Chapter 8

- Uniry operator (++i,i++,--i,i--)

  Void operator ++( ) { } --- > prefix (++obj)

  Or --

  Void operator ++(int){ } ----> postfix (obj++)

  Or --

- Temporary object

  First form -----> Class_name temp;

  Return temp;

  Second form ----> Class_name (int x):c(x){ }

  Constructor need to return temporary obj.
  Func( ) {
  Return class_name(x);
  This called un named obj (use constructor)

- Binary operator ---> A+B

Form-->

Class distance;

d1 operator + (distance d2)

{ feet = d2.feet}

المعامل الاول المعامل الثانى

Another operators can overload As: - / * > < == =

Note : * left operand -> access directly

Right operand -> function arugment

Ex. Dist 3 = dist1 + dist2

Main attribute

. Always operand before (sign) carry the other operand

EX. D1+= d2;

This operator overload will be void as we only add data of d2 to d1 .(change itself)

- Subscript operator [ ] --> can use as overloaded 3 ways can make same function of [ ]

We can make this function of subscript in 3 ways:

1.seperate get and put functions

. x=obj.get (i)

.x=obj.put(i)

2. single access return by reference

Ex. Int & func( ) --> return by reference

Can make operation or change it's value

Function--> (put & get)

X=obj. access(i) can get & put

3. overload [ ] operator ---> return by reference

Form---> int & operator [ ] (int x) { return arr[x]; }

X=obj[indx]

- converstion

1-converstion from user define to datatype (casting)

Operator datatype ( ) const{

Datatype x;

Return x; }

Datatype y= obj;

Or

Y=static-cast<datatype>(obj)

2- convert from datatype to userdefine

(AS pass parameter to constructor)

1-- > constructor (int x)

{ inch =x ; } ----> no return (bec. This is constructor)

Class_name obj(y); or obj= y;

3- from c-string to string (additions)

Form----> operator char*( ){ return str;}

Important: (little confuse**)**

4- convert between obj of different classes

We should call the class we need before we use it .

Class1 obj1, class2 obj2;

1. Obj1= obj2 --> source of overloading

Form ---> class2: operator class1 ( ) const { }

(=) sign with class1

2. Obj1(obj2)
3. In this way will need func to access private data of obj2 ---> (parameter of function)

constructor doesn't return value

Form-->

class1_name(class x){ }; --> constructor of calss1

EX. Obj1(obj1) --> maintain auto by compile if they made with same class

EX. Obj1(obj2) --> hand made as they diff.

Obj1=obj2 --> diff style in writing

- UML Diagram

Some rules:

1. Association : one to one corresponding between (obj& real life)
2. Navigability : detect navigability of the association between classes

---> unidirectional Association

---> bidirectional Association

Gidlines-->

1. Make similar meaning

AS: (+) overaload to sum not subtract

2.Use similar syntax

AS: use (+) not(~) for summation operation

Avoid Ambigoity

AS: avoid doing some converstion in more than one way.

Finally-->

. :: ?: -> * & (can’t overloaded).