* overloading the binary operators as member functions

* ال prototype بتكون جوا ال class :-

```
returntype operator* (const classname &)const;
```

* function definition :-

```
returntype classname:: operator* (const classname &other obj)const
{
    //algorithm to perform the operation
    return vlaue;
}
```

* overloading the Arithmetic or relational operators :-

* Function prototype :-

```
friend returntype operator* (const classname &, const
                                       classname &);
```

* Function definition :-

returntype operator** (const classname &firstobj,
const classname &secondobj)

{

// algorithm to perform the operation
return value;

}

→ overloading the stream insertion operator (<<) :-

* Function prototype :-

friend ostream & operator << (ostream &, const
classname &);

* Function definition :-

ostream & operator << (ostream & osobj, const classnam
& cobj)

{

return osobject;

{

overloading the stream extraction operator (>>) :-

*Function prototype :-

```
friend istream & operator>>(istream &, classname &);
```

*Function definition :-

```
istream & operator>> (istream & is object, classname & cobj)
{
    return is object;
}
```

overloading the Assignment operator (=) :-

*Function prototype :-

```
const classname & operator=(const classname &);
```

*Function definition :-

```
const classname & classname :: operator=(const classname &
{   if(this != & right object)
    {   return *this;
    }
```

* overloading Unary operators *

↝ If the unary operator is a member of the class, it has no parameters

↝ If the operator function is a nonmember it has one parameter.

* overloading the Increament operators :-

① pre-increament / member function

classname operator++(); ⇒ Prototype

```
classname classname:: operator++()
{
    return *this;
}
```

=

definition

② pre-increment / nonmemberfunction :-

friend classname operator++(classname &); ← prototype

* Function difinition :-

```
classname operator++(classname & incobj)
{
    return incobj;
}
```

③ post-increment / memberfunction :-

* prototype :-

```
classname operator++(int );
```

* definition :-

```
{
  classname temp = *this;
  return temp;
}
```

(4) post-increment / nonmember function :-

* Function prototype :-

```
friend classname operator++(classname&, int);
```

* Function difinition :-

```
classname operator++(classname & incobj, int u)
{
classname temp = incobj;

 return temp;
}
```

(Decrement) الاربع اشكال هدول بقدر أكررهم بـ

※ سلايدات ـ الدكتورة هبة Ch13

※ راجع سلايد 25-26 من Ch13
27

overloading the Array Index (subscript) operator ([ ])

* non constant arrays :-

Type & operator [ ](int index);

* constant arrays :-

const Type & operator [ ](int index) const;

* Function overloading :- اكثر من function الهم نفس الاسم بس بختلفوا بال parameters list

نوع ال D.T لل formal parameters

عدد ال formal Parameters

* ما بيفرق عنا ال return D.T.