School Of Mechanical & Manufacturing Engineering, NUST

Department of Mechanical Engineering

CS-114 - Fundamental of Programing

Lab manual # 09

Course Instructor: Dr Jawad Khan

Lab Instructor: Muhammad Affan

Student Name: AATIKA KAMRAN

CMS ID: 464185

DATE:

12/12/2023

1. Make 2D Array in C++ and print left diagonal and right diagonal sum of a 3x3 matrix

```cpp
#include <iostream>
using namespace std;
int main()
{ int count = 1;
 int ar[3][3]={};
 for (int i =0; i<3;i++)
 {
    for (int j =0; j<3;j++)
 { ar[i][j]=count;
 count++ ;
      cout<<ar[i][j]<<"\t";
    }
    cout<<endl;
}

int leftsum = 0;
for(int i=0;i<3;i++){
        leftsum += ar[i][i];

        }
         cout<<"sum of left diagonal = " << leftsum<<endl;

        int rightsum = 0;
for(int i=0;i<3;i++){
        rightsum += ar[i][2-i];

        }
         cout<<"sum of right diagonal = " << rightsum<<endl;
    return 0;
}
```

```
1       2       3
4       5       6
7       8       9
sum of left diagonal = 15
sum of right diagonal = 15
```

2. Write a function to add two 2D arrays of size 3x3.

```
{
    int x=1;
    int a1[3][3], a2[3][3];
    cout<<"first matrix"<<endl;
    for( int i=0; i<3; i++) {
        for (int j =0; j<3;j++) {
            a1[i][j]=x;
            x++;
            cout<<a1[i][j]<<" ";
        }
        cout<<endl;
    }

    cout<<"second matrix"<<endl;
    for( int i=0; i<3; i++) {
        for (int j =0; j<3;j++) {
            a2[i][j]=x;
            x++;

            cout<<a2[i][j]<<" ";
        }
        cout<<endl;
    }
    int s[3][3];
    for( int i=0; i<3; i++) {
        for (int j =0; j<3;j++) {
        s[i][j]=a1[i][j]+a2[i][j];
        }

    }
        cout<<"sum is "<<endl;
    for( int i=0; i<3; i++) {
        for (int j =0; j<3;j++) {
            cout<<s[i][j]<<" ";
            }
        cout<<endl;
    }
return 0;
```

```
first matrix
1 2 3
4 5 6
7 8 9
second matrix
10 11 12
13 14 15
16 17 18
sum is
11 13 15
17 19 21
23 25 27
```

3. Using 2D arrays in C++, take transpose of a 3x3 matrix. Make a transpose function

```cpp
void transpose(int a[3][3], int trans[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            trans[i][j] = a[j][i];
        }
    }
}

int main() {
    int a[3][3];
    int transposed[3][3];

    cout << "Enter 9 elements of the matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> a[i][j];
        }
    }

    cout << "Original matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }

    transpose(a, transposed);

    cout << "Transpose matrix:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << transposed[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```

mpile Log | Debug | Find Results | Clo

```
Enter 9 elements of the matrix:
1
2
3
4
5
6
7
8
9
Original matrix:
1 2 3
4 5 6
7 8 9
Transpose matrix:
1 4 7
2 5 8
3 6 9
```

4. Using 2D arrays in C++, implement 3x3 matrix multiplication. Make a function.

```
    void inputarray(int a[3][3]) {
        for (int i=0;i<3;i++) {
            for(int j=0;j<3;j++) {
                cin>>a[i][j];
            }
        }
    }
void displayarray(int a[3][3]) {
        for (int i=0;i<3;i++) {
            for(int j=0;j<3;j++) {
                cout<<a[i][j]<<" ";

            }
            cout<<endl;
        }
    }
    void product(int a1[3][3],int a2[3][3],int p[3][3]) {
            for (int i=0;i<3;i++) {
            for(int j=0;j<3;j++) {
                p[i][j]=0;
                for(int k=0;k<3;k++) {
                    p[i][j] += a1[i][k]*a2[k][j];
                }

            }


    } }

    int main () {
        int array1[3][3],array2 [3][3], productarray[3][3];
        cout<<"enter first matrix " <<endl;
        inputarray(array1);
            cout<<"enter second matrix " <<endl;
        inputarray(array2);
            cout<<" first matrix : " <<endl;
        displayarray(array1);
            cout<<" second matrix : " <<endl;
        displayarray( array2);
        product(array1,array2,productarray);
        cout<<" multiplied matrix : " <<endl;
        displayarray(productarray);
```

```
enter first matrix
1
2
3
4
5
6
7
8

9
enter second matrix

98
7
6
5
4
3

2
1
2
 first matrix :
1 2 3
4 5 6
7 8 9
 second matrix :
98 7 6
5 4 3
2 1 2
 multiplied matrix :
114 18 18
429 54 51
744 90 84

------------------------
```

5. Print the multiplication table of 15 using recursion.

```
// }
void table(int number , int scope,int i=1) {
    if (i<scope) {

    cout<<"table :"<<endl;
    cout<<number<<" x "<<i<<" = "<<number*i<<endl;
    table(number,scope,i+1);
    }
}

int main() {
    int x,n;
    cout<<"enter the number whose table is to be printed "<<endl;
    cin>>n;
    cout<<"enter the number till which you want to print table "<<endl;
    cin>>x;
    table(n,x);
```

```
enter the number whose table is to be printed
12
enter the number till which you want to print table
5
table :
12 x 1 = 12
table :
12 x 2 = 24
table :
12 x 3 = 36
table :
12 x 4 = 48
```

1. Write a C++ program to take inverse of a 3x3 matrix using its determinant and adjoint

```
{int m[3][3],adj[3][3];
float inv[3][3];
int det;
cout<<" enter elements of matrix "<<endl;
    for (int i=0;i<3;i++) {
        for(int j=0;j<3;j++) {
            cin>>m[i][j];
        }
        cout<<"  matrix : "<<endl;
            for (int i=0;i<3;i++) {
            for(int j=0;j<3;j++) {
                cout<<m[i][j]<<" ";
            }
            cout<<endl;
        }
    int a=m[0][0],b=m[1][0],c=m[2][0],d=m[0][1],e=m[1][1],f=m[2][1],g=m[0][2],h=m[1][2],i=m[2][2];
    det=a*(e*i-f*h)-b*(d*i-f*g) +c*(d*h-e*g);
    adj[0][0] =e*i-f*h;
    adj[1][0]=-1*(b*i-c*h);
    adj[2][0]=b*f-c*e;
    adj[0][1]=-1*(d*i-f*g);
    adj[1][1]=a*i-c*g;
    adj[2][1]=-1-1*(a*f-c*d);
    adj[0][2]=d*h-e*g;
    adj[1][2]=-1*(a*h-b*g);
    adj[2][2]=a*e-b*g;

    if (det==0) {
        cout<<"Inverse does not exist"<<endl;
    }
    else{
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++ ) {
                inv[i][j]=adj[i][j]/det;
            }
        }
    }
    cout<<" INVERSE: "<<endl;
    for(int i=0;i<3;i++){
            for(int j=0;j<3;j++ ) {
                cout<<inv[i][j]<<" ";
            }
            cout<<endl;
        }
```

```
 enter elements of matrix
1
0
-3
4
2
-1
0
3
2
  matrix :
1 0 -3
4 2 -1
0 3 2
 INVERSE:
-0.241379 0.310345 -0.206897
0.275862 -0.0689655 0.37931
-0.413793 0.137931 -0.482759
```