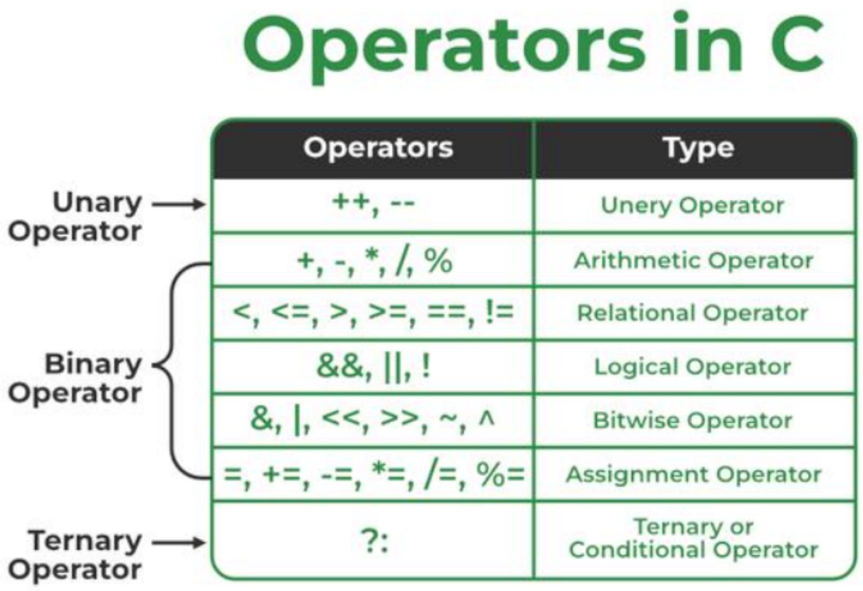
Operator and its Types in C

Opeartor: operator is a symbol that tells the compiler to perform specific mathematical, conditional, or logical functions. It is a symbol that operates on a value or a variable. For example, + and - are the operators to perform addition and subtraction in any C program.3

Operand: **an operand is any object capable of being manipulated. For example, in "1 + 2" the "1" and "2" are the operands and the plus symbol is the operator.**


Unary Operator
Binary Operator
Ternary Operator

Operators in C

Operators	Type
++, --	Unery Operator
+, -, *, /, %	Arithmetic Operator
<, <=, >, >=, ==, !=	Relational Operator
&&, ||, !	Logical Operator
&, |, <<, >>, ~, ^	Bitwise Operator
=, +=, -=, *=, /=, %=	Assignment Operator
?:	Ternary or Conditional Operator

Unary Operators in C: A unary operator **operates on only a single operand.**

++ (increment operator)

--(decrement operator)

Increment ++ increases the value by 1 whereas decrement -- decreases the value by 1. These two operators are unary operators, meaning they only operate on a single operand.

Example 2: Increment and Decrement Operators

```
// Working of increment and decrement operators
#include <stdio.h>
void main()
{
    int a = 10, b = 100;
    float c = 10.5, d = 100.5;

    printf("++a = %d \n", ++a);
    printf("--b = %d \n", --b);
    printf("++c = %f \n", ++c);
    printf("--d = %f \n", --d);


}
```

Output

```
++a = 11
--b = 99
++c = 11.500000
--d = 99.500000
```

Binary Operators in C: Binary Operators in the C programming language are ones that operate on two operands.

Arithmatic Operators: An arithmetic operator performs mathematical operations such as addition, subtraction, multiplication, division etc on numerical values (constants and variables).

Operator	Name of the Operator	Arithmetic Operation	Syntax	x=11 y=5 output
+	Addition	Add two operands.	x + y	16
–	Subtraction	Subtract the second operand from the first operand.	x – y	6
*	Multiplication	Multiply two operands.	x * y	55
/	Division	Divide the first operand by the second operand.	x / y	2
%	Modulus	Calculate the remainder when the first operand is divided by the second operand.	x % y	1

C Relational Operators

A relational operator checks the relationship between two operands. If the relation is true, it returns 1; if the relation is false, it returns value 0.

Relational operators are used in decision making and loops.

