SINGLY LINKED LIST

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
struct node
{
int data;
struct node *next;
};
struct node *start = NULL;
struct node *create_ll(struct node *);
struct node *display(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *insert_before(struct node *);
struct node *insert_after(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_node(struct node *);
struct node *delete_after(struct node *);
struct node *delete_list(struct node *);
struct node *sort_list(struct node *);
int main()
{
int option;
clrscr();
do
{
printf("\n\n *****MAIN MENU *****");
```

```
printf("\n 1: Create a list");
printf("\n 2: Display the list");
printf("\n 3: Add a node at the beginning");
printf("\n 4: Add a node at the end");
printf("\n 5: Add a node before a given node");
printf("\n 6: Add a node after a given node");
printf("\n 7: Delete a node from the beginning");
printf("\n 8: Delete a node from the end");
printf("\n 9: Delete a given node");
printf("\n 10: Delete a node after a given node");
printf("\n 11: Delete the entire list");
printf("\n 12: Sort the list");
printf("\n 13: EXIT");
printf("\n\n Enter your option : ");
scanf("%d", &option);
switch(option)
{
case 1: start = create_ll(start);
printf("\n LINKED LIST CREATED"); break;
case 2: start = display(start); break;
case 3: start = insert_beg(start); break;
case 4: start = insert_end(start); break;
case 5: start = insert_before(start);break;
case 6: start = insert_after(start); break;
case 7: start = delete_beg(start); break;
case 8: start = delete_end(start); break;
case 9: start = delete_node(start); break;
case 10: start = delete_after(start); break;
case 11: start = delete_list(start);
printf("\n LINKED LIST DELETED"); break;
case 12: start = sort_list(start); break;
```

```
}
}while(option !=13);
getch();
return 0;
}
struct node *create_ll(struct node *start)
{
struct node *new_node, *ptr;
int num;
printf("\n Enter –1 to end");
printf("\n Enter the data : ");
scanf("%d", &num);
while(num!=–1)
{
new_node = (struct node*)malloc(sizeof(struct node));
new_node -> data=num;
if(start==NULL)
{
new_node -> next = NULL;
start = new_node;
}
else
{
ptr=start;
while(ptr–>next!=NULL)
ptr=ptr–>next;
ptr–>next = new_node;
new_node–>next=NULL;
}
printf("\n Enter the data : ");
```

```
scanf("%d", &num);
}
return start;
}
struct node *display(struct node *start)
{
struct node *ptr;
ptr = start;
while(ptr != NULL)
{
printf("\t %d", ptr -> data);
ptr = ptr -> next;
}
return start;
}
struct node *insert_beg(struct node *start)
{
struct node *new_node;
int num;
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
new_node -> next = start;
start = new_node;
return start;
}
struct node *insert_end(struct node *start)
{
struct node *ptr, *new_node;
int num;
```

```
printf("\n Enter the data : ");
scanf("%d", &num);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
new_node -> next = NULL;
ptr = start;
while(ptr -> next != NULL)
ptr = ptr -> next;
ptr -> next = new_node;
return start;
}
struct node *insert_before(struct node *start)
{
struct node *new_node, *ptr, *preptr;
int num, val;
printf("\n Enter the data : ");
scanf("%d", &num);
printf("\n 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)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next = new_node;
new_node -> next = ptr;
return start;
```

```
}
struct node *insert_after(struct node *start)
{
struct node *new_node, *ptr, *preptr;
int num, val;
printf("\n Enter the data : ");
scanf("%d", &num);
printf("\n Enter the value after which the data has to be inserted : ");
scanf("%d", &val);
new_node = (struct node *)malloc(sizeof(struct node));
new_node -> data = num;
ptr = start;
preptr = ptr;
while(preptr -> data != val)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next=new_node;
new_node -> next = ptr;
return start;
}
struct node *delete_beg(struct node *start)
{
struct node *ptr;
ptr = start;
start = start -> next;
free(ptr);
return start;
}
struct node *delete_end(struct node *start)
```

```
{
struct node *ptr, *preptr;
ptr = start;
while(ptr -> next != NULL)
{
preptr = ptr;
ptr = ptr -> next;
}
preptr -> next = NULL;
free(ptr);
return start;
}
struct node *delete_node(struct node *start)
{
struct node *ptr, *preptr;
int val;
printf("\n Enter the value of the node which has to be deleted : ");
scanf("%d", &val);
ptr = start;
if(ptr -> data == val)
{
start = delete_beg(start);
return start;
}
else
{
while(ptr -> data != val)
{
preptr = ptr;
ptr = ptr -> next;
```

```
}

preptr -> next = ptr -> next;

free(ptr);

return start;

}

}

struct node *delete_after(struct node *start)

{

struct node *ptr, *preptr;

int val;

printf("\n Enter the value after which the node has to deleted :
");

scanf("%d", &val);

ptr = start;

preptr = ptr;

while(preptr -> data != val)

{

preptr = ptr;

ptr = ptr -> next;

}

preptr -> next=ptr -> next;

free(ptr);

return start;

}

struct node *delete_list(struct node *start)

{

struct node *ptr;

ptr=start;

while(ptr -> next != NULL)

{

printf("\n %d is to be deleted next", ptr -> data);
```

```
start = delete_beg(ptr);

ptr = ptr -> next;

}

return start;

}
struct node *sort_list(struct node *start)

{

struct node *ptr1, *ptr2;

int temp;

ptr1 = start;

while(ptr1 -> next != NULL)

{

ptr2 = ptr1 -> next;

while(ptr2 != NULL)

{

if(ptr1 -> data > ptr2 -> data)

{

temp = ptr1 -> data;

ptr1 -> data = ptr2 -> data;

ptr2 -> data = temp;

}

ptr2 = ptr2 -> next;

}

ptr1 = ptr1 -> next;

}

return start;

}
```
OUTPUT:

1.Insert Beg
2.Insert Middle
3.Insert End
4.Delete Beg
5.Delete Middle
6.Delete End

```
7.Find
8.Traverse
9.Exit
Enter your choice : 1
Enter the element : 40
Enter your choice : 1
Enter the element : 30
Enter your choice : 1
Enter the element : 20
Enter your choice : 1
Enter the element : 10
Enter your choice : 8
10 20 30 40
Enter your choice : 7
Enter the element : 30
Element found...!
Enter your choice : 1
Enter the element : 5
Enter your choice : 8
5 10 20 30 40
Enter your choice : 3
Enter the element : 45
Enter your choice : 8
5 10 20 30 40 45
Enter your choice : 2
Enter the position element : 20

Enter the element : 25
Enter your choice : 8
5 10 20 25 30 40 45
Enter your choice : 4
The deleted item is 5
Enter your choice : 8
10 20 25 30 40 45
Enter your choice : 6
The deleted item is 45
Enter your choice : 8
10 20 25 30 40
Enter your choice : 5
Enter the element : 30
The deleted item is 30
Enter your choice : 8
10 20 25 40
Enter your choice : 9
```