Programming In C

Part III

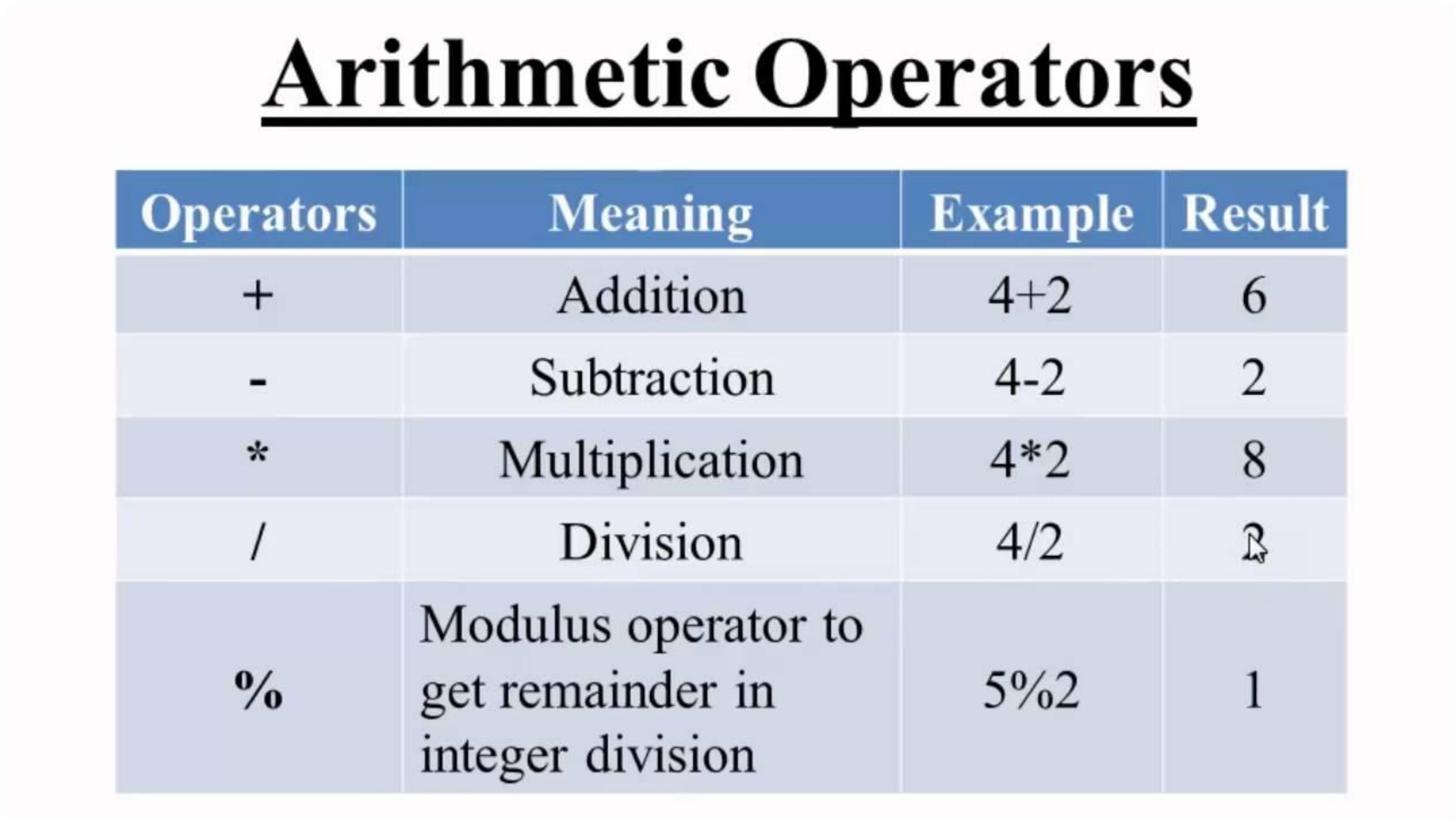
Operators, and Expressions

C-Operators

- Operators are used to manipulate data
  - Perform specific mathematical or logical functions.
- C language provides the following types of operators:
  - Arithmetic Operators
  - Relational Operators
  - Logical Operators
  - Assignment Operators
  - Increment/Decrement Operators
  - Other Operators (conditional and sizeof)

