Literate Programming with LyX & Data Structures in C

EE14B127

18th August 2015

Abstract

In this lab session, we will write a C program that implements linked lists. Data structures play a important role in programming and linked list is one of the important data structures. The linked list implemented in the assignment will be used in the spice program.

1 Implementation of Linked List in C (Extra Credit Included)

```c
#include <stdio.h> // Library File Access
#include <stdlib.h> // Library File Access

/* Declaring a structure called Node */
typedef struct Node{
    struct Node *next;
    int a;
    int b;
}Node;
/* Declaring a structure called List */
typedef struct List {
    Node *head;
    int count;
}List;
/* Function for making the List Head */
List* createListHead(Node* new){
    List *list = malloc(sizeof(List)); // Allocation of memory to struct List
    list->head = new; // Head points to new
    list->count = 0; // Initialise count of head to 0
    return list; // Return the struct List list
}
Node* createNode(int a,int b){
    Node *new =(Node *) malloc(sizeof(Node)); // Allocation of memory to struct Node
    new->a = a; // Assign value a to new
    new->b = b; // Assign value b to new
    new->next = NULL; // new point to NULL
    return new; // Return the struct Node new
}

void add(List *list,Node *node){
    node->next = list->head; // Node points where head points
    list->head = node; // Head points to node
    list->count += 1; // Count of list increased by 1
}
void displayReverse(Node* head){
    Node *temp = head; // temp points where head points
    if(temp==NULL) // If temp points to NULL
        return; // Return nothing
    displayReverse(temp->next); // Recursive call of display
    printf("%d %d\n",temp->a,temp->b); //Printing the details of node
}
```

```
void display(List *list){
    Node* current = list->head; // current points where head points
    Node* next; // Declaring pointer next
    while(current!=NULL){
        printf("%d %d\n",current->a,current->b); //Displaying details of current
        next = current->next; // next points where current points
        free(current); // Deleting the struct current
        current = next; //current now points to next
    }
    list = NULL; // list points to NULL
}

int main(){
    int testcases;
    int a,b;
    scanf("%d",&testcases);
    scanf("%d %d",&a,&b);
    Node *new = createNode(a,b); // Node new created
    List *list = createListHead(new); // List list created
    while(testcases-1){
        int a, b;
        scanf("%d %d",&a,&b);
        Node *new = createNode(a,b); // Node new created
        add(list,new); // Node new added to the linked list
        testcases --; // testcases decremented by 1
    }
    printf("\n");
    printf("%s\n","Printing Linked List in Reverse(Using Recursion)" );
    Node *head = list->head;
    displayReverse(head); // Reverse printing of linked list
    printf("\n");
    printf("%s\n","Printing Linked List" );
    display(list); // Printing of linked list
    return 0;
}
```

2 Output of the C Program

For the given input :

```
5
2 3
-3 3
4 5
6 7
8 9
```

The output obtained ie the printing of linked list in forward and reverse :

```
Printing Linked List in Reverse(Using Recursion)
2 3
-3 3
4 5
6 7
8 9
Printing Linked List
8 9
6 7
4 5
-3 3
2 3
```