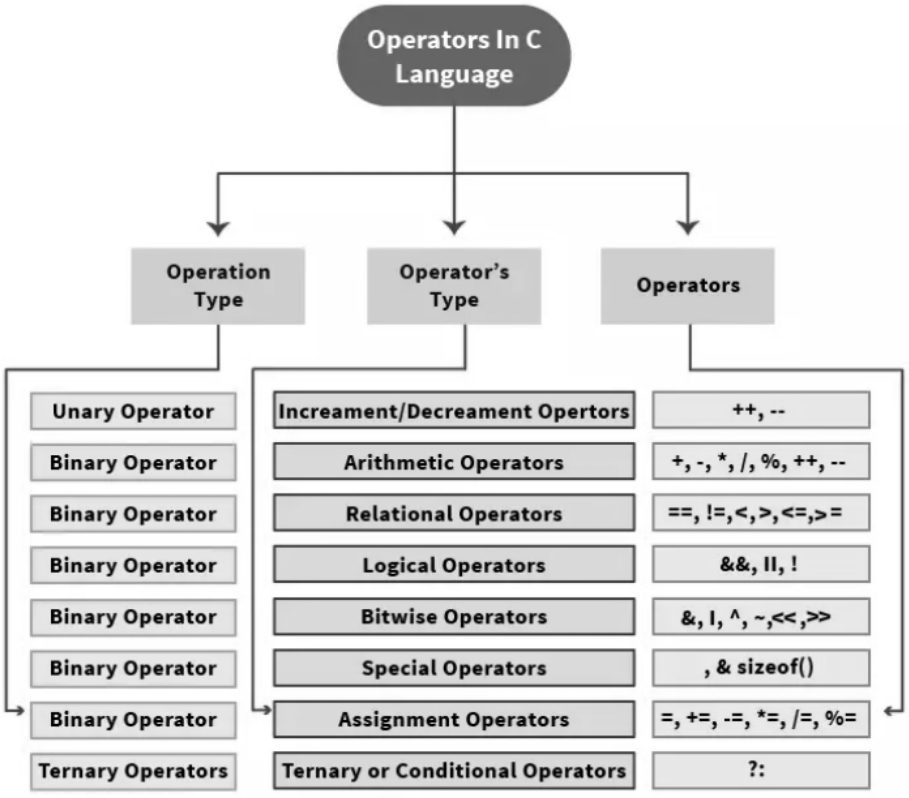
Algorithm Development and Programming Fundamentals
MCA SEM-1

Operators in C Language

Operators in C are symbols which work on operands. Operator in C language is used to perform specific mathematical or logical computations on the operands and it reduces a single value.

Operators in C language are classified into several categories:

- Arithmetic Operators
- Relational Operators
- Shift Operators
- Logical Operators
- Bitwise Operators
- Ternary or Conditional Operators
- Assignment Operators
- Misc Operators
- Special Operators



Arithmetic Operators

An arithmetic operator is used to perform arithmetic/mathematical operations on operands. Some of the arithmetic operators are (+, -, *, /, %, ++, --)

Operator	Name of Operator	What it does	How it is used
+	Unary Plus	Add two Operands	a+b
-	Unary Minus	Subtracts the second operand from the first.	a-b
*	Multiplication	Multiplies both operands.	a*b
/	Division	Divides numerator by de-numerator.	a/b
%	Modulus	return remainder, after an integer division.	a%b
++	Increment Operator	increases the integer value by one.	a++
- -	Decrement Operator	decreases the integer value by one.	a- -

Relational Operators

Relational operators help in making a relationship or comparison between two operands with which they are used. Hence, relational operators help us make decisions in the program and their end result is either true or false. Some of the relation operators are (==, !=, <, >, <=, >=)

Operator	Name of Operator	What it does	Return value
==	Equality Operator	checks if a == b	Boolean/Integer
!=	Not equal to	checks if a != b	Boolean/Integer
<	Less than	checks if a < b	Boolean/Integer
>	Greater than	checks if a > b	Boolean/Integer
<=	Less than or equal to	checks if a<=b	Boolean/Integer
>=	Greater than or equal to	checks if a>=b	Boolean/Integer

Logical Operators

The logical operators are used when we want to check or test more than one condition and make decisions. Some of the logical operators are(&&, ||, !).

Example:

```
(a > b) && x == 100
```

The logical expression given above is true only if a > b is true and x == 100 is true. if either (or both) of them are false, the expression is false.

Operator	Name of the operator	What it does	How it is used/output
&&	logical AND	returns true if both side operands value is true otherwise returns false	Boolean/Integer
||	logical OR	returns true if one of the operand's value is true or both of the operand's values is true otherwise returns false	Boolean/Integer
!	logical Not	returns true if the condition in consideration is not satisfied Otherwise returns false	Boolean/Integer

Bitwise Operators

A Bitwise operator is used for the manipulation of data at the bit level. These operators are not applied for the float and double datatype. Bitwise operator first converts the integer into its binary representation then performs its operation. Bitwise operators consist of two digits, either 0 or 1. Some of the bitwise operators are (&, | , ^, ~). Shift Bitwise operators are used to shift the bits right to left. Some of the shift bitwise operators are(<<, >>)

A	B	A & B (Bitwise AND)	A | B (Bitwise OR)	A ^ B (Bitwise XoR)
1	1	1	1	0
0	1	0	1	1
1	0	0	1	1
0	0	0	0	0

Example:

a = 5, b = 6
a & b = 4 (In Decimal) a | b = 7 (In Decimal) a ^ b = 3 (In Decimal)

a's binary representation is 0101(5) and b's binary representation is 0110(6)

```
AND Operation     OR Operation      XOR Operation
   0101              0101              0101
& 0110            | 0110            ^ 0110
   ------            ------            ------
   0100 = 4          0111  = 7         0011 = 3
```

Operator	Name of Operator	What it does	How it is used
&	bitwise AND	bitwise AND operator do AND of every corresponding bits of both operands and output 1 (true) if both operands have 1 at that position otherwise 0(false).	a & b
|	bitwise OR	bitwise OR operator do OR operation of every corresponding bits of both operands and output 0 (false) if both operands