

05-02-24

- Q write a program to implement doubly linked list with primitive operations
- create a doubly linked list.
 - Insert a new node to left of node
 - Delete the node based on a specific value.

```
→ struct node
{
    struct node *next;
    int data;
    struct node *prev;
};

struct node *create_ll (struct node *start)
{
    struct node *new_node, *ptr;
    int num;
    printf("In Enter -1 to end:");
    printf("In Enter data:");
    scanf("%d", &num);
    while (num != -1)
    {
        if (start == NULL)
        {
            new_node = (struct node *) malloc
                (sizeof(struct node));
            new_node->prev = NULL;
            new_node->data = num;
        }
    }
}
```

```
new_node -> next = NULL;
```

```
start = new_node;
```

```
}
```

```
else
```

```
{
```

```
ptr = start;
```

```
new_node = (struct node *) malloc (sizeof(struct  
node));
```

```
new_node -> data = num;
```

```
new_node -> next = NULL;
```

```
start =
```

```
while (ptr -> next != NULL)
```

```
ptr = ptr -> next;
```

```
ptr -> next = new_node;
```

```
new_node -> prev = ptr;
```

```
new_node -> next = NULL;
```

```
}
```

```
printf("In Enter the data:");
```

```
scanf("%d", &num);
```

```
}
```

```
return start;
```

```
}
```

```
struct node *display(struct node * start)
```

```
{
```

```
struct node *ptr;
```

```
ptr = start;
```

```
while (ptr != NULL)
```

```
{
```

```
    printf("%d", ptr->data);
```

```
    ptr = ptr->next;
```

```
}
```

```
return start;
```

```
}
```

```
struct node *insert-before (struct node *ptr,
```

```
{
```

```
    struct node *new-node, *ptr;
```

```
    int num, val;
```

```
    printf("In Enter the data: ");
```

```
    scanf("%d", &num);
```

```
    printf("In Enter the value before which the data  
        has to be inserted ");
```

```
    scanf("%d", &val);
```

```
    new-node = (struct node *) malloc (sizeof (struct  
        node));
```

```
    new-node->data = num;
```

```
    ptr = start;
```

```
    while (ptr->data != val)
```

```
    {
```

```
        ptr = ptr->next;
```

```
    }
```

```
    new-node->next = ptr;
```

```
    new-node->prev = ptr->prev;
```

```
ptr->prev->next = new-node;
```

```
ptr->prev = new-node;
```

```
return start;
```

4.

```
struct node *delete_position( struct node  
                                * start)
```

```
{
```

```
    struct node * ptr;
```

```
    int val;
```

```
    ptr = start;
```

```
    printf("In Enter the value to be  
           deleted : ");
```

```
    scanf("%d", &val);
```

```
    while (ptr->data != val)  
    {
```

```
        ptr = ptr->next;
```

```
    }
```

```
    ptr->prev->next = ptr->next;
```

```
    ptr->next->prev = ptr->prev;
```

```
    free(ptr);
```

```
    return start;
```

```
}
```

Output

1. create a list
2. Display the list.
3. Add a node before a given node
4. Delete a given node
5. Exit.

Enter your choice: 1.

Enter -1 to end

Enter the data: 2

Enter the data: 3

Enter the data: 5

Enter the data: -1

Doubly Linked list created

1. create a list
2. Display the list.
3. Add a node before a given node
4. Delete a given node
5. Exit.

Enter your ~~choice~~ choice: 2

2 3 5

1. create a list
2. Display the list.
3. Add a node before a given node
4. Delete a given node
5. Exit.

Enter your choice: 3

Enter the data to be inserted: 4

Enter the value before which ^{data} to be inserted: 4

1. Create a list
2. Display the list
3. Add a node before a given node
4. Delete a given node
5. Exit

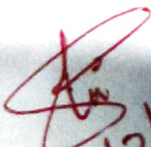
Enter your choice: 4.

Enter the value to be deleted: 3.

1. Create a list
2. Display the list
3. Add a node before a given node
4. Delete a given node
5. Exit

Enter your choice: 2.

2 4 5.


4/3/24