**Batch:B4**

**RollNo.:16010122221**

**ExperimentNo. 2**

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

Title: Implementation of differentoperationsonLinkedList–creation, insertion,

deletion,traversal,searchinganelement

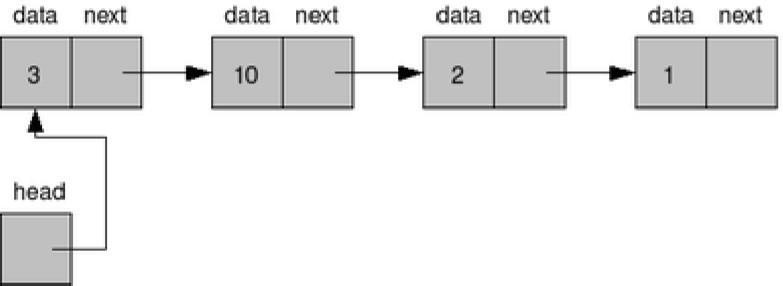
Objective: To understand the advantage of linked list over other structures like arrays in implementing the general linear list

Expected Outcome of Experiment:

|  |  |
| --- | --- |
| CO | Outcome |
| CO1 | Tounderstandtheadvantageoflinkedlistoverotherstructureslikearraysinimplementingthe general linear list |

Books/Journals/Websitesreferred:[www.geeksforgeeks.com](http://www.geeksforgeeks.com/)[www.javapoint.com](http://www.javapoint.com/)[www.leetcode.com](http://www.leetcode.com/)Introduction:

Alinearlistisalistwhereeachelementhasauniquesuccessor.Therearefourcommonoperations associated with linear list: insertion, deletion, retrieval, and traversal. Linear lists can be divided into two categories: general list and restricted list. In general, the data can be inserted or deleted without any restriction whereas in restricted list there is restrictions for these operations. Linked lists 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 It has a pointer to the next element, or to NULL if the current element is the tail(endofthelist).Thispointerpointstoastructureofthesametypeasitself.ThisStructurethatcontains 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 insert nor removal of elements for 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

LinkedList:

Linked List is an *Abstract Data Type (ADT)* that holds a collection of **Nodes,** the nodes can be accessed in a sequential way. Linked List doesn’t provide random access to a Node.

Usually, those Nodes are connected to the next node and/or with the previous one, this gives the linked effect. When the Nodes are connected with only the next pointer list

is called Singly Linked List and when it’s connected by the next and previous the list is called Doubly Linked List.

Value definition:

Abstract Typedef linked list/node<<int number, struct linked list \*next>>Condition: None

Operator Definition

1. Abstract void beginsert<< >>Pre-condition:none

Post-condition: New node of linked list are linked with it’s next node; and tail apparatus, successively creating circular linked list .Take she nondeath argument

1. Abstract void lastinsert<< >>Pre-condition:none

Post-condition: Node of linked list are linked with it’s tail as ptr thus and tail toitshead

1. Abstract linked\_list/node begin\_delete<< >>Pre-condition:None

Post-condition:Thefunctionthendeletesthe nodeaptlyatthebeginning.

1. Abstract linked\_list/node last\_delete<< >>Pre-condition:None

Post-condition: The function then deletes the node aplly at the ending.

1. Abstractvoiddisplay<<>>

Pre-condition : The next node for tail node must point to head

Post-condition: All nodes of the linked list is printed from head to tail; demonstrating that the linked list previously created is indeed ,circular linked

1. Abstract linked list/node searching<<node \*head>>Pre-condition:None

Post-condition: Requests the user to enter the value of the data stored at a particular node which is to be searched. The function then searches the LinkedList to find the data. Returns the index of the node where the data is stored.

