**PRACTICAL FILE**

**ON**

**DATA AND FILE STRUCTURES USING C**

**MASTER OF COMPUTER APPLICATIONS**

**REGULAR PROGRAM**

**Offered by**

****

***Indira Gandhi Delhi Technical University for Women***

**(Established by Govt. of Delhi Wide Act 09 of 2012)**

**Submitted to :**

Ms.Nidhi Arora

**Submitted by :**

Bhavana Sinha

02804092018

MCA 1st Year

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.no** | **Topic** | **Date** | **Signature** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

OPERATIONS ON 1D ARRAY

1. Insert n element

//1.Insert n elements in an array

#include<stdio.h>

int main()

{

int i=0,n,a[20];

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

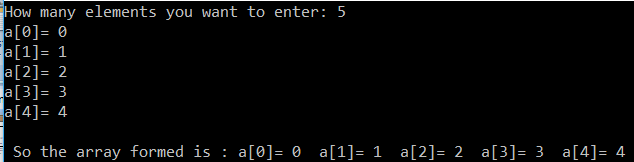
printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

return 0;

}



2. Delete an element

//delete an element of 1d array

#include<stdio.h>

int main()

{

int i=0,n,a[20],pos;

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

printf("\n Enter the position from where you want to delete the element : ");

scanf("%d",&pos);

for(i=pos;i<n;i++)

a[i]=a[i+1];

n--;

printf("\n So the array formed after deletion of an element is : ");

for(i=0;i<n;i++)

{

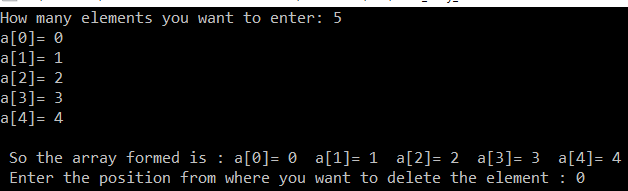
printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

return 0;

}



3. Search for a given value-i)Linear

//search an element in 1d array

#include<stdio.h>

int main()

{

int i=0,n,a[20],ele,pos=-1,found=0;

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

printf("\n Enter the element you want to search : ");

scanf("%d",&ele);

for(i=0;i<n;i++)

{

if(a[i]==ele)

{

found=1;

pos=i;

printf("\n %d is found at position %d in the given array",ele,pos);

break;

}

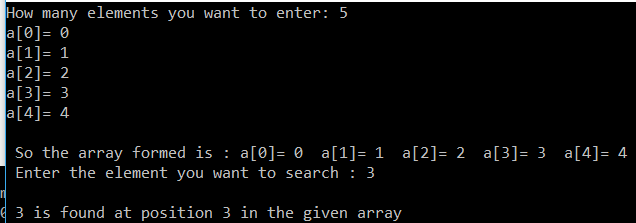
}

if(found==0)

printf("\n %d is not found in the given array",ele);

return 0;

}



ii)Binary

//binary search

#include<stdio.h>

int main()

{

int i=0,n,a[20],ele,pos=-1,found=0,first,last,mid;

printf("How many elements you want to enter: ");

scanf("%d",&n);

printf("\nPlease!!enter a sorted array\n");//precondition for binary search

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

printf("\n Enter the element you want to search : ");

scanf("%d",&ele);

first=0,last=n-1;

while(first<=last)

{

mid=(first+last)/2;//has to be inside while loop

if(a[mid]==ele)

{

found=1;

pos=mid;

printf("\n %d is found at position %d in the given array",ele,pos);

break;

}

if(ele<a[mid])//search in left portion

last=mid-1;

else if(ele>a[mid])//search in right portion

first=mid+1;

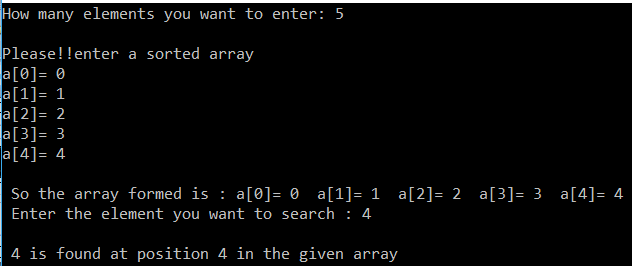
}

if(first>last && found==0)

printf("\n %d is not found in the given array",ele);

return 0;

