Unit–3

❖ Operators and Expression

- Introduction of different types of operators and their symbolic representation
- Properties of operator
- Priority of operator and their clubbing
- Comma and conditional operator
- Arithmetic operators
- Relational operators
- Assignment operators and expressions
- Logical operators
- Bitwise operators
- formatted input and output in 'C'

❖ What is operator? List various operators available in C.

- **Def:-**An operator is a symbol that tells the computer to perform certain operation on operands.
- Operators are always associated with operands. Depending upon number of operands, there are basically three types of operators:
  - For example: A+B
  - Here, A and B are variable (operands) and + is operator.
  - **Unary operator**: The operator which has one operand is known as unary operator.
    - Example : -a
  - **Binary operator:** The operator which has two operands is known as binary operator.
    - Example :a+b
  - **Ternary operator**: The operator which has three operands is known as ternary operator.
    - Example: (a>b)?a:b
- **Various operator available in C are as follows:**
  - Arithmetic operator
  - Relational operator
  - Logical operator
  - Assignment operator
  - Increment and decrement operator
  - Conditional (ternary)operator
  - Bitwise operator
  - Special operators

❖ Arithmetic operator or Explain Arithmetic operators with example?

- Arithmetic operators are perform Arithmetic calculation

Operator Name	Meaning
+	Addition
-	Subtract
*	Multiply
/	Divide
%	Modulo Division

- **Example:**
  - int a, b, c for doing arithmetic operation on that we require that addition operator and the result of them will store in variable c the syntax is c= a+b
  - If both operands are integer then we found the integer result. And is call integer arithmetic. i.e. int a=4, b=7 then a+b = 11
  - When Operations like Addition (+), subtraction (-), multiplication (*), division (/) in when the both operands are integers the result is always integers. If one or both operands are real the result is real.
- The arithmetic operator support by c language as bellow.
  - **Unary plus(+):**
    - It is used to indicate positive value.
    - Example :+a
  - **Binary plus(+):**
    - It adds two operands.
    - Example :a+b
  - **Multiplication(*):**
    - It multiplies two operands
    - Example :a*b
  - **Division(/) :**
    - It divides one operand by another operand.
    - Example: a/b
  - **Modulo operator (%):**
    - It divides one operand by another operand and produce reminder of the division.
    - Example :a%b
    - Modulo operator does not work with floating point data. It only works with the integer data.
    - Now assume that a=5 and b=2 then
      1) +a=5
      2) –b=-2
      3) a+b =7
      4) a-b =3
      5) a*b =10
      6) a/b =2
      7) a%b =1
- **Example:-**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a=20,b=10;
    clrscr();
    printf("a+b=%d",(a+b));
    getch();
}
```

❖ **Relational operators or Explain Relational operators with example?**

- Relational operators are used for comparison.
- If we want to take decision based on certain condition at that time we have to use relational operator.
- An expression, which contains relational operator, is known as relational expression.
- The relational expression returns either TRUE (1) or FALSE (0) based on result of condition.
- The relational operator support by c language is given bellow.
- **Equal to(= =):**
  - It compares two operands and return TRUE (1) if both operands are equal.
- **Not equal to (! =)**
  - It compares two operands and return TRUE (1) if both operands are not equal.
- **Less than(<):**
  - It compares two operands and return TRUE (1) if operand1 is less than operand2.
- **Less than equal to(<=):**
  - It compares two operands and return TRUE (1) if operand1 is less than or equal to operand2.
- **Greater than(>):**
  - It compares two operands and return TRUE (1) if operand1 is greater than operand2.
- **Greater than equal to(>=):**
  - It compares two operands and return TRUE (1) if operand1 is greater than or equal to operand2.
- **Example:**
  - Suppose a=5, b=7

    a==b (5==7) false (**Note**: it is not same as a=b.)
    a!=b (5!=7) true
    a<b (5<7) true
    a<=b (5<=7) true
    a>b (5>7) false
    a>=b (5>=7) false
- **Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a=20,b=10;
    clrscr();
    printf("a>b=%d",(a>b));
    getch();
}
```

❖ **Bitwise operators OR Explain Logical operators with example?**

- Logical operators are used when we want to combine more than one condition.
- An expression, which combines more than one condition, is known as logical expression or compound relational expression.
- The logical expression returns either true (1) or false (0) value based on conditions and truth table.
- The logical operator supported by c language is given bellow:

- **Logical AND (&&):**
- **Logical OR(||):**
- **Logical NOT (!):**

Operator	Description
&&	Called Logical AND operator. If both the operands are non zero then condition becomes true.
||	Called Logical OR Operator. If any of the two operands are non zero then condition becomes true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

  - **Examples :**
    - Suppose a=5, b=3, c=6;

```
(a>b) && (a>c)      -false
(a>b) || (a>c)      -true
!(a>b)              -false
```

❖ **Assignment operators and expressions OR Explain Assignment operators with example?**

- The assignment operator assigns the result of an expression to a variable.
- Example: a=5 will assign the value 5 to variable a.
- In addition to this assignment operator C has a set of short hand assignment operator.
- The short hand operator takes the following form:
- V op = exp.
- Where v is a variable, exp is an expression or a value and op is a binary arithmetic operator.
- Here op = is known as short hand assignment operator.
- Thus the statement v op = exp is same as v = v op exp.