Algorithm for creation, insertion, deletion, traversal and searching an element in assigned LinkedList type:

Insertion Operation

beginsert(): Begincreateanewnodenode

-> data := data if thelistisempty,thenhead

:= node next of node =headelsetemp:=head

while next of temp is not head, dotemp := next of temp done next ofnode:=headnextoftemp:=nodehead:= node end if

End

Deletion Operation

deleteFirst():Begin

if head is null, then it isUnderflow and return else ifnextofhead=head,thenhead

:=nulldeallocateheadelseptr

:=head

whilenext ofptris not head,do

ptr := next of ptr next ofptr = next of headdeallocate head head :=nextof ptr end if

End

Display List Operation

display ():Begin

if head is null, then Nothing toprint and return else ptr := headwhile next of ptr is not head, dodisplaydataofptrptr:=nextofptrdisplay dataofptrendif

End

Searching:

Step1: SET PTR =HEAD

Step2: Set I=0

STEP 3: IF PTR = NULL WRITE"EMPTYLIST"

GOTO STEP 8ENDOF IF

STEP4:REPEATSTEP5TO7 UNTILPTR! =NULL

STEP 5: if ptr → data = itemwritei+1 End of IF

STEP6: I=I+1

STEP 7: PTR = PTR → NEXT[ENDOFLOOP]

STEP8: EXIT

Implementation of an application using linked lists:

Application : Addition of polynomials Code:

#include <bits/stdc++.h>usingnamespacestd;

//Nodestructurecontainingpowerandcoefficientof

// variablestructNode{

int coeff;intpow;

structNode\*next;

};

// Functiontocreatenewnode

voidcreate\_node(int x,inty,struct Node\*\*temp)

{

struct Node \*r, \*z;z = \*temp;

if(z ==NULL) {

r = (struct Node\*)malloc(sizeof(struct Node));r->coeff =x;

r->pow= y;

\*temp =r;

r->next = (struct Node\*)malloc(sizeof(struct Node));r= r->next;

r->next=NULL;

}

else{

}

}

r->coeff = x;r->pow= y;

r->next = (struct Node\*)malloc(sizeof(struct Node));r= r->next;

r->next=NULL;

//FunctionAddingtwopolynomialnumbers

voidpolyadd(struct Node\*poly1,structNode\*poly2,

structNode\*poly)

{

while(poly1->next&&poly2->next) {

//Ifpowerof1stpolynomial isgreaterthen2nd,

// then store 1st as it is and move its pointerif(poly1->pow> poly2->pow) {

poly->pow = poly1->pow;poly->coeff = poly1->coeff;poly1= poly1->next;

}

//Ifpower of2ndpolynomial isgreaterthen1st,

// then store 2nd as it is and move its pointerelseif (poly1->pow< poly2->pow) {

poly->pow = poly2->pow;poly->coeff = poly2->coeff;poly2= poly2->next;

}

//Ifpower ofbothpolynomialnumbersissamethen

// add their coefficientselse{

poly->pow=poly1->pow;

poly->coeff = poly1->coeff + poly2->coeff;poly1= poly1->next;

poly2= poly2->next;

}

// Dynamically create new nodepoly->next

= (struct Node\*)malloc(sizeof(struct Node));poly =poly->next;

poly->next=NULL;

}

while (poly1->next || poly2->next) {if(poly1->next) {

poly->pow = poly1->pow;poly->coeff = poly1->coeff;poly1= poly1->next;

}

if(poly2->next) {

poly->pow = poly2->pow;poly->coeff = poly2->coeff;poly2= poly2->next;

}

poly->next

= (struct Node\*)malloc(sizeof(struct Node));poly =poly->next;

poly->next=NULL;

}

}

// DisplayLinkedlist

voidshow(structNode\*node)

{

while(node->next !=NULL){

printf("%dx^%d", node->coeff, node->pow);node= node->next;

if(node->coeff >= 0) {

if(node->next!=NULL)

printf("+");

}

}

}

// Driver codeintmain()

{

struct Node \*poly1 = NULL, \*poly2 = NULL, \*poly = NULL;int num;

intp,q;

cout << "Enter degree: ";cin>> num;

//Createfirstlist

for(int i=num;i>=0;i--)

{

cout<<"Forpolynomial1:\n";

cout<<"Enter coefficient: ";cin>>p;

q=i;

create\_node(p,q,&poly1);

}

cout<<"\n\n";

// Createsecondlist

for(int j=num;j>=0;j--)

{

cout<<"Forpolynomial2:\n";

cout<<"Enter coefficient: ";cin>>p;

q=j;

create\_node(p,q,&poly2);

