CHAPER 5:

OPERATOR OVERLOADING.

5hours.

Operator Overloading:

- When an Operator is overloaded with multiple jobs, It is known as operator overloading.
- It is the important technique that has enhanced the power of extensibility of C++.
- It is a way to implement compile time polymorphism.
- **Operator overloading** allows you to redefine the way **operator** works for user-defined types only (objects, structures) not for primitive or built_in type.

Overloadable Operator:

- We can overload the following operators.

+	-	*	/	%	^
&	|	~	!	,	=
<	>	<=	>=	++	--
<<	>>	==	!=	&&	||
+=	-=	/=	%=	^=	&=
|=	*=	<<=	>>=	[]	()
->	->*	new	new []	delete	delete []

- Following operators are not overloadable:

1. Class member access operator (., .*)
2. Scope resolution operator (::)
3. Size of operator (sizeof)
4. Conditional operator (?:)
5. run_time type information operator(type id)

Rules of Operator Overloading:

- Any symbol can be used as function name:
  1. If it is valid operator in C language.
  2. If it is preceded by operator keyword.
- Operators cannot be overloaded for built in types only. At least one operand must be user defined type.
- Assignment (=), subscript ([]), function call ("()"), and member selection (->) operators must be defined as member functions. All other operators can be either member functions or a non member functions.
- Some operators like (assignment)=, (address)& and comma (,) are by default overloaded.

Syntax Operator Overloading:

```
 return_type  operator  operator_symbol ( arg_list )
{
 //body of function;
}
E.g. Complex operator +()
{
}
```

Unary Operating Overloading:

- The operator which operates on single operand (data) are called unary operator.
- E.g. int x=2;

```
        a++;

        ++a;

         int b= -a;
```

Syntax:

```
return_type operator operator_symbol () //prefix
return_type operator operator_symbol (int)  //postfix
```

Binary Operator Overloading:

- Binary operator operates on two operands (data) .
- The binary operator function can be defined by either a non static member function taking one argument or a non member function (usually global function) taking two argument.

Operator Overloading With Member and Non Member Function:

- We will perform these topics on whiteboard….

Data Conversion:

1.Basic – User Defined (Primitive type to class type):

To perform this conversion, the idea is to use the constructor to perform type conversion during the object creation.

2. User Defined – Basic (Class type to primitive type) :

In this conversion, the **source** type is a class object and the **destination** type is primitive data type. To perform this conversion, the idea is to use the **casting operator** to perform type conversion.

The normal form of an casting operator (Syntax):

```
 operator  typename () {
   // Code
     return  (type-data);   }
```

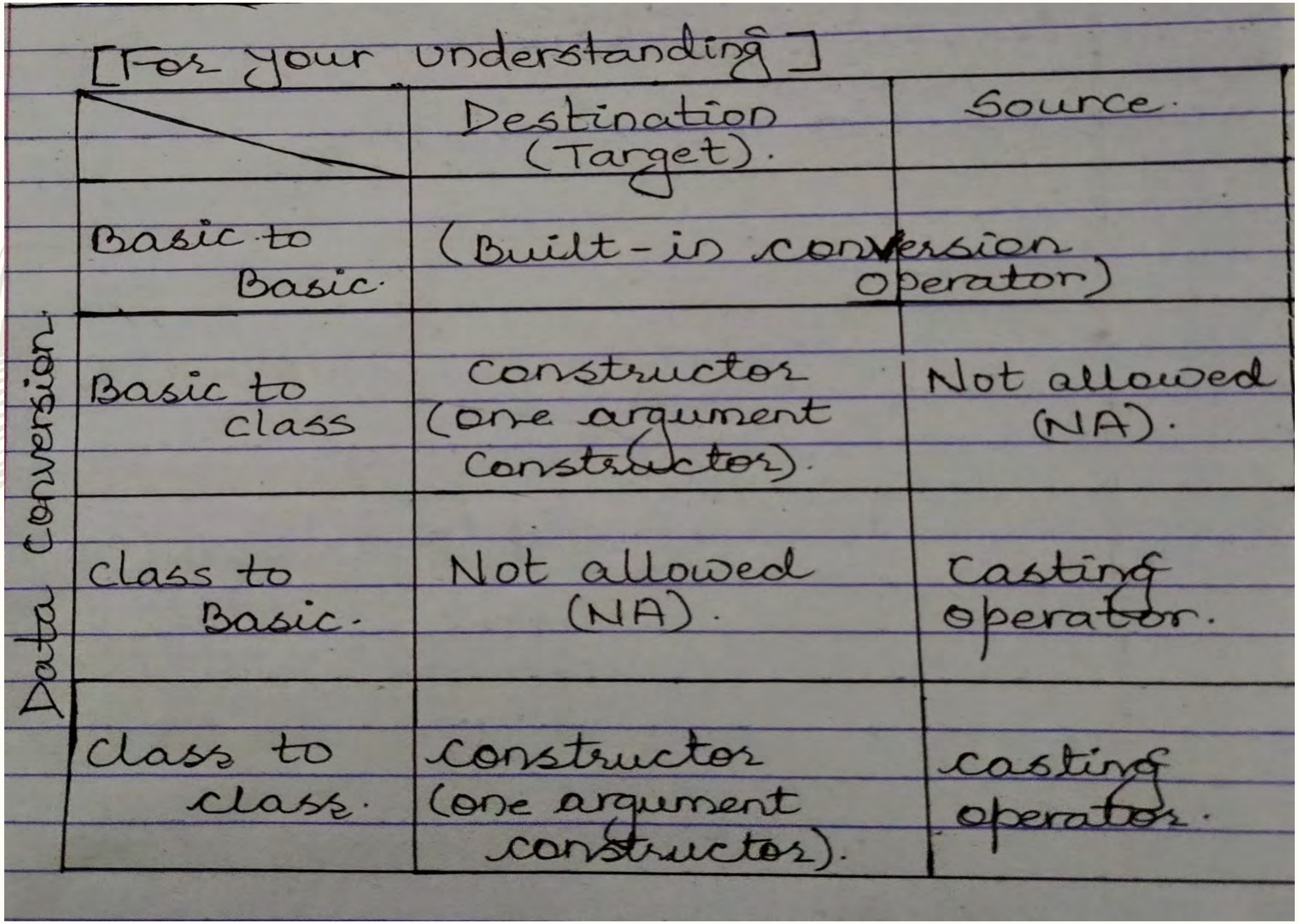
Now, this function converts a **user-defined data type** to a **primitive data type**. For Example, the operator **float()** converts a class object to type float, the operator **int()** converts a class type object to type int, and so on.

3. User Defined –User Defined : In this type, one class type is converted into another class type. It can be done in 2 ways :

1.Using constructor.

2.Using casting operator.

[**we will program for all data conversion types in class.]



[For your understanding]

Data Conversion

	Destination (Target).	Source.
Basic to Basic.	(Built-in conversion Operator)	
Basic to class	constructor (one argument constructor).	Not allowed (NA).
class to Basic.	Not allowed (NA).	casting operator.
class to class.	constructor (one argument constructor).	casting operator.

Explicit Constructor:

Q. Why explicit constructor is used?

Ans. Explicit constructor is used to avoid implicit call to the constructor.

[we will program explicit constructor in class.]

Assignment: new delete overloading , assignment overloading , string manipulation in oo , << >>operator overloading etc….