AIM: To create a Singly Linked List for student data and perform a. insertion b. deletion c. search d. modify

THEORY:

Singly linked list is a basic linked list type. Singly linked list is a collection of nodes linked together in a sequential way where each node of singly linked list contains a data field and an address field which contains the reference of the next node. Singly linked list can contain multiple data fields but should contain at least single address field pointing to its connected next node.

To perform any operation on a linked list we must keep track/reference of the first node which may be referred by head pointer variable. In singly linked list address field of last node must contain a NULL value specifying end of the list.

Singly linked list is probably the easiest data structure to implement. Insertion and deletion of element can be done easily. Insertion and deletion of elements doesn't require movement of all elements when compared to an array.

Each node of a singly linked list follows a common basic structure. In a node we can store more than one data fields but we need at least single address field to store the address of next connected node.

struct node {

int data; // Data

struct node \* next; // Address

};

SOURCE CODE:

/\*

PROGRAM BY ANOM DEVGUN

GR: 21810017

Successfully implemented SLL and its various operations with edge cases resolved.

\*/

#include<iostream>

#include<cstdio>

#include<cstdlib>

using namespace std;

struct node

{

int data;

struct node \*next;

}\*head = NULL;

void create() //Function to create nodes at end of list

{

struct node \*nn,\*t;

nn= new node;

cout<<"Enter Data\n";

cin>>nn->data;

nn->next = NULL;

if(head == NULL)

head = nn;

else

{

t=head;

while(t->next != NULL)

t = t->next;

t->next = nn;

}

}

void beg()

{

struct node \*nn,\*t= NULL;

nn= new node;

cout<<"Enter Data\n";

cin>>nn->data;

if(head == NULL)

{

head = nn;

nn->next = NULL;

}

else{

nn->next = head;

head = nn;

}

}

void disp() //Function to display data of nodes;

{

struct node \*t;

t=head;

if(t == NULL)

{

cout<<"List is empty\n";

return;

}

while(t != NULL)

{

cout<<"Data: "<<t->data<<"\n";

t=t->next;

}

}

void sear() //Function to search data in list

{

struct node \*t;

int dat,sr=1;

t = head;

cout<<"Enter Data to be searched\n";

cin>>dat;

if(t == NULL)

{

cout<<"The list is empty.\n Now exiting to main menu.\n";

return;

}

while(t->data != dat)

{

t = t->next;

sr++;

if(t -> data != dat && t -> next == NULL)

{

cout<<"Search data not found in list.\n";

return;

}

}

cout<<dat<<" Found at node "<<sr<<"\n";

}

void del() //Function to delete data from list

{

struct node \*t,\*a;

int dat,sr=1;

t = head;

cout<<"Enter Data to be searched and deleted\n";

cin>>dat;

if(t == NULL)

{

cout<<"The list is empty.\n Now exiting to main menu.\n";

return;

}

while(t->data != dat)

{

a=t;

t = t->next;

sr++;

if(t -> data != dat && t -> next == NULL)

{

cout<<"Search data not found in list.\n";

return;

}

}

if(head->data == dat )

{

head = head->next;

}

cout<<dat<<" Found at node "<<sr<<"\n";

cout<<"Node and data deleted\n";

a->next = t->next;

delete(t);

}

void mod() //Function to modify data in list

{

struct node \*t;

int dat,sr=1;

t = head;

cout<<"Enter Data to be modified\n";

cin>>dat;

if(t == NULL)

{

cout<<"The list is empty.\n Now exiting to main menu.\n";

return;

}

while(t->data != dat)

{

t = t->next;

sr++;

if(t -> data != dat && t -> next == NULL)

{

cout<<"Search data not found in list.\n";

return;

}

}

cout<<dat<<" Found at node "<<sr<<"\n";

cout<<"Enter new data\n";

cin>>t->data;

}

int main()

{

int ch;

while(1)

{

cout<<"\n1)For Entering data(at list end)\n";

cout<<"2)For Displaying linked list data\n";

cout<<"3)for adding at beginning of list\n";

cout<<"4)To search data from list\n";

cout<<"5)To delete data from list\n";

cout<<"6)To modify data in list \n";

cout<<"7)To exit\n";

cin>>ch;

cout<<"\n";

switch(ch)

{

case 1:

create();

break;

case 2:

disp();

break;

case 3:

beg();

break;

case 4:

sear();

break;

case 5:

del();

break;

case 6:

mod();

break;

case 7:

cout<<"\nNow exiting\n";

exit(1);

break;

default:

cout<<"Invalid choice.\n";

}

}

}

Time Complexity: O(n).

CONCLUSION:

We studied the fundamentals of linked lists. Linked lists are dynamic data structures where nodes can be added, deleted or updated .It is one of most efficient data structure to implement when traversing in one direction is required.