**Batch: A1 Roll No. 1911004**

**Experiment No. 1**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of Faculty:**

|  |
| --- |
| **Title:** Implementation of different operations on Linked List – concatenate, reverse, count no. of nodes. |

**Objective:** To understand the advantage of linked list over other structures like arrays in implementing the general linear list.

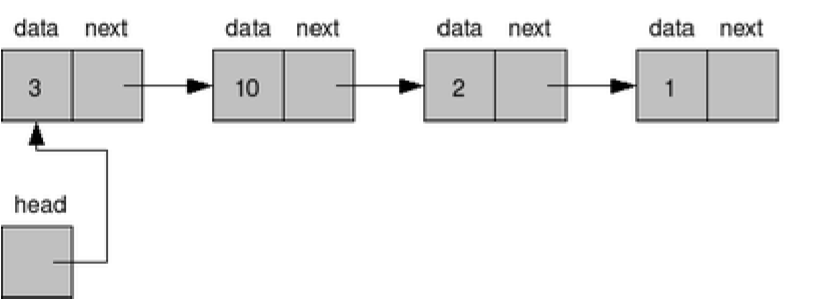
**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **2** | Use linear and non-linear data structure in domain like compiler construction, DBMS, etc. |

**Books/ Journals/ Websites referred:**

1. Data Structures Pseudocode Approach with C,Richard F.Gilberg & Behrouz A. Forouzan, second edition, CENGAGE learning.
2. Data Structures Using C & C++, Rajesh K. Shukla, Wiley- india.

**Abstract**: - A linear list is a list where each element has a unique successor. There are four common operations associated with linear list: insertion, deletion, retrieval, and traversal. Linear list can be divided into two categories: general list and restricted list. In general list the data can be inserted or deleted without any restriction whereas in restricted list there is restrictions for these operations. Linked list and arrays are commonly used to implement general linear list. A linked list is simply a chain of structures which contain a pointer to the next element. It is dynamic in nature. Items may be added to it or deleted from it at will :



A list item has a pointer to the next element, or to NULL if the current element is the tail (end of the list). This pointer points to a structure of the same type as itself. This Structure that contains elements and pointers to the next structure is called a Node.

**Related Theory: -**

In computer science, a linked list is a linear collection of data elements, whose order is not given by their physical placement in memory. Instead, each element points to the next.

It is a data structure consisting of a collection of nodes which together represent a sequence. In its most basic form, each node contains: data, and a reference to the next node in the sequence.

This structure allows for efficient insertion or removal of elements from any position in the sequence during iteration.

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.

**Advantages over arrays:**  
**Dynamic Data Structure**

Linked list is a dynamic data structure 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.

**No Memory Wastage**

As size of linked list can increase or decrease at run time so there is no memory wastage. In case of array there is lot of memory wastage, like if we declare an array of size 10 and store only 6 elements in it then space of 4 elements are wasted. There is no such problem in linked list as memory is allocated only when required.

**Insertion and Deletion**

Insertion and deletion of nodes is easier when compared with arrays. Unlike array here we don’t have to shift elements after insertion or deletion of an element. In linked list we just have to update the address present in next pointer of a node.

**Drawbacks of linked lists over arrays :**

**Memory Usage**

More memory is required to store elements in linked list as compared to array. Because in linked list each node contains a pointer and it requires extra memory for itself.

**Traversal**

Elements or nodes traversal is difficult in linked list. We can not randomly access any element as we do in array by index. For example if we want to access a node at position n then we have to traverse all the nodes before it. So, time required to access a node is large.

**Reverse Traversing**

In linked list reverse traversing is quiet difficult. In case of doubly linked list its easier but extra memory is required for back pointer hence wastage of memory.

**Linked List ADT Type Definition:**

Abstract Data type (ADT) is a type (or class) for objects whose behavior is defined by a set of value and a set of operations.

The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented.

It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.

It is called “abstract” because it gives an implementation independent view. The process of providing only the essentials and hiding the details is known as abstraction

.A list contains elements of same type arranged in sequential order and following operations can be performed on the list.

insert() – Insert an element at any position of the list.

delete() – Delete/Remove the first occurrence of any element from a non-empty list.

deleteAt() – Delete/Remove the element at a specified location from a non-empty list.

search() –check if element is in the list at any given position.

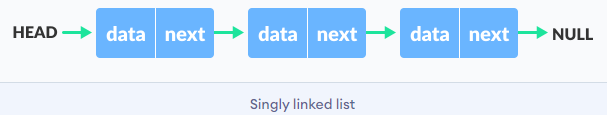
size() – Return the number of elements in the list.

isEmpty() – Return true if the list is empty, otherwise return false.

Destroy() - destroy entire ADT LINK LIST

**Chosen Linked list type and Related Theory: -**

**SINGLY LINK LIST**



A linked list is a sequence of data structures, which are connected via pointers

Link List is contains many nodes in it .A node is a collection of two sub-elements or parts. A **data** part that stores the element and a **next** part that stores the link to the next node.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

* Linked List contains a link element called head
* Each link carries a data field and a link field called next.
* Each link is linked with its next link using its next link.
* Last link carries a link as null to mark the end of the list.

