Apex College

BCIS Program

Affiliated to Pokhara University

Data Structure & Algorithms
Lab Report
11
Doubly Linked List

Date: 17.02.2022

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

Submitted to:
Pravakar Ghimire, &
Anmol Shrestha
Apex College

Lab 11 Objectives

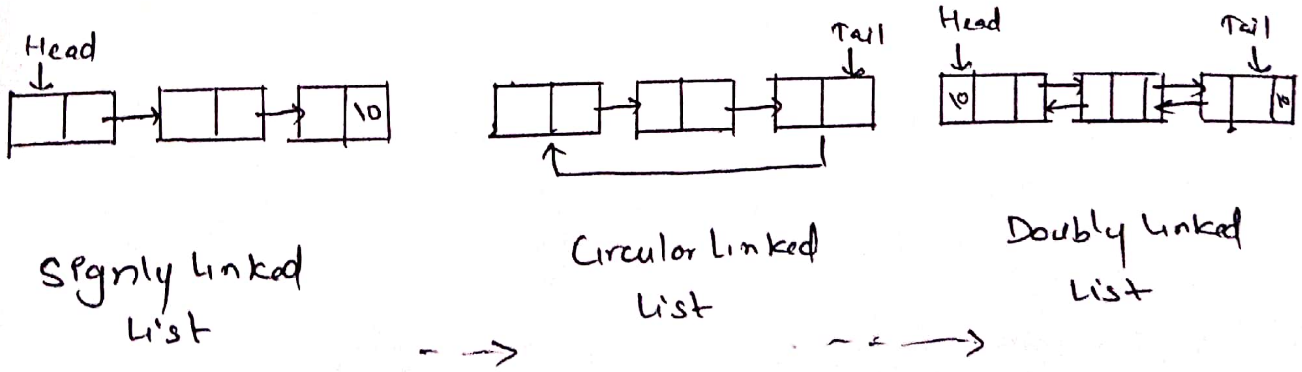
- To understand about doubly linked list
- To implement the various operations of doubly linked list.

Introduction

Doubly Linked List is a Data Structure, which is a variation of the Linked List, in which the traversal is possible in the both directions,

As we realized the circular linked list just solved the more time taken process of traversal of singly linked list by connecting last node with first node. But, ~~the~~ It does not solve the problem of accessing the second last node ~~as~~ we expect.

So, the Doubly Linked List introduced to traverse back from both directions.



A 'C' program to perform different operations in Doubly linked list

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

struct bnode {
    int data;
    struct bnode *next;
    struct bnode *prev;
};
typedef struct bnode bnode;

~~struct~~ bnode *tail = NULL;
bnode *head = NULL;

void insert_from_beg(int item){
  bnode *temp;
  temp = (bnode*) malloc (size of(bnode));
  temp -> data = item;
  temp -> prev = NULL;
  if (head == NULL){
     temp -> next = NULL;
     head = temp;
     tail = temp;
  }else{
     temp -> next = head;
     head -> prev = temp;
     head = temp;
  }
}
```

```
void insert_from_end(int item) {
  bnode *temp;
   temp = (bnode *) malloc (sizeof (bnode));
   temp -> data = item;
   temp -> next = NULL;

    if (head == NULL) {
        temp -> prev = NULL;
        head = temp;
        tail = temp;
    };
    else {
        temp -> prev = tail;
        tail -> next = temp;
        tail = temp;
    }
}
void insert_from_sp_pos (int item, int pos) {
  bnode *temp *temp1;
  int i;
   temp = (bnode *) malloc (sizeof (bnode));
   temp -> data = item;
   temp1 => head;

   for (i=1; i<pos -1 ; i++).
       temp1 = temp1 -> next;

   temp -> next = temp1 -> next;
   temp => prev = temp1;
   temp1 -> next => prev = temp;
   temp1 -> next = temp;
}
```

```
void del-from.beg (·){
    int Item = -1;
    bnode *temp ;
    if (head = NULL)
        printf("List is empty \n");

    else if (head == tail){
        temp = head ;
        head = NULL;
        tail = NULL;
        item = temp ->data;
        free (temp);
        printf("%d is deleted \n", item);
    }
    else {
        temp = head ;
        head = temp -> next;
        head ->prev = NULL;
        item = temp -> data;
        free (temp);
        printf.("%d is deleted from list.\n", item);
    }
}

void del-from-end (){
    int item = -1;
    node *temp;
    if (head == NULL)
        printf ("List is empty\n");
else if (head == tail){
    temp =head;
    head =NULL;
    tail = NULL;
    item = temp-data;
    free (temp);
    printf("%d is deleted ", item);
}
else {
    temp =tail;
    tail = temp ->prev;
    tail->next = NULL;
    item = temp ->data;
    free (temp);
    printf("%d is deleted.\n",item);
}
}

void del-from-sp-pos (int pos){
  bnode *temp;
  int i, item=-1;
  temp =head;
  for (i=1; i<pos-1 ; i++)
      temp = temp->next;
  temp-next ->prev = temp ->prev;
  temp ->prev ->next = temp ->next;
  item = temp ->data ;
  free (temp);
  printf("%d is deleted \n",item);
}
```

```
void search (int item){
 bnode *temp;
 int i = 0, flag;
 temp = head;
 if (temp == NULL){
     printf("\n list is empty \n");

 else {
     while (temp != NULL){
        if (temp->data == item) {
            printf(" % found at %d position \n", item, i+1);
            flag =0;
            break;
        }
        else
            flag=1;
        }
        i++;
        temp = temp->next;
    }
    if (flag == 1)
        printf(" % is not found \n", item);
  }
}

void display () {
 bnode *temp;
 temp = ~~tail~~ head;
 if (temp == NULL)
    printf("List is empty. \n");
```

```
else {
    printf("Nodes of Doubly linked list
    while (temp! = NULL){
        printf("%d \t", temp->data);
        temp = temp->next;
    }
}

main(){
  int n, pos, ch;

  while (1){
     printf("Enter your choice : \t");
     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. Search, \n 8. Display \n 9. Exiting");

     Scanf("%d", & ch);

     switch (ch){
        case 1:
          printf("\n Enter number:");
          scanf("%d", &n);
          insert-from-beg(n);
          break;

        case 2:
          printf("\n Enter number:");
          scanf("%d", &n);
          insert-from-end(n);
          break;

        case 3:
           printf("\n Enter number :");
           scanf("%d", &n);
        printf("\n Enter the position:");
        scanf("%d", &pos);
        break;

    case 4:
        del-from-beg ( );
        break;

    case 5:
        del-from-end();
        break;

    case 6:
        printf("\n Enter the positon to delete:");
        scanf("%d", &pos);
        del-from-sp-pos (pos);
        break;

    case 7:
        printf("\n Enter the elenent to search:");
        Scanf("%d ", &n);
        search (n);
        break;

    Case 8:
        display( );
        break;

    case 9:
        Exit(0);

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

Activities

In this lab, we performed various operations of Doubly linked list.

1. Insertion Operations
2. Deletion Operations
3. Display / Traversal Operation
4. Search operation

Conclusion

I learned about the implementation of Doubly Linked List & its various operations.