Lecture 9
OOP

Khola Naseem
khola.naseem@uet.edu.pk

Operator overloading:

- Binary Operators

```
class number
{
    int num1;
    public:
        number();

};
number::number()
{
    num1=5;
}
int main(int argc, char** argv) {
    number obj1;
    number obj2;
    cout<<obj1+obj2;
}
```

Error:

E:\UET\Spring 23\OOP\Class\Operator overloading.cpp	**In function 'int main(int, char**)':**
E:\UET\Spring 23\OOP\Class\Operator overloading.cpp	[Error] no match for 'operator+' (operand types are 'number' and 'number')

Operator overloading:

- Binary Operators

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        number operator +(const number &n2)
        {
            number n3;
            n3.num1=num1+n2.num1;
            return n3;

        }
        void display()
        {
            cout<<"Answer is: "<<num1;
        }

};
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;
    //obj3=obj1+obj2;
    obj3=obj1+obj2+obj2; //output 17
    obj3.display();
}
```

Output:

```
Answer is: 17
--------------------------------
```

Credit: Khola Naseem

Operator overloading:

- Stream insertion<<, and extraction operator>>

```
class number
{
    int num1;
    int num2;

    public:
        number(int num,int num2);
};
number::number(int num,int num_2)
{
    num1=num;
    num2=num_2;
}
int main(int argc, char** argv) {
    number obj1(2,3);
    cout<<obj1;
}
```

```
In function 'int main(int, char**)':
[Error] no match for 'operator<<' (operand types are 'std::ostream {aka std::basic_ostream<char>}' and 'number')
```

Credit: Khola Naseem

overloaded operator :

Some of the operators there are many more

- +=
- !=
- >=
- <=
- && and ||
- []
- ()

overloaded operator :

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        number operator +(const number &n2)
        {
            number n3;
            n3.num1=num1+n2.num1;
            return n3;
        }
        void display()
        {
            cout<<"Answer is: "<<num1;
        }
        //friend number operator +(const number &n2,const number &n1);
};
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;
    //obj3=obj1+obj2;
    obj3=obj1+obj2+obj2; //output 17

    obj3.display();
}
```

overloaded operator :

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        number operator +(int num3)
        {
            number n3;
            n3.num1=num3+num1;
            return n3;

        }
        void display()
        {
            cout<<"Answer is: "<<num1;
        }

};
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;
    //obj3=obj1+obj2;
    obj3=obj1+1;
    obj3.display();
}
```

```
Answer is: 6
--------------------------------
Process exited after 0.1402 seconds with return value 0
Press any key to continue . . .
```

Credit: Khola Naseem

overloaded operator :

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        number operator +(int num3)
        {
            number n3;
            n3.num1=num3+num1;
            return n3;

        }
        void display()
        {
            cout<<"Answer is: "<<num1;
        }

};
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;
    //obj3=obj1+obj2;
    obj3=1+obj1;
    obj3.display();
}
```

Operator overloading with the help of friend function

- output:

```
Answer is: 6
```

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        friend number operator +( int num3, const number &c1);

        void display()
        {
            cout<<"Answer is: "<<num1;
        }

};
    number operator +( int num3, const number &c1)
        {
            number n3;
            n3.num1=c1.num1+num3;
            return n3;

        }
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;

    obj3=1+obj1;
    obj3.display();
}
```

Operator overloading with the help of friend function

- +

```
class number
{
    int num1;
    public:
        number();
        number(int num);
        number operator +(const number &n2)
        {
            number n3;
            n3.num1=num1+n2.num1;
            return n3;
        }
        void display()
        {
            cout<<"Answer is: "<<num1;
        }
        friend number operator +(const number &n2,const number &n1);
};
        number operator +(const number &n2,const number &n1)
        {
            number n3;
            n3.num1=n1.num1+n2.num1;
            return n3;
        }
number::number()
{
    num1=5;
}
number::number(int num)
{
    num1=num;
}
int main(int argc, char** argv) {
    number obj1,obj2(6),obj3;
    //obj3=obj1+obj2;
    obj3=obj1+obj2+obj2; //output 17
    obj3.display();
}
```

Operator overloading:

- Stream insertion<<, and extraction operator>>