Operator	Description	Example
=	assigns values from right side operands to left side operand	a=b
+=	adds right operand to the left operand and assign the result to left	a+=b is same as a=a+b
-=	subtracts right operand from the left operand and assign the result to left operand	a-=b is same as a=a-b
*=	multiply left operand with the right operand and assign the result to left operand	a*=b is same as a=a*b
/=	divides left operand with the right operand and assign the result to left operand	a/=b is same as a=a/b
%=	calculate modulus using two operands and assign the result to left operand	a%=b is same as a=a%b

- **Advantage of short hand assignment operator are**
  - Program writing becomes faster particularly when a variable name is long in size.
  - The statement is more efficient.
  - The statement is more concise and easier to read.

❖ **Explain Increment & Decrement operators with example?**

- Increment and decrement operators are unary operator because they have only one operand.
- **Increment operator:**
  - Increment operator adds one to its operand.
    - Example: m=5 then m++ will increment the value of m by one. Thus the value of m becomes 6 now.
  - There are two type of increment operator available.
  - **Prefix increment operator:**
    - When prefix increment operator is used in expression then it will first increment the value of its operand and then assign it to variable.
    - **Example**:
      - m = 5
      - y = ++m
      - Will gives m=6 and y=6
  - **Postfix increment operator:**
    - When postfix increment operator is used in expression then it will first assign the value of its operand to the variable and then Decrement the value of its operand.
    - **Example**:
      - m = 5
      - y = m++
      - Will gives m=6 and y=5
    - **Example:-**

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a=10;
        clrscr();
        a++;
        printf("%d",a);
        getch();
}
```

- **Decrement operator:**
  - Decrement operators subtract one from its operand.
  - Example: m=5 then m-- will Decrement the value of m by one. Thus the value of m becomes 4 now.
  - There are two type of Decrement operator available.
  - **Prefix Decrement operator:**
    - When prefix Decrement operator is used in expression then it will first Decrement the value of its operand and then assign it to variable.
    - **Example**:
      - m = 5
      - y = --m
      - Will gives m=4 and y=4

- **Postfix Decrement operator:**
  - When postfix Decrement operator is used in expression then it will first assign the value of its operand to the variable and then Decrement the value of its operand.
  - **Example:**
    - m = 5
    - y = m--
    - Will gives m=4 and y=5
  - **Example:**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a=10;
    clrscr();
    a--;
    printf("%d",a);
    getch();
}
```

❖ **Explain conditional operators with example?**

- The conditional operator in C is called by two more names
  - **Ternary Operator OR ? : Operator**
- It is actually the if condition that we use in C language, but using conditional operator, we turn if condition statement into more simple line of code conveying the same thing.
- The syntax of a conditional operator is :
  - **expression 1 ? expression 2 : expression 3**
- **Explanation-**
  - The question mark **"?"** in the syntax represents the if part.
  - The first expression (expression 1) is use to check true or false condition only.
  - If that condition (expression 1) is true then the expression on the left side of **" : "** i.e expression 2 is executed.
  - If that condition (expression 1) is false then the expression on the right side of **" : "** i.e expression 3 is executed.
- **Example**: (a>b)? a: b; (Same as:- (a>b) ? printf("a is max") : printf("b is max");)
  - Here if the value of a greater than b then it prints *a is max* otherwise it prints *b is max*.
- **Example:-**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a, b, c, max;
    clrscr();
    printf("Enter any three number: ");
    scanf("%d%d%d",&a,&b,&c);
    max=a>b ? (a>c?a:c) : (b>c?b:c);
    printf("Largest Number is: %d",max);
```