Operator	Meaning of Operator	Example
==	Equal to	`5 == 3` is evaluated to 0
>	Greater than	`5 > 3` is evaluated to 1
<	Less than	`5 < 3` is evaluated to 0
!=	Not equal to	`5 != 3` is evaluated to 1
>=	Greater than or equal to	`5 >= 3` is evaluated to 1
<=	Less than or equal to	`5 <= 3` is evaluated to 0

C Assignment Operators

An assignment operator is used for assigning a value to a variable. The most common assignment operator is `=`

Operator	Example	Same as	a=10
			b=5
			output

=	a = b	a = b	a=5
+=	a += b	a = a+b	a=15
-=	a -= b	a = a-b	a= 5
*=	a *= b	a = a*b	a=50
/=	a /= b	a = a/b	a=2
%=	a %= b	a = a%b	a=0

C Logical Operators

An expression containing logical operator returns either 0 or 1 depending upon whether expression results true or false. Logical operators are commonly used in decision making in C programming.

Operator	Meaning	Example
&&	Logical AND. True only if all operands are true	If c = 5 and d = 2 then, expression `((c==5) && (d>5))` equals to 0.
||	Logical OR. True only if either one operand is true	If c = 5 and d = 2 then, expression `((c==5) || (d>5))` equals to 1.
!	Logical NOT. True only if the operand is 0	If c = 5 then, expression `!(c==5)` equals to 0.

Bitwise operators:

The bitwise operators are the operators used to perform the operations on the data at the bit-level. When we perform the bitwise operations, then it is also known as bit-level programming. It consists of two digits, either 0 or 1. It is mainly used in numerical computations to make the calculations faster.

We have different types of bitwise operators in the C programming language. The following is the list of the bitwise operators:

Operator	Meaning of operator
&	Bitwise AND operator
|	Bitwise OR operator
^	Bitwise exclusive OR operator
~	One's complement operator (unary operator)
<<	Left shift operator
>>	Right shift operator

Let's look at the truth table of the bitwise operators.

X	Y	X&Y	X|Y	X^Y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	1

Ternary Operator: We use the ternary operator in C to run one code when the condition is true and another code when the condition is false. For example,

```
(age >= 18) ? printf("Can Vote") : printf("Cannot Vote");
```

Syntax of Ternary Operator

The syntax of ternary operator is :

```
testCondition ? expression1 : expression 2;
```

The `testCondition` is a boolean expression that results in either **true** or **false**. If the condition is

- `true` - **expression1** (before the colon) is executed
- `false` - **expression2** (after the colon) is executed

Example: C Ternary Operator

```
#include <stdio.h>
void main() {
  int age;
  // take input from users
  printf("Enter your age: ");
  scanf("%d", &age);

  // ternary operator to find if a person can vote or not
  (age >= 18) ? printf("You can vote") : printf("You cannot vote");
}
```

output 1

Enter your age 17

You cannot vote

Operator Precedence:

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.

