**MATRIX**

#include<stdio.h>

int main()

{ int i,j,temp;

int a[10][10], b[10][10],res[10][10],ch,r,c;

printf("Welcome to the program \n Enter the no of rows and coulomns of first matrix\n");

scanf("%d%d",&r,&c);

printf("Enter the data elements of the first matrix\n");

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

{

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

{

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

}

}

printf("Enter the no of rows and coulomns of second matrix\n");

scanf("%d%d",&r,&c);

printf("Enter the data elements of the second matrix\n");

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

{

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

{

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

}

}

printf("For additon of matrices press 1\n For substraction of matrices press 2\n For multiplication of matrices press 3\nFor Transpose of matrices press 4 \n To exit the Program press 5\n");

scanf("%d",&ch);

switch(ch)

{

case 1:printf("The resultant Matrix is\n");

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

{

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

{

res[i][j]=a[i][j]+b[i][j];

}

}

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

{

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

{

printf("%3d",res[i][j]);

}printf("\n");

}

break;

case 2:printf("The resultant Matrix is\n");

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

{

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

{

res[i][j]=a[i][j]-b[i][j];

}

}

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

{

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

{

printf("%3d",res[i][j]);

}printf("\n");

}

break;

case 3:printf("The resultant Matrix is\n");

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

{

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

{

res[i][j]=a[i][j]\*b[i][j];

}

}

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

{

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

{

printf("%3d",res[i][j]);

}printf("\n");

}

break;

case 4:printf("Enter the no of matrix you would like to Transpose");

scanf("%d",&temp);

switch(temp)

{

case 1:

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

{

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

{

printf("%3d",a[j][i]);

}printf("\n");

}break;

case 2:

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

{

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

{

printf("%3d",b[j][i]);

}printf("\n");

}break;

default:printf("wrong choice");

break;

}

break;

case 5: break;

default:printf("Wrong choice");

break;

}

}

**INSERTION SORT**

#include<stdio.h>

int main()

{

int i,j,n ,t;

printf("enter number of elements");

scanf("%d",&n);

int a[100];

printf("Enter the elements\n");

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

{

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

}

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

{

j=i;

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

{

t=a[j];

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

a[j-1]=t;

}

j--;

}

printf("The sorted array is");

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

{

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

}

}

**QUICKSORT**

#include<stdio.h>

#include<stdlib.h>

int partition(int a[100],int p,int l)

{

int x,i,j,temp;

x=a[l];

i=p-1;

for(j=p;j<=l;j++)

{

if(a[j]<a[l])

{

i++;

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

i++;

temp=a[i];

a[i]=a[l];

a[l]=temp;

return i;

}

void quicksort(int a[], int p, int l)

{

int q;

if(p<l)

{

q=partition(a,p,l);

quicksort(a,p,q-1);

quicksort(a,q+1,l);

}

}

int main()

{

int a[100],f,l,i,n;

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

scanf("%d",&n);

printf("Enter the Elements :");

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

{

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

}

quicksort(a,0,n);

printf("The sorted Elements are :");

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

{

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

}

}

**MERGESORT**

#include<stdio.h>

void merge(int a[10], int s,int e)

{

int i,j,k,temp[100];

i=s;

int mid=(s+e)/2;

j=mid+1;

k=s;

while(i<=mid&&j<=e)

{

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

{

temp[k++]=a[i++];

}

else{

temp[k++]=a[j++];

}

}

while(i<=mid) {

temp[k++]=a[i++];

}

while(j<=e){

temp[k++]=a[j++];

}

for(i=s;i<=e;i++){

a[i]=temp[i];

}

}

void msort(int a[10],int s,int e)

{

if(s==e)

{

return;

}

int mid=(s+e)/2;

msort(a,s,mid);

msort(a,mid+1,e);

merge(a,s,e);

}

int main(){

int n,i,a[10];

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

scanf("%d",&n);

printf("enter the elements :");

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

{

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

}

msort(a,0,n-1);

printf("The sorted array is :");

for(i=0;i<

n;i++)

{

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

}

}

**POSTFIX**

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

return 0;

}

**STACK**

#include <stdio.h>

#include <conio.h>

#define MAX 10

int stack[MAX],topA=-1,topB=MAX;

void pushA(int val)

{

if(topA==topB-1)

{

printf("\n OVERFLOW");

}

else

{

topA+= 1;

stack[topA] = val;

}

}

int popA()

{

int val;

if(topA==-1)

{

printf("\n UNDERFLOW");

}

else

{

val = stack[topA];

topA--;

}

return val;

}

void display\_stackA()

{

int i;

if(topA==-1)

{

printf("\n Stack A is Empty");

}

else

{

for(i=topA;i>=0;i--)

printf("\t %d",stack[i]);

}

}

void pushB(int val)

{

if(topB-1==topA){

printf("\n OVERFLOW");}

else

{

topB -= 1;

stack[topB] = val;

}

}

int popB()

{

int val;

if(topB==MAX)

{

printf("\n UNDERFLOW");

}

else

{

val = stack[topB];

topB++;

}

}

void display\_stackB()

{

int i;

if(topB==MAX){

printf("\n Stack B is Empty");}

else

{

for(i=topB;i<MAX;i++)

{

printf("\t %d",stack[i]);

}

}

}

void main()

{

int option, val;

do

{

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

printf("\n 1. PUSH IN STACK A");

printf("\n 2. PUSH IN STACK B");

printf("\n 3. POP FROM STACK A");

printf("\n 4. POP FROM STACK B");

printf("\n 5. DISPLAY STACK A");

printf("\n 6. DISPLAY STACK B");

printf("\n 7. EXIT");

printf("\n Enter your choice");

scanf("%d",&option);

switch(option)

{

case 1: printf("\n Enter the value to push on Stack A : ");

scanf("%d",&val);

pushA(val);

break;

case 2: printf("\n Enter the value to push on Stack B : ");

scanf("%d",&val);

pushB(val);

break;

case 3: val=popA();

if(val!=-999)

printf("\n The value popped from Stack A = %d",val);

break;

case 4: val=popB();

if(val!=-999)

printf("\n The value popped from Stack B = %d",val);

break;

case 5: printf("\n The contents of Stack A are : \n");

display\_stackA();

break;

case 6: printf("\n The contents of Stack B are : \n");

display\_stackB();

break;

}

}while(option!=7);

getch();

}