3) Conversion Operator: We can also write conversion operators that can be used to convert one type to another type.   
 Overloaded conversion operators must be a member method. Other operators can either be the member method or the global method.

|  |
| --- |
| #include <iostream>  using namespace std;  class Fraction  {  private:      int num, den;  public:      Fraction(int n, int d) { num = n; den = d; }        // Conversion operator: return float value of fraction      operator float() const {          return float(num) / float(den);      }  };    int main() {      Fraction f(2, 5);      float val = f;      cout << val << '\n';      return 0;  } |

Output

0.4

**Binary arithmetic operator such as +, -, \*, / must be explicitly return a value. They must not attempt to change their own arguments.**

#include<iostream>

using namespace std;

int a,b,c;

class A

{

    private:

      int a;

    public:

      A()

      {

        a=0;

      }

      A(int x)

      {

        a=x;

      }

      //Operator Overtloading function

      A operator + (A y)

      {

        A sum;

        sum=a+y.a;

      }

      void dispaly()

      {

        cout<<"a="<<a<<endl;

      }

};

int main()

{

    //Addition between two built-in data type

    a=5;

    b=10;

    c=a+b;

    cout<<"c="<<c<<endl;

    //Addition between two user-define data type

    A obj;

    A obj1(5);

    A obj2(10);

    obj=obj1+obj2;

    //obj=obj1+(obj2);

    obj.dispaly();

    return 0;

}