}



4.Sort the value using i)bubble

//bubble sort

#include<stdio.h>

int main()

{

int i=0,j,temp,n,a[20],pos;

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

//bubble sort logic

for(i=0;i<n;i++)

{

for(j=0;j<n-i;j++)

{

if(a[j]>a[j+1])

{//swapping

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\n So the array formed after sorting in ascending order is : ");

for(i=0;i<n;i++)

{

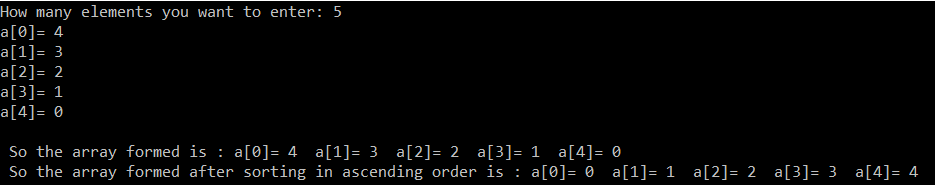
printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

return 0;

}



ii)selection

#include <stdio.h>

int main()

{

int array[20], n, c, d, position, swap;

printf("Enter number of elements\n");

scanf("%d", &n);

printf("Enter %d integers\n", n);

for (c = 0; c < n; c++)

scanf("%d", &array[c]);

for (c = 0; c < (n - 1); c++)

{

position = c;

for (d = c + 1; d < n; d++)

{

if (array[position] > array[d])

position = d;

}

if (position != c)

{

swap = array[c];

array[c] = array[position];

array[position] = swap;

}

}

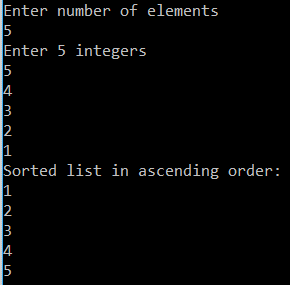
printf("Sorted list in ascending order:\n");

for (c = 0; c < n; c++)

printf("%d\n", array[c]);

return 0;

}



iii)insertion

#include <stdio.h>

int main()

{

int n, a[20], i, j, temp;

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

for (i = 1 ; i <= n - 1; i++) {

j = i;

while ( j> 0 && a[j-1] > a[j]) {

temp = a[j];

a[j] = a[j-1];

a[j-1] = temp;

j--;

}

}

printf("\n So the array formed after sorting in ascending order is : ");

for(i=0;i<n;i++)

{

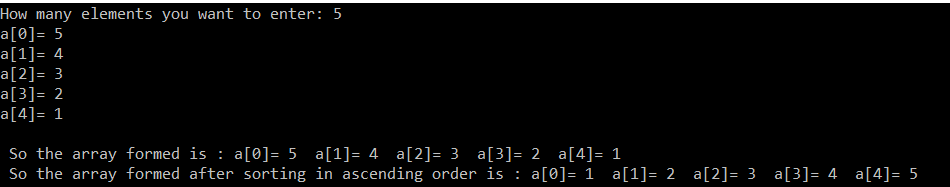
printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

return 0;

}



5.Transverse/print the elements of array

#include<stdio.h>

int main()

{

int i=0,n,a[20];

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

return 0;

}

6.Count the frequency of each element in the array

#include <stdio.h>

int main()

{

int arr[100], freq[100];

int size, i, j, count;

/\* Input size of array \*/

printf("Enter size of array: ");

scanf("%d", &size);

/\* Input elements in array \*/

printf("Enter elements in array: ");

for(i=0; i<size; i++)

{

scanf("%d", &arr[i]);

/\* Initially initialize frequencies to -1 \*/

freq[i] = -1;

}

for(i=0; i<size; i++)

{

count = 1;

for(j=i+1; j<size; j++)

{

/\* If duplicate element is found \*/

if(arr[i]==arr[j])

{

count++;

/\* Make sure not to count frequency of same element again \*/

freq[j] = 0;

}

}

/\* If frequency of current element is not counted \*/

if(freq[i] != 0)

{

freq[i] = count;

}

}

/\*

\* Print frequency of each element

\*/

printf("\nFrequency of all elements of array : \n");

for(i=0; i<size; i++)