```
    getch();
}
```

❖ **Explain Bitwise operators with example?**

- The bitwise operators are used to manipulate data at bit level.
- These operators are used for testing bits or shifting them left or right.
- They cannot be applied to float and double.
- The bitwise operators supported by C language is given below:
  - **Bitwise AND(&):**
  - **Bitwise OR(|):**
  - **Bitwise Exclusive OR(^):**
  - **Shift left(<<):**
  - **Shift right(>>):**
  - **One 's complement(~):**

Operator	Name	Example	Result	Description
a & *b*	and	3 & 5	1	1 if both bits are 1.
a | *b*	or	3 | 5	7	1 if either bit is 1.
a ^ *b*	xor	3 ^ 5	6	1 if both bits are different.
~*a*	not	~3	-4	Inverts the bits.
n << *p*	left shift	3 <<< 2	12	Shifts the bits of *n* left *p* positions. Zero bits are shifted into the low-order positions.
n >> *p*	right shift	5 >> 2	1	Shifts the bits of *n* right *p* positions. If *n* is a 2's complement signed number, the sign bit is shifted into the high-order positions.
n >>> *p*	right shift	-4 >>> 28	15	Shifts the bits of *n* right *p* positions. Zeros are shifted into the high-order positions.

- **Example:**
  - Write c program to multiply and divide the given number by2. Using bitwise operator.

OR

  - Write a c program to multiply and divide the given number by2. Using left shift (<<) and right shift (>>) operator

OR

  - Write a c program to multiply and divide the given number by2. Without using multiply (*) operator**.**

```
#include<stdio.h>
#include<conio.h>
void main ()
{
    int a, mul,div;
    clrscr ();
     printf ("\nEnter value of A:");
     scanf ("%d",&a);
     mul=a<<1;
     div=a>>1;
    printf ("A:%d \n Multiply by 2=%d \n Divide by 2=%d",a,mul,div);
```

```
    getch ();
}
```

❖ **Explain special operators with example?**

- The special operators supported by C language are given below:
- **Comma operator(,)**
  - The comma operator can be used to link related expression together.
  - The expressions are evaluated left to right.
  - The value of the right most expression is the value of the combined expression.
  - **Example**:
    - Z= (x=10, y=5, x+y);
  - First the value of 10 is assigned to x, then value of 5 is assigned to y. finally the value of x+y (10+5=15) is assigned to z.

- **Size of operator:**
  - The size of operator is used to find how many bytes the operand occupies (used).
  - The operand may be a variable, a constant or data type.
  - The general syntax of size of operator is:
    - **Syntax: Size of (operand)**
  - **Examples:**
    - Size of (a); // here it return 2 bytes if a is of type int.
    - Size of (2.5) // here it return 4 bytes because 2.5 is float.
    - Size of (double) // here it return 8 bytes.
  - The size of operator is a compile time operator.
  - The common use of size of operator are as given below:
    - To determine the length of array and structure when their size of are not known to the programmer.
    - To allocate memory space dynamically to variables during execution of the program.
  - **Example:-**

```
#include<stdio.h>
#include<conio.h>
void main()
{
    int a;
    clrscr();
    printf("Size of Integer: %d bytes\n",sizeof(a));
    getch();
}
```

- **Pointer operator(&,*):**
  - Pointer operator are used to declared the pointer variable and access the value stored at particular memory location.

- **Member selection operator (., ->):**
  - The member selection operators are used to access the member of structure.

❖ **Formatted input and output in 'C'**

➢ **Formatted Output**

- The function printf() is used for formatted output to standard output based on a format specification.
- The format specification string, along with the data to be output, are the parameters to the printf() function.
- **Syntax:** printf (format, data1, data2,……..);
- In this syntax format is the format specification string.
- This string contains, for each variable to be output, a specification beginning with the symbol % followed by a character called the conversion character.
- **Example:**
  - printf ("%c", data1);
- The character specified after % is called a conversion character because it allows one data type to be converted to another type and printed.

➢ **Formatted Input**

- The function scanf() is used for formatted input from standard input and provides many of the conversion facilities of the function printf().
- **Syntax:** scanf (format, num1, num2,……);
- The function scnaf() reads and converts characters from the standards input depending on the format specification string and stores the input in memory locations represented by the other arguments (num1, num2,….).
- For Example:scanf(" %c %d",&Name, &Roll No);
- **Note**: the data names are listed as &Name and &Roll No instead of Name and Roll No

➢ Commonly used scanf And Printf . format codes are:

Code	**Meaning**
%C	Read a single character
%d	Read decimal integer
%f	Read a floating point value
%e	Read a floating point value
%O	Read a octal integer
%s	Read a string
%u	Read an unsigned decimal integer
%x	Read a hexadecimal integer
%[“]	Read a string of word(s)

❖ **Properties of operator** OR Operators Precedence in C

- Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated.
- Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.
- For example, x = 7 + 3 * 2;
- here, x is assigned 13, not 20 because operator * has a higher precedence than +,
- so it first gets multiplied with 3*2 and then adds into 7.
- Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right