DSA - Experiment 4

Name: Divyesh Khunt **Sapid:**60009210116 **Batch: A/3**

Aim: To create and implement a linked list in c programming

Theory:

Linked is a data structure which uses dynamic memory allocation. Unlike arrays, linked list elements are not stored at a contiguous location; the elements are linked using pointers. They include a series of connected nodes. Here, each node stores the data and the address of the next node.

Advantages of Linked Lists over arrays:

• Dynamic Array.
• Ease of Insertion/Deletion.

Drawbacks of Linked Lists:

• Random access is not allowed. We have to access elements sequentially starting from the first node(head node). So we cannot do a binary search with linked lists efficiently with its default implementation.
• Extra memory space for a pointer is required with each element of the list.
• Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

Time Complexity :

For searching is O(n)

For insertion and deletion is O(1)

CODE:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<stdlib.h>
struct node {
 int data ;
 struct node *next ;
};
struct node *start = NULL ;
struct node *create_ll (struct node *) ;
struct node *insert_begin (struct node *) ;
struct node *insert_end (struct node *) ;
```

```
struct node *insert_before (struct node *) ;
struct node *insert_after (struct node *) ;
struct node *display (struct node *) ;
struct node *delete_beg (struct node *) ;
struct node *delete_end (struct node *) ;
struct node *delete_after (struct node *) ;
struct node *delete_node (struct node *) ;
struct node *delete_list (struct node *) ;
struct node *sort_ll (struct node *) ;

int main () {

 int ch ;

 printf("Press 1 to create a linked list\n") ;
 printf("Press 2 to insert at the begin\n") ;
 printf("Press 3 to insert at the end\n") ;
 printf("Press 4 to insert before an element\n") ;
 printf("Press 5 to insert after an element\n") ;
 printf("Press 6 to display the linked list\n") ;
 printf("Press 7 to delete the first element\n") ;
 printf("Press 8 to delete the last element\n") ;
 printf("Press 9 to delete after an element\n") ;
 printf("Press 10 to delete a particular node\n") ;
 printf("Press 11 to delete the entire linked list\n") ;
 printf("Press 12 to sort the linked list\n") ;
 printf("Press 13 to exit this program\n") ;

 do {
 scanf("%d", &ch) ;
 switch (ch) {

 case 1 :
 start = create_ll (start) ;
 printf("list is created\n") ;
 break ;

 case 2 :
 start = insert_begin (start) ;
 break ;

 case 3 :
 start = insert_end (start) ;
```

```
break ;

case 4 :
start = insert_before (start) ;
break ;

case 5 :
start = insert_after (start) ;
break ;

case 6 :
start = display (start) ;
break ;

case 7 :
start = delete_beg (start) ;
break ;

case 8 :
start = delete_end (start) ;
break ;

case 9 :
start = delete_after (start) ;
break ;

case 10 :
start = delete_node (start) ;
break ;

case 11 :
start = delete_list (start) ;
break ;

case 12 :
start = sort_ll (start) ;
break ;

}

}while(ch != 13) ;
}
```

```
struct node *create_ll(struct node *start) {

 struct node *nn, *ptr ;
 int x ;
 printf("enter -1 to stop\n") ;
 printf("enter a number: \n") ;
 scanf("%d", &x) ;

 while(x != -1) {
 nn = (struct node *)malloc(sizeof(struct node)) ;
 nn -> data = x ;
 if(start == NULL) {
 nn -> next = NULL ;
 start = nn ;
 }
 else {
 nn -> next = NULL ;
 ptr = start ;
 while(ptr -> next != NULL) {
 ptr = ptr -> next ;
 }
 ptr -> next = nn ;
 }
 printf("Enter a no: ") ;
 scanf("%d", &x) ;
 }
 return start ;
}


struct node *insert_begin (struct node * start) {

 struct node *nn ;
 int x ;
 printf("Enter a number: ") ;
 scanf("%d" , &x) ;
 nn = (struct node *)malloc(sizeof(struct node)) ;
 nn -> data = x ;
 nn -> next = start ;
 start = nn ;
 return start ;
}
```

```
struct node *insert_end (struct node *start) {
 struct node *nn, *ptr ;
 int x ;
 printf("Enter a no: ") ;
 scanf("%d", &x) ;
 nn = (struct node *)malloc(sizeof(struct node)) ;
 nn -> data = x ;
 ptr = start ;
 while(ptr -> next != NULL) {
 ptr = ptr -> next ;
 }
 ptr -> next = nn ;
 nn -> next = NULL ;

 return start ;
}


struct node *insert_before (struct node *start) {
 struct node *nn, *pp, *ptr ;
 int x, val ;
 printf("enter a number: ") ;
 scanf("%d", &x) ;
 nn = (struct node *)malloc(sizeof(struct node)) ;
 nn -> data = x ;

 printf("enter the number you want to insert: ") ;
 scanf("%d", &val) ;
 ptr = start ;

 while(ptr -> data != val) {
 pp = ptr ;
 ptr = ptr -> next ;

 pp -> next = nn ;
 nn -> next = ptr ;