{

if(freq[i] != 0)

{

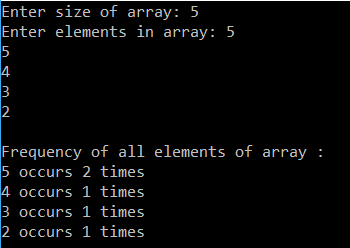
printf("%d occurs %d times\n", arr[i], freq[i]);

}

}

return 0;

}



7. Find the kth smallest and kth largest element in the array

i)//kth smallest

#include<stdio.h>

int main()

{

int i=0,j,temp,n,a[20],pos,k;

printf("How many elements you want to enter: ");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("a[%d]= ",i);

scanf("%d",&a[i]);

}

printf("\n So the array formed is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

//bubble sort logic

for(i=0;i<n;i++)

{

for(j=0;j<n-i;j++)

{ //if(a[j]<a[j+1])

if(a[j]>a[j+1])

{//swapping

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\n So the array formed after sorting in ascending order is : ");

for(i=0;i<n;i++)

{

printf("a[%d]= %d ",i,a[i]);

printf(" ");

}

printf("\nEnter the value of K for kth smallest: ");

scanf("%d",&k);

for(i=0;i<n;i++)

{

if(i==k)

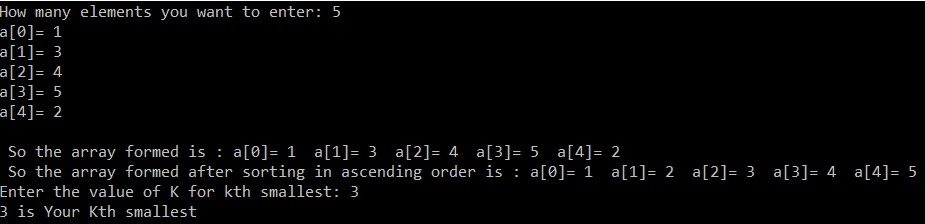
{printf("%d is Your Kth smallest",a[i-1]);

break;}

}

return 0;

}



ii)// kth largest

#include <stdio.h>

void main ()

{

int number[30];

int i, j, a, n,k;

printf("How many elements you want to enter : ");

scanf("%d", &n);

printf("Enter the numbers : \n");

for (i = 0; i < n; ++i)

scanf("%d", &number[i]);

/\* sorting begins ... \*/

for (i = 0; i < n; ++i)

{

for (j = i + 1; j < n; ++j)

{

if (number[i] < number[j])

{

a = number[i];

number[i] = number[j];

number[j] = a;

}

}

}

printf("The numbers arranged in descending order are given below :\n");

for (i = 0; i < n; ++i)

{

printf("%d\n", number[i]);

printf(" ");

}

printf("\nEnter the value of K for kth largest: ");

scanf("%d",&k);

for(i=0;i<n;i++)

{

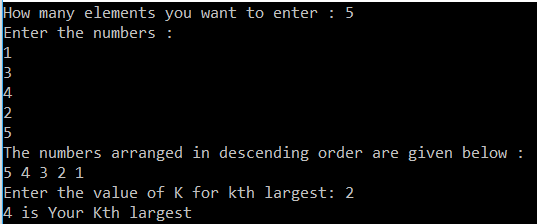
if(i==k)

{printf("%d is Your Kth largest",number[i-1]);

break;}

}

}



8. Merge two sorted arrays

//assumed that user will enter array in ascending order

#include <stdio.h>

void main()

{

int array1[50], array2[50], array3[100], m, n, i, j, k = 0;

printf("\n Enter size of array Array 1: ");

scanf("%d", &m);

printf("\n Enter sorted elements of array 1: \n");

for (i = 0; i < m; i++)

{

scanf("%d", &array1[i]);

}

printf("\n Enter size of array 2: ");

scanf("%d", &n);

printf("\n Enter sorted elements of array 2: \n");

for (i = 0; i < n; i++)

{

scanf("%d", &array2[i]);

}

i = 0;

j = 0;

while (i < m && j < n)

{

if (array1[i] < array2[j])

{

array3[k] = array1[i];

i++;

}

else

{

array3[k] = array2[j];

j++;

}

k++;

}

if (i >= m)

{

while (j < n)

{

array3[k] = array2[j];

j++;

k++;

}

}

if (j >= n)

{

while (i < m)

{

array3[k] = array1[i];

i++;

k++;

}

}

printf("\n After merging: \n");

for (i = 0; i < m + n; i++)

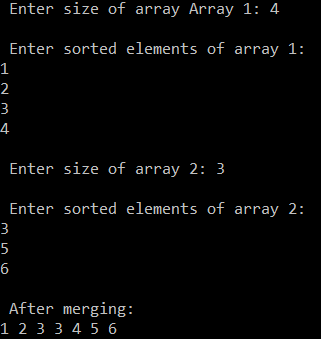
{

printf("%d", array3[i]);

printf(" ");

}

}



9. Concatenate two arrays(merging two unsorted array-just appending 2nd array to 1st)

#include<stdio.h>

void main()

{

int a1[20],a2[20],a3[40];

int i,n1,n2,n3,index=0;

printf("How many elements you want to enter in array 1: ");

scanf("%d",&n1);

for(i=0;i<n1;i++)

{

printf("a1[%d]= ",i);

scanf("%d",&a1[i]);

}

printf("\n So the array1 formed is : ");

for(i=0;i<n1;i++)

{

printf("%d ",a1[i]);

printf(" ");

}

printf("\nHow many elements you want to enter in array 2: ");

scanf("%d",&n2);

for(i=0;i<n2;i++)

{

printf("a2[%d]= ",i);

scanf("%d",&a2[i]);

}

printf("\n So the array2 formed is : ");

for(i=0;i<n2;i++)

{

printf("%d ",a2[i]);

printf(" ");

}

n3=n1+n2;//no. of elements in array3

for(i=0;i<n1;i++)

{

a3[index]=a1[i];

index++;

}

for(i=0;i<n2;i++)

{

a3[index]=a2[i];

index++;

}

printf("\n So the array3-concatenated array formed is : ");

for(i=0;i<n3;i++)

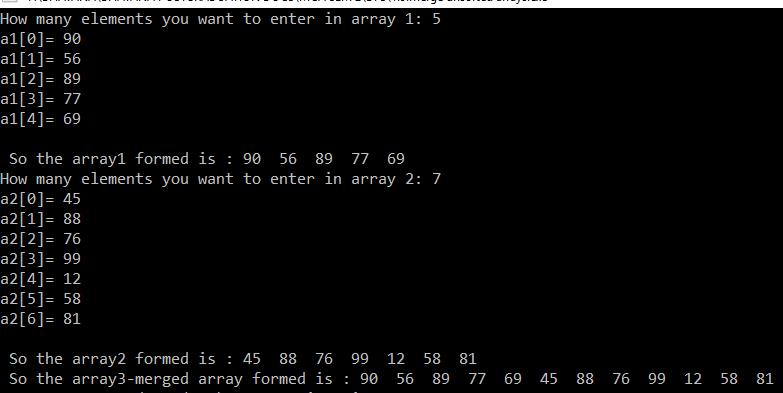
{

printf("%d ",a3[i]);

printf(" ");

}

}



10. Reverse the contents of the array

#include <stdio.h>

int main()

{

int n, i, j, a[20], b[20];

printf("Enter the number of elements in array: ");

scanf("%d", &n);

printf("Enter array elements\n");

for (i = 0; i < n ; i++)

scanf("%d", &a[i]);

//Copying elements into array b starting from end of array a

for (i = n - 1, j = 0; i >= 0; i--, j++)

b[j] = a[i];

//Copying reversed array into the original.Here we are modifying original array, this is optional.

for (i = 0; i < n; i++)

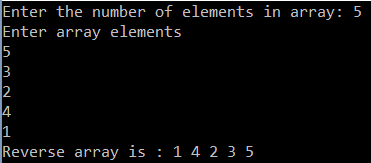
a[i] = b[i];

printf("Reverse array is : ");

for (i = 0; i < n; i++)

{printf("%d", a[i]);printf(" ");}

return 0;}



OPERATIONS ON 2D ARRAY

1.Create & Display the 2d array 2. Add 3.Subtract 4.multiply two 2d arrays 5.Transpose a 2d array 6. Search the given element and print its index if found.

#include<stdio.h>

void read\_matrix(int mat[2][2],int,int);

void sum\_matrix(int mat1[2][2],int mat2[2][2],int,int);

void diff\_matrix(int mat1[2][2],int mat2[2][2],int,int);

void mul\_matrix(int mat1[2][2],int mat2[2][2],int,int);

void transpose\_matrix(int mat[2][2],int ,int);

void display\_matrix(int mat[2][2],int r ,int c);//r=rows,c=column

int search(int mat[2][2], int r, int c,int x);//x is elemnt to be found

int main()

{

int op,row,col,ch,x;

int mat1[2][2],mat2[2][2];

do{

printf("\nPERFORM FOLLOWING OPERATIONS ON 2D ARRAY");

printf("\n1.Read two matrices");

printf("\n2.Add two matrices");

printf("\n3.Subtract two matrices");

printf("\n4.Multiply two matrices");

printf("\n5.Transpose of matrix");

printf("\n6.Display two matrices");

printf("\n7.Search for an elemnt in the matrix");

//printf("\n Exit");

printf("\nEnter your option :");

scanf("%d",&op);

switch(op)

{

case 1: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

break;

case 2: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

printf("\n first matrix :\n");

display\_matrix(mat1,row,col);

printf("\n second matrix :\n ");

display\_matrix(mat2,row,col);

printf("\nSum of two matrices :\n");

sum\_matrix(mat1,mat2,row,col);

break;

case 3: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

printf("\n first matrix :\n");

display\_matrix(mat1,row,col);

printf("\n second matrix :\n ");

display\_matrix(mat2,row,col);

printf("\nDifference of two matrices :\n");

diff\_matrix(mat1,mat2,row,col);

break;

case 4: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

printf("\n first matrix :\n");

display\_matrix(mat1,row,col);

printf("\n second matrix :\n ");

display\_matrix(mat2,row,col);

printf("\nProduct of two matrices :\n");

if(col==row)

mul\_matrix(mat1,mat2,row,col);

else printf("\n To multiply 2 matrices,number of col in 1st matrix should be equal to no. of rows in 2nd matrix");

break;

case 5: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

printf("\n first matrix :\n");

display\_matrix(mat1,row,col);

printf("\n second matrix :\n ");

display\_matrix(mat2,row,col);

printf("\nTranspose of 1st matrix :\n");

transpose\_matrix(mat1,row,col);

printf("\nTranspose of 2nd matrix :\n");

transpose\_matrix(mat2,row,col);

break;

case 6:printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter first matrix :\n");

read\_matrix(mat1,row,col);

printf("\nEnter second matrix :\n ");

read\_matrix(mat2,row,col);

printf("\n first matrix :\n");

display\_matrix(mat1,row,col);

printf("\n second matrix :\n ");

display\_matrix(mat2,row,col);

break;

case 7: printf("\nEnter the no. of rows and columns of the matrix : ");

scanf("%d %d",&row,&col);

printf("\nEnter matrix :\n");

read\_matrix(mat1,row,col);

printf("\n matrix :\n");

display\_matrix(mat1,row,col);

search(mat1, row, col,x);

break;

default:printf("\nPlease chose a valid option dear!!\n");

}

printf("\nDo you want to continue(enter 0 for no and 1 for yes):");

scanf("%d",&ch);

}while(ch!=0);

return 0;

}

void read\_matrix(int mat[2][2],int r,int c)

{

int i,j;

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

printf("mat[%d][%d]= ",i,j);

scanf("%d",&mat[i][j]);

printf(" ");

}

}

}

void sum\_matrix(int mat1[2][2],int mat2[2][2],int r,int c)

{

int i,j,sum[2][2];

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

sum[i][j]=mat1[i][j]+mat2[i][j];

}

}

display\_matrix(sum,r,c);

}

void diff\_matrix(int mat1[2][2],int mat2[2][2],int r,int c)

{

int i,j,diff[2][2];

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

diff[i][j]=mat1[i][j]-mat2[i][j];

}

}

display\_matrix(diff,r,c);

}

void mul\_matrix(int mat1[2][2],int mat2[2][2],int r,int c)

{

int i,j,mul[2][2],k;

