Lesson Number : 12

Operators :

→ An operator is a symbol that tells the computer to perform certain mathematical or logical manipulations.

→ Operators are used in programs to manipulate data and variables.

→ They are usually form a part of the mathematical or logical expressions.

→ C operators can be classified into several different categories. They include ;

1. Arithmetic operators
2. Assignment operators
3. Increment and Decrement operators
4. Relational operators.
5. Logical operators
6. Conditional operators
7. Comma operator } Special operators.
8. Sizeof operator }
9. Bitwise operator.

1. Arithmetic operators:

→ Arithmetic operators are used for numeric calculations.

→ They are of two types

a. Unary arithmetic operators
b. Binary arithmetic operators

a. Unary arithmetic operators :

→ Unary operators ~~are~~ require only one operand

→ For example : +x -Y

Here '-' changes the sign of operand Y

b. Binary arithmetic operators :

→ Binary operators require two operands.

→ There are five binary arithmetic operators.

Operators	purpose
+	Addition
−	Subtraction or Unary minus
*	Multiplication
/	Division
%	Module division.

→ % (modulus operator) cannot be applied with floating point operands. There is no exponent operator in C. However there is a library function pow() to carry out exponentiation operation.

Integer Arithmetic :

→ When both operands are integers then the arithmetic operation with these operands is called integer arithmetic and the resulting value is always an integer.

→ Let us take two variables a and b. The value of a = 17 and b = 4, the results of the following operations are.

Expression	Result
a+b	21
a-b	13
a*b	68
a/b	4
a%b	1

Example : /* program to understand the integer arithmetic operations */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a=17, b=4;
    clrscr();
    printf("Sum = %d\n", a+b);
    printf("Difference = %d\n", a-b);
    printf("product = %d\n", a*b);
    printf("Quotient = %d\n", a/b);
    printf("Remainder = %d\n", a%b);
}
```

Output :

```
Sum = 21
Difference = 13
product = 68
Quotient = 4
Remainder = 1
```

Example : /* Show the integer use of integer arithmetic to Convert a given number of days into months and days */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int months, days;
    clrscr();
```

```
printf ("Enter Days\n");
scanf ("%d", &days);
months = days/30;
days = days % 30;
printf ("Months = %d Days = %d", months, days);
getch();
}
```

Output : Enter Days

265

Months = 8 Days = 25

Floating point arithmetic : (or Real arithmetic).

→ When both operands are of float type then the arithmetic operation with these operands is called floating point arithmetic

→ Let us take two variables a and b. The value of a = 12.4 and b = 3.1, the results of the following operations are as :

Expression	Result
a+b	15.5
a-b	9.3
a*b	38.44
a/b	4.0

→ The modulus operator (%) cannot be used with floating point numbers

Example : /* program to understand the floating point arithmetic operations */

```
#include <stdio.h>
#include <conio.h>
void main()
 {
    float a=12.4, b=3.8;
    clrscr();
  printf("Sum = %f\n", a+b);
  printf("Difference = %f\n", a-b);
  printf("product = %f\n", a*b);
  printf("Rem Division = %f\n", a/b);
  getch();
 }
```

output : Sum = 16.20
Difference = 8.60
product = 47.12
Division = 3.26

Mixed mode arithmetic :

→ when one operand is of integer type and other is of floating type, then the arithmetic operation with these operands is known as mixed mode arithmetic and the resulting value is float type.

→ let us take two variables a and b. The value of a=12 and b=2.5, the results of the following operations are as:

Expressions	Results
a+b	14.5
a-b	9.5
a*b	30.0
a/b	4.8

→ Sometimes, mixed mode arithmetic can help in getting exact results. For example, the result of expression 5/2 will be 2, since integer arithmetic is applied. If we want exact result we can make one of the operands float type. For example, 5.0/2 or 5/2.0, both gives result 2.5.

2. Assignment operators:

→ A value can be stored in a variable, with the use of assignment operator.

→ This assignment operator "=" is used in assignment expressions and assignment statements.

→ The operand on the left hand side should be a variable, while the operand on the right hand side can be any variable, constant or expression.

→ The value of right hand operand is assigned to the left hand operand.

→ Example:

```
X = 8     /* 8 is assigned to X */
Y = 5     /* 5 is assigned to Y */
S = X+Y-2 /* The value of expression X+Y-2 is
             assigned to S */
Y = X     /* X is assigned to Y */
X = Y     /* The value of Y is assigned to X */
```

→ In addition, C has a set of 'shorthand' assignment operators of the form;

```
v op = exp ;
```

where v is a variable, exp is an expression and op is a C binary arithmetic operator. The operator op= is known as the shorthand assignment operator or Compound assignment operator.

→ The assignment statement

v op = exp ; is equivalent to

```
v = v op (exp);
```

with v evaluated only once.

→ Example: x += y+1 is equivalent to x = x+y+1.

Here, the shorthand operator += means 'add y+1 to x' or 'increment x by y+1'

Statement with simple assignment operator	Statement with shorthand operator
a = a+1	a += 1
a = a-1	a -= 1
a = a*(n+1)	a *= n+1
a = a/(n+1)	a /= n+1
a = a%b	a %= b

(Shorthand assignment operators)

→ The use of shorthand assignment operators has three advantages.

1. What appears on the left-hand side need not be repeated and therefore it becomes easier to write.

Assignment : what is operator ? Name various types of operators used in C programming.

→ An operator is a symbol that tells the computer to perform certain mathematical and logical manipulations.

→ Name of various types of operators in C are ;

1. Arithmetic operators
2. Assignment operator
3. Increment and decrement operator.
4. Relational operators.
5. Logical operators
6. Conditional operator
7. Comma operator } special operators
8. Sizeof operator
9. Bitwise operator.

Assignment : what is assignment operator and short-hand assignment operator ?

→ A value can be stored in a variable, with the use of assignment operator.

→ The assignment operator "=" is used in assignment expression and assignment statements.

→ The operand on the left hand side should be a variable and the operand on the right hand side can be any variable, constant or expression.

→ The value of right hand side operand is assigned to the left hand side operand.

→ Example : `x = 9 /* 9 is assigned to x */`

`S = X + Y - Z /* The value of expression X+Y-Z is assigned to S */`

→ The C programming has a set of shorthand assignment operators of the form ;

V op = exp ;

where v is a variable, exp is an expression and op is a C binary arithmetic operator.

→ The operator op= is known as short hand assignment operator.

→ The V op= exp is equivalent to

V = V op (exp); with v evaluated only once.

→ Example: x = x + Y + 1 is equivalent to

x + = Y + 1

Assignment: what are the advantages of using short hand assignment operator ?

→ The use of shorthand assignment operator has three advantages.

1) what appears on the left hand side need not be repeated and therefore it becomes easier to write.

2) The statement is more concise and easier to read.

3) The statement is more efficient.

Assignment: write the following terms using short hand assignment operator.

a = a + 1 , a = a * (n+1) , a = a % b , x = x + t + 10

soln: a = a + 1 → a + = 1

a = a * (n+1) → a * = n+1

a = a % b → a % = b

x = x + t + 10 → x + = t + 10

2. The statement is more concise and easier to read.

3. The statement is more efficient.

[References : 1. E. Balagurusami
2. Ashok N. Kamthan

Lesson Number : 13

3. Increment and Decrement operators :

→ C has two useful operators increment (++) and decrement (--).

→ These are unary operators because they operate on a single operand.

→ The increment operator (++) increments the value of the variable by 1 and decrement operator (--) decrements the value of the variable by 1.

→ ++x is equivalent to x = x+1
--x is equivalent to x = x-1

These operators should be used only with variables; they can't be used with constants on expressions. For example, the expression ++5 or ++(x+y+z) are invalid.

→ These operators are of two types.

1. prefix increment / decrement - operator is written before the operand (e.g. ++x or --x).

2. postfix increment / decrement - operator is written after the operand (e.g. x++ or x--).

1. prefix increment / decrement :

→ Here first the value of variable is incremented / decremented then the new value is used in the operations.

→ Let us take a variable x whose value is 3

→ The statement Y = ++x ; means first increment the value of x by 1, then assign the value of x to Y.

This single statement is equivalent to these two statement

```
X = X+1 ;
Y = X ;
```

Here now value of X is 4 and value of Y is 4.

→ The statement Y = --X ; means first decrement the value of X by 1, then assign the value of X to Y.

→ This statement is equivalent to these two statements.

```
X = X-1 ;
Y = X ;
```

Here now value of X is 3 and value of Y is 3.

Example : /* program to understand the use of prefix increment / decrement operators */

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int x=8;
  clrscr();
  printf ("x= %d\t", x);
  printf ("x = %d\t", ++x); /* prefix increment */
  printf ("x = %d\t", x);
  printf ("x = %d\t", --x); /* prefix decrement */
  printf ("x = %d\t", x);
getch();
}
```

Output : x = 8 x = 9 x = 9 x = 8 x = 8

2. postfix increment / decrement :

→ Here first the value of a variable is used in the operation and then increment / decrement is performed.

→ Let us take a variable x whose value is 3.

→ The statement Y = x++; means, first the value of x is assigned to Y and then x is incremented. This statement is equivalent to these two statements.

```
Y = X ;
X = X+1 ;
```

Hence now value of Y is 3 & X is 4.

→ The statement Y = X-- ; means, first the value of x is assigned to Y and then x is decremented. This statement is equivalent to these two statements.

```
Y = X ;
X = X-1 ;
```

Hence now value of Y is 4 and x is 3.

Example : /* program to understand the use of postfix increment / decrement operators */

```
#include <stdio.h>
#include <conio.h>
Void main()
 {
   int x = 8 ;
   clrscr();
  printf ("x = %d\t", x);
```

```
printf ("x = %d\t", x++); /* postfix increment */
printf ("x = %d\t", x);
printf ("x = %d\t", x--); /* postfix decrement */
printf ("x = %d\t", x);
getch();
}
```

output : x = 8 x = 8 x = 9 x = 9 x = 8

4. Relational operators :

→ Relational operators are used to compare values of two expressions depending on their relations.

→ An expression that contains relational operators is called relational expression.

→ If the relation is true then the value of relational expression is 1 and if the relation is false then the value of expression is 0.

→ The relational operators are ;

Operators	Meaning
<	less than
<=	less than or equal to
==	equal to
!=	Not equal to
>	Greater than
>=	Greater than or equal to.

→ Let us take two variables a=9 and b=5 and form simple relational expression with them.

Expressions	Relation	value of Expression
a < b	False	0
a <= b	False	0
a == b	False	0
a != b	True	1
a > b	True	1
a >= b	True	1
a == 0	False	0
b != 0	True	1
a > 8	True	1
2 > 4	False	0

→ The relational operators are generally used in if...else construct and loops.

Example : /* program to understand the use of relational operator */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b;
    clrscr();
    printf("Enter the values for a and b: \n");
    scanf("%d %d", &a, &b);
    if (a<b)
        printf("%d is less than %d \n", a, b);
    if (a<=b)
        printf("%d is less than or equal to %d \n", a, b);
```

```
if (a==b)
  printf("%d is equal to %d\n", a,b);
if (a!=b)
    printf("%d is not equal to %d\n", a,b);
if (a>b)
    printf("%d is greater than %d\n", a,b);
if (a>=b)
      printf("%d is greater than or equal to %d\n", a,b);
getch();
}
```

output : Enter the values for a and b :

12 7

12 is not equal to 7

12 is greater than 7

12 is greater than or equal to 7.

→ It is important to note that the assignment operator (=) and equality operator (==) are entirely different.

→ Assignment operator is used for assigning values while equality operator is used to compare expressions.

5. Logical operators :

→ Logical operators are also called as Boolean operators.

→ An expression that combines two or more expressions is termed as a logical expression. For combining these expressions we use logical operators.

→ These operators return 0 for false and 1 for true.

→ The operands may be constants, variables or expressions.

→ C has 3 logical operators.

operators	Meaning
&&	AND
||	OR
!	NOT

→ Here logical NOT is a unary operator while the other two are binary operators.

→ In C, any non-zero value is regarded as true and zero value is regarded as false.

i) AND (&&) operator :

→ This operator gives the net result true if both the conditions are true, otherwise the result is false.

Condition1	condition2	Result
False	False	False
False	True	False
True	False	False
True	True	True.

(Boolean Table)

→ Let us take 3 variables a = 10, b = 5, c = 0.

Suppose we have a logical expression :-

(a == 10) && (b < a)

Here both the conditions a == 10 and b < a are true, and hence this whole expression is true. Since the logical operators return 1

for true hence the value of this expression is 1.

