G. H. Raisoni College Of Engineering And Management, Wagholi Pune Assignment no :- 4 2021- 2022			
Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	04-10-2021
Subject Name /Code	Object Oriented Programming/ UTIL201/UITP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Assignment No:-4

Aim:→ Create a class MAT of size m*n. Define all possible matrix operations for MAT type object

Theory :→

▶ Overloading in C++

If we crecte two or more members having the same name but different in numbers or type of parameter, it is known as C++ overloading.

In C++ we can overload:→ methods, constructors, and indexed properties

It is because This members have parameters only.

▶ Types OF Overloading in C++

- Function overloading
- operator overloading

◉ Function overloading

Function overloading is defined as the process of having two or more Function with the same name, but different in parameters is known as function overloading in C++. In Function overloading, the Function is redefined by using either different eol Types

of arguments or different number of arguments. It is only through these differences compilers can differentiate between the functions.

The advantage of function overloading is that it increases the readability of the program because you don't need to use different names for the same action.

Example of function overloading

```
// program of function overloading when number
// of arguments vary.

#include<iostream>
using namespace std;
class cal {
public:
static int add(int a, int b) {
return a+b ; }
static int add (int a, int b, int c)
{ return a+b+c ; }

};
int main(void) {
cal C;                    // class object declaration
cout << C.add(10,20) << endl;
cout << C.add(12,20,23);
return 0; }
```

Output

```
30
55
```

Function overloading and Ambiguity:

When the compiler is unable to decide which function is to be invoked among the overloaded function this situation is known as function overloading.

When compiler shows the ambiguity error. the compiler does not run the program.

Causes of Function Overloading & Ambiguity

- Type conversion.
- Function with default argument
- Function with pass by reference.

Program code

```
#include<iostream>
#include<iomanip>
using namespace std;
class mat
{
 float **m;
 int rs,cs;
 public:
 mat(){}
 void creat(int r,int c);
 friend istream & operator >>(istream &,mat &);
 friend ostream & operator <<(ostream &,mat &);
 mat operator+(mat m2);
 mat operator-(mat m2);
 mat operator*(mat m2);
};
void mat::creat(int r,int c)
{
 rs=r;
 cs=c;
 m=new float *[r];
 for(int i=0;i<r;i++)
 m[i]=new float [r];
}
istream & operator>>(istream &din, mat &a)
{
```

