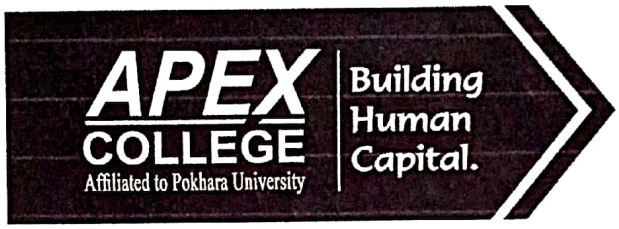
Apex College

BCIS Program

Affiliated to Pokhara University



Data Structure & Algorithms
Lab Report

Q1

Implementing multiple operations of Singly Linked List

Date: __-06-2022

Submitted by:
Ishwor Shrestha
Roll no.: 2018-BCIS-414

Submitted to:
Pravakar Ghimire, &
Anmol Shrestha
Apex College

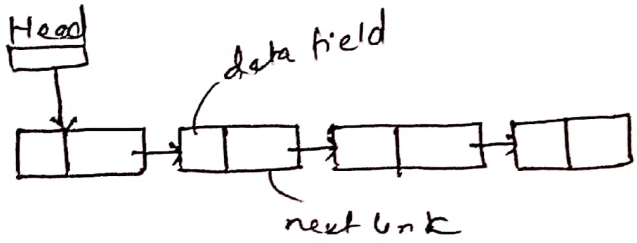
Lab 8 Objectives

- To understand about Linked List in data structure.
- To implement various operations of Singly Linked List.

Introduction

- LinkedList is a linear collection of data nodes like an array but those nodes are located dynamically. The nodes are connected to each others with their link. Each node contains data and link(s) field(s) (i.e. nodes are linked using pointers.

The singly linked list is a collection of nodes where each node contains one pointer to the next node in sequence.



A program to implement various operations in a singly linked list.

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

void insert_from_beg(node **head, int item);
struct node {
    int data;
    struct node *next;
};
typedef struct node node;
```

```
void insert_from_beg(node **head, int item){
    node *temp;
    temp = (node*) malloc(sizeof(node));
    temp->data = item;
    temp->next = *head;
    *head = temp;
}

void insert_from_end(node **head, int item){
    node *temp, temp1;
    temp = (node*) malloc(sizeof(node));
    temp->data = item;
    temp->next = NULL;
    if(*head == NULL)
        *head = temp;
    else{
        temp1 = *head;
        while(temp1->next != NULL)
            temp = temp->next;
        temp1->next = temp;
    }
}

void insert_from_sp_pos(node *head, int item, int pos){
    int i;
    node *temp, *temp1;
    temp = (node*) malloc(sizeof(node));
    temp->data = item;
    temp1 = *head;
    for(i=0; i<pos-1; i++)
        temp1 = temp1->next;
    temp->next = temp1->next;
    temp1->next = temp;
}
```

```
int del_from_beg (node **head){
  int item = -1;
  node *itemp;
  if (*head == NULL)
     printf("List is empty \n");
  else {
     temp = *head;
     *head = temp -> next;
     item = temp -> data;
     free(temp);
  }
  return item;
}

int del_from_end (node **head){
  int item=-1;
  node *temp, *temp1;
  if (*head == NULL)
     printf("List is empty.\n");
  else if ((*head) -> next == NULL) {
     temp = *head;
     *head = NULL;
     item = temp-> data;
     free(temp);
  }
  else {
     temp = *head;
     while (temp->next != NULL) {
        temp1 = temp;
        temp = temp -> next;
     }
```

```
    temp1 -> next = NULL;
    item = temp -> data;
    free (temp);
  }
  return item;
}

int del_from_sp_pos (node **head, int pos){
  int i, item = -1;
  node *temp, *temp1;
  temp = *head;
  for (i=1; i<pos; i++){
    temp1 = temp;
    temp = temp -> next;
  }
  temp1 -> next = temp -> next;
  item = temp -> data;
  free (temp);
  return item;
}

void traverse (node **head) {
  node *temp;
  temp = *head;
  printf ("The elements in Singly Linked List.\n");
  while (temp != NULL){
    printf ("%d \t", temp -> data);
    temp = temp -> next;
  }
}
```

```
void main(){
    node *head;
    int ch, item, pos;
    head = NULL;
    while(1){
        printf("Enter your choice of option:\n");
        printf("1.Insert from beginning.\n
                2. Insert from end.\n
                3. Insert from specific position.\n
                4. Delete from beginning.\n
                5. Delete from end.\n
                6. Delete from specific position.\n
                7. Traverse and display all elements\n
                8. Exit.\n");
        scanf("%d", &ch);
        switch(ch){
        case 1:
            printf("Enter data:");
            scanf("%d", &item);
            insert_from_beg(&head, item);
            break;

        case 2;
            printf("Enter data.\n");
            scanf("%d", &item);
            insert_from_end(&head, item);
            break;
        Case 3:
            printf("Enter data:");
            scanf("%d", &item);
            printf("Enter position:");
            scanf("%d", &pos);
            insert_from_sp_pos(&head, item, pos);
            break;
```

```
case 4:
    item = del_from_beg (&Head);
    printf("%d is deleted.\n",item);
    break;

case 5:
    item = del_from_end (&Head);
    printf("%d is deleted \n",item);
    break;

case 6:
    printf("Enter positon to be deleted: ");
    scanf("%d", &pos);
    item = del_from_spr_pos (&Head, pos);
    printf("%d is deleted.\n", item);
    break;

case 7:
    traverse (&Head);
    break;

case 8:
    exit (0);
    break;

default:
    printf("Invalid choice.\n");
}
}
```

Activities.

In this lab 7, we performed various operations of singly linked list:

1. Insertion of a node from beginning of current singly linked list
2. Insertion of a node from end of list
3. Insertion of a node from a specific position (n).
4. Deletion of a node from beginning of singly linked list
5. Deletion of a node from end
6. Deletion of a node from a specific position (n).
7. Traverse all nodes and display all nodes of Singly Linked list.

Conclusion

I learned about the all possible operations in a singly linked list.