleted in between \*/ { tmp = q->link; q->link = tmp->link; free( tmp ); continue; }  q = q->link; } /\*End of while\*/  if ( q->link->dest == u ) /\* last element deleted \*/ { tmp = q->link; free( tmp ); q->link = NULL; }  ptr = ptr->next; } /\*End of while\*/ } /\*End of delnode\_edge()\*/  insert\_edge( char u, char v )  struct node \* locu, \*locv;  struct edge \*ptr, \*tmp; locu = find( u ); locv = find( v );  if ( locu == NULL ) { printf( "Source node not present ,first insert node %c\n", u ); return ; } if ( locv == NULL ) { printf( "Destination node not present ,first insert node %c\n", v ); return ; }  tmp = malloc( sizeof( struct edge ) ); tmp->dest = v; tmp->link = NULL;  if ( locu->adjacency == NULL ) /\* item added at the begining \*/ { locu->adjacency = tmp; return ; } ptr = locu->adjacency;  while ( ptr->link != NULL ) ptr = ptr->link; ptr->link = tmp; } /\*End of insert\_edge()\*/  struct node \*find( char item ) {  struct node \*ptr, \*loc; ptr = start;  while ( ptr != NULL ) { if ( item == ptr->name ) { loc = ptr; return loc; } else ptr = ptr->next; }  loc = NULL; return loc; } /\*End of find()\*/  del\_edge( char u, char v ) { struct node \* locu, \*locv;  struct edge \*ptr, \*tmp, \*q; locu = find( u );  if ( locu == NULL ) { printf( "Source node not present\n" ); return ; } if ( locu->adjacency->dest == v ) { tmp = locu->adjacency; locu->adjacency= locu->adjacency->link; /\* first element deleted \*/ free( tmp ); return ; } q = locu->adjacency; while ( q->link->link != NULL ) { if ( q->link->dest == v ) /\* element deleted in between \*/ { tmp = q->link; q->link = tmp->link; free( tmp ); return ; }  q = q->link; } /\*End of while\*/  if ( q->link->dest == v ) /\* last element deleted \*/ { tmp = q->link; free( tmp ); q->link = NULL; return ; }  printf( "This edge not present in the graph\n" ); } /\*End of del\_edge()\*/ display() {  struct node \* ptr;  struct edge \*q;  ptr = start;  while ( ptr != NULL ) { printf( "%c ->", ptr->name ); q = ptr->adjacency;  while ( q != NULL ) { printf( " %c", q->dest ); q = q->link; }  printf( "\n" ); ptr = ptr->next;  } } /\*End of display()\*/ |  |  |

[**list of Data Stru**](http://www.itstudentjunction.com/data-structures-programs-in-c.htm)