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

mul[i][j]=0;

for(k=0;k<c;k++)/////

mul[i][j]+=mat1[i][k]\*mat2[k][j];

}

}

display\_matrix(mul,r,c);

}

void transpose\_matrix(int mat[2][2],int r,int c)

{

int i,j,trans[2][2];

for(i=0;i<r;i++)

{

for(j=0;j<c;j++)

{

trans[i][j]=mat[j][i];//just interchange row and col that is j and i

}

}

display\_matrix(trans,r,c);

}

void display\_matrix(int mat[2][2],int r,int c)

{

int i,j;

for(i=0;i<r;i++)

{ printf("\n");

for(j=0;j<c;j++)

{

printf("\t mat[%d][%d]=%d",i,j,mat[i][j]);

}

}

}

int search(int mat[2][2], int r, int c,int x)

{

int i = 0, j = r-1; //set indexes for top right element

printf("\nenter the element to be searched :") ;

scanf("%d",&x);

while ( i < r && j >= 0 )

{

if ( mat[i][j] == x )

{

printf("%d Found at %d, %d",x, i, j);

return 1;

}

if ( mat[i][j] > x )

j--;

else // if mat[i][j] < x

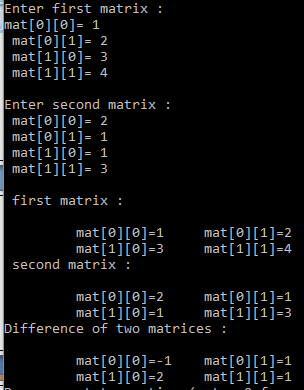
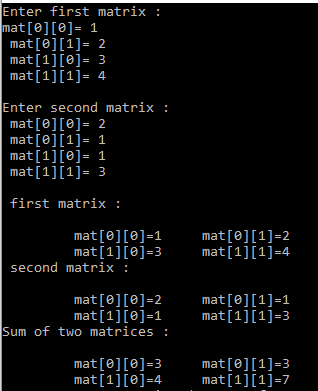
i++;

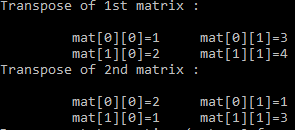
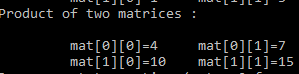
}

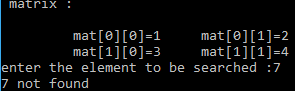
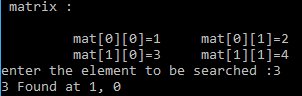
printf("%d not found",x);

return 0; // if ( i==n || j== -1 )

}







OPERATIONS ON SINGLY LINKED LIST

1.Create the linked list

2.Insert a node in the front, at the end, and at any other position in the list

3.Delete the first node, last node and any other node from the list.

4.Search the node with given value in the list

5.Count the no. of nodes in the list (iteratively).

6.Traverse and print all the values of the node

7.Find the length of the list recursively

8.Find the middle element of the list

9.Reverse linked list in place(without using additional memory)

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node{

int data;

struct node \*next;

};

typedef struct node NODE;

NODE \*start=NULL;

/\*----Function Prototypes-----\*/

void create();

void display();

void insert\_begin();

void insert\_end();

void insert\_pos();

void delete\_begin();

void delete\_end();

void delete\_pos();

void search(int x);

void count\_iterative();

int count\_recursive(NODE \*start);

void find\_mid();

void reverse();

/\*-----------------------------\*/

int main()

{

int op,ch,x,c;

do{

printf("\nPERFORM FOLLOWING OPERATIONS ON SINGLY LINKED LIST");

printf("\n1.Create a node");

printf("\n2.Traverse and print all the values of node");

printf("\n3.Insert a node in the beginning");

printf("\n4.Insert anode at the end");

printf("\n5.Insert a node at any random position");

printf("\n6.Delete a node from the beginning");

printf("\n7.Delete a node from end");

printf("\n8.Delete a node from any random position");

printf("\n9.Search the node with the given value in the list");

printf("\n10.Count the no. of nodes(length of linked list) iterativey");

printf("\n11.Count the no. of nodes(length of linked list) recursively");

printf("\n12.Find the middle element of the list");

printf("\n13.Reverse linked list in place(without using additional memory)");

//printf("\n Exit");

printf("\nEnter your option :");

scanf("%d",&op);

switch(op)

