7. *Operator Overloading: Issues & Mechanism*

Operator Overloading

- Ease of Use & Readability

Rules:

- Overload only existing Operators
- Can't change existing Operator definitions
  - 5 + 2 is 7, period!
- Operator precedence rules apply - can't change

Operator Overloading is merely a function - a special functions though

Examples using Complex class

```
class Complex {
   double rep, imp;
public:
   Complex (double rp=0, double ip=0) { set (rp, ip); }
   void set (double rp, double ip)
   {    rep = rp;    imp = ip; }
   void get(double& rp, double& ip) const
   { rp = rep;       ip = imp; }
   ...
};
```

Overloading the + Operator - as member function

Adding two Complex numbers:

C = A + B;

```
// Operator+ as a member function of Complex class
Complex Complex::operator+(const Complex& b) const
{
   Complex temp;
   temp.rep = rep + b.rep;
   temp.imp = imp + b.imp;
   return temp;
}
```

Overloading the + Operator - as global function

```
// Operator+ as a non-member function of Complex class
Complex operator+(const Complex& a, const Complex& b)
{
    double realpartofa, realpartofb, impartofa, impartofb;

    a.get(realpartofa, impartofa);
    b.get(realpartofb, impartofb);

    Complex temp;
    temp.set(realpartofa+realpartofb, impartofa + impartofb);
    return temp;
}
```

Mechanism involved in resolving a call to Operator Overloading

C = A + B;

is equivalent to one of the following

- C = A.operator+(B);
  - The operator + is associated with the left operand object.
  - Expects to see a member function operator+ in class Complex which takes an object of type Complex as argument
- C = operator+(A, B);
  - The operator + is associated with neither object.
  - Expects to see a global function operator+ which takes two objects of type Complex as argument.

Exercise on operator+

- What are the possible ways to provide the following feature:

  A is a Complex number.

  C = A + 2.1; // Add 2.1 (double) to the
  // real part of A.

Exercise on operator+ : Solution

- Provide

  Complex operator+(double val) const;

  as a member function of Complex
- Provide

  Complex operator+(const Complex& a, double val);

  as a global function
- No need for any function if one of the following exists:
- Complex Complex::operator+(const Complex&) const;
- Complex operator+(const Complex&, const Complex&);

Since

Complex (double=0, double=0); can convert 2.1 to a Complex object

Another Exercise on operator+

- What are the possible ways to provide the following feature:

  A is a Complex number.

  C = 2.1 + A; // Add 2.1 (double) to the
  // real part of A.

Another Exercise on operator+ : Solution

- Recollect that 2.1 + A is equivalent to one of the following:
  - 2.1.operator+(A);
  - operator+(2.1, A);

The first one is not possible since you can't redefine + on double - built in datatype.

Only option (not considering type conversion): provide

Complex operator+(double val, const Complex& a);

Writing the operator+ for 2.1 + A

```
Complex operator+(double val, const Complex& A)
{
   double realpartofa, impartofa;

   a.get(realpartofa, impartofa);        // Function call Overhead

   Complex temp;
   temp.set(val + realpartofa, impartofa);      // Function call Overhead
   return temp;
}
```

Eliminating Overhead - that is what friends are for ?!

```
class Complex
{  ...
   friend Complex operator+(double val, const Complex& a);
}

Complex operator+(double val, const Complex& A)
{
   Complex temp;
   temp.rep = val + a.rep;       // Direct access to A's data, and temp's
   temp.imp = a.imp;             // Direct access to A's data, and temp's
   return temp;
}
```

Should I write a member function or a global friend function?

- Pure object-oriented languages allow only member functions. In C++ you may have a choice
- Some functions should be members
  - operator=
- Member do not introduce global names - use these in absence of other reasons
- If implicit type conversion is desired, for all operands of an operation, use global functions.
- If an operation modifies an operand, rather than merely returning a result, use member.

Cascading Operators

- D = A + B + C;
  - Where A, B and C are Complex
- D = A + B + 2.1;
- D = 2.1 + A + 3.2;

All that it takes is a proper return type in the operator overloaded function.

Lab Work: Details provided on-line.