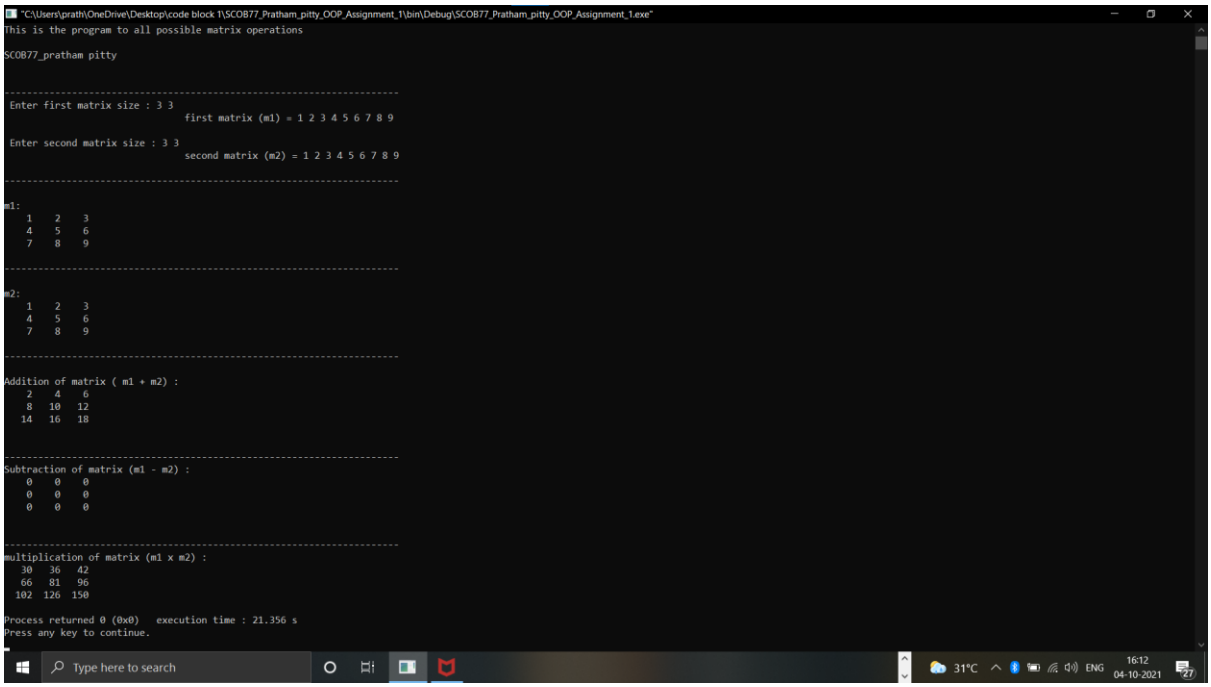
```
 int r,c;
 r=a.rs;
 c=a.cs;
 for(int i=0;i<r;i++)
 {
 for(int j=0;j<c;j++)
 {
 din>>a.m[i][j];
 }
 }
 return (din);
}
ostream & operator<<(ostream &dout,mat &a)
{
 int r,c;
 r=a.rs;
 c=a.cs;
 for(int i=0;i<r;i++)
 {
 for(int j=0;j<c;j++)
 {
 dout<<setw(5)<<a.m[i][j];
 }
 dout<<"\n";
 }
return (dout);
}
mat mat::operator+(mat m2)
{
 mat mt;
 mt.creat(rs,cs);
```

```
 for(int i=0;i<rs;i++)
 {
 for(int j=0;j<cs;j++)
 {
 mt.m[i][j]=m[i][j]+m2.m[i][j];
 }
 }
 return mt;
}
mat mat::operator-(mat m2)
{
 mat mt;
 mt.creat(rs,cs);
 for(int i=0;i<rs;i++)
 {
 for(int j=0;j<cs;j++)
 {
 mt.m[i][j]=m[i][j]-m2.m[i][j];
 }
 }
 return mt;
}
mat mat::operator*(mat m2)
{
 mat mt;
 mt.creat(rs,m2.cs);
 for(int i=0;i<rs;i++)
 {
 for(int j=0;j<m2.cs;j++)
 {
 mt.m[i][j]=0;
```

```
 for(int k=0;k<m2.rs;k++)
 mt.m[i][j]+=m[i][k]*m2.m[k][j];
 }
 }
 return mt;
 }
int main()
{
 mat m1,m2,m3,m4,m5;
 int r1,c1,r2,c2;
 cout<<"\n-------------------------------------------------------------------\n";
 cout<<"\nThis is the program to all possible matrix operations \n";
 cout<<"\nSCOB77_pratham pitty\n\n";
 cout<<"\n-------------------------------------------------------------------\n";
 cout<<" Enter first matrix size : ";
 cin>>r1>>c1;
 m1.creat(r1,c1);
 cout<<"\t\t\t\tfirst matrix (m1) = ";
 cin>>m1;
 cout<<"\n Enter second matrix size : ";
 cin>>r2>>c2;
 m2.creat(r2,c2);
 cout<<"\t\t\t\tsecond matrix (m2) = ";
 cin>>m2;
 cout<<"\n-------------------------------------------------------------------\n";
 cout<<" \nm1:"<<endl;
 cout<<m1;
 cout<<"\n-------------------------------------------------------------------\n";
 cout<<"\nm2: "<<endl;
 cout<<m2;
 cout<<"\n-------------------------------------------------------------------";
```

```
 cout<<endl<<endl;
 if(r1==r2 && c1==c2)
 {
 m3.creat(r1,c1);
 m3=m1+m2;
 cout<<"Addition of matrix ( m1 + m2) : "<<endl;
 cout<<m3<<endl;
 m4.creat(r1,c1);
 m4=m1-m2;
 cout<<"\n-----------------------------------------------------------------------\n";
 cout<<"Subtraction of matrix (m1 - m2) :"<<endl;
 cout<<m4<<endl;
 }
 else
 cout<<" Summation & substraction are not possible n"<<endl
 <<"Two matrices must be same size for summation & substraction "<<endl;
if(c1==r2)
{

 m5=m1*m2;
 cout<<"\n-----------------------------------------------------------------------\n";
 cout<<"multiplication of matrix (m1 x m2) : "<<endl;
 cout<<m5;
}
else
cout<<" Multiplication is not possible "<<endl
<<" column of first matrix must be equal to the row of second matrix ";
return 0;
}
```


"C:\Users\prath\OneDrive\Desktop\code block 1\SCOB77_Pratham_pitty_OOP_Assignment_1\bin\Debug\SCOB77_Pratham_pitty_OOP_Assignment_1.exe"
This is the program to all possible matrix operations
SCOB77_pratham pitty
Enter first matrix size : 3 3
first matrix (m1) = 1 2 3 4 5 6 7 8 9
Enter second matrix size : 3 3
second matrix (m2) = 1 2 3 4 5 6 7 8 9
m1:
1 2 3
4 5 6
7 8 9
m2:
1 2 3
4 5 6
7 8 9
Addition of matrix (m1 + m2) :
2 4 6
8 10 12
14 16 18
Subtraction of matrix (m1 - m2) :
0 0 0
0 0 0
0 0 0
multiplication of matrix (m1 x m2) :
30 36 42
66 81 96
102 126 150
Process returned 0 (0x0) execution time : 21.356 s
Press any key to continue.
Type here to search
31°C
ENG
16:12
04-10-2021