Expressions	(True && False)	Result	value
(a==10) && (b>a)	true && false	false	0
(b>=a) && (b==3)	false && false	false	0
a && b	true && true	true	1
a && c	true && false	false	0

→ In the last two statements, we have taken only variables. Since nonzero values are regarded as true and zero value is regarded as false, so variable a and b are considered true and variable c is considered false.

ii) OR (||) operator :

→ This operator gives the net result false. if both the conditions have the value false, otherwise the result is true.

Condition 1	Condition 2	Result
False	False	False
False	True	True
True	False	True
True	True	True

(Boolean Table)

→ Let us take 3 variables a=10, b=5, c=0.

Consider the logical expression (a>=b) || (b>15).

This gives result true because one condition is true.

Expressions	(True || False)	Result	value
a || b	true || true	true	1
a || c	true || False	true	1
(a<9) || (b>10)	false || false	false	0
(b!=7) || c	true || False	True	1

iii) NOT (!) operator :

→ This is a unary operator and it negates the value of the condition.

→ If the value of the condition is false then it gives the result true. If the value of condition is true then it gives the result false.

Condition	Result
False	True
True	False

(Boolean table)

→ Let us take 3 variables a=10, b=5, c=0. suppose logical expression is: !(a==10). The value of the condition (a==10) is true. NOT operator negates the value of the condition. Hence the result is false.

Expression	(!True/! False)	Result	value
!a	! true	false	0
!c	! false	true	1
!(b>c)	! true	false	0
!(a && c)	! false	True	1

Example : /* program to display 1 if inputed number is between 1-100, otherwise 0. use the logical AND (&&) Operator */

```
# include <stdio.h>
# include <conio.h>
void main()
{
   int x, z;
   clrscr();
   printf ("Enter number :");
   scanf ("%d", &x);
   z = (x>=1 && x<=100 ? 1 : 0);
   printf ("z = %d", z);
 getch();
}
```

output : Enter number : 5
z = 1

Example : /* program to display 1 if inputed no. is either 1 or 100 otherwise 0. use logical OR (||) operator */

```
# include <stdio.h>
# include <conio.h>
void main()
{  int x, z;
   clrscr();
   printf ("Enter number\n :");
   scanf ("%d", &x);
   z = ( x==1 || x==100 ? 1 : 0);
```

```
printf ("z = %d", z);
getch();
}
```

output : Enter number : 1
z = 1

Example : /* program to display 1 if inputed number is except 100 otherwise 0. use logical NOT (!) operator */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int x, z;
    clrscr();
    printf ("Enter number\n :");
    scanf ("%d", &x);
    z = (x != 100 ? 1 : 0);
    printf ("z=%d", z);
    getch();
}
```

output : Enter number : 100
z = 0

Assignment : what will be the value of x and Y in the following printf statement ?

```
void main()
  {
    int x=10;
     Y= ++x;
     printf(" x = %d Y = %d\n", x, Y);
  }
```

output : x = 11 . Y = 11

Assignment : what is the o/p of the following program ?

```
void main()
{
  int x, Y, z;
   Y = 2;
   x = 2;
   x = 2 * ( Y++);
   z = 2 * (++Y);
  printf ( "\n  x = %d  Y = %d  z = %d", x, Y, z);
}
```

output : x = 4 Y = 4 z = 8

Assignment : what will be the value of K after execution of the following program ?

```
void main()
   {  int K = 8;
    printf (" K = %d \n", K++ - K++);
    }
```

output : K = 0

Assignment: what will be the value of b. after execution ?

```
void main()
{
    int b, K=8;
    b = (K++ - K++ - K++ , K++);
    printf ("b = %d\n", b);
}
```

output : b =

Assignment: what will be the value of x, Y and z ?

```
void main()
{
    int x, Y, z;
    x = 8++;
    Y = ++X++;
    z = (x+Y)--;
    printf (" x = %d  Y = %d  z = %d \n", x, Y, z);
}
```

output :

Assignment: what is the value of c after execution of the program ?

