Mechi Multiple Campus

(Tribhuvan University)
Bhadrapur, Jhapa

Lab Report of

Data Structures and Algorithm (CACS-201)

Implementation of Singly Linked List

Faculty of Humanities & Social Sciences

Tribhuvan University

Kritipur, Nepal

Submitted By

Name: Santosh Bhandari

Roll No: 58

Submitted To

Mechi Multiple Campus

Department of Bachelor in Computer Application

Bhadrapur, Jhapa, Nepal

Introduction to Linked list and Singly Linked list

Linked list

A linked list is a dynamic list. It is implemented by using concept of pointer and Dynamic Memory Allocation. In a linked list data are stored in a nodes. Each node is divided into two parts, The first contains the data and the second part contains address of the next node.

Singly linked list

Singly linked list is a most common type of linked list. The node of a singly linked list is divided into two parts, the first part contains the data and the second part contains the pointer to the next node. The pointer field of last node contains NULL. It contains an additional pointer, START that points the first node.

Algorithms to Insert and Delete Data in Singly Linked list

Insert Data in Singly linked list

The insertion of data in singly linked list can have three cases which are:-

Insertion at the Beginning

① START
② [Allocate a new node]
newnode = (nodeType *) malloc (sizeof(nodeType))
③ Read the data (el) to Insert
④ Assign newnode → info = el
⑤ newnode → next = START
⑥ START = newnode
⑦ END

Insertion at the End

① START
② [Allocate a new node]
newnode = (nodeType *) malloc (sizeof(nodeType))
③ Read the data(el) to insert
④ Assign newnode → info = el

⑤ If START = NULL [When there is no nodes]
{
START = newnode
newnode → next = NULL
}
else {
lastnode = START
While (lastnode → next ≠ NULL)
{
lastnode = lastnode → next
}
lastnode → next = newnode
newnode → next = NULL
}
⑥ END

Insertion at the Specified Position

① START
② [Allocate a newnode]
newnode = (nodeType *) malloc (sizeof (nodeType))
③ Read the data(el) to insert
④ Assign newnode → info = el
⑤ Read the position (pos) to insert
⑥ Set previousnode = START, i = 1
⑦ While (i < pos - 1) {
previousnode = previousnode → next
i = i + 1
}
⑧ nextnode = previousnode → next
⑨ newnode → next = nextnode
⑩ previousnode → next = newnode
⑪ END

Delete Data in Singly Linked List

The deletion of data in singly linked list can have three cases which are:

Deleting the first node

1. START
2. If START = NULL
   then, print UNDERFLOW and exit
3. firstnode = START
4. second node = firstnode → next
5. START = secondnode
6. free(firstnode)
7. END

Deleting the Last Node

1. START
2. If START = NULL
   then, print UNDERFLOW and Exit
3. Set lastnode = START
4. While(lastnode → next ≠ NULL) {
   secondlastnode = lastnode
   lastnode = lastnode → next
   }
5. secondlastnode → next = NULL
6. free(lastnode)
7. END

Deleting the Node at the Specified position

① START
② If START = NULL
    then, print UNDERFLOW and exit
③ Read the position of the node (pos) to be deleted
④ Set specifiednode = START and i = 1
⑤ While (i < pos) {
    previousnode = specifiednode
    specifiednode = specifiednode → next
    nextnode = specifiednode → next
    i = i + 1
  }
⑥ previousnode → next = nextnode
⑦ Deallocate (specifiednode)
⑧ END

