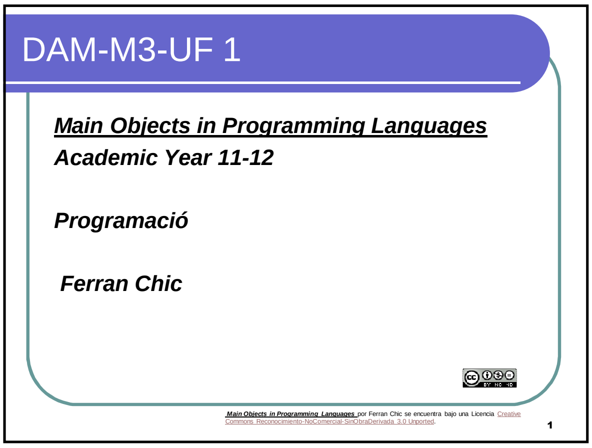


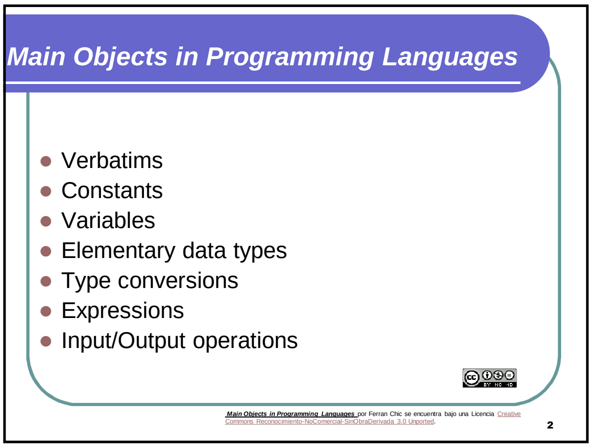
DAM-M3-UF 1

Main Objects in Programming Languages

Academic Year 11-12

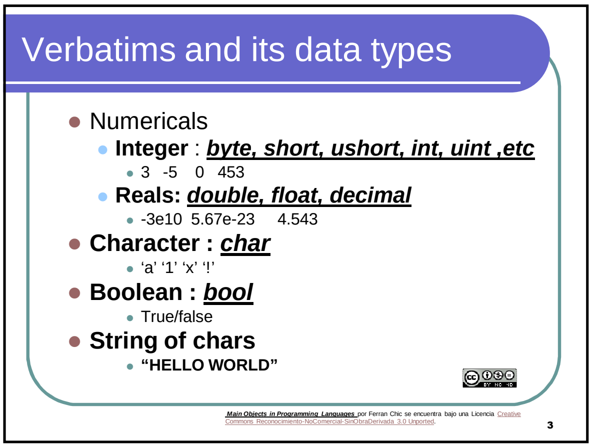
Programació

Ferran Chic



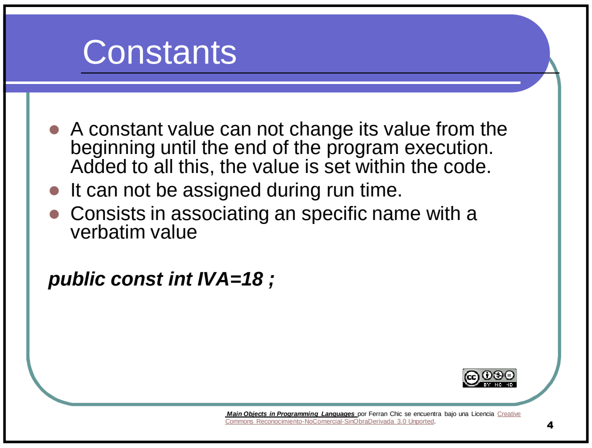
Main Objects in Programming Languages

- Verbatims
- Constants
- Variables
- Elementary data types
- Type conversions
- Expressions
- Input/Output operations



Verbatims and its data types

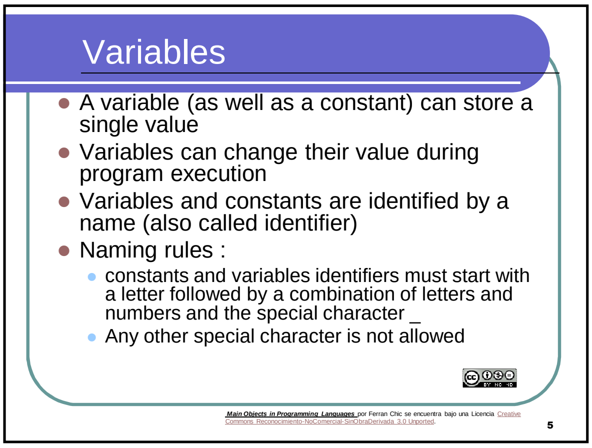
- Numericals
  - **Integer** : ***byte, short, ushort, int, uint ,etc***
    - 3 -5 0 453
  - **Reals: *double, float, decimal***
    - -3e10 5.67e-23 4.543
- **Character : *char***
    - 'a' '1' 'x' '!'
- **Boolean : *bool***
    - True/false
- **String of chars**
    - **"HELLO WORLD"**



Constants

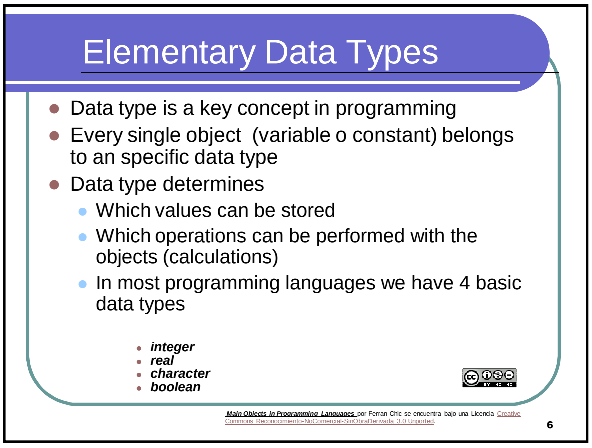
- A constant value can not change its value from the beginning until the end of the program execution. Added to all this, the value is set within the code.
- It can not be assigned during run time.
- Consists in associating an specific name with a verbatim value

public const int IVA=18 ;



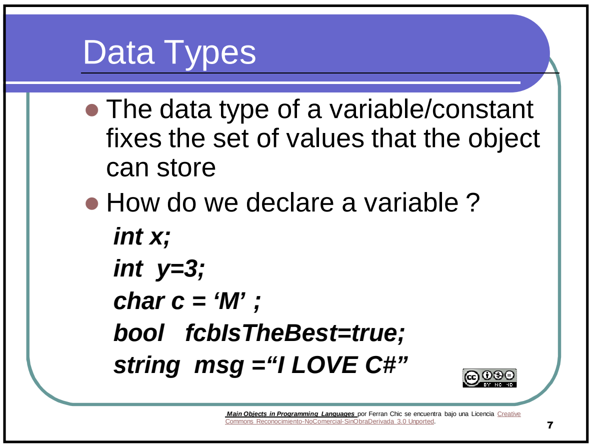
Variables

- A variable (as well as a constant) can store a single value
- Variables can change their value during program execution
- Variables and constants are identified by a name (also called identifier)
- Naming rules :
  - constants and variables identifiers must start with a letter followed by a combination of letters and numbers and the special character _
  - Any other special character is not allowed



Elementary Data Types

- Data type is a key concept in programming
- Every single object (variable o constant) belongs to an specific data type
- Data type determines
  - Which values can be stored
  - Which operations can be performed with the objects (calculations)
  - In most programming languages we have 4 basic data types
    - ***integer***
    - ***real***
    - ***character***
    - ***boolean***



Data Types

- The data type of a variable/constant fixes the set of values that the object can store
- How do we declare a variable ?

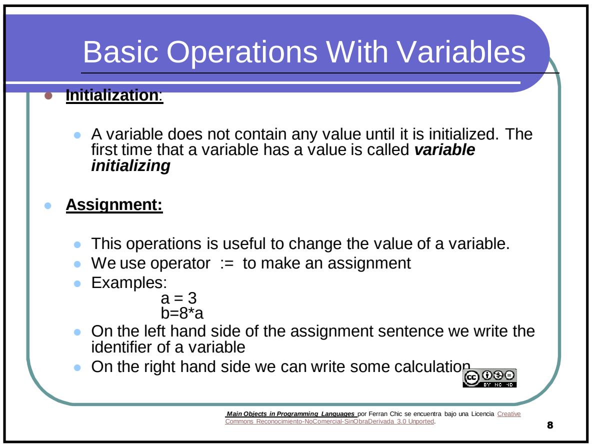
int x;

int y=3;

char c = ‘M’ ;

bool   fcbIsTheBest=true;

string msg =“I LOVE C#”

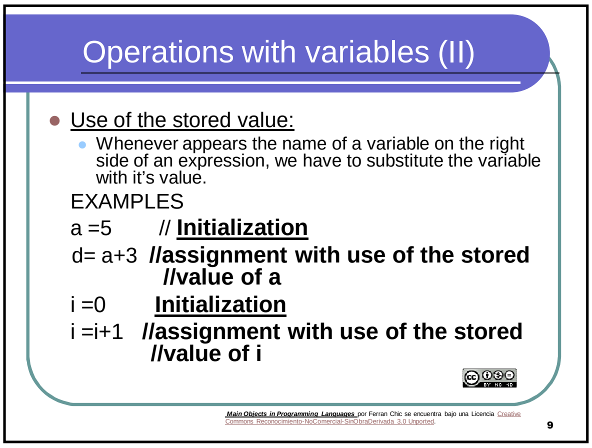


Basic Operations With Variables

- **Initialization**:
  - A variable does not contain any value until it is initialized. The first time that a variable has a value is called ***variable initializing***
- **Assignment:**
  - This operations is useful to change the value of a variable.
  - We use operator := to make an assignment
  - Examples:

```
a = 3
b=8*a
```

  - On the left hand side of the assignment sentence we write the identifier of a variable
  - On the right hand side we can write some calculation



Operations with variables (II)

- Use of the stored value:
  - Whenever appears the name of a variable on the right side of an expression, we have to substitute the variable with it’s value.

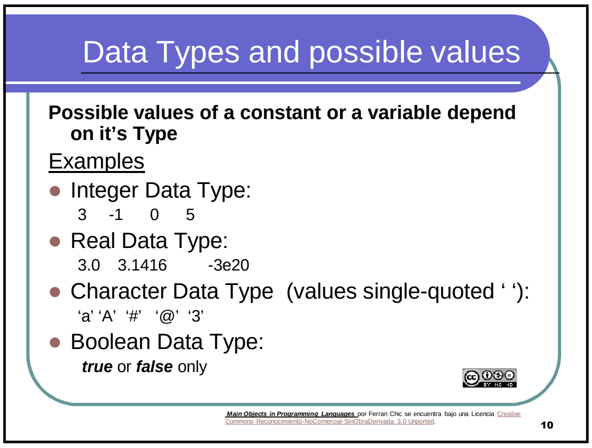
EXAMPLES

a =5   // **Initialization**

d= a+3 **//assignment with use of the stored //value of a**

i =0   **Initialization**

i =i+1   **//assignment with use of the stored //value of i**

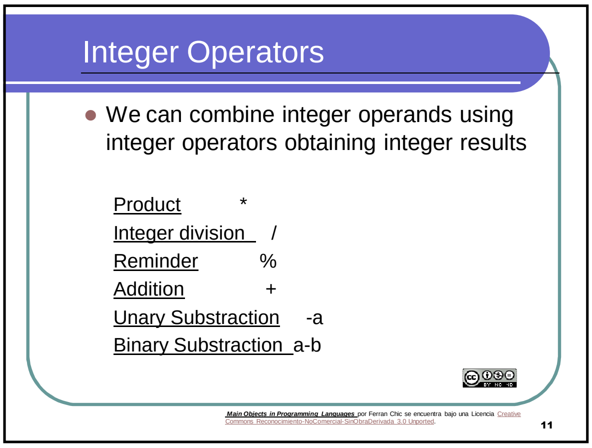


Data Types and possible values

Possible values of a constant or a variable depend on it's Type

Examples

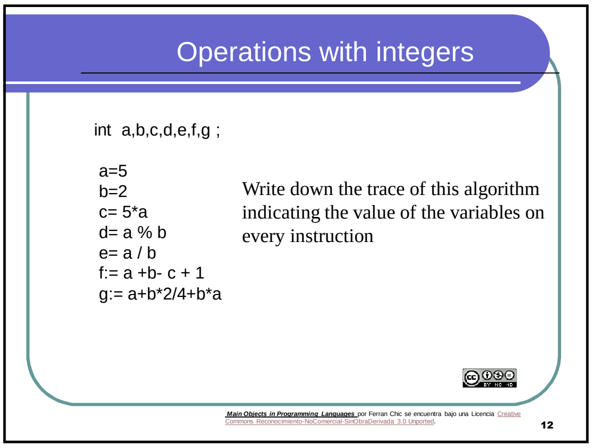
- Integer Data Type:
  3 -1 0 5
- Real Data Type:
  3.0 3.1416 -3e20
- Character Data Type (values single-quoted ' '):
  'a' 'A' '#' '@' '3'
- Boolean Data Type:
  true or ***false*** only



Integer Operators

- We can combine integer operands using integer operators obtaining integer results

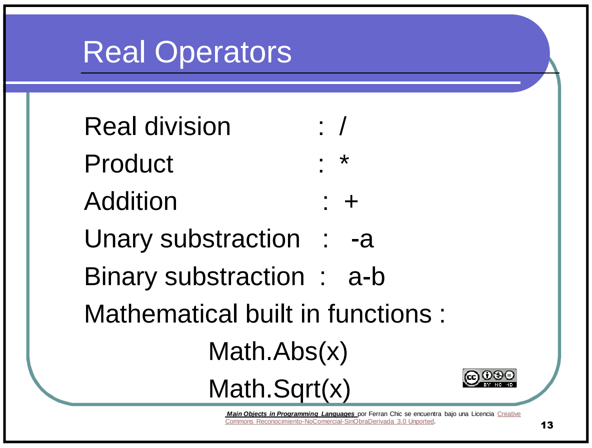
Product	*
Integer division	/
Reminder	%
Addition	+
Unary Substraction	-a
Binary Substraction	a-b



Operations with integers

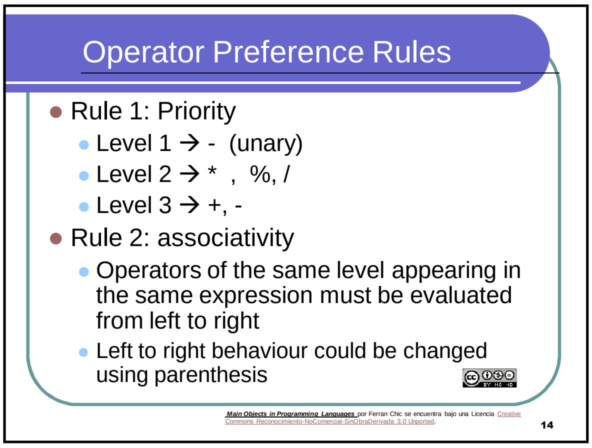
```
int  a,b,c,d,e,f,g ;

 a=5
 b=2
 c= 5*a
 d= a % b
 e= a / b
 f:= a +b- c + 1
 g:= a+b*2/4+b*a
```

Write down the trace of this algorithm indicating the value of the variables on every instruction



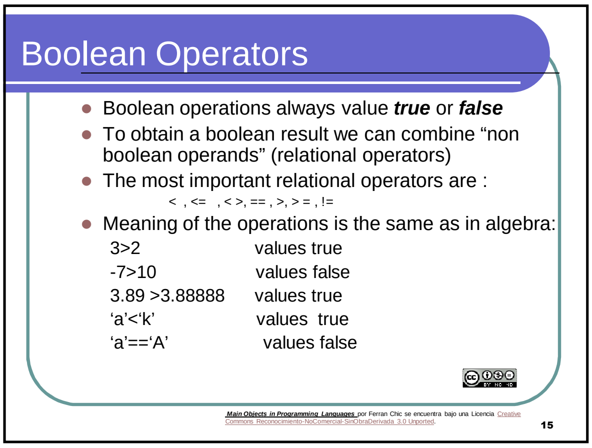
Real Operators

Real division : /
Product : *
Addition : +
Unary substraction : -a
Binary substraction : a-b
Mathematical built in functions :
Math.Abs(x)
Math.Sqrt(x)



Operator Preference Rules

- Rule 1: Priority
  - Level 1 → - (unary)
  - Level 2 → * , %, /
  - Level 3 → +, -
- Rule 2: associativity
  - Operators of the same level appearing in the same expression must be evaluated from left to right
  - Left to right behaviour could be changed using parenthesis

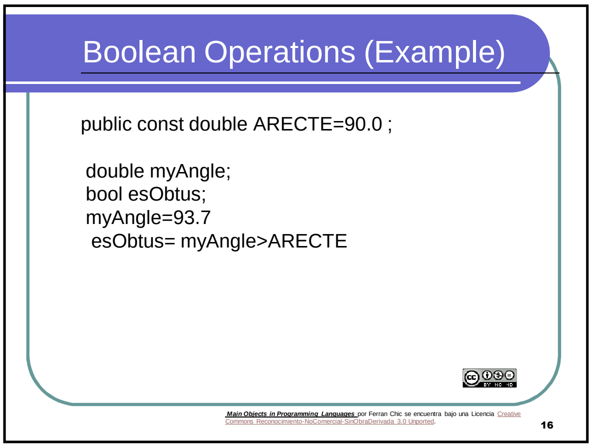


Boolean Operators

- Boolean operations always value ***true*** or ***false***
- To obtain a boolean result we can combine “non boolean operands” (relational operators)
- The most important relational operators are :

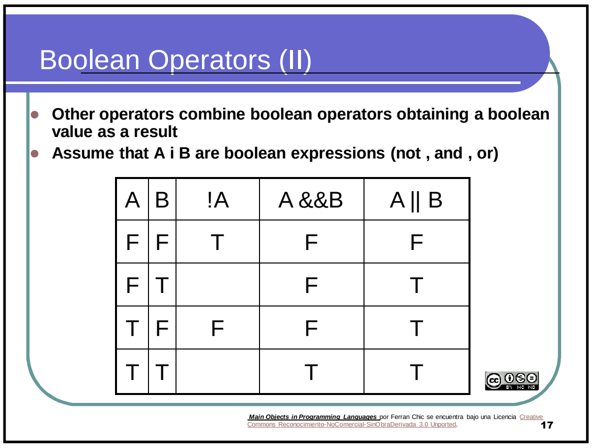
  < , <= , < >, == , >, > = , !=
- Meaning of the operations is the same as in algebra:

3>2	values true
-7>10	values false
3.89 >3.88888	values true
‘a’<‘k’	values true
‘a’==‘A’	values false



Boolean Operations (Example)

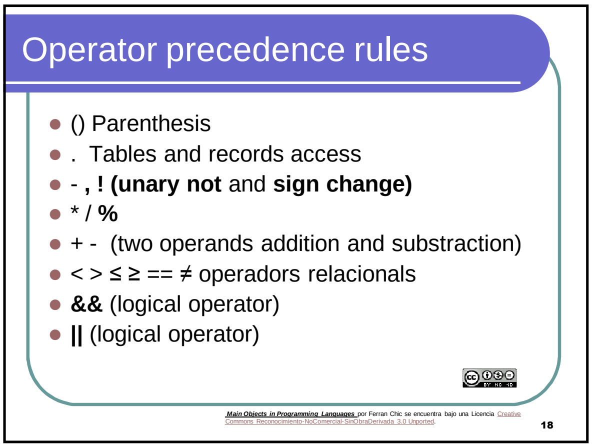
```
public const double ARECTE=90.0 ;

 double myAngle;
 bool esObtus;
 myAngle=93.7
  esObtus= myAngle>ARECTE
```



Boolean Operators (II)

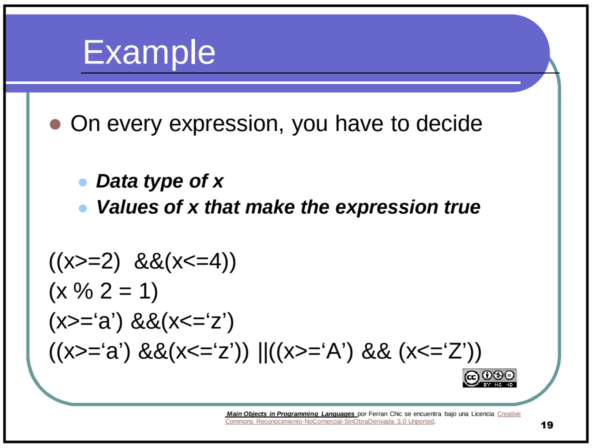
- Other operators combine boolean operators obtaining a boolean value as a result
- Assume that A i B are boolean expressions (not , and , or)