{

case 1: create();

break;

case 2: display();

break;

case 3: insert\_begin();

break;

case 4: insert\_end();

break;

case 5: insert\_pos();

break;

case 6: delete\_begin();

break;

case 7: delete\_end();

break;

case 8: delete\_pos();

break;

case 9: printf("\n Enter the element to be searched :");

scanf("%d",&x);

search(x);

break;

case 10: count\_iterative();

break;

case 11: c=count\_recursive(start);

printf("There are %d node(s) in the given linked list",c);

break;

case 12:find\_mid();

break;

case 13:reverse();

break;

default:printf("\nPlease chose a valid option dear!!\n");

}

printf("\nDo you want to continue(enter 0 for no and 1 for yes):");

scanf("%d",&ch);

}while(ch!=0);

return 0;

}

void create()

{

NODE \*temp,\*ptr;

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

if(temp==NULL)

{

printf("\nOut of Memory Space i.e Overflow\n");

exit(0);

}

printf("\nEnter the data value for the node:\t");

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

temp->next=NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next;

}

ptr->next=temp;

}

}

void display()

{

NODE \*ptr;

if(start==NULL)

{

printf("\nList is empty\n");

return;

}

else

{

ptr=start;

printf("\nThe List elements are:\n");

while(ptr!=NULL)

{

printf("%d\t",ptr->data);

ptr=ptr->next ;

}//end of while

}//end of else

}

void insert\_begin()

{

struct node \*temp;

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

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the data value for the node:\t" );

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

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

temp->next=start;

start=temp;

}

}

void insert\_end()

{

struct node \*temp,\*ptr;

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

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the data value for the node:\t" );

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

temp->next =NULL;

if(start==NULL)

{

start=temp;

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

ptr=ptr->next ;

}

ptr->next=temp;

}

}

void insert\_pos()

{

struct node \*ptr,\*temp;

int i,pos;

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

if(temp==NULL)

{

printf("\nOut of Memory Space:\n");

return;

}

printf("\nEnter the position for the new node to be inserted:\t");

scanf("%d",&pos);

printf("\nEnter the data value of the node:\t");

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

temp->next=NULL;

if(pos==0)

{

temp->next=start;

start=temp;

}

else

{

for(i=0,ptr=start;i<pos-1;i++)

{

ptr=ptr->next;

if(ptr==NULL)

{

printf("\nPosition not found:[Handle with care]\n");

return;

}

}

temp->next =ptr->next ;

ptr->next=temp;

}//end of else

}

void delete\_begin()

{

struct node \*ptr;

if(ptr==NULL)

{

printf("\nList is Empty:\n");

return;

}

else

{

ptr=start;

start=start->next ;

printf("\nThe deleted element is :%d\t",ptr->data);

free(ptr);

}

}//end of delete\_begin()

void delete\_end()

{

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\nList is Empty:");

exit(0);

}

else if(start->next ==NULL)

{

ptr=start;

start=NULL;

printf("\nThe deleted element is:%d\t",ptr->data);

free(ptr);

}

else

{

ptr=start;

while(ptr->next!=NULL)

{

temp=ptr;

ptr=ptr->next;

}

temp->next=NULL;

printf("\nThe deleted element is:%d\t",ptr->data);

free(ptr);

}

}//end of delete\_begin()

void delete\_pos()

{

int i,pos;

struct node \*temp,\*ptr;

if(start==NULL)

{

printf("\nThe List is Empty:\n");

exit(0);

}

else

{

printf("\nEnter the position of the node to be deleted:\t");

scanf("%d",&pos);

if(pos==0)

{

ptr=start;

start=start->next ;

printf("\nThe deleted element is:%d\t",ptr->data );

free(ptr);

}

else

{

ptr=start;

for(i=0;i<pos;i++)

{

temp=ptr;

ptr=ptr->next ;

if(ptr==NULL)

{

printf("\nPosition not Found:\n");

return;

}

}

temp->next =ptr->next ;

printf("\nThe deleted element is:%d\t",ptr->data );

free(ptr);

}

}//end of else

}//end of delete\_pos()

void search(int x)

{

int flag=0;

NODE \*ptr = start;

while(ptr != NULL)

{

if (ptr->data == x)

