OOP Assignment No.- 04

Assignment No.-(4)

Page :
Date :

AIM - Create class of size m*n. Define all possible matrix operations for MAT type objects.

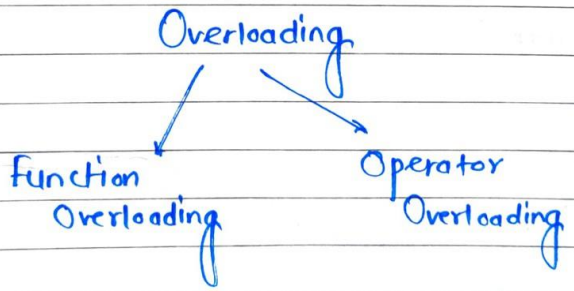
Theory:-

If we create two or more members having the same name but different in number or type of parameter, it is known as C++ overloading. In C++, we can overload :

- Methods
- Constructors, and
- indexed properties

It is because these members have parameters only

- Types of Overloading in C++ are :



* C++ Function Overloading :-

Function overloading is defined as process of having two or more function with the same name, but different in parameters known as function overloading in C++.

Camlin

rudraksh.karpe.cs@ghrcem.raisoni.net

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

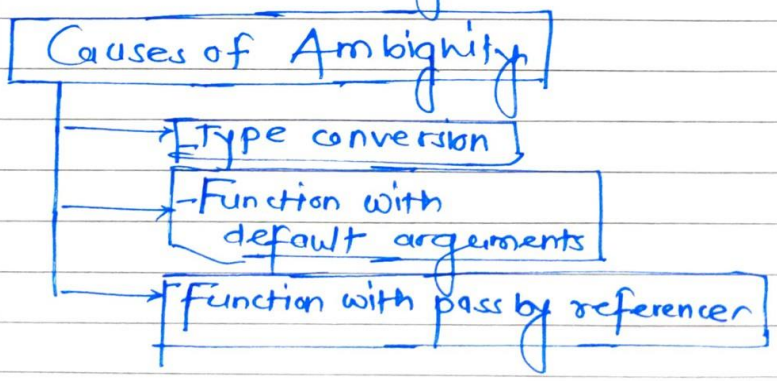
(*) Function Overloading and Ambiguity:

- 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 the compiler shows the ambiguity error, the compiler doesn't run the program.

Causes of function overloading :

- Type conversion
- Function with default arguments
- 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;
}
```

```
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<<" Enter first matrix size : ";
cin>>r1>>c1;
m1.creat(r1,c1);
cout<<"m1 = ";
cin>>m1;
cout<<" Enter second matrix size : ";
cin>>r2>>c2;
m2.creat(r2,c2);
cout<<"m2 = ";
cin>>m2;
cout<<" m1:"<<endl;
cout<<m1;
cout<<" m2: "<<endl;
cout<<m2;
cout<<endl<<endl;
if(r1==r2 && c1==c2)
 {
        m3.creat(r1,c1);
                m3=m1+m2;
        cout<<" m1 + m2: "<<endl;
        cout<<m3<<endl;
       m4.creat(r1,c1);

        m4=m1-m2;
       cout<<" m1 - m2:"<<endl;
```

```
            cout<<m4<<endl<<endl;


      }
   else
    cout<<" Summation & substraction are not possible n"<<endl
       <<"Two matrices must be same size for summation &   substraction
"<<endl<<endl;
if(c1==r2)
{
          m5=m1*m2;
          cout<<" 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;
}
```

Output:

Inputs:

```
PS R:\GHRCEM\OOP LAB> cd "r:\GHRCEM\OOP LAB\" ; if ($?) { g++ LAB_4.cpp -o LAB_4 } ; if ($?) { .\LAB_4 }
 Enter first matrix size : 3 3
m1 =
5 6 8

2 5 6

3 2 1
 Enter second matrix size : 3 3
m2 =
5 6 7

3 6 2

1 2 9
```

Operations performed on Matrix:

```
m1:
    5     6     8
    2     5     6
    3     2     1
m2:
    5     6     7
    3     6     2
    1     2     9


m1 + m2:
   10    12    15
    5    11     8
    4     4    10

m1 - m2:
    0     0     1
   -1    -1     4
    2     0    -8


m1 x m2:
   51    82   119
   31    54    78
   22    32    34
```

rudraksh.karpe.cs@ghrcem.raisoni.net