Linked List Program in C

A linked list is a sequence of data structures, which are connected together via links. Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array.

Implementation in C

Live Demo

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

struct node {
   int data;
   int key;
   struct node *next;
};

struct node *head = NULL;
struct node *current = NULL;

//display the list
void printList() {
   struct node *ptr = head;
   printf("\n[ ");

   //start from the beginning
   while(ptr != NULL) {
      printf("(%d,%d) ",ptr->key,ptr->data);
      ptr = ptr->next;
   }

   printf(" ]");
}

//insert link at the first location
void insertFirst(int key, int data) {
   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));

   link->key = key;
   link->data = data;

   //point it to old first node
   link->next = head;

   //point first to new first node
   head = link;
}

//delete first item
struct node* deleteFirst() {

   //save reference to first link
   struct node *tempLink = head;

   //mark next to first link as first
   head = head->next;

   //return the deleted link
   return tempLink;
}

//is list empty
bool isEmpty() {
   return head == NULL;
```

```
}

int length() {
   int length = 0;
   struct node *current;

   for(current = head; current != NULL; current = current->next) {
      length++;
   }

   return length;
}

//find a link with given key
struct node* find(int key) {

   //start from the first link
   struct node* current = head;

   //if list is empty
   if(head == NULL) {
      return NULL;
   }

   //navigate through list
   while(current->key != key) {

      //if it is last node
      if(current->next == NULL) {
         return NULL;
      } else {
         //go to next link
         current = current->next;
      }
   }

   //if data found, return the current Link
   return current;
}

//delete a link with given key
struct node* delete(int key) {

   //start from the first link
   struct node* current = head;
   struct node* previous = NULL;

   //if list is empty
   if(head == NULL) {
      return NULL;
   }

   //navigate through list
   while(current->key != key) {

      //if it is last node
      if(current->next == NULL) {
         return NULL;
      } else {
         //store reference to current link
         previous = current;
         //move to next link
         current = current->next;
      }
   }

   //found a match, update the link
   if(current == head) {
      //change first to point to next link
      head = head->next;
   } else {
      //bypass the current link
```

```
      previous->next = current->next;
   }

   return current;
}

void sort() {

   int i, j, k, tempKey, tempData;
   struct node *current;
   struct node *next;

   int size = length();
   k = size ;

   for ( i = 0 ; i < size - 1 ; i++, k-- ) {
      current = head;
      next = head->next;

      for ( j = 1 ; j < k ; j++ ) {

         if ( current->data > next->data ) {
            tempData = current->data;
            current->data = next->data;
            next->data = tempData;

            tempKey = current->key;
            current->key = next->key;
            next->key = tempKey;
         }

         current = current->next;
         next = next->next;
      }
   }
}

void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;

   while (current != NULL) {
      next  = current->next;
      current->next = prev;
      prev = current;
      current = next;
   }

   *head_ref = prev;
}

void main() {
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);

   printf("Original List: ");

   //print list
   printList();

   while(!isEmpty()) {
      struct node *temp = deleteFirst();
      printf("\nDeleted value:");
      printf("(%d,%d) ",temp->key,temp->data);
   }

   printf("\nList after deleting all items: ");
```

```
   printList();
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);

   printf("\nRestored List: ");
   printList();
   printf("\n");

   struct node *foundLink = find(4);

   if(foundLink != NULL) {
      printf("Element found: ");
      printf("(%d,%d) ",foundLink->key,foundLink->data);
      printf("\n");
   } else {
      printf("Element not found.");
   }

   delete(4);
   printf("List after deleting an item: ");
   printList();
   printf("\n");
   foundLink = find(4);

   if(foundLink != NULL) {
      printf("Element found: ");
      printf("(%d,%d) ",foundLink->key,foundLink->data);
      printf("\n");
   } else {
      printf("Element not found.");
   }

   printf("\n");
   sort();

   printf("List after sorting the data: ");
   printList();

   reverse(&head);
   printf("\nList after reversing the data: ");
   printList();
}
```

If we compile and run the above program, it will produce the following result −

Output

```
Original List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10) ]
Deleted value:(6,56)
Deleted value:(5,40)
Deleted value:(4,1)
Deleted value:(3,30)
Deleted value:(2,20)
Deleted value:(1,10)
List after deleting all items:
[ ]
Restored List:
[ (6,56) (5,40) (4,1) (3,30) (2,20) (1,10) ]
Element found: (4,1)
List after deleting an item:
[ (6,56) (5,40) (3,30) (2,20) (1,10) ]
Element not found.
List after sorting the data:
[ (1,10) (2,20) (3,30) (5,40) (6,56) ]
List after reversing the data:
[ (6,56) (5,40) (3,30) (2,20) (1,10) ]
```

Useful Video Courses

Azure Data Lake Online Training

42 Lectures 1.5 hours

Ravi Kiran

More Detail

Data Structure Online Training

141 Lectures 13 hours

Arnab Chakraborty

More Detail

Oracle Data Guard Online Training

26 Lectures 8.5 hours

Parth Panjabi

More Detail

Big Data & Hadoop Online Training

65 Lectures 6 hours

Arnab Chakraborty

More Detail

Python With Data Science

75 Lectures 13 hours

Eduonix Learning Solutions

More Detail