Note 3:
Arithmetic, Relational, Equality, Logical, Unary, Assignment, Comma, and Sizeof Operators

1 Operators in C

An operator is a symbol that specifies the mathematical, logical, or relational operation to be performed. C language supports different types of operators, which can be used with variables and constants to form expressions.

1. Arithmetic Operators
2. Relational Operators
3. Equality Operators
4. Logical Operators
5. Unary Operators
6. Assignment Operators
7. Comma Operator
8. Sizeof Operator

1.1 Arithmetic Operators

1. **Multiply**:
   Operator - *
   Syntax - a * b
2. **Divide**:
   Operator - '/'
   Syntax - a / b

3. **Addition**:
   Operator - '+'
   Syntax - a + b

4. **Subtraction**:
   Operator - '-'
   Syntax - a - b

There is a fifth Arithmetic operator in C besides the 4 basic mathematical ones. Its called the Modulo

5. **Modulo**:
   Operator - "%"
   Syntax - a % b

The modulo returns the remainder of an integer division, and only applies to integers and not floats or doubles.
For example:

```
16 / 3 = 5
BUT
16 % 3 = 1, since 1 is the remainder when 16 is divided by 3.
```

1.2 Relational Operators

A relational operator, also known as an comparison operator is an operator that compares two values.
There are 4 relational operators and they are listed below:

1. '<' - Less than
2. '>' - Greater than
3. '<=' - Less than or equal to.
4. '>=' - Greater than or equal to.

When the relational operator is used between two values, and if the expression is true, it will return a 1, if the expression is false, it will return a 0.
For ex:

```
int x = 5, y = 2;
printf("%d > %d = %d", x, y, x>y);
```

this expression will printout a 1 since 5 is greater than 1.

1.3 Equality Operators

There are two equality operators:

1. == - Returns 1 if both operands of both sides of the operator are equal, otherwise returns 0. Can be used to compare more than just numbers. For example, comparing values of two different arrays etc.
2. '!=' - Returns 1 if the operands do not have the same value, and returns 0 if they do have the same value.

1.4 Logical Operators

C language supports 3 logical operators -

1. **Logical AND ()**:
   The logical AND operator is a binary operator, which simultaneously evaluates two values or relational expressions. If both the operands are true, then the whole expression evaluates to true. If both or one of the operands is false, then the whole operation evaluates to false.

2. **Logical OR (||)**:
   Logical OR returns a false value if both the operands are false. Otherwise it returns a true value if one 'or' both the values are true.

3. **Logical NOT(!)**:
   The logical NOTE operator takes a single expression and negates the value of the expression. In other words it just reverses the value of the expression. If the expression produces a zero, it reverses it to a one and vice versa.

1.5 Unary Operators

Unary operators act on single operands. There are three unary operators:

1. **Unary minus**:
   Unary minus(-) operator is different from the binary arithmetic subtraction operator. When an operand is preceded by a minus sign, the unary operator negates its value.
   For example if a number is positive, the operator turns it into a negative and vice versa

   ```
   int a, b = 10;
   a = -(b);
   ```

   when we print out the value of a we get the value as -10.

2. **Increment and Decrement Operators**:
   The increment operator increments the value of the operand by 1. Similary the decrement operator decrements the value of the operand by 1.

   The increment operators have two variants, postfix and prefix.

   In a prefix expression (++x or –x), the operator is applied before an operand is fetched for computation, and thus, the new value is used for the computation.

   For example:

   ```
   int a = 10
   printf("%d", ++a);
   ```

   Here the value of a gets printed out as 11.

   In a postfix expression (x++ or x–), the computation is carried out first and then the new value is fetched and updated.

   For example:

   ```
   int a = 10;
   printf("%d", a);
   printf("%d", a++);
   printf("%d", a)
   ```

   Here the output is as follows:

   ```
   10
   10
   11
   ```

1.6 Assignment Operators

In C, the assignment operator (=) is responsible for assigning values to the variables.

This is just the basic way to assign values to variables.

```
int x;
x = 10;

int x = 2, y = 3, sum = 0;
sum = x + y;
```

The assignment operator has left to right associativity,

```
a = b = c = 10
```

first the value of 10 is assigned to c, value of c is assigned to b, finally value of b is assigned to a.

There are many other assignment operators what we will study in the coming chapters while actively using them in loops etc.

1.7 Comma Operator

The comma operator in C takes two operands. It works by evaluating the first and discarding its value, and then evaluates the second and returns the value as the result of the expression. They are evaluated in left-to-right sequence.

1.8 Sizeof Operator

The sizeof operator is a unary operator used to calculate the size of data types.

```
int a = 10;
int result = sizeof(a);
```

The value of result will be 2, meaning the size of an integer type data is 2 bytes. (It may be 4 depending on the user system.)