Arithmetic Operators

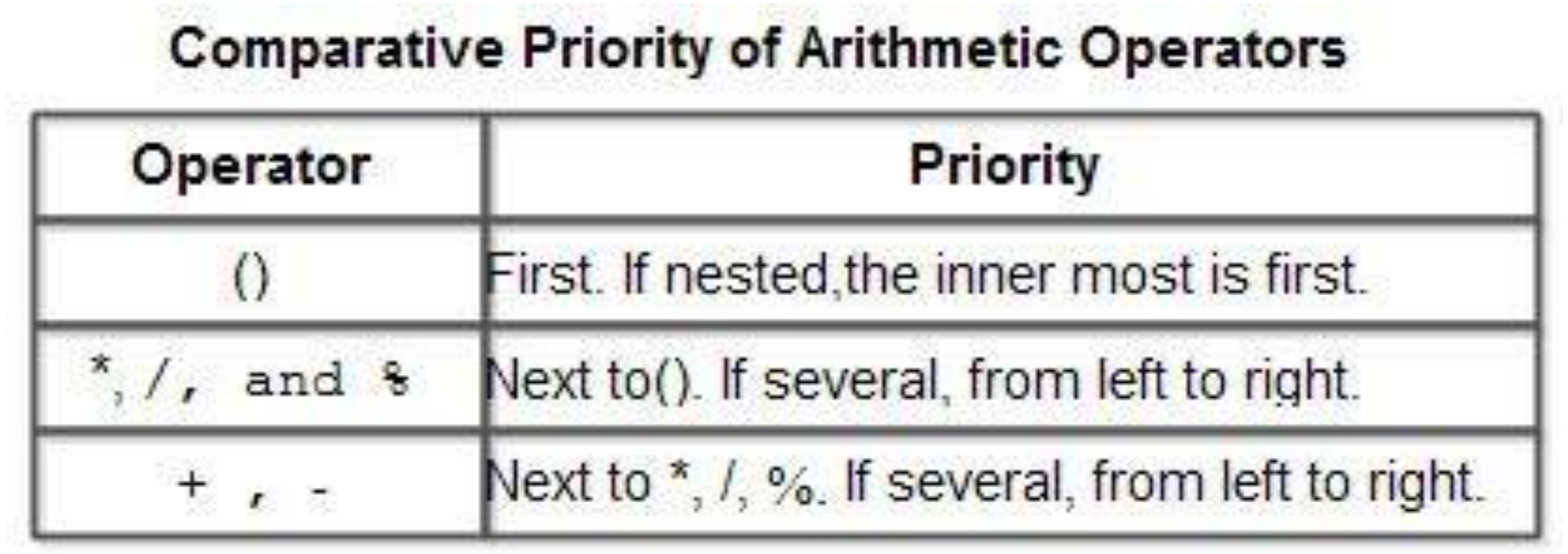
- Arithmetic operators are used to perform numerical calculations among the values.



Arithmetic Operators

Operators	Meaning	Example	Result
+	Addition	4+2	6
-	Subtraction	4-2	2
*	Multiplication	4*2	8
/	Division	4/2	2
%	Modulus operator to get remainder in integer division	5%2	1

Precedence of Arithmetic Operators



Comparative Priority of Arithmetic Operators

Operator	Priority
()	First. If nested,the inner most is first.
*, /, and %	Next to(). If several, from left to right.
+ , -	Next to *, /, %. If several, from left to right.

Precedence of Arithmetic Operators

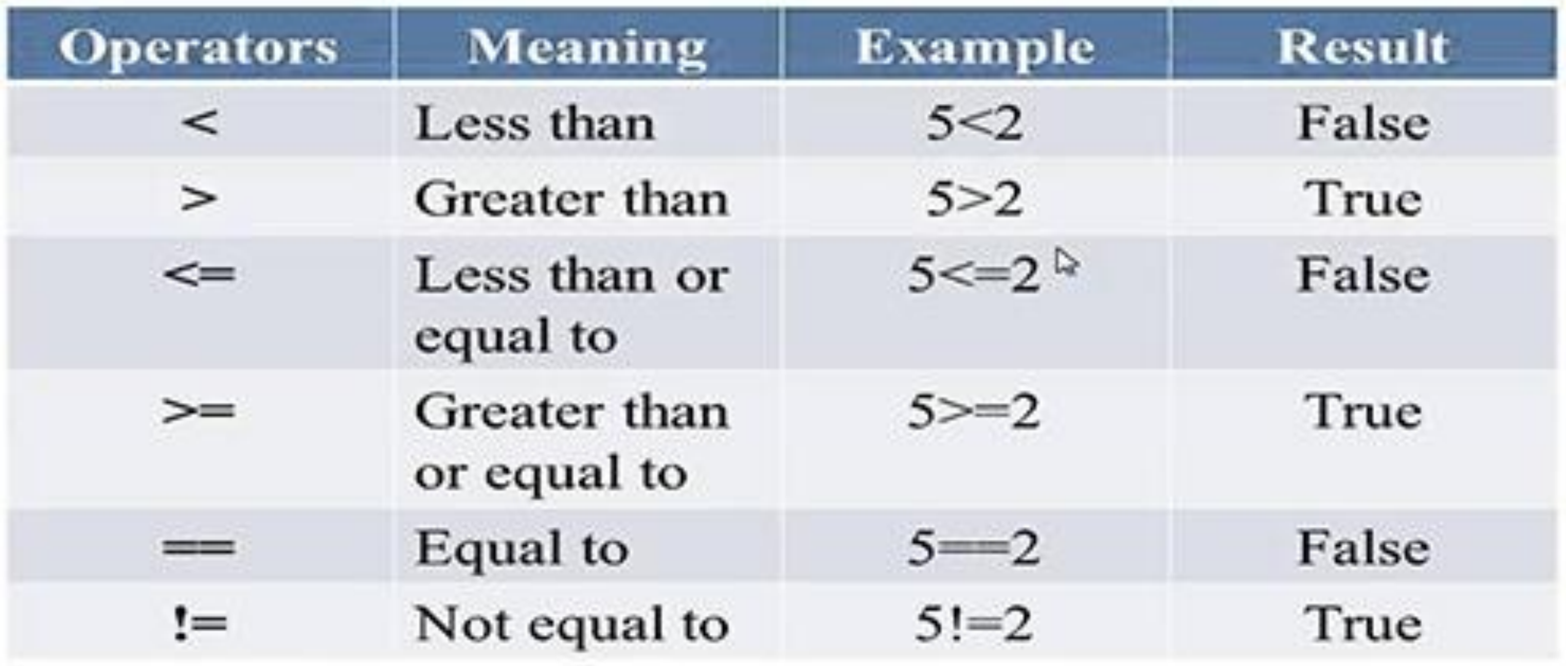
- (*,/, and %) are executed first, followed by (+, and -).
- Operators of the same precedence are executed sequentially(from left to right)

