Operator

What is operator?

Answer: An operator is a symbol that operates on a value or a variable (অপারেটর এমন একটি প্রতীক যা একটি মান বা একটি ভেরিয়েবলের উপর পরিচালিত হয়)

**Type of operator in C++**

1. **Arithmetic Operators**
2. **Relational Operator**
3. **Logical Operator**
4. **Assignment Operator**
5. **Miscellaneous Operator**
6. **Bitwise Operator**
7. **Arithmetic Operator :**

Answer : An arithmetic operator is a mathematical function that takes two operands and performs a calculation on them(গাণিতিক অপারেটর একটি গাণিতিক ফাংশন যা দুটি অপারেন্ড নেয় এবং তাদের উপর একটি গণনা সম্পাদন করে)

Example : These operators are + (**addition**), - (**subtraction**), \* (**multiplication**), / (**division**), and % (modulo)

1. **Relational Operator :**

Answer: **Relational operator** is an operator that’s defines some kind of **relation** between two entities (রিলেশনাল অপারেটর এমন একটি অপারেটর যা দুটি সত্তার মধ্যে কোনও প্রকারের সম্পর্ককে সংজ্ঞায়িত করে)

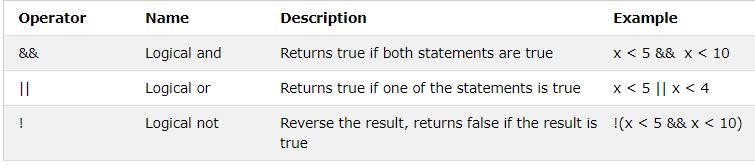
**Example:**



**3. Logical Operator**

A **logical operator** is a symbol or word used to connect two or more expressions (লজিকাল অপারেটর একটি প্রতীক বা শব্দ যা দুটি বা ততোধিক এক্সপ্রেশনকে সংযুক্ত করতে ব্যবহৃত হয়)

**Example:**



**4. Assignment Operator:**

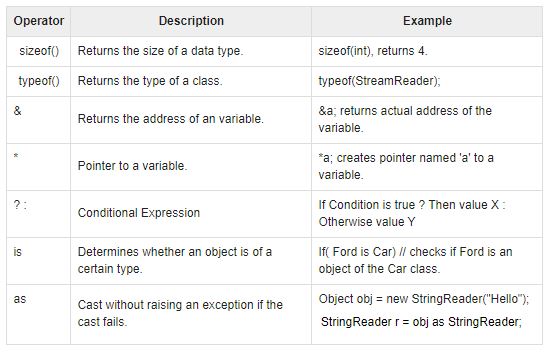
**Assignment operators** store a value in the object specified by the left operand(বরাদ্দ অপারেটরগুলি বাম অপারেণ্ড দ্বারা নির্দিষ্ট বস্তুর মধ্যে মূল্য সঞ্চয় করে।)

**Example :**

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| = | Simple assignment operator. Assigns values from right side operands to left side operand | C = A + B will assign the value of A + B to C |
| += | Add AND assignment operator. It adds the right operand to the left operand and assign the result to the left operand. | C += A is equivalent to C = C + A |
| -= | Subtract AND assignment operator. It subtracts the right operand from the left operand and assigns the result to the left operand. | C -= A is equivalent to C = C - A |
| \*= | Multiply AND assignment operator. It multiplies the right operand with the left operand and assigns the result to the left operand. | C \*= A is equivalent to C = C \* A |
| /= | Divide AND assignment operator. It divides the left operand with the right operand and assigns the result to the left operand. | C /= A is equivalent to C = C / A |
| %= | Modulus AND assignment operator. It takes modulus using two operands and assigns the result to the left operand. | C %= A is equivalent to C = C % A |
| <<= | Left shift AND assignment operator. | C <<= 2 is same as C = C << 2 |
| >>= | Right shift AND assignment operator. | C >>= 2 is same as C = C >> 2 |
| &= | Bitwise AND assignment operator. | C &= 2 is same as C = C & 2 |
| ^= | Bitwise exclusive OR and assignment operator. | C ^= 2 is same as C = C ^ 2 |
| |= | Bitwise inclusive OR and assignment operator. | C |= 2 is same as C = C | 2 |

**5. Miscellaneous Operator:**

**Miscellaneous operator** is conditional operator which has 3 operands; the first operand is always evaluated first. If nonzero, the second operand is evaluated, and that is the value of the result. Otherwise, the third operand is evaluated, and that is the value of the result (বিবিধ অপারেটর শর্তসাপেক্ষ অপারেটর যার 3 টি অপারেন্ড রয়েছে, প্রথম অপারেন্ডটি সর্বদা প্রথমে মূল্যায়ন করা হয়। যদি ননজারো হয় তবে দ্বিতীয় অপারেন্ডকে মূল্যায়ন করা হয় এবং এটি ফলাফলের মান। অন্যথায়, তৃতীয় অপারেন্ড মূল্যায়ন করা হয়, এবং এটি ফলাফলের মান)

****

**6. Bitwise Operation :**

**Bitwise operators** perform operations on integer data at the individual bit-level (বিটওয়াইজ অপারেটরগুলি পৃথক বিট-স্তরে পূর্ণসংখ্যার ডেটাতে অপারেশন করে।**)**

**BITWISE OPERATORS** are **used** for manipulating data at the bit level

Example :



**Bitwise AND operator &**

The output of bitwise AND is 1 if the corresponding bits of two operands is 1. If either bit of an operand is 0, the result of corresponding bit is evaluated to 0.