```
class number
{
    int num1;
    int num2;

    public:
        number(int num,int num2);
};
number::number(int num,int num_2)
{
    num1=num;
    num2=num_2;
}
int main(int argc, char** argv) {
    number obj1(2,3);
    cout<<obj1;
}
```

```
In function 'int main(int, char**)':
[Error] no match for 'operator<<' (operand types are 'std::ostream {aka std::basic_ostream<char>}' and 'number')
```

Credit: Khola Naseem

Operator overloading:

- Stream insertion<<
- Definition:

```
ostream& operator<<(ostream& os, const ClassName& obj)
{
    // Print the data members of obj using os like you would using cout

    // Return the output stream object so the operator may be cascaded correctly

    return os;
}
```

Operator overloading:

- Stream insertion<<
- Example:

```
class Box1{
    private:
    float length;
    float width;
    float height;
    public:
        Box1()
        {
        length=1.3;
        width=2.3;

        }
        friend ostream& operator<<(ostream& os, const Box1& obj);
};
ostream& operator<<(ostream& os, const Box1& obj)
{
    os<<"length is "<<obj.length<<endl;
    os<<"width is "<<obj.width<<endl;

    return os;
}
int main(int argc, char** argv) {
    Box1 b;

    cout<<b;
}
```

output:

```
length is 1.3
width is 2.3
```

Credit: Khola Naseem

Operator overloading:

- Stream insertion<< and steam extraction >>
- Example:

```
class Box1{
    private:
    float length;
    float width;

    public:
        Box1()
        {
        length=1.3;
        width=2.3;

        }
        friend ostream& operator<<(ostream& os, const Box1& obj);
        friend istream& operator>>(istream& is, Box1& obj);
};
ostream& operator<<(ostream& os, const Box1& obj)
{
    os<<"length is "<<obj.length<<endl;
    os<<"width is "<<obj.width<<endl;

    return os;
}
istream& operator>> (istream& is, Box1& obj)
{
    cout<<"enter length"<<endl;
    is>>obj.length;
    cout<<"enter Width"<<endl;
    is>>obj.width;

    return is;
}
int main(int argc, char** argv) {
    Box1 b;
    cin>>b;
    cout<<b;
}
```

Credit: Khola Naseem

Cascading call to overloaded operator :

- Example:

```
class Box1{
    private:
    float length;
    float width;

    public:
        Box1()
        {
        length=1.3;
        width=2.3;

        }
        friend ostream& operator<<(ostream& os, const Box1& obj);
        friend istream& operator>>(istream& is, Box1& obj);
};
ostream& operator<<(ostream& os, const Box1& obj)
{
    os<<"length is "<<obj.length<<endl;
    os<<"width is "<<obj.width<<endl;

    return os;
}
istream& operator>> (istream& is, Box1& obj)
{
    cout<<"enter length"<<endl;
    is>>obj.length;
    cout<<"enter Width"<<endl;
    is>>obj.width;

    return is;
}
int main(int argc, char** argv) {
    Box1 b;
    cin>>b;
    cout<<"the value of b is \n"<<b<<endl;
}
```

Credit: Khola Naseem

Operator overloading with the help of friend function

```
class counter
{
    int val1;

    public:
        counter();
        friend counter operator ++(counter &c);
        friend counter operator ++(counter &c,int unused);
        void display()
        {
            cout<<"Answer is: "<<val1;
        }
};
counter::counter()
{
    val1=5;

}
//prefix
counter operator ++(counter &c)
        {

            c.val1++;
            return c;

        }
        //postfix
counter operator ++(counter &c, int unused)
        {
          counter c9=c;
            c.val1++;
            return c9;

        }
int main(int argc, char** argv) {
    counter c,c2;
    c2=++c;
    counter c3=c++;
    c2.display();
    cout<<"value of c3   " ;
    c3.display();
}
```

Credit: Khola Naseem

Reference material

For Practice Questions, refer to these books

- C++ Programming From Problem Analysis To Program Design, 5th Edition, D.S.Malik. Chapter 12.
- C++ How to Program, Deitel & Deitel, 5th Edition, Prentice Hall.
- Object Oriented Programming in C++ by Robert Lafore.
- Object Oriented Software Construction, Bertrand Meyer's
- Object-Oriented Analysis and Design with applications, Grady Booch et al, 3Rd Edition, Pearson, 2007
- Web