  2+3-4+5 = ((2+3)-4)+5) = ((5-4)+5) = (1+5) = 6
- Parenthesis can be used to override the evaluation or

  (2+3)-(4+5) = 5-9 = -4

Relational Operators

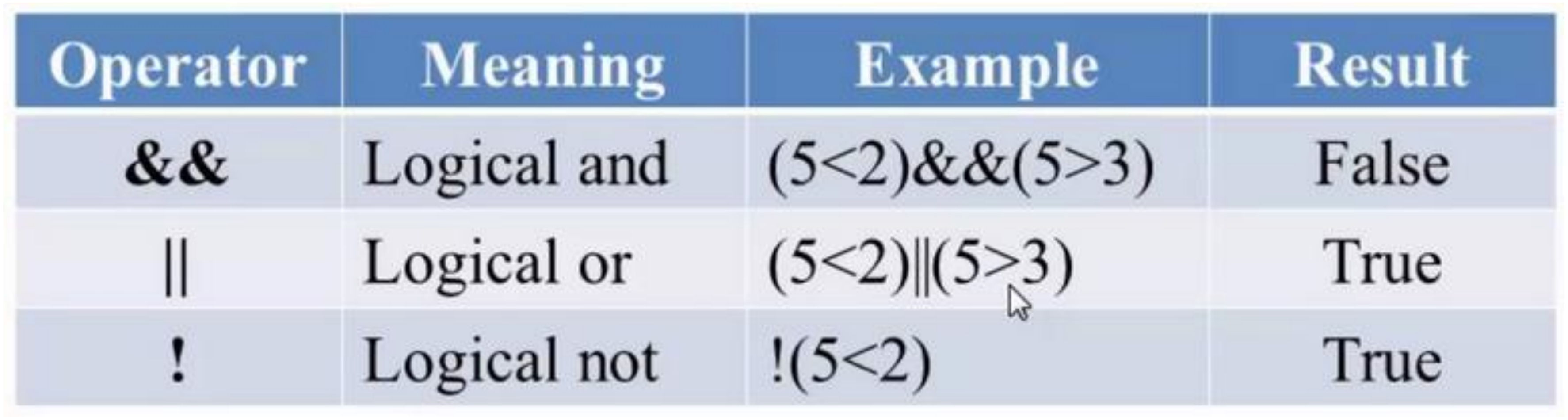
- Relational Operators are used to compare two quantities and take certain decision depending on their relation.
  - The specified relation is either true or false.



Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

Logical Operators

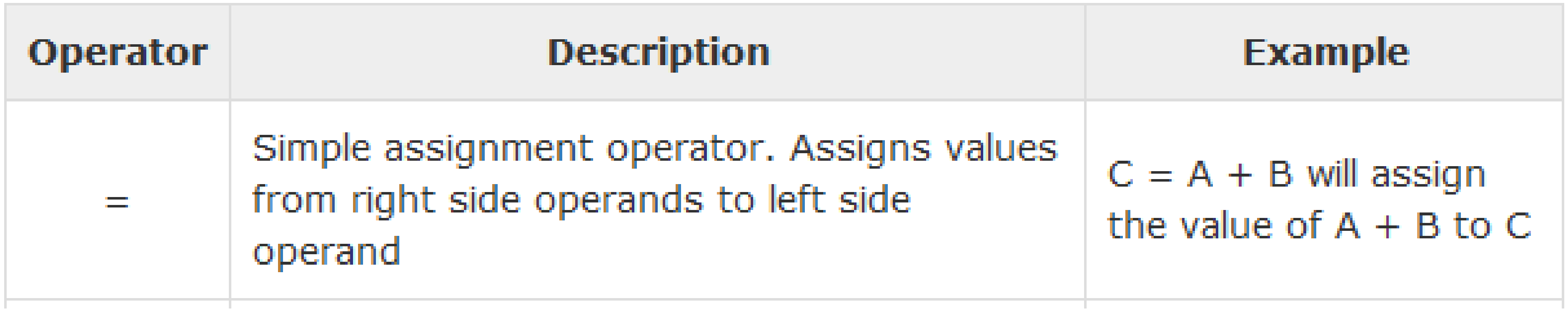
- These operators are used for testing more than one condition and making decisions. C language has three logical operators they are:



Operator	Meaning	Example	Result
&&	Logical and	(5<2)&&(5>3)	False
||	Logical or	(5<2)||(5>3)	True
!	Logical not	!(5<2)	True

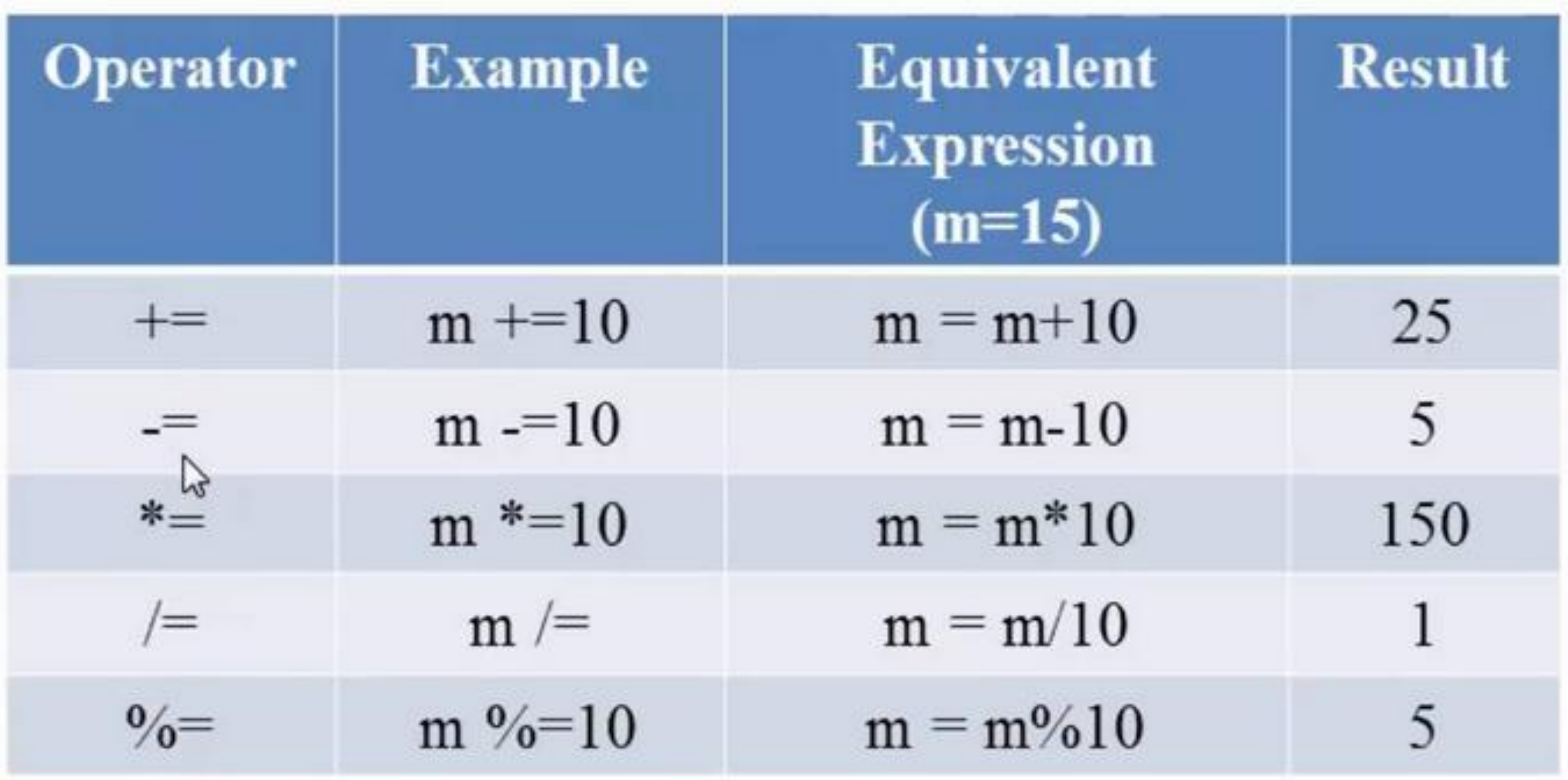
Assignment Operators

- An assignment operator is used for assigning the result of an expression to a variable.
- The most common assignment operator is the equal sign (=) which refers to (←) in algorithms.



Operator	Description	Example
=	Simple assignment operator. Assigns values from right side operands to left side operand	C = A + B will assign the value of A + B to C

Other Assignment Operators



Operator	Example	Equivalent Expression (m=15)	Result
+=	m +=10	m = m+10	25
-=	m -=10	m = m-10	5
*=	m *=10	m = m*10	150
/=	m /=	m = m/10	1
%=	m %=10	m = m%10	5

Implicit Data type Conversion

- If the type of the values in an expression are not the same, data type conversion is made.
- All the data types of the variables in that expression are upgraded implicitly to the data type of the variable with largest data type according to the flowing order:

  bool -> char -> int -> float -> double

Example

```
int a, x;

float z, y;

z = x + y;

/* x is first converted to float then x+y is evaluated and
   assigned to z*/

a = x + y;

/* x is first converted to float then x+y is evaluated. The
   result is then converted to int and assigned to a*/

z= a / x;

/* a/y is first evaluated. The result is then converted to
   float and assigned to z */
```

Example

```
int a, x; // x=3
float z, y; // y=2.000000
z = x + y;
//z = 3+2.000000 = 3.000000+2.000000 = 5.000000
/* x is first converted to float then x+y is evaluated and
    assigned to z*/
a = x + y;
//a = 3+2.000000 = 3.000000+2.000000 = 5
/* x is first converted to float then x+y is evaluated. The result
    is then converted to int and assigned to a*/
z= a / x;
//z = 5/3 = 1.000000
/* a/y is first evaluated. The result is then converted to float
    and assigned to z */
```

Explicit Data type Conversion

- Explicit type conversion is the process where the user can define the type to which the result is made of a particular data type.
- The syntax in C:

```
(type) expression;
```

Example

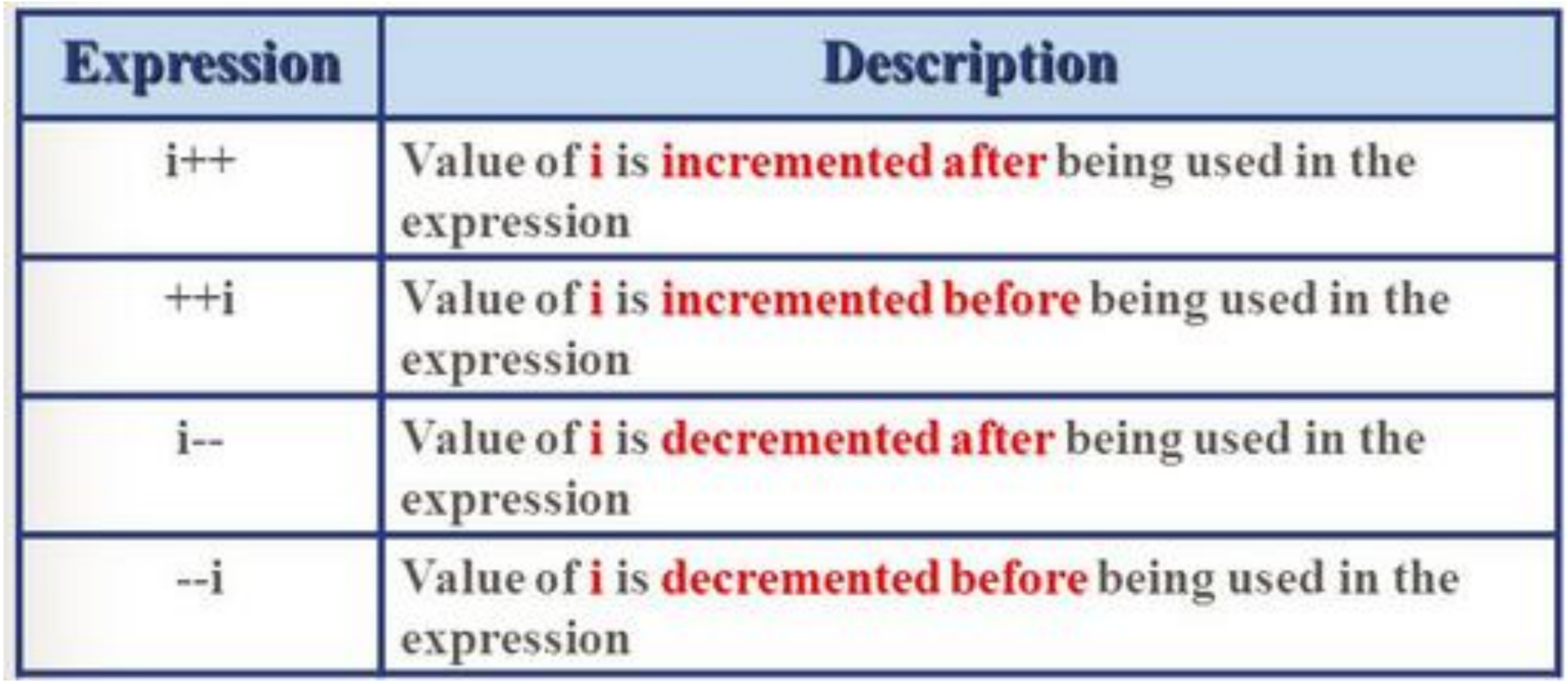
```
1) int  x=7, y=5 ;
    float z;
    z =x / y; /*Here the value of z is 1.000000*/

2) int  x=7, y=5;
    float z;
   z = (float)x / y; /*Here the value of z is 1.400000*/
```

Increment/Decrement Operators

- Two most useful operators which are present in C are increment and decrement operators.
- Operators: **++** and **--**
- The operator ++ **adds one** to the operand
- The operator -- **subtracts one** from the operand.

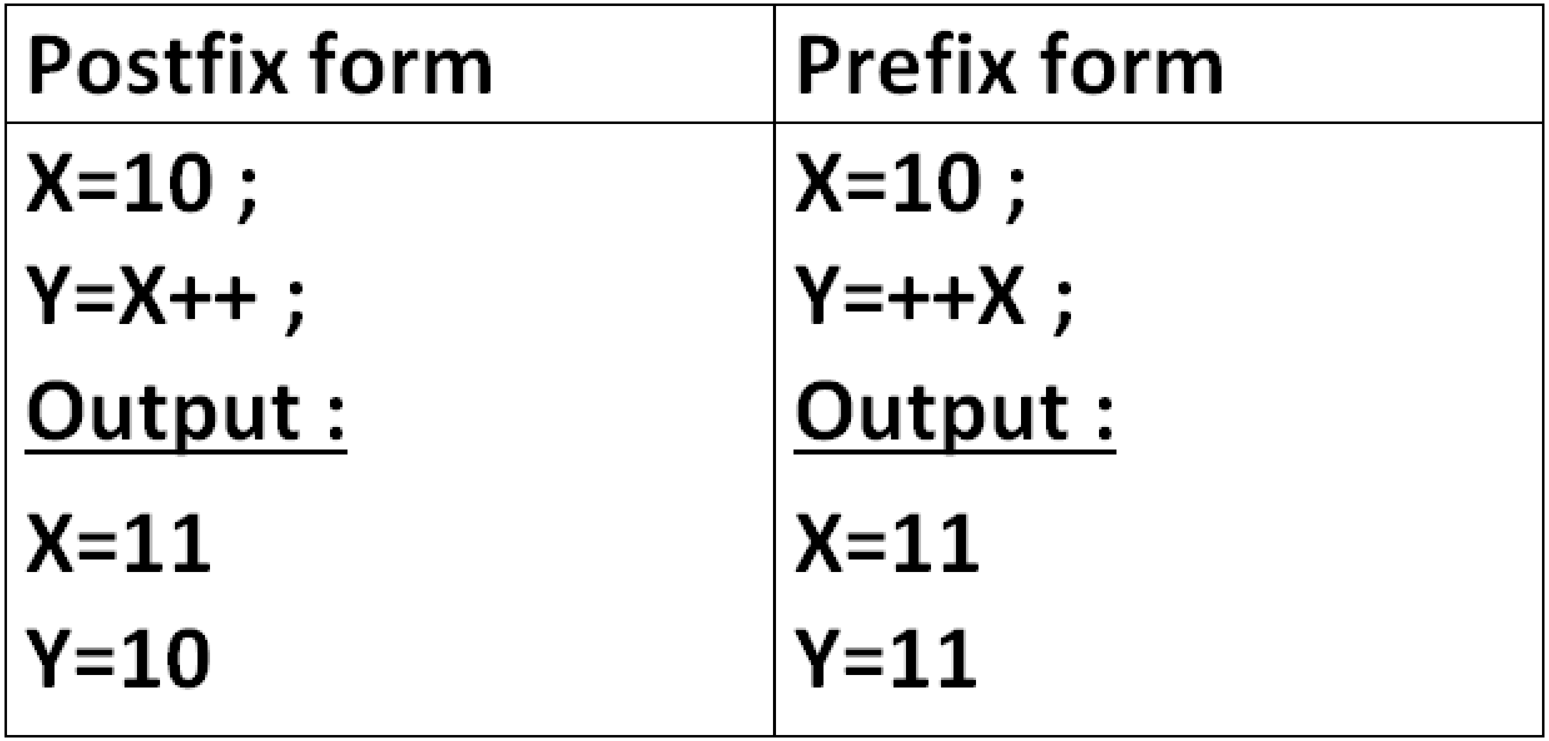
Prefix and Postfix

- Increment and decrement operators can be either prefix or postfix forms



Expression	Description
i++	Value of **i** is **incremented after** being used in the expression
++i	Value of **i** is **incremented before** being used in the expression
i--	Value of **i** is **decremented after** being used in the expression
--i	Value of **i** is **decremented before** being used in the expression