{flag=1;

break;

}

ptr = ptr->next;

}

if(flag==1)printf("%d found",x);

else printf(" Sorry,%d not found",x);

}

void count\_iterative()

{

int count = 0; // Initialize count

NODE \*ptr = start; // Initialize ptr

while (ptr != NULL)

{

count++;

ptr = ptr->next;

}

printf("\nThere are %d node(s) in this linked list.",count);

}

int count\_recursive(NODE \*start)

{

// Base case

if (start == NULL)

return 0;

// count is 1 + count of remaining list

return 1 + count\_recursive(start->next);

}

void find\_mid()

{

//NODE \*start;

NODE \*slow\_ptr = start;

NODE \*fast\_ptr = start;

//if (start!=NULL)

// {

while (fast\_ptr != NULL && fast\_ptr->next != NULL)

{

slow\_ptr = slow\_ptr->next;

fast\_ptr = fast\_ptr->next->next;

}

printf("The middle element is %d \n", slow\_ptr->data);

// }

}

void reverse()

{

struct node \*prevNode, \*curNode;

if(start != NULL)

{

prevNode = start;

curNode = start->next;

start = start->next;

prevNode->next = NULL; // Make first node as last node

while(start != NULL)

{

start = start->next;

curNode->next = prevNode;

prevNode = curNode;

curNode = start;

}

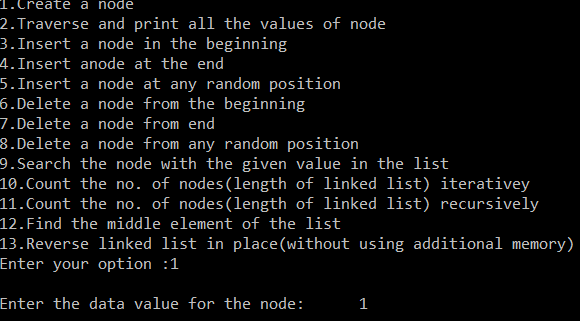
start = prevNode; // Make last node as start

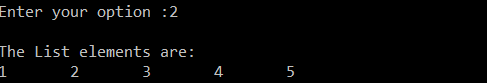
printf("SUCCESSFULLY REVERSED LIST\n");

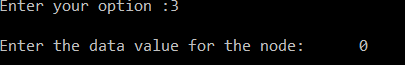
display();

}

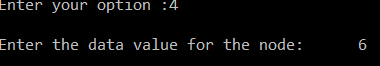
}



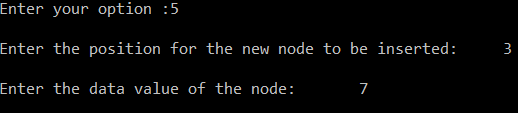










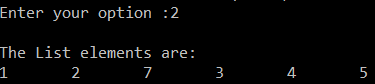


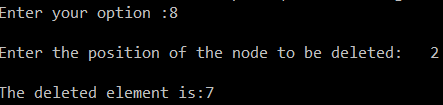


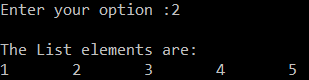


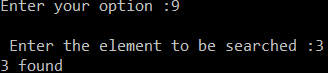


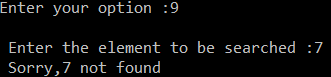








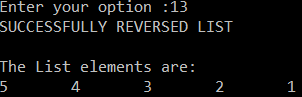












10.Split linked list into two equal halves.

11.Concatenate 2 linked list

12.Merge 2 linked list

13.Find Union and intersection of 2 linked list

OPERATIONS ON CIRCULAR LINKED LIST

1.Create circular linked list

2.Insert a node

3.Delete a node

4.Traverse/print the value of nodes

5.Count no. of nodes

6.Search node with given value

7.Reverse circular linked list without using additional memory

OPERATIONS ON DOUBLY LINKED LIST

1.Create a node

2.Insert a node(at front ,end and any random position)

3. Delete a node(from front ,end and any random position)

4.Display the contents of linked list

5.Count no. of nodes

OPERATIONS ON STACK

1.Perform push,pop and traverse on stack using

a) an array

b) a linked list

2.Reverse the contents of stack using linked list or array by

a) recursion

b) iteration