Let us suppose the bitwise AND operation of two integers 12 and 25.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bit Operation of 12 and 25

00001100

& 00011001

\_\_\_\_\_\_\_\_

00001000 = 8 (In decimal)

**Example #1: Bitwise AND**

#include <stdio.h>

int main()

{

int a = 12, b = 25;

printf("Output = %d", a&b);

return 0;

}

**Output**

Output = 8

**Bitwise OR operator |**

The output of bitwise OR is 1 if at least one corresponding bit of two operands is 1. In C Programming, bitwise OR operator is denoted by |.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise OR Operation of 12 and 25

00001100

| 00011001

\_\_\_\_\_\_\_\_

00011101 = 29 (In decimal)

**Example #2: Bitwise OR**

#include <stdio.h>

int main()

{

int a = 12, b = 25;

printf("Output = %d", a|b);

return 0;

}

**Output**

Output = 29

## Bitwise XOR (exclusive OR) operator ^

The result of bitwise XOR operator is 1 if the corresponding bits of two operands are opposite. It is denoted by ^.

12 = 00001100 (In Binary)

25 = 00011001 (In Binary)

Bitwise XOR Operation of 12 and 25

00001100

^ 00011001

\_\_\_\_\_\_\_\_

00010101 = 21 (In decimal)

**Example #3: Bitwise XOR**

#include <stdio.h>

int main()

{

int a = 12, b = 25;

printf("Output = %d", a^b);

return 0;

}

**Output**

Output = 21

## Bitwise complement operator ~

Bitwise compliment operator is an unary operator (works on only one operand). It changes 1 to 0 and 0 to 1. It is denoted by ~.

35 = 00100011 (In Binary)

Bitwise complement Operation of 35

~ 00100011

\_\_\_\_\_\_\_\_

11011100 = 220 (In decimal)

### Twist in bitwise complement operator in C Programming

The bitwise complement of 35 (~35) is -36 instead of 220, but why?

For any integer n, bitwise complement of n will be -(n+1). To understand this, you should have the knowledge of 2's complement.

### 2's Complement

Two's complement is an operation on binary numbers. The 2's complement of a number is equal to the complement of that number plus 1. For example:

Decimal Binary 2's complement

0 00000000 -(11111111+1) = -00000000 = -0(decimal)

1 00000001 -(11111110+1) = -11111111 = -256(decimal)

12 00001100 -(11110011+1) = -11110100 = -244(decimal)

220 11011100 -(00100011+1) = -00100100 = -36(decimal)

Note: Overflow is ignored while computing 2's complement.

The bitwise complement of 35 is 220 (in decimal). The 2's complement of 220 is -36. Hence, the output is -36 instead of 220.

### Bitwise complement of any number N is -(N+1). Here's how:

bitwise complement of N = ~N (represented in 2's complement form)

2'complement of ~N= -(~(~N)+1) = -(N+1)

### Example #4: Bitwise complement

#include <stdio.h>

int main()

{

printf("Output = %d\n",~35);

printf("Output = %d\n",~-12);

return 0;

}

**Output**

Output = -36

Output = 11

## Shift Operators in C programming

There are two shift operators in C programming:

* Right shift operator
* Left shift operator.

### Right Shift Operator

Right shift operator shifts all bits towards right by certain number of specified bits. It is denoted by >>.

212 = 11010100 (In binary)

212>>2 = 00110101 (In binary) [Right shift by two bits]

212>>7 = 00000001 (In binary)

212>>8 = 00000000

212>>0 = 11010100 (No Shift)

## Left Shift Operator

Left shift operator shifts all bits towards left by a certain number of specified bits. The bit positions that have been vacated by the left shift operator are filled with 0. The symbol of the left shift operator is <<.

212 = 11010100 (In binary)

212<<1 = 110101000 (In binary) [Left shift by one bit]

212<<0 = 11010100 (Shift by 0)

212<<4 = 110101000000 (In binary) =3392(In decimal)

### Example #5: Shift Operators

#include <stdio.h>

int main()

{

int num=212, i;

for (i=0; i<=2; ++i)

printf("Right shift by %d: %d\n", i, num>>i);

printf("\n");

for (i=0; i<=2; ++i)

printf("Left shift by %d: %d\n", i, num<<i);

return 0;

}

Right Shift by 0: 212

Right Shift by 1: 106

Right Shift by 2: 53

Left Shift by 0: 212

Left Shift by 1: 424

Left Shift by 2: 848