 }
 return start ;
}
```

```
struct node *insert_after (struct node *start) {
 struct node *nn, *pp, *ptr ;
 int x ,val ;
 printf("Enter a no: ") ;
 scanf("%d", &x) ;

 nn = (struct node *)malloc(sizeof(struct node)) ;
 nn -> data = x ;
 printf("Enter the value: ") ;
 scanf("%d", &val) ;
 pp = start ;
 ptr = start ;

 while(pp -> data != val) {

 pp = ptr ;
 ptr = ptr -> next ;
 }

 pp -> next ;
 nn-> next = ptr ;
 return start ;
}



struct node *display (struct node *start) {
 struct node *ptr ;
 ptr = start ;
 while (ptr != NULL) {
 printf("Data is : %d\n", ptr -> data) ;
 ptr = ptr -> next ;
 }
 return start ;
}



struct node *delete_beg (struct node *start) {
 struct node *ptr ;
 ptr = start ;
 start = start -> next ;
 printf("Data to be deleted is: %d\n", ptr -> data) ;
```

```
 free(ptr) ;
 return start ;
}



struct node *delete_end (struct node * start) {
 struct node *ptr, *pp ;
 pp = start ;
 ptr = start ;

 while(ptr -> next != NULL) {
 pp = ptr ;
 ptr = ptr -> next ;
 }
 pp -> next = NULL ;
 printf("Node to be deleted is: %d\n", ptr -> data) ;
 free(ptr) ;
 return start ;
}



struct node *delete_after (struct node *start) {
 int val ;
 struct node *pp, *ptr ;
 ptr = start ;
 pp = start ;
 printf("Enter a value after which u want to delete a node: ") ;
 scanf("%d", &val) ;
 while(pp -> data != val) {
 pp = ptr ;
 ptr = ptr -> next ;
 }
 pp -> next - ptr -> next ;
 printf("deleted node is %d\n", ptr -> data ) ;
 free(ptr) ;
 return start ;
}
```

```
struct node *delete_node (struct node *start) {
 struct node *ptr = start, *pp;
 int x ;
 printf("Enter a value to delete: ") ;
 scanf("%d", &x) ;
 while(ptr -> data != x) {
 pp = ptr ;
 ptr = ptr -> next ;
 }
 pp -> next = ptr -> next ;
 free(pp) ;
 return start ;
}



struct node *delete_list (struct node *start) {
 while(start != NULL) {
 printf("Data deleted %d\n", start -> data) ;
 start = delete_beg(start) ;
 }
 return start ;
}


struct node *sort_ll (struct node *start) {
 struct node *ptr1, *ptr2 ;
 int temp ;
 ptr1 = start ;
 while (ptr2 -> 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 ;
}
```

OUTPUTS:

Creation and display of LL

```
Press 1 to create a linked list
Press 2 to insert at the begin
Press 3 to insert at the end
Press 4 to insert before an element
Press 5 to insert after an element
Press 6 to display the linked list
Press 7 to delete the first element
Press 8 to delete the last element
Press 9 to delete after an element
Press 10 to delete a particular node
Press 11 to delete the entire linked list
Press 12 to sort the linked list
Press 13 to exit this program
1
enter -1 to stop
enter a number:
23
Enter a no: 46
Enter a no: 32
Enter a no: 78
Enter a no: 1
Enter a no: -1
list is created
6
Data is : 23
Data is : 46
Data is : 32
Data is : 78
Data is : 1
```

Deletion of data

```
Press 3 to insert at the end
Press 4 to insert before an element
Press 5 to insert after an element
Press 6 to display the linked list
Press 7 to delete the first element
Press 8 to delete the last element
Press 9 to delete after an element
Press 10 to delete a particular node
Press 11 to delete the entire linked list
Press 12 to sort the linked list
Press 13 to exit this program
1
enter -1 to stop
enter a number:
5
Enter a no: 4
Enter a no: 3
Enter a no: -1
list is created
7
Data to be deleted is: 5
6
Data is : 4
Data is : 3
```

```
Press 1 to create a linked list
Press 2 to insert at the begin
Press 3 to insert at the end
Press 4 to insert before an element
Press 5 to insert after an element
Press 6 to display the linked list
Press 7 to delete the first element
Press 8 to delete the last element
Press 9 to delete after an element
Press 10 to delete a particular node
Press 11 to delete the entire linked list
Press 12 to sort the linked list
Press 13 to exit this program
1
enter -1 to stop
enter a number:
1
Enter a no: 2
Enter a no: 3
Enter a no: 4
Enter a no: 5
Enter a no: -1
list is created
11
Data deleted 1
Data to be deleted is: 1
Data deleted 2
Data to be deleted is: 2
Data deleted 3
Data to be deleted is: 3
Data deleted 4
Data to be deleted is: 4
Data deleted 5
Data to be deleted is: 5
```

CONCLUSION:

Thus, in this article, we have understood the concept of Stack data structure and its implementation using Arrays in C.