Program Code

```
#include<stdio.h>
#include<malloc.h>
#include<process.h>
struct node{
        int info;
        struct node *next;
};
struct node *start=NULL;
void insert_first(int);
void insert_last(int);
void insert_specposition(int);
void delete_first();
void delete_last();
void delete_specposition();
void display();
void search();
void main(){
        int data,ch;
        top:
        printf("\nMenu for the program\n");
        printf("\n1 to insert data as the first node");
        printf("\n2 to insert data as the last node");
        printf("\n3 to insert data at the specified position");
        printf("\n4 to delete the first node");
        printf("\n5 to delete the last node");
        printf("\n6 to delete the specified node");
        printf("\n7 to display the linked list");
        printf("\n8 to search a particular data in the linked list");
        printf("\n9 to exit");
        printf("\nEnter Your Choice(1,2,3,4,5,6,7,8,9): ");
        scanf("%d",&ch);
        switch(ch)
        {
                case 1:
                        printf("\nInput the data to insert:");
                        scanf("%d",&data);
                        insert_first(data);
                        goto top;
                case 2:
                        printf("\nInput the data to insert:");
                        scanf("%d",&data);
                        insert_last(data);
                        goto top;
                case 3:
                        printf("\nInput the data to insert:");
                        scanf("%d",&data);
                        insert_specposition(data);
                        goto top;
```

```
                case 4:
                        delete_first();
                        goto top;
                case 5:
                        delete_last();
                        goto top;
                case 6:
                        delete_specposition();
                        goto top;
                case 7:
                        display();
                        goto top;
                case 8:
                        search();
                        goto top;
                case 9:
                        exit(0);
                default:
                        printf("\nInvalid Choice");
                        goto top;
        }
}
void insert_first(int data){
        struct node *newnode;
        newnode=(struct node *)malloc(sizeof(struct node));
        if(newnode==NULL){
                printf("\nOut of memory space\n");
        }else{
                newnode->info=data;
                if(start==NULL){
                        newnode->next=NULL;
                        start=newnode;
                }else{
                        newnode->next=start;
                        start=newnode;
                }
                printf("\n%d is successfully inserted as the first node\n",data);
        }
}
void insert_last(int data){
        struct node *newnode, *last;
        newnode=(struct node *)malloc(sizeof(struct node));
        if(newnode==NULL){
                printf("\nOut of memory space");
        }else{
                newnode->info=data;
                if(start==NULL){
                        newnode->next=NULL;
                        start=newnode;
                }else{
```

```
                newnode->next=NULL;
                last=start;
                while(last->next!=NULL){
                        last=last->next;
                }
                last->next=newnode;
        }
        printf("\n%d is successfully inserted as the last node\n",data);
    }
}
void insert_specposition(int data){
    int pos,i;
    struct node *newnode,*previous;
    newnode=(struct node *)malloc(sizeof(struct node));
    if(newnode==NULL){
        printf("\nOut of memory space");
    }else{
        newnode->info=data;
        printf("\nInput the position of the node to inset data:");
        scanf("%d",&pos);
        if(pos==1){
            if(start==NULL){
                newnode->next=NULL;
                start=newnode;
            }else{
                newnode->next=start;
                start=newnode;
            }
        }else{
            previous=start;
            i=1;
            while(i<pos-1){
                previous=previous->next;
                if(previous==NULL){
                    printf("\nSorry insert position in between
existing nodes");
                    return;
                }
            }
            newnode->next=previous->next;
            previous->next=newnode;
        }
        printf("\n%d is successfully inserted at the %d position",data,pos);
    }
}
void delete_first(){
    struct node *temp;
    if(start==NULL){
        printf("\nList is empty \n");
    }else{
```

```
                temp=start;
                start=temp->next;
                printf("\nThe deleted element is %d\n",temp->info);
                free(temp);
        }
}
void delete_last(){
        struct node *temp, *last;
        if(start==NULL){
                printf("\nList is empty \n");
        }else{
                temp=start;
                last=start;
                while(temp->next!=NULL)  {
                        last=temp;
                        temp=temp->next;
                }
                last->next=NULL;
                printf("\nThe deleted element is %d\n",temp->info);
                free(temp);
        }
}
void delete_specposition(){
        int pos,i;
        struct node *temp, *previous;
        if(start==NULL){
                printf("\nList is empty \n");
        }else{
                printf("\nInput the position of the data to delete\n");
                scanf("%d",&pos);
                if(pos==1){
                        temp=start;
                        start=start->next;
                        printf("\nThe deleted data is %d\t",temp->info);
                        free(temp);
                }else{
                        i=1;
                        temp=start;
                        while(i<=pos-1){
                                previous=temp;
                                temp=temp->next;
                                if(temp==NULL){
                                        printf("\nSorry insert position in between
existing nodes");
                                        return;
                                }
                                i=i+1;
                        }
                        previous->next=temp->next;
                        printf("\nThe deleted element is %d\n",temp->info);
```

```
                free(temp);
            }
        }
}
void display(){
        struct node *ptr;
        if(start==NULL){
                printf("\nThe list is empty");
        }else{
                ptr=start;
                printf("\nThe elements of the linked list are:\n");
                while(ptr!=NULL){
                        printf("%d\t",ptr->info);
                        ptr=ptr->next;
                }
        }
}
void search(){
        int el,search=0;
        struct node *temp;
        if(start==NULL){
                printf("\nThe list is empty");
        }else{
                printf("\nInput the data to search:");
                scanf("%d",&el);
                temp=start;
                while(temp!=NULL){
                        if(temp->info==el){
                                search=1;
                                break;
                        }
                        temp=temp->next;
                }
                if(search==1){
                        printf("\nSearch is successful");
                }else{
                        printf("\nSearch is unsuccessful");
                }
        }
}
```

Output of the Program

```
Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 1

Input the data to insert:10

10 is successfully inserted as the first node

Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 1

Input the data to insert:20

20 is successfully inserted as the first node
Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 3

Input the data to insert:1

Input the position of the node to inset data:1

1 is successfully inserted at the 1 position
Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 7

The elements of the linked list are:
1       20      10
```

```
Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 6

Input the position of the data to delete
2

The deleted element is 20

Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 7

The elements of the linked list are:
1        10
Menu for the program

1 to insert data as the first node
2 to insert data as the last node
3 to insert data at the specified position
4 to delete the first node
5 to delete the last node
6 to delete the specified node
7 to display the linked list
8 to search a particular data in the linked list
9 to exit
Enter Your Choice(1,2,3,4,5,6,7,8,9): 8

Input the data to search:10

Search is successful
```

Conclusion

Singly linked list is the type of linked list. The node of singly linked list is divided into two parts, the first store the data and the second parts stores the address of the next node. The last node of singly linked list contains the NULL value in second part.