

## Operator Overloading

- Operators are the building blocks for expressions, thus they are essential part in any high level programming language.
- Usually, operators work on the built in data types.

For example, + operator can add integers, floats, Booleans but can NOT add objects of classes

Also, > can compare characters, doubles, integers but can NOT compare objects of classes

- How can the programmer extend the capabilities of the operators to operate on objects??  
⇒ This could be done by operator overloading
- Operator Overloading is a mechanism that helps the programmer to let the operators do operations that do not exist on their original definitions. (like extending the capabilities of + operator to add objects)
- General guidelines and precautions that must be taken into consideration when overloading an operator:
  1. When an operator is overloaded, none of its original meaning is lost , instead the capabilities of the operator is extended.
  2. Operator overloading is done using special function called **operator#**. Where # is the operator that we wish to overload (ex. operator+ can extend to meaning of the + operator)
  3. The **operator#** function can be a- member function b- friend function c- global function that is non-member non-friend function
  4. You can NOT create new operators; rather you can overload the existing ones.
  5. Operator is not overloaded implicitly, if you overload the + and the = operator , this do not mean that the operator += is already overloaded.
  6. When an operator is overloaded, it can work with classes but the **precedence**, **associativity**, **arity** could NOT get changed.
  7. Four operators can NOT be overloaded are: **. . \* :: ? :**
  8. = () and [] operators can NOT be overloaded using friend function.
  9. << and >> must overloaded as friend function.