A	B	!A	A &&B	A || B
F	F	T	F	F
F	T		F	T
T	F	F	F	T
T	T		T	T



Operator precedence rules

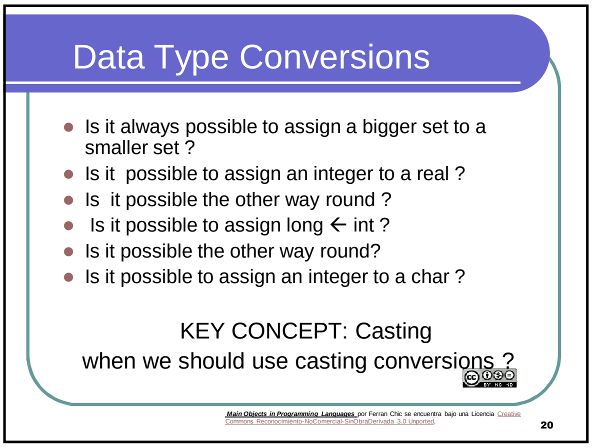
- () Parenthesis
- . Tables and records access
- **- , ! (unary not** and **sign change)**
- *** / %**
- + - (two operands addition and substraction)
- < > ≤ ≥ == ≠ operadors relacionals
- **&&** (logical operator)
- **||** (logical operator)



Example

- On every expression, you have to decide
  - ***Data type of x***
  - ***Values of x that make the expression true***

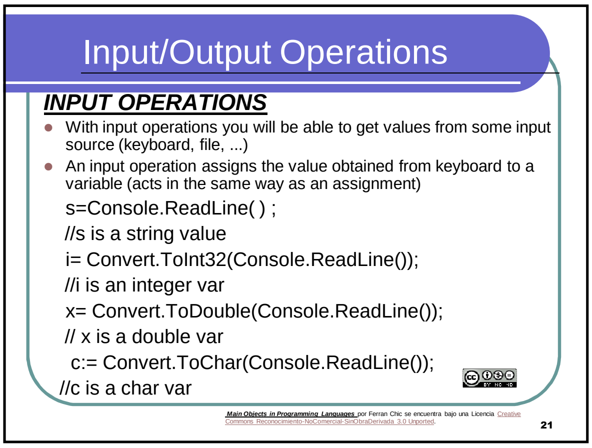
((x>=2) &&(x<=4))
(x % 2 = 1)
(x>='a') &&(x<='z')
((x>='a') &&(x<='z')) ||((x>='A') && (x<='Z'))



Data Type Conversions

- Is it always possible to assign a bigger set to a smaller set ?
- Is it possible to assign an integer to a real ?
- Is it possible the other way round ?
- Is it possible to assign long ← int ?
- Is it possible the other way round?
- Is it possible to assign an integer to a char ?

KEY CONCEPT: Casting
when we should use casting conversions ?

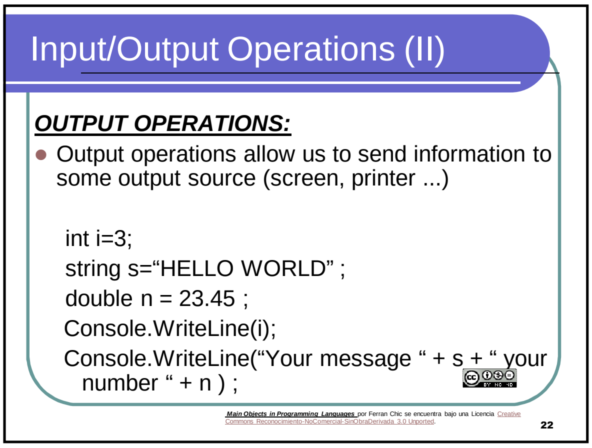


Input/Output Operations

INPUT OPERATIONS

- With input operations you will be able to get values from some input source (keyboard, file, ...)
- An input operation assigns the value obtained from keyboard to a variable (acts in the same way as an assignment)

```
s=Console.ReadLine( ) ;
//s is a string value
i= Convert.ToInt32(Console.ReadLine());
//i is an integer var
x= Convert.ToDouble(Console.ReadLine());
// x is a double var
 c:= Convert.ToChar(Console.ReadLine());
//c is a char var
```



Input/Output Operations (II)

OUTPUT OPERATIONS:

- Output operations allow us to send information to some output source (screen, printer ...)

```
int i=3;
string s=“HELLO WORLD” ;
double n = 23.45 ;
Console.WriteLine(i);
Console.WriteLine(“Your message “ + s + “ your
   number “ + n ) ;
```