Following are the basic operations supported by a list.

* **Insertion** − Adds an element at the beginning of the list.
* **Deletion** − Deletes an element at the beginning of the list.
* **Display** − Displays the complete list.
* **Search** − Searches an element using the given key.
* **Delete** − Deletes an element using the given key.

**Linked list implementation algorithm:**

1. **Start**
2. **Create struct node{**

**int data;**

**struct node \*next;**

**};**

1. **Struct node \* head,temp,p1,p2;**
2. **Insert(**

**struct node \*newN;**

**head=0;**

**repeat following till ans!=0**

**newN=(struct node\*)malloc(sizeof(struct node));**

**newN->next=0;**

**take in newN->data**

**if(head==0)**

**head=newN;**

**temp=newN;**

**else**

**temp->next=newN;**

**temp=newN;**

**ask if user wants to enter more data**

1. **Delete()**

**temp=head;**

**p1=head;**

**ask user for delete element**

**int del,flag=0,i=0;**

**while(temp!=0)**

**if((temp->data)==del)**

**flag=1;**

**p1->next=temp->next;**

**p2=temp->next;**

**free(temp);**

**break;**

**p1=temp;**

**temp=temp->next;**

**p2=temp->next;**

**i++;**

**if(flag!=0)**

**printf("\n The %d element deleted at position %d\n",del,i+1);**

**else**

**printf("\n The %d element not in list ",del);**

1. **Search()**

**temp=head;**

**int search,flag=0,i=0;**

**search is asked from user**

**while(temp!=0)**

**if((temp->data)==search)**

**flag=1;**

**break;**

**temp=temp->next;**

**i++;**

**if(flag!=0)**

**printf("\n The %d element found at position %d\n",search,i+1);**

**else**

**printf("\n The %d element not in list ",search);**

1. **Display()**

**temp=head;**

**printf("\t");**

**if(temp!=0)**

**while(temp!=0){**

**printf(temp->data);**

**temp=temp->next;**

**printf(" NULL");**

**else**

**printf("No Elements");**

1. **Destroy()**

**temp=head;**

**while(temp!=0){**

**head=head->next;**

**free(temp);**

**temp=head;**

1. **main()**

**in do while loop ask for user ‘s choice & do needed operation**

1. **End**

**Source Code:**

**#include <stdio.h>**

**struct node{**

**int data;**

**struct node \*next;**

**}; //user defined struct node**

**struct node \*head,\*temp,\*p1,\*p2;**

**void insert() //insert the nodes**

**{**

**int ans;**

**struct node \*newN;**

**head=0;**

**do{**

**newN=(struct node\*)malloc(sizeof(struct node));**

**newN->next=0;**

**printf("Enter the data = ");**

**scanf("%d",&newN->data);**

**if(head==0)**

**{**

**head=newN;**

**temp=newN;**

**}**

**else{**

**temp->next=newN;**

**temp=newN;**

**}**

**printf("Do you want to add more elements ? [ 1=YES / 0=NO ] : ");**

**scanf("%d",&ans);**

**}while(ans!=0);**

**}**

**void display(){**

**temp=head;**

**printf("\t");**

**if(temp!=0){**

**while(temp!=0){**

**printf("%d --> ",temp->data);**

**temp=temp->next;**

**}printf(" NULL");**

**}else{**

**printf("No Elements");**

**}**

**}**

**void deletion(){ //delete the nodes**

**temp=head;**

**p1=head;**

**printf("Enter the elements data you want to delete = ");**

**int del,flag=0,i=0;**

**scanf("%d",&del);**

**while(temp!=0){**

**if((temp->data)==del){**

**flag=1;**

**p1->next=temp->next;**

**p2=temp->next;**

**free(temp);**

**break;**

**}**

**p1=temp;**

**temp=temp->next;**

**p2=temp->next;**

**i++;**

**}**

**if(flag!=0)**

**printf("\n The %d element deleted at position %d\n",del,i+1);**

**else**

**printf("\n The %d element not in list ",del);**

**}**

**void search(){ //search the node**

**temp=head;**

**printf("Enter the elements data you want to search = ");**

**int search,flag=0,i=0;**

**scanf("%d",&search);**

**while(temp!