}

cout<<"\n\n";

printf("1st Number: ");show(poly1);

printf("\n2nd Number: ");show(poly2);

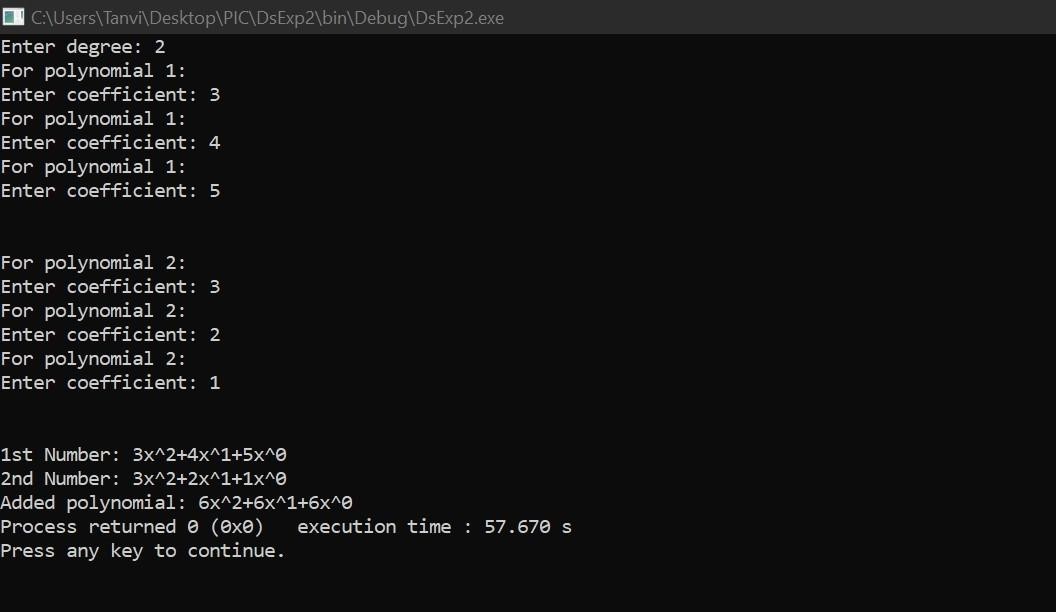
poly=(structNode\*)malloc(sizeof(structNode));

// Function add two polynomial numberspolyadd(poly1,poly2, poly);

// Display resultant Listprintf("\nAdded polynomial: ");show(poly);

return0;

}

Output:

Conclusion:

We have successfully understood algorithms for creation, traversal, insertion, deletion and searching an element in LinkedList. Written code for application of linked list.

Postlabquestions:

1. Writethedifferencesbetweenlinkedlistandlineararray

|  |  |
| --- | --- |
| **LinearArray** | **LinkedList** |
| Anarrayisacollectionofelementsofasimilardata type. | A linked list is a collection of objectsknown as a node where node consists oftwoparts, i.e., data and address. |
| Array works with a static memory. Herestaticmemorymeansthatthememorysizeis fixed and cannot be changed at the runtime. | The Linked list works with dynamicmemory.Here,dynamicmemorymeansthat the memory size can be changed atthe run time according to ourrequirements. |
| Array takes more time while performinganyoperationlikeinsertion,deletion,etc. | Linked list takes less time whileperforminganyoperationlikeinsertion,deletion,etc. |
| Inthecaseofanarray,memoryisallocatedat compile-time. | In the case of a linked list, memory isallocatedat run time. |
| Memoryutilizationisinefficientinanarray | Memoryutilizationisefficientwhenitcomesto linked lists |

1. Namesomeapplications whichusealinkedlist.

* Implementationofstacks andqueues.
* Implementation of graphs: Adjacency list representation of graphs is most popularwhichis uses alinked list to storeadjacent vertices.
* Dynamicmemoryallocation: Weusealinked listof freeblocks.
* Maintainingadirectoryofnames.
* Image viewer – Previous and next images are linked, hence can be accessed by nextandprevious button.
* Usefulforimplementationof aqueue.We can maintainapointer tothelastinsertednodeand the front canalways beobtainedas nextof last