KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

DATA STRUCTURE AND ANALYSIS OF ALGORITHM

KCA 253 : Session 2020-21

EXPERMINT – 4

PROGRAM

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
typedef struct dlklist{
int info;
struct dlklist *prev;
struct dlklist *next;
}node;

int main()
{
  int data,choice,after,before;
  node *tail,*head,*ptr,*loc;
  head=tail=NULL;
  while(1)
  {
         printf("\n1.INSERT AT BEGINNING .\n2.INSERT AT END.\n3.TRAVERSING
.\n4.TRAVERSING IN REVERSE .\n5.INSERT BEFORE \n6.INSERT AFTER
ELEMENT.");
         printf("\n7.DELETE_BEGINNING .\n8. DELETE_FROM_END");
         scanf("%d",&choice);
switch(choice)
         {
         case 1:printf ("enter data to insert");
                scanf ("%d",&data);
                ptr = (node*)malloc(sizeof(node));
                ptr -> next = NULL;
                ptr -> prev = NULL;
                ptr -> info = data;
                if ( head == NULL )
                  {
                      head=tail=ptr;
                       }
                      else
                        {
                                ptr->next=head;
                  head->prev=ptr;
                                head=ptr;
                        }  break;
    case 2:
          printf("enter data to insert");
    scanf("%d",&data)    ;
       ptr=(node*)malloc(sizeof(node));
       ptr->next=NULL;
       ptr->prev=NULL;
       ptr->info=data;
        if(head==NULL)
           {
            head=tail=ptr;
         }
        else
         {
             tail->next=ptr;
          ptr->prev=tail;
             tail=ptr;
         }
     break;
case 3:
   loc=head;
    while(loc!=NULL)
       {
          printf("%d ->",loc->info);
             loc=loc->next;
       }
           break;
case 4:
    loc=tail;
    while(loc!=NULL)
          {
               printf("%d->",loc->info);
               loc=loc->prev;
          }
          break;
case 5:
    printf("enter element before need to insert\n");
    scanf("%d",&before);
    printf("enter element you need to insert\n");
    scanf("%d",&data);
    loc=head;
       while(loc!=NULL)
       {
         if(loc->info==before)
             {
           printf("hello");
           break;
             }
             loc=loc->next;
        }
            if(loc!=NULL)
            {
             ptr=(node*)malloc(sizeof(node));
             ptr->info=data;
             if(loc->prev!=NULL)
              {
                ptr->prev=loc->prev;
                ptr->next=loc;
             (loc->prev)->next=ptr;
                loc->prev=ptr;
       }
     else
              if(loc->prev==NULL)
         {
           ptr->prev=NULL;
           ptr->next=loc;
           loc->prev=ptr;
           head=ptr;
         }
        else
          {
          printf("element not found");
          }
        }
        break;
case 6:
    printf("enter elements after that need to insert new node\n");
    scanf("%d",&after);
        printf("enter element you need to insert\n");
        scanf("%d",&data);
        loc=head;
        while(loc!=NULL)
         {
             if(loc->info==after)
               {
                   break;
                  }
             loc=loc->next;
            }
        if(loc!=NULL)
            {
                 ptr=(node*)malloc(sizeof(node));
                 ptr->info=data;
```

```
                    if(loc->next!=NULL)
                        {
                            ptr->prev=loc;
                            ptr->next=loc->next;
                            (loc->next)->prev=ptr;
                            loc->next=ptr;
                       }
                else
                   if(loc->next==NULL)
                        {
                         ptr->prev=loc;
                         ptr->next=NULL;
                         loc->next=ptr;
                         tail=ptr;
                         }
                  else
                    {
                         printf("element not found");
                    }
             }
             break;
    case 7:node *loc,*temp;
         if(head == NULL)
             {
                 printf("linked list is empty");
             }
                 else
                  if(head == tail)
                   {
                        temp = head ;
                        free (temp);
                        head = tail = NULL ;
                   }
                   else
                    {
                       loc = head ;
                       head = head -> next ;
                       head -> prev = NULL ;
                       free ( loc ) ;
                        }
             break ;
case 8 :
          if (head == NULL)
                  {
                       printf (" list is empty ");
                    }
                    else
                      if(head == tail)
                       {
                       loc = head ;
                       free ( loc );
                       head = tail = NULL ;
                                }
                                else
                                 {
                                        loc = tail ;
                                        tail = tail -> prev ;
                                        tail -> next = NULL ;
                                        free ( loc ) ;
                                 }
                       break ;
        case 9:exit(1);
        default:printf("\nwrong choice");
   }

   }
return 0;

}
```

OUTPUT:

```
1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END1
enter data to insert2

1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
```

```
8. DELETE_FROM_END1
enter data to insert3

1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END2
enter data to insert4

1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END3
3 ->2 ->4 ->
1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END4
4->2->3->
1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END5
enter element before need to insert
2
enter element you need to insert
3
hello
1.INSERT AT BEGINNING .
```

```
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END7

1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END3
3 ->2 ->4 ->
1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END8

1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END3
3 ->2 ->
1.INSERT AT BEGINNING .
2.INSERT AT END.
3.TRAVERSING .
4.TRAVERSING IN REVERSE .
5.INSERT BEFORE
6.INSERT AFTER ELEMENT.
7.DELETE_BEGINNING .
8. DELETE_FROM_END
```

KIET Group of Institutions, Ghaziabad

Department of Computer Applications

(An ISO – 9001: 2015 Certified & 'A' Grade accredited Institution by NAAC)

DATA STRUCTURE AND ANALYSIS OF ALGORITHM

KCA 253 : Session 2020-21

COMPILED BY YASH AGRAWAL MCA 2 B