=0){**

**if((temp->data)==search){**

**flag=1;**

**break;**

**}**

**temp=temp->next;**

**i++;**

**}**

**if(flag!=0)**

**printf("\n The %d element found at position %d\n",search,i+1);**

**else**

**printf("\n The %d element not in list ",search);**

**}**

**void destroy(){ //destroy entire linklist**

**temp=head;**

**while(temp!=0){**

**head=head->next;**

**free(temp);**

**temp=head;**

**}**

**}**

**int main(void) { //main() to call all function**

**int ch=-1;**

**do{**

**printf("\nOPERATIONS ON SINGLY LINK - LISTS\n");**

**printf("1-INSERT\n2-DELETE\n3-SEARCH\n");**

**printf("4-DISPLAY\n5-DESTROY\n0-EXIT\n");**

**printf("Enter your choice = ");**

**scanf("%d",&ch);**

**switch(ch){**

**case 1:**

**insert();**

**display();**

**break;**

**case 2:**

**deletion();**

**display();**

**break;**

**case 3:**

**search();**

**display();**

**break;**

**case 4:**

**display();**

**break;**

**case 5:**

**destroy();**

**display();**

**break;**

**case 0:**

**printf("Exiting ...\n");**

**break;**

**default:**

**printf("Invalid Choice\n");**

**break;**

**}**

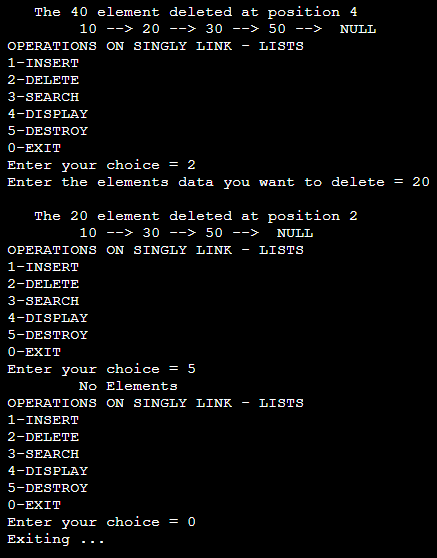
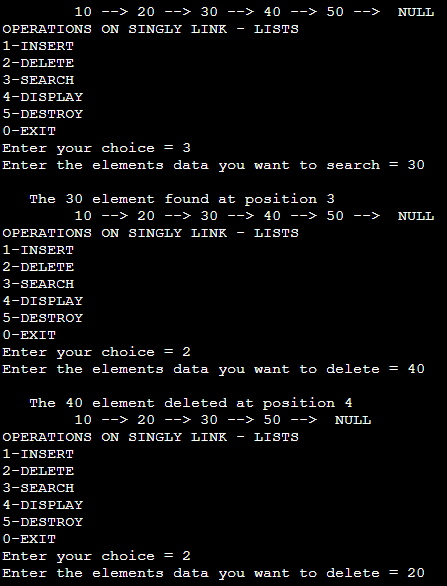
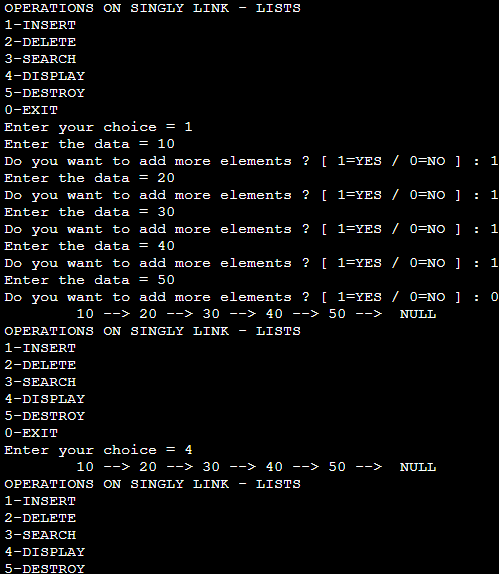
**}while(ch!=0);**

**printf("\n\n\n");**

**return 0;**

**}**

**Output:**



**Post Lab questions**

**1. Explain your program logic, classes and methods used.**

First we create a struct with the int data member & struct node \* next;

**Struct node{**

**int data ;**

**struct node\* next;**

**};**

Using Menu Driven Approach we ask user to select any of the operation

1. **insert() - insert the node with data in link list**

* **we ask user to enter the data till he wants**
* **we then call display to display that link list**

1. **deletion() – delete the given node at any location**

* **we ask user to enter the data to delete**
* **this element is searched in entire linked - list**
* **if found it is deleted**
* **else not found Message display**

1. **searching() – find & search the element entered in link list**

* **we ask user to enter the data to search**
* **this element is searched in entire linked - list**
* **if found print location**
* **else not found Message display**

1. **displaying() – display entire link list**

* **temp = head**
* **repeat till temp != 0**

**print temp-> data**

1. **destroy() – destroy entire link list**

* **temp = head**
* **repeat till temp != 0**

**print free(temp)**

1. **exit() – exit from the given program**

* **break;**

1. **main() – to call all above function & run menu driven program**

**//call to all function in menu driven program**

1. **Explain the Importance of the approach followed by you.**

**This program was implemented using menu driven approach . We had to also perform various task in this experiment too that was main reason to follow the Menu Driven Approach.**

**It was assumed by me that the data in the link list would be integer only.**

**The various function created where display() ,search() , delete() , insert() , exit () &destroy().**

**In main using menu driven approach user is asked for his choice & that function is called**

**insert() - insert the node with data in link list**

**deletion() – delete the given node at any location**

**searching() – find & search the element entered in link list**

**displaying() – display entire link list**

**destroy() – destroy entire link list**

**exit() – exit from the given program**

**// this is an easy approach followed using SINGLY LINK LIST**

**Conclusion:**

We understood the concept of the Link – List & implemented it in C ; getting the necessary and required correct output.