Postfix Vs Prefix form



Postfix form	Prefix form
X=10 ;	X=10 ;
Y=X++ ;	Y=++X ;
<u>Output :</u>	<u>Output :</u>
X=11	X=11
Y=10	Y=11

Conditional Operator

- The conditional operator is used to construct conditional expression of the form:

Syntax:

identifier=(test_expression)?expression1:expression2;

Meaning:

If test_expression is true then dentifier=expresion1, otherwise identifier=expression2.

- Examples:

```
x=(y>0)?y:-y;// if y>0 then x=y else x=-y
min=(x<y)?x:y;// if x<y then min=x else min=y
```

Sizeof Operator

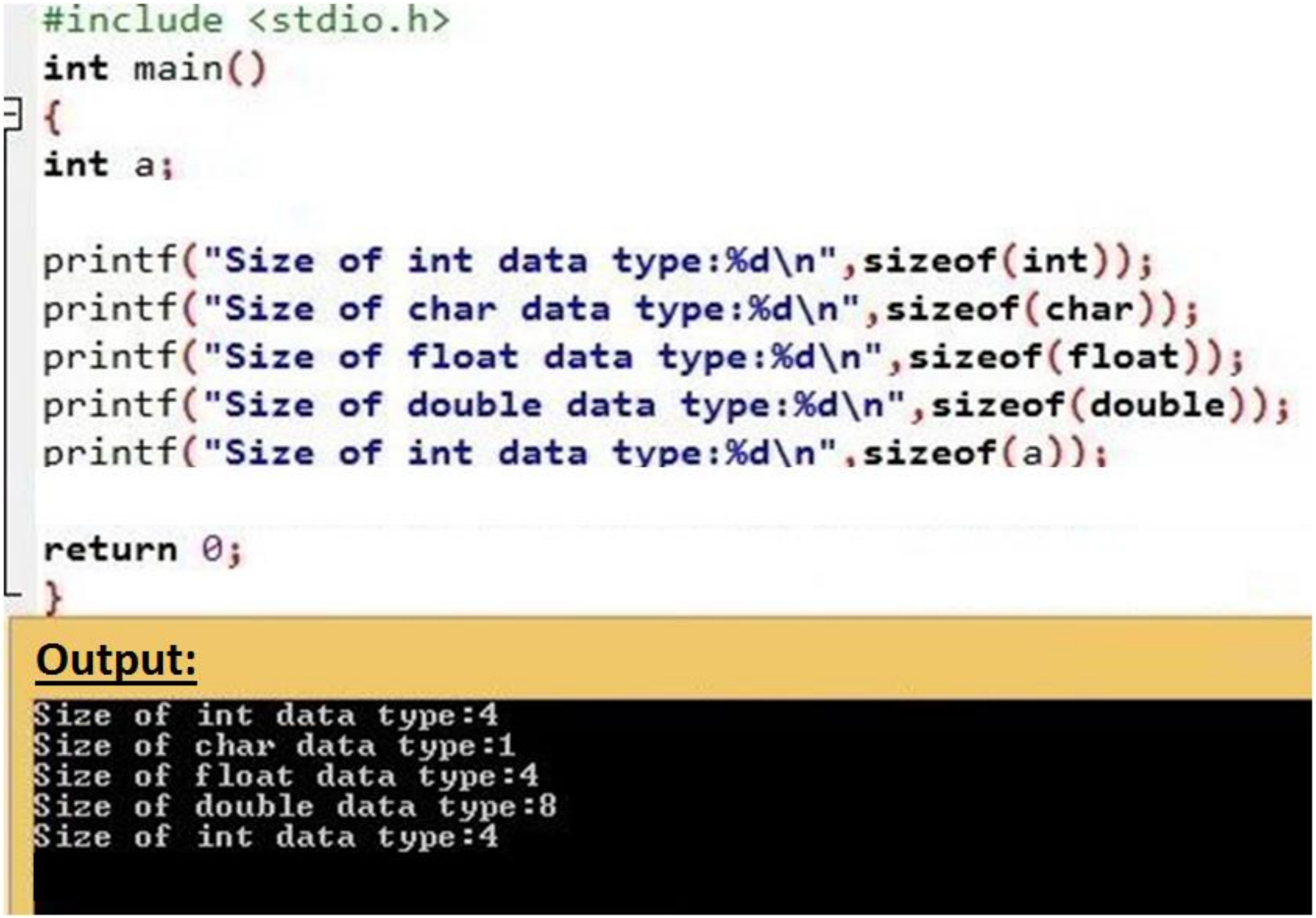
- Sizeof is an operator used to return the number of bytes the operand occupies.

<u>Example:</u>

```
int i , j;
j = sizeof(i); // j=4 because i is an integer and occupies 4 bytes.
```

Sizeof Operator

- <u>Another Example:</u>

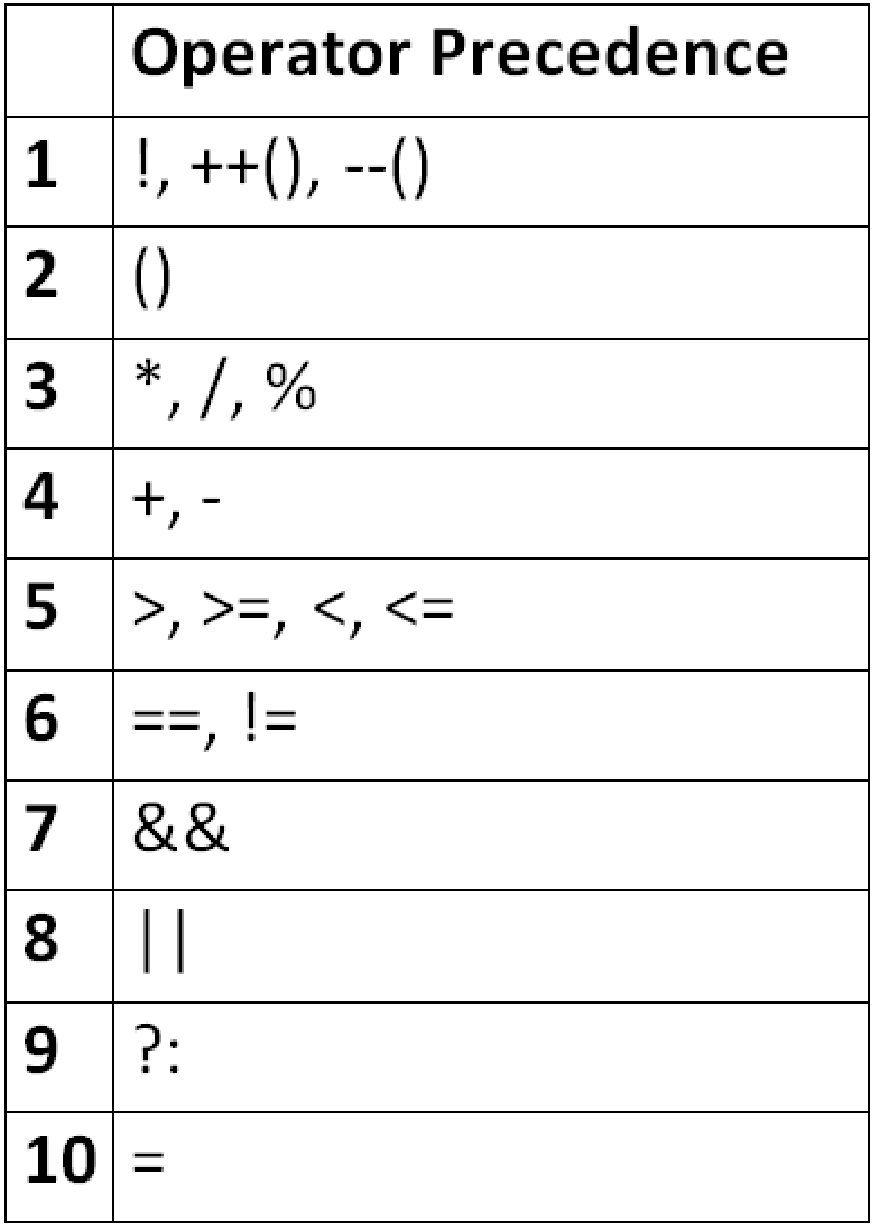


```c
#include <stdio.h>
int main()
{
int a;

printf("Size of int data type:%d\n",sizeof(int));
printf("Size of char data type:%d\n",sizeof(char));
printf("Size of float data type:%d\n",sizeof(float));
printf("Size of double data type:%d\n",sizeof(double));
printf("Size of int data type:%d\n",sizeof(a));

return 0;
}
```

<u>Output:</u>

```
Size of int data type:4
Size of char data type:1
Size of float data type:4
Size of double data type:8
Size of int data type:4
```

Precedence of C-operators



	Operator Precedence
1	!, ++(), --()
2	()
3	*, /, %
4	+, -
5	>, >=, <, <=
6	==, !=
7	&&
8	||
9	?:
10	=