Yash Mahajan 04 SE IT-B 191061

Page : 1
Date : / /

Experiment - 1: Study of linked list

Aim :-
Write a C program to implement singly linked lists with insert and delete operations.

Objectives :-
After study of this experiment, the student will be able to
- Understand how to create a linked list.
- Implement an algorithm using computer to create solve the given problem.

Outcomes :-
Develop algorithms to construct linked list.

Prerequisite :-
Array and Linked list.

Pre - Experiment Exercise :-

Brief Theory :-

A linked list, in simple terms, is a linear collection of data elements. These data elements are called as nodes.
Linked lists provide an efficient way of storing related data and perform basic operations such

Page:
Date: / /

as insertion, deletion and updation of information at the cost of extra space required for ~~sorting~~ storing the address of the next node.
Each node of the linked list comprises of two items - the data and the the reference to the next node. The last node has the reference to null for singly linked lists.

Program:-

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>

struct node
{
   int data;
   struct node* next;
};

struct node *start = NULL;

struct node *display(struct node *start){

   if (start==NULL)
   {
       printf("\n The list is empty");
   }
   else
   {
      printf("\n The elements of list are:-\n");
      struct node *ptr;
       ptr = start;
       while(ptr != NULL){
           printf(" %d\t", ptr -> data);
           ptr = ptr -> next;
       }
   }
   printf("\n");
   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){

   if (start==NULL){
      printf("\n The list is empty");
   }
   else{
       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_after(struct node *start){

   if (start==NULL){
      printf("\n The list is empty");
```

```
    }

    else{
        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;

        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){
    if (start == NULL)
    {
        printf("\n The list is empty");
    }
    else{
        struct node *ptr;
        ptr =start;
        start = start -> next;
        free(ptr);
    }
```

```
    return start;
}

struct node *delete_end(struct node *start){

    if (start == NULL)
    {
        printf("\n The list is empty");
    }
    else{
        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){

    if (start == NULL)
    {
        printf("\n The list is empty");
    }
    else{
        struct node *ptr, *preptr;
        int val;
        printf("\n Enter the node 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;
}

int main(){
    int ch;
    do
    {
        printf("\nEnter 1 for Displaying");
        printf("\nEnter 2 for Inserting at beginning");
        printf("\nEnter 3 for Inserting at end");
        printf("\nEnter 4 for Inserting after given value");
        printf("\nEnter 5 for Deleting first node");
        printf("\nEnter 6 for Deleting last node");
        printf("\nEnter 7 for Deleting node");
        printf("\nEnter 8 for Exit\n");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch (ch){
            case 1: start = display(start);
                break;
            case 2: start = insert_beg(start);
                start = display(start);
                break;
            case 3: start = insert_end(start);
```

```
                start = display(start);
                break;
            case 4: start = insert_after(start);
                start = display(start);
                break;
            case 5: start = delete_beg(start);
                start = display(start);
                break;
            case 6: start = delete_end(start);
                start = display(start);
                break;
            case 7: start = delete_node(start);
                start = display(start);
                break;
            default:
                break;
            }
    } while (ch !=8);

    return 0;
}
```

Output:-

```
yashmmahajan19@penguin:~/Work/DSA$ ./LinkedList

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:2

 Enter the data : 1

 The elements of list are:-
 1

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:3

 Enter the data : 2

 The elements of list are:-
 1       2

Enter 1 for Displaying
Enter 2 for Inserting at beginning
```

```
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:3

 Enter the data : 4

 The elements of list are:-
 1       2       4

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:3

 Enter the data : 5

 The elements of list are:-
 1       2       4       5

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
```

```
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:4

 Enter the data : 3

 Enter the value after which the data has to be inserted : 2

 The elements of list are:-
 1       2       3       4       5

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:5

 The elements of list are:-
 2       3       4       5

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit
```

```
Enter your choice:6

 The elements of list are:-
 2       3       4

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:7

 Enter the node to be deleted:3

 The elements of list are:-
 2       4

Enter 1 for Displaying
Enter 2 for Inserting at beginning
Enter 3 for Inserting at end
Enter 4 for Inserting after given value
Enter 5 for Deleting first node
Enter 6 for Deleting last node
Enter 7 for Deleting node
Enter 8 for Exit

Enter your choice:8
yashmmahajan19@penguin:~/Work/DSA$
```

Yash Mahajan 04 SE IT-B 191061

Page :
Date : / /

Post - Experiments Exercise:

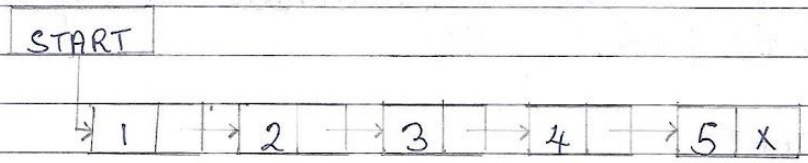
Question:

1. Explain types of linked list and state it's advantages:

→ A linked list is a linear collection of data elements ~~is which~~ called as nodes in which the linear representation is given by links from one node to the next node. The types of linked lists are :-

a) Singly linked list :-

A singly linked is the simplest type of linked list in which every node contains some data and a pointer to the next node of the same data type. By saying that the node contains a pointer to the next node, we mean that the node stores the address of the next in sequence. A singly linked list allows the traversal of data only in one way.

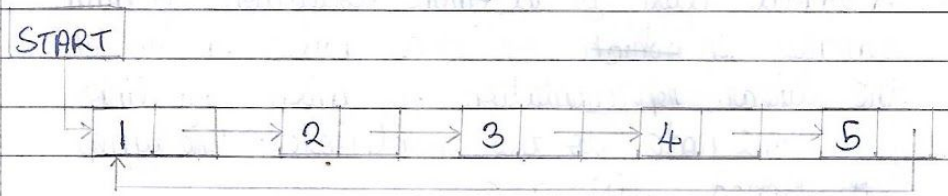


Singly linked list.

b) Circular linked list:-

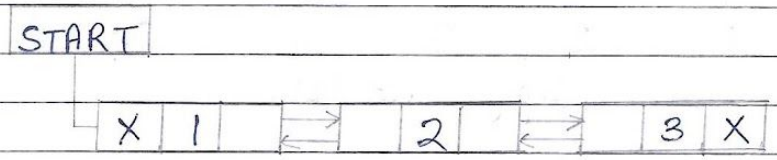
In a circular linked list, the last node contains the pointer of the first node. We can have a circular singly list as well as a circular doubly linked list. While

traversing a circular linked list, we can begin at any node and traverse the list in any direction until we reach the same node where we started. Thus a circular linked list has no end and no begining.



Circular linked list

c) Doubly linked lists :-
A doubly linked list or a two-way linked list ~~has~~ is a complex type of linked list which contains a pointer to the next node as well as the previous node in sequence. Therefore it consists of three parts - data, a pointer to the next node, and a pointer to the previous node.



Doubly linked list.

The advantages of linked list are :-
1) Linked lists are dynamic data structures

so it can grow and shrink at runtime by allocating and deallocating memory. So there is no need to give initial size of linked list.

2) Insertion and deletion of nodes is easier. Unlike arrays we don't have to shift each element after insertion or deletion of an element

3) ~~dfo~~ ~~fo~~ Size of linked list can increase or decrease at runtime so there is no memory wastage.

Conclusion :-

~~i)~~ ~~Summary of experiment~~ :-

The aim of the experiment to study linked lists was achieved by writing a C program to create a linked list and perform insertion and deletion operations on the same. Thus we have studied how to create and perform various operations on a linked list.

Linked lists are dynamic data structures which means they can grow or shrink. Insertion and deletion operation on liked list is easier compared to arrays. Memory wastage is less as we can increase or decrease the number of nodes in runtime.

Some important applications of linked list are implementation of stacks and queues and graphs, performing arithmetic operations of ~~graph~~

Page:
Date: / /

long integers, manipulation of polynomials by storing constants in the node of linked list.

Refrences:-

1. Ellis Horowitz, Sartaj Sahni; Fundamentals of Data Structures; Galgotia Publications; 2010

2. S.K Srivastava, Deepali Srivastava; Data structure through C in depth; BPB publications; 2011.

3. Reema Thareja; Data Structures using C; Oxford.