```
void main()
{
    int a, b, C;
    a = 9;
    b = 10;
    C = ( b<a || b>a)
    check();
    printf ("\n C = %d", C);
}
```

output : C = 1

Lesson Number : 14 [References : 1. S. Srivasthava
2. Ashok N. Kamthane]

6. Conditional operator :

→ The conditional operator contains a condition followed by two statements or values.

→ If condition is true, the first statement is executed otherwise the second statement is executed.

→ The conditional operator (?) and (:) are sometimes called ternary operators because they take three arguments.

→ The syntax of conditional operator is ;

```
Condition ? (expression 1) : (expression 2) ;
```

→ Two expressions are separated by a colon. If the condition is true expression 1 gets evaluated otherwise expression 2. The condition is always written before ? mark.

→ Example : a > b ? a : b

Here first the expression a > b is evaluated, if the value is true then value of variable a becomes the value of conditional expression otherwise the value of b becomes the value of conditional expression.

Suppose a = 5 and b = 8 and we use the above conditional expression in a statement as :

```
max = a > b ? a : b ;
```

First the expression a > b is evaluated, since it is false so the value of b becomes the value of conditional expression.

Again if `a < b ? printf("a is smaller") : printf("b is smaller");`

since a < b is true, so the first printf function is executed

Example : /* program to print the larger of two numbers using conditional operator */

```
#include <stdio.h>
#include <conio.h>
Void main()
{
  int a, b, max ;
  clrscr() ;
  printf ( " Enter the values for a and b :") ;
  scanf ( " %d %d", &a, &b) ;
  max = (a > b) ? a : b ;
  printf ( " Larger of %d and %d is %d \n", a, b, max) ;
  getch () ;
}
```

output : Enter the values for a and b : 12 7
Larger of 12 and 7 is 12

7. Comma Operator :

→ The comma operator is used to separate two or more expressions.

→ Comma operator has the lowest priority among all the operators.

→ It is not essential to enclose the expressions with comma operators within the parenthesis.

→ The separated expressions are evaluated from left to right and the type and value of the rightmost expression is the type and value of the compound expression.

→ For example, consider the expression

a=8, b=7, c=9, a+b+c

Here we have combined 4 expressions. Initially 8 is assigned to the variable a, then 7 is assigned to the variable b, 9 is assigned to variable c and after this a+b+c is evaluated which becomes the value of the whole expression. So the value of the above expression is 24.

→ Now consider the statement

sum = (a=8, b=7, c=9, a+b+c);

Here, the value of the whole expression on right side will be assigned to variable sum.

→ Without the use of comma operator, the above task would have been done in 4 statements.

```
a=8;
b=7;
c=9;
sum = a+b+c;
```

Example: /* program to understand the use of comma operator */

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int a, b, c, sum;
   clrscr();
   sum = (a=8, b=7, c=9, a+b+c);
   printf("sum = %d\n", sum);
   getch();
}
```

output : Sum = 24

Example : /* program to interchange the value of two variables using comma operator */

```
# include <stdio.h>
# include <conio.h>
void main()
{
   int a=8, b=7, temp ;
   printf (" a = %d , b = %d \n", a,b);
   temp = a , a=b , b = temp ;
   printf (" a = %d , b = %d \n" , a,b);
   getch();
}
```

output : a=8 , b=7
a=7 , b=8

8. Sizeof operator :

→ Sizeof is an unary operator.

→ The sizeof() operator gives the bytes occupied by a variable.

→ The number of bytes occupied varies from variable to variable depending upon its data types

→ For example sizeof(int) gives the bytes occupied by the int data type i.e. 2.

Example : /* program to understand the sizeof operator */

```
# include <stdio.h>
# include <conio.h>
```

```
void main()
{
   int var ;
   clrscr();
  printf (" size of int = %d\n", sizeof (int));
  printf (" size of float = %d\n", sizeof (float));
  printf (" size of var = %d\n", sizeof (var));
  printf (" size of an integer constant = %d\n", sizeof (45));
 getch();
}
```

output : Size of int = 2
Size of float = 4
Size of var = 2
Size of an integer constant = 2

→ Generally sizeof operator is used to make portable program. i.e. programs that can be run on different machines. For example, if we write our program assuming int to be of 2 bytes, then it won't run correctly on a machine on which int is of 4 bytes. So to make general code that can run on all machines we can use sizeof operator.

9. Bitwise operators:

→ C has the ability to support the manipulation of data at the bit level.

→ Bitwise operators are used for operations on individual bits.

→ Bitwise operators operate on integers only, such as int, char, short, long int etc.

Before the execution of the program :- The number entered through the keyboard is 8 and its corresponding binary number is 1000.

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

After the execution of the program :-

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Shifting two bits right means, the input number is to be divided by 2^s where s is the no. of shifts i.e. in short

$$Y = n/2^s$$

where n = Number
S = Number of position to be shifted.

As per the program $Y = 8/2^2 = 8/4 = 2$.

Example: /* program to shift inputed data by two bits to left */

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int x, Y;
   clrscr();
   printf("Read the integer from the keyboard (x):-");
   scanf("%d", &x);

   x << 2;
   Y = x;
  printf("The left shifted data is = %d", Y);

  getch();
}
```

Before execution of the program :-

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

After execution of the program :-

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Shifting two bits left means, the input number is to be multiplied by 2^S where S is the no. of shifts i.e. in short

$$Y = n * 2^S$$

where n = Number
S = No. of position to be shifted.

So As per the program $Y = 2 * 2^2 = 2 * 4 = 8$

Output : Read the integer from the keyboard (x) :- 2
The left shifted data is = 8.

Example : /* program to use Bitwise AND operator between the two integers and display the results. */

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int a, b, c;
  clrscr();
  printf(" Read a and b from the keyboard (a & b) :-");
  scanf ("%d %d", &a, &b);
  c = a & b ;
  printf (" The answer after ANDing is (c) = %d", c);
```

```
getch();
}
```

output : Read a and b from the keyboard (a & b) :- 8 4
The answer after ANDing is (c) = 0 ..

a = 8

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

b = 4

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	14	13	12	11	10	09	08	01	06	05	04	03	02	01	00

Now a & b i.e. c = 0.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Because in AND operation;

Inputs		output
X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Example : /* program to operate OR operation on two integers and display the result */

```
# include <stdio.h>
# include <conio.h>
void main()
{
```

```
int a,b,c;
clrscr();
printf("Read a and b:-");
scanf("%d %d", &a, &b);
c = a | b;
printf("The ORing operation bet^n a & b is : %d", c);
getch();
}
```

output : Read a and b :- 8 4

The ORing operation bet^n a & b is : 12

a = 8

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

b = 4

0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

c = 12

0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Because in <u>OR</u> operation;

Inputs		Output
X	Y	Z
0	0	0
1	0	1
0	1	1
1	1	1

Example : /* program with Exclusive OR ('XOR) operation betn the two integers and display the result */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a, b, c;
    clrscr();
    printf(" Read a & b :");
    scanf(" %d %d", &a, &b);
    c = a ^ b;
    printf(" The data after XOR operation is : %d", c);
    getch();
}
```

Output : Read a & b : 8 2

The data after XOR operation is : 10

a = 8

0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

b = 2

0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

c = 10

0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Because in XOR operation;

Inputs		outputs
X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

Example : /* pgm to understand the one's complement (~) */

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int a,b;
    clrscr();
    printf("Enter the value of a:");
    scanf("%d",&a);
    b=~a;
    printf("The data after one's complement is : %d",b);
    getch();
}
```

output : Enter the value of a : 25
The data after one's complement is : 65510

a = 25

0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

b = 65510

1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00

Because in one's complement (~) :

Input	output
X	Y
0	1
1	0

Assignment : what is the o/p of the following program ?

```
#include <stdio.h>
void main()
{
    int a, b, min;
    printf ("Enter the value of a, b : \n");
    scanf ("%d %d", &a, &b);
    min = a < b ? a : b;
    printf ("Smaller = %d\n", min);
}
```

output : Enter the value of a, b : 12 15
Smaller = 12

Assignment : what is the output of the following program ?

```
void main()
{
    int a=8, b=7, temp;
    printf ("a= %d  b= %d\n", a, b);
    temp = a;
    a = b;
    b = temp;
    printf ("a= %d  b= %d\n", a, b);
}
```

output : a = 8 b = 7
a = 7 b = 8

Assignment : write the syntax for ternary operator .

→ Conditional operator is also known as ternary operator.

→ Syntax : Condition ? (expression1) : (expression2) ;

Assignment : what is the o/p of the following program ?

```
void main()
{
   int a ;
  float average;
printf ( " size of int is %d", sizeof(a));
printf ("size of average is %d", sizeof(average));
}
```

output : size of int is 2 size of average is 4

Assignment : what is the value of Y if x = 8 and x << 2

```
Y = X ;
```

soln: x = 8

x << 2 , i.e. x is left shifted by 2.

we know shifting 2 bits left means the input number is to be multiplied by 2^s where s is the no. of shifts i.e. in short

$Y = n * 2^S$

$= 8 * 2^2$

$= 8 * 4$

$= 32$