Operators

1. The **& (bitwise AND)** in C or C++ takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
2. The **| (bitwise OR)** in C or C++ takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
3. The **^ (bitwise XOR)** in C or C++ takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.
4. The **<< (left shift)** in C or C++ takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
5. The **>> (right shift)** in C or C++ takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
6. The **~ (bitwise NOT)** in C or C++ takes one number and inverts all bits of it

```
C Program to demonstrate use of bitwise operators
#include <stdio.h>
int main()
{
    // a = 5(00000101), b = 9(00001001)
    unsigned char a = 5, b = 9;

    // The result is 00000001
    printf("a = %d, b = %d\n", a, b);
    printf("a&b = %d\n", a & b);

    // The result is 00001101
    printf("a|b = %d\n", a | b);

    // The result is 00001100
    printf("a^b = %d\n", a ^ b);

    // The result is 11111010
    printf("~a = %d\n", a = ~a);

    // The result is 00010010
    printf("b<<1 = %d\n", b << 1);

    // The result is 00000100
    printf("b>>1 = %d\n", b >> 1);

    return 0;
}
```

Output

```
a = 5, b = 9

a&b = 1

a|b = 13

a^b = 12

~a = 250

b<<1 = 18
```

Conditional or Ternary Operator (?:) in C/C+

The conditional operator is kind of similar to the if-else statement as it does follow the same algorithm as of if-else statement but the conditional operator takes less space and helps to write the if-else statements in the shortest way possible.

Syntax:
The conditional operator is of the form

variable = Expression1 ? Expression2 : Expression3

```
if(Expression1)

{

   variable = Expression2;

}

else

{

   variable = Expression3;

}
```

Here, Expression1 is the condition to be evaluated. If the condition(Expression1) is True then Expression2 will be executed and the result will be returned.

Otherwise, if the condition(**Expression1**) is false then **Expression3** will be executed and the result will be

```
// C program to find largest among two
// numbers using ternary operator

#include <stdio.h>

int main()
{
    // variable declaration
    int n1 = 5, n2 = 10, max;

    // Largest among n1 and n2
    max = (n1 > n2) ? n1 : n2;

    // Print the largest number
    printf("Largest number between"
           " %d and %d is %d. ",
           n1, n2, max);

    return 0;
}
```

Output:

```
Largest number between 5 and 10 is 10.
```

2. Calculator program

```
#include <stdio.h>

int main() {

   char operator;

   double first, second;

   printf("Enter an operator (+, -, *,): ");

   scanf("%c", &operator);

   printf("Enter two operands: ");

   scanf("%lf %lf", &first, &second);

   switch (operator) {

   case '+':

      printf("%.1lf + %.1lf = %.1lf", first, second, first + second);

      break;

   case '-':

      printf("%.1lf - %.1lf = %.1lf", first, second, first - second);

      break;

   case '*':

      printf("%.1lf * %.1lf = %.1lf", first, second, first * second);

      break;

   case '/':

      printf("%.1lf / %.1lf = %.1lf", first, second, first / second);

      break;

      // operator doesn't match any case constant

   default:

      printf("Error! operator is not correct");

   }

   return 0;

}
```

OUTPUT

```
Enter the operator(‘+,-,*,/) :  *

Enter the operands: 4 5

4.0*5.0=20.0

Enter the operator(‘+,-,*,/): +

Enter the operands: 4 5

4.0+5.0=9.0

Enter the operator(‘+,-,*,/): +

Enter the operands: 4 5

4.0-5.0= -1.0
```