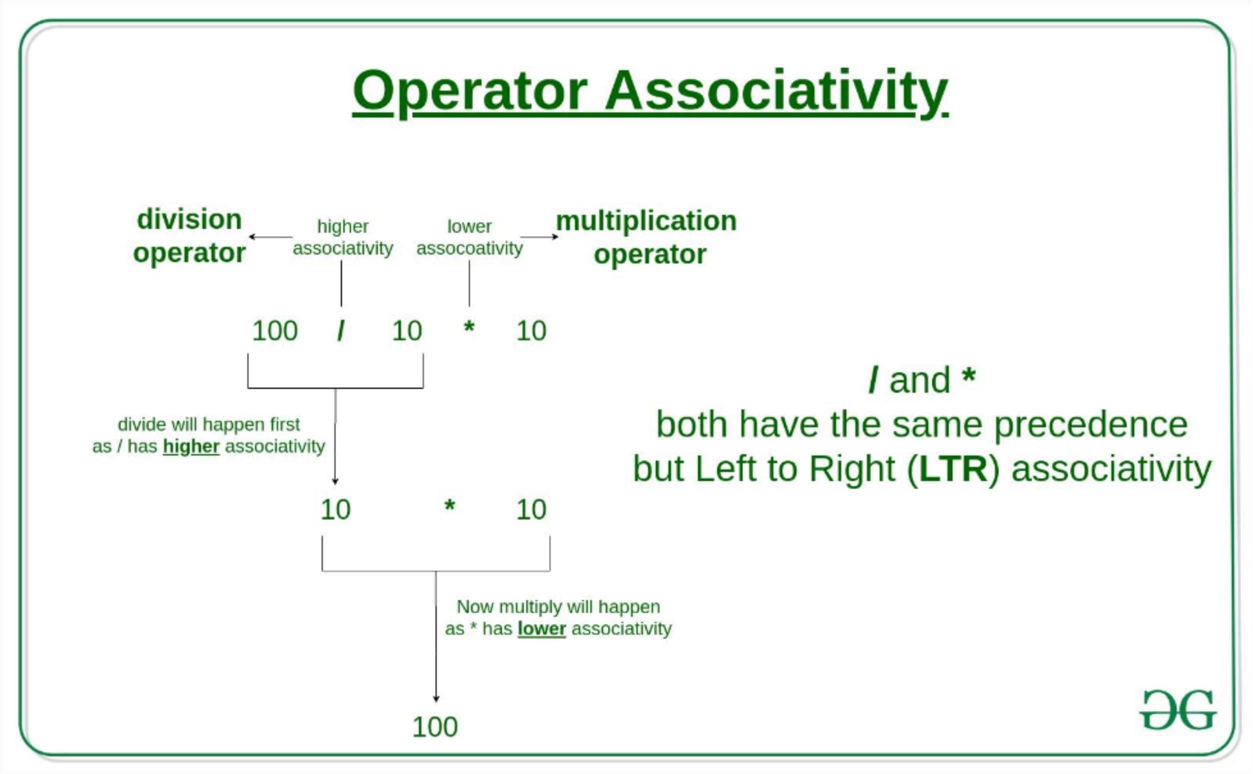
For example, x = 7 + 3 * 2; here, x is assigned 13, not 20 because operator * has a higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

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

Operator Associativity:

Operators Associativity is used when two operators of same precedence appear in an expression. Associativity can be either **L**eft **t**o **R**ight or **R**ight **t**o **L**eft.

For example: '*' and '/' have same precedence and their associativity is **L**eft **t**o **R**ight, so the expression "100 / 10 * 10" is treated as "(100 / 10) * 10


Operator Associativity
division operator
higher associativity
lower assocoativity
multiplication operator
100 / 10 * 10
divide will happen first as / has higher associativity
10 * 10
Now multiply will happen as * has lower associativity
100
/ and * both have the same precedence but Left to Right (LTR) associativity

Unit 2 Notes Part 2

Obj:stands for object files

Exe:stands for executable files (difference already told in unit 1)

Define if Else in C:

In C programming language, if-else statement is used to perform the operations based on some specific condition. If the given condition is true, then the code inside the if block is executed, otherwise else block code is executed. It specifies an order in which the statements are to be executed.

Syntax

```
if (condition or expression) {
     // statement(s) will execute if the condition or
expression is true
} else {
     // statement(s) will execute if the condition or
expression is false
}
```

If Else in C

Program for even or odd number

```
#include<stdio.h>
#include<conio.h>
void main()
{
int num;
printf(“enter any no”);
scanf(“%d”,&num);
If num%2==0
printf(“even number”);
Elese
printf(“odd number”);
getch();
}
```

Program for leap year

```
#include<stdio.h>
#include<conio.h>
void main()
{
int year;
clrscr();
Printf(“enter any no”);
Scanf(“%d”,&year);
If(year %400==0 || year%4==0 && year %100 !=0)
printf(“leap year”);
else
printf(“not leap year”)
getch();
}
```

Other if else program

1.Program to find number is positive or negative using if else
2.Program to enter age and find if eligible to vote or not using if else.
3.Program to find greatest of 2 numbers using if else

Switch Case in C:

Switch statement in C tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed.

Each case in a block of a switch has a different name/number which is referred to as an identifier.

If a case match is NOT found, then the default statement is executed, and the control goes out of the switch block.

Rules for switch statement in C language

1) The *switch expression* must be of an integer or character type.

2) The *case value* must be an integer or character constant.

3) The *case value* can be used only inside the switch statement.

Switch case program for calculator:

```
#include <stdio.h>
 void  main()
{
    int num1,num2,ch,result;

    printf("Enter first number: ");
    scanf("%d",&num1);
    printf("Enter second number: ");
    scanf("%d",&num2);

    printf("Choose operation to perform 1.add 2.substract 
3.Multiply ");
    scanf(" %d",&ch);

        switch(ch)
    {
        case 1:
            result=num1+num2;
printf(“addition is %d”,result);
            break;

        case 2:
                        result=num1-num2;
printf(“substraction is %d”,result);
            break;


        case 3:
                        result=num1*num2;
printf(“Multiply is %d”,result);
            break;

        case 4:
                        result=num1/num2;
printf(“Division is %d”,result);
            break;

                default:
            printf("Invalid operation.\n");
    } }
```

```
/**
 * C program to print day of week using switch case
 */

#include <stdio.h>

void main()
{
    int week;

    /* Input week number from user */
    printf("Enter week number(1-7): ");
    scanf("%d", &week);

    switch(week)
    {
        case 1:
            printf("Monday");
            break;
        case 2:
            printf("Tuesday");
            break;
        case 3:
            printf("Wednesday");
            break;
        case 4:
            printf("Thursday");
            break;
        case 5:
            printf("Friday");
            break;
        case 6:
            printf("Saturday");
            break;
        case 7:
            printf("Sunday");
            break;
        default:
            printf("Invalid input! Please enter week number between 1-7.");
    }


}
```

Output: enter week number(1-7)

1

Monday

Program to swap two numbers without using third variable:

```
#include <stdio.h>

void main()
```

```
{
int a,b;
printf(“enter 2 numbers\n”);
scanf(“%d%d”,&a,&b);
a=a+b;
b=a-b;
a=a-b;
printf(“swapped variable a=%d b=%d”,a,b);
}
```

Mixed mode arithmetic

An operation between an integer and a floating point yields a floating point result. In this operation, the integral value is first converted to floating point value then the operation is performed. Let's take two variables `a` and `b`, such that `a = 14` and `b = 2.5`. The following table shows arithmetic operations performed on `a` and `b`.

Expression	Result
`a + b`	`16.500000`
`a - b`	`12.500000`
`a * b`	`35.000000`
`a / b`	`5.600000`

As we can see, when `14` is divided by `2.5` the fractional part is not lost because arithmetic operation between an `int` and a `double` yields a `double` value. So we can use Mixed mode arithmetic to solve the problem we encountered while dividing `10/4`. To get the correct answer simply make one of the operands involved in the operation a floating point number. For example, `10/4.0` or `10.0/4` both will give the correct result i.e `2.5`.

type conversion

In C programming, we can convert the value of one data type (`int`, `float`, `double`, etc.) to another. This process is known as **type conversion**. Let's see an example

```
#include <stdio.h>

void main() {

  int number = 34.78;

  printf("%d", number);

  }

// Output: 34
```

In C, there are two types of type conversion:

1. Implicit Conversion

2. Explicit Conversion

Implicit Conversion	Explicit type Conversion
Implicit Conversion is done automatically.	Explicit Conversion is done programatically.
In Implicit conversion, no data loss take place during the data conversion.	In explicit conversion, data loss may or may not be take place d data conversion. Hence there is a risk of information loss.
No possibility of throwing exception during the conversion and therefore is called type safe.	It might throw error if tried to do without type casting.
Implicit conversion do not require any special syntax.	Explicit conversion do require cast operator to perform conversion.
Example : Conversion of smaller number to larger number is implicit conversion. Conversion of integer type data to float. float i=0; int j=10; i=j; // This is implicit conversion since float is larger than integer,hence no loss of data & no exception.	Example : Conversion of larger number to smaller number is explicit conv **float k;** **int i=11,j=5;** **k= (float)i/j;** **printf("%d",k);** **Output:2.2**

Explicit Conversion Example:

float k;

int i=11,j=5;

k= (float)i/j;

printf("%d",k);

Output:2.2

Without type conversion the answer will be 2.00