Data Structures

Dilip Kumar Maripuri
Computer Applications

Data Structures

Session : Traverse Operation; Insert Operations: At Front, At Rear; Delete Operations: At Front, At Rear

Dilip Kumar Maripuri

Computer Applications

Data Structures
Operations on Lists

- **Traverse a Singly Linked List**

Algorithm Display_LinkedList(head):

```
1. if Head = NULL
    1.1 Display ”Empty List”
    1.2 return
2. current ← head
3. while current $\neq$ NULL do
    3.1 print current.data              // Process the data of the current node
    3.2 current ← current.link
4. end while
```

End Algorithm

```c
void Display(NODE Head)
{
    if(Head==NULL)
         printf("Empty List");
    else
    {
         printf("\n HEAD-> ");
         for(NODE temp = Head; temp != NULL;
            temp=temp->link)
            printf(" %d ->",temp->data);
             printf("  NULL \n");
    }
}
```

Operations on Lists

- **Insert at the Beginning of Singly Linked List**

Algorithm Insert_Front(head, data):

```
1.  Set $new\_node \leftarrow$ create_node(data)
2.  if $new\_node \neq$ NULL then
    2.1 Set $new\_node.link \leftarrow$ head
    2.2 Set head $\leftarrow new\_node$
3.  end if
4.  return head
```

End Algorithm

```c
NODE ins_front(NODE Head, int data)
{
    NODE new_node = create_node(data);
    if(new_node != NULL)
    {
        new_node->link=Head;
        Head = new_node;
    }
    return new_node;
}
```

- **Delete First Node of Singly Linked List**

Algorithm Delete_Front(head):

```
1. if head = NULL then
    1.1 Print "Empty List"                    // No nodes to delete
    1.2 return head
2. end if
3. Set temp $\leftarrow$ head
4. Set head $\leftarrow$ head.next
5. Print "Deleting temp.data"
6. Free the memory allocated to temp
7. return head
```

End Algorithm

```
NODE del_front(NODE Head)
{
    NODE temp;
    if(Head == NULL)
        printf("\n\t\t Empty List");
    else
    {   temp = Head; Head = Head->link;
        printf("\n Deleting %d",temp->data);
        free(temp);
    }
    return Head;
}
```

Data Structures

Operations on Lists

- **Insert Node at the End of Singly Linked List**

Algorithm Insert_Last(head, data):

```
1.  Set $new\_node \leftarrow$ create_node(data)
2.  if $new\_node \neq$ NULL then
      2.1   if head = NULL then
        2.1.1  return $new\_node$            // List is empty, new node becomes the head
      2.2   end if
      2.3   Set $temp \leftarrow$ head
      2.4   while $temp.link \neq$ NULL do
        2.4.1  Set $temp \leftarrow temp.link$
      2.5   end while
      2.6   Set $temp.link \leftarrow new\_node$
3.  end if
4.  return head
```

End Algorithm

```
NODE ins_last(NODE Head, int data)
{
    NODE temp, new_node = create_node(data);
    if(new_node != NULL)
    {
        if(Head==NULL)
            return new_node;
        for(temp = Head; temp->link!= NULL; temp
            =temp->link);
        temp->link=new_node;
    }
    return Head;
}
```

- **Delete Last Node of Singly Linked List**

Algorithm Delete_Last(head):

```
1. if head = NULL then
      1.1 Print "Empty List"
      1.2 return NULL
2. end if
3. if head.link = NULL then                    // Only one node in the list
      3.1 Print "Deleted Node head.data"
      3.2 Free the memory allocated to head
      3.3 return NULL
4. end if
```

- **Delete Last Node of Singly Linked List**

Algorithm Delete_Last(head):

```
5.  Set $curr \leftarrow$ head
6.  while $curr.link.link \neq$ NULL do
      6.1 Set $curr \leftarrow curr.link$
7.  end while
8.  Print "Deleted Node $curr.link.data$"
9.  Free the memory allocated to $curr.link$
10. Set $curr.link \leftarrow$ NULL
11. return head
```

End Algorithm

```
NODE Delete_Last(NODE Head) {
    if (Head == NULL) {
        printf("\n\t\t Empty List");
        return NULL;
        }
    if (Head->link == NULL) {
        printf("\n\t Deleted Node %d", Head->
           data);
        free(Head);            return NULL;
        }
    NODE curr = Head;
    while (curr->link->link != NULL)
        curr = curr->link;
    printf("\n\t Deleted Node %d", curr->link->
       data);
    free(curr->link);
    curr->link = NULL;
    return Head;
}
```

PES
UNIVERSITY
CELEBRATING 50 YEARS

Thank You

Dilip Kumar Maripuri
Associate Professor
Department of Computer Applications

dilip.maripuri@pes.edu
8073212026