INDEX

Ex. No.:6

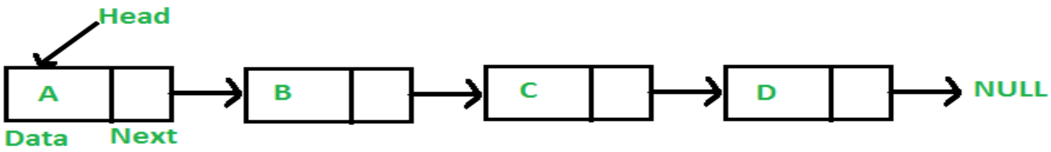
Date :

Implementation of Singly Linked List

Aim: To write a C-program to implement Singly Linked List (SLL).

Singly Linked List:

- It is a basic type of linked list.
- Each node contains data and pointer to next node and last node's pointer is NULL.
- Limitation of SLL is that we can traverse the list in only one direction, forward direction.



Representation of Singly Linked List

```
Struct node
{
        int info;
        Struct node *next;
};
Typedef struct node NODE;
NODE *start;
```

An algorithm to insert a node at the beginning of the singly linked list:

let *head be the pointer to first node in the current list

```
1. Create a new node using malloc function
        NewNode = (NodeType*) malloc (sizeof(NodeType));
2. Assign data to the info field of new node
```

```
    NewNode->info = newItem;
3. Set next of new node to head
    NewNode->next = head;
4. Set the head pointer to the new node
    head = NewNode;
5. End
```

An algorithm to insert a node at the end of the singly linked list:

let *head be the pointer to first node in the current list

```
1. Create a new node using malloc function
    NewNode=(NodeType*)malloc(sizeof(NodeType));
2. Assign data to the info field of new node
    NewNode->info=newItem;
3. Set next of new node to NULL
    NewNode->next=NULL;
4. if (head ==NULL)then
    Set head =NewNode and exit.
5 Else set temp = head
     while(temp->next!=NULL)
    temp=temp->next;        //increment temp
7. Set temp->next=NewNode;
8. End
```

An algorithm to delete the first node of the singly linked list:

let *head be the pointer to first node in the current list

```
1. If(head==NULL) then
    print “Void deletion” and exit
2. Store the address of first node in a temporary variable
    temp = head;
```

```
3. Set head to next of head.
        head=head->next;
4. Free the memory reserved by temp variable.
        free(temp);
5. End
```

An algorithm to delete the last node of the singly linked list:

let *head be the pointer to first node in the current list

```
1. If(head==NULL) then                      // list is empty
        print “Void deletion” and exit
2. else if(head->next==NULL) then           // list has only one node
        printf(“%d” ,head->info);           //print deleted item
        free(head);
3. else
        set temp=head;
        while (temp->next->next != NULL)
                set temp = temp->next;
        free(temp->next);
        Set temp->next=NULL;
4. End
```

Program

```
/*Linear Linked List implementation\*/
#include<stdio.h>
#include<conio.h>
#include<malloc.h>    //for malloc function
//#include<process.h>   //for exit function
struct node
{
    int info;
    struct node *next;
};
typedef struct node NodeType;
NodeType *head=NULL;
```

```
void insert_atfirst(int);
void insert_givenposition(int);
void insert_atend(int);
void delet_first();
void delet_last();
void delet_nthnode();
void info_sum();
void count_nodes();

int main()
{
    int choice;
    int item;
    //clrscr();
    do
    {
        printf("\n Menu for program:\n");
        printf("\n 1: Insert at Beginning \n 2: Insert at given position \n 3: Insert at last \n 4: Delete first node\n 5: Delete last node\n 6: Delete nth node\n 7: Display Items\n 8: Count Nodes\n 9: Exit\n");
        printf("\n Enter your choice\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("\nEnter item to be inserted");
                scanf("%d", &item);
                insert_atfirst(item);
                break;
            case 2:
                printf("\nEnter item to be inserted");
                scanf("%d", &item);
                insert_givenposition(item);
                break;
            case 3:
                printf("\nEnter item to be inserted");
                scanf("%d", &item);
                insert_atend(item);
                break;
            case 4:
                delet_first();
                break;
            case 5:
                delet_last();
                break;
            case 6:
                delet_nthnode();
                break;
            case 7:
                info_sum();
                break;
            case 8:
                count_nodes();
                break;
            case 9:
                exit(1);
                break;
            default:
                printf("invalid choice\n");
            break;
        }
    }while(choice<10);
    getch();
    return 0;
}
/***********function definitions*************/
void insert_atfirst(int item)
{
    NodeType *nnode;
    nnode=(NodeType*)malloc(sizeof(NodeType));
    nnode->info=item;
    nnode->next=head;
    head=nnode;
}

void insert_givenposition(int item)
{
    NodeType *nnode;
    NodeType *temp;
    temp=head;
    int p,i;
    nnode=( NodeType *)malloc(sizeof(NodeType));
    nnode->info=item;
    if (head==NULL)
    {
        nnode->next=NULL;
        head=nnode;
    }
    else
    {
        printf("Enter Position of a node at which  you want to insert an new node\n");
        scanf("%d",&p);
    for(i=1;i<p-1;i++)
    {
    temp=temp->next;
    }
    nnode->next=temp->next;
    temp->next=nnode;
    }
}
void insert_atend(int item)
{
    NodeType *nnode;
    NodeType *temp;
    temp=head;
    nnode=( NodeType *)malloc(sizeof(NodeType));
    nnode->info=item;
    if(head==NULL)
    {
        nnode->next=NULL;
        head=nnode;
    }
    else
    {
       while(temp->next!=NULL)
       {
     temp=temp->next;
       }
       nnode->next=NULL;
       temp->next=nnode;
    }
}
void delet_first()
{
    NodeType *temp;
    if(head==NULL)
    {
        printf("Void deletion|n");
        return;
    }
    else
    {
         temp=head;
         head=head->next;
         free(temp);
    }
}
 void delet_last()
 {
    NodeType *hold,*temp;
    if(head==NULL)
    {
        printf("Void deletion|n");
        return;
    }
    else if(head->next==NULL)
    {
         hold=head;
         head=NULL;
         free(hold);
    }
    else
    {
        temp=head;
       while(temp->next->next!=NULL)
       {
          temp=temp->next;
       }
       hold=temp->next;
       temp->next=NULL;
       free(hold);
    }
 }
void delet_nthnode()
 {
    NodeType *hold,*temp;
    int pos, i;
    if(head==NULL)
    {
        printf("Void deletion|n");
        return;
    }
    else
    {
       temp=head;
       printf("Enter position of node which node is to be deleted\n");
        scanf("%d",&pos);
       for(i=1;i<pos-1;i++)
        {
          temp=temp->next;
        }
       hold=temp->next;
       temp->next=hold->next;
       free(hold);
    }
 }
void info_sum()
 {
     NodeType *temp;
     temp=head;
     while(temp!=NULL)
    {
        printf("%d\t",temp->info);
        temp=temp->next;
    }
 }
void count_nodes()
 {
     int cnt=0;
     NodeType *temp;
     temp=head;
     while(temp!=NULL)
     {
        cnt++;
        temp=temp->next;
     }
     printf("total nodes=%d",cnt);

 }
```

Output:

Conclusion: