Modulus Operator in C

The symbol % denotes the modulus operator in C. It is an arithmetic operator. Modulus Operator in C is an important part of the GATE CSE syllabus.

Syntax of Modulus Operator in C

Let's assume that the variables p and q in a particular code are both integers. The modulus expression changes to:

p%q

The potentials for return values when p is divided by q, a remnant is left over. For example, the following return values are potential:

- The outcome of the equation is zero if and only if p is entirely divided by q. (0).
- The outcome will be an integer value that is non-zero if q cannot divide p entirely. In other words, the remaining amount will fall between [1, x-1].
- Division by zero causes a compile-time error if p is 0.
- If q is 0 and p is an integer value, we also receive a compile-time error.

Working of Modulus Operator in C

Since it always discovers a remainder of the two accessible numbers or integers concerning the numerator and denominator, the operation of this type of operator depends on the user's value. Various questions over the years have been formulated in GATE question papers based on the working of the Modulus Operator in C. Remember that we are discussing the residual, not the numerator-denominator pair's quotient. For instance:

- 9% 2 will leave us with a remainder of 1. It's because there is a residual of 1 when we divide 9 by 2, which results in a quotient of 3.
- Similarly, 8% 5 will leave us with 3 as the remaining number. Again, it is because when we divide 8 by 5, the quotient is 1, yet there is still a residual of 3.
- 6% 3 will result in 0 as 3 divides 6 completely, and no remainder is left.

Calculation of Modulus Operator in C

Let's examine the % operator's internal calculation in some C code currently in use. These calculations are important for the GATE exam as well. If p and q are integers, then the resolution of p% q is p - (p/q) * q. Now, we will assume values for p and q and then apply the modulus operator to them to arrive at the solution.

Example:

If p = 53 and q = 17, then

p % q >> p – (p/q) * q

53 % 17 >> 53– (53/17) * 17

53– (3) * 17

53 – 51

2

Thus, p % q here is 2.

Example for Modulus Operator in C

Being familiar with the working of the modulus operator, let us understand it better using a sample C code for modulus operation:

```
#include<stdio.h>

#include<conio.h>

void main()

{

int x, y;

int res; // store the resultant of modulus expression

x = 15;

y = 2;

res = x % y; // define modulus expression

printf(" Modulus returns a remainder: %d", res);

res = y % x; // define modulus expression

printf(" \n Modulus returns a remainder: %d", res);

x= 20;

y = 5;

res = x % y; // define modulus expression

printf(" \n Modulus returns a remainder: %d", res);

getch();
```

}

The output of the above code after execution will be:

- Modulus returns a remainder: 1
- Modulus returns a remainder: 2
- Modulus returns a remainder: 0

Exceptional Restrictions of the Modulus Operator in C

So far, we have thoroughly discussed the modulus operator usage in C, along with the help of examples. However, there are a few restrictions of the modulus operator, which are as follows:

- The % operator cannot be applied to a set of floating numbers (float or double). The compiler will produce a compile-time error if we try to use this operator with any floating-point variables or even constants.
- When employing the modulo operator on any two integers, it turns out that the sign of the result is machine-dependent for all the negative operands. It's because the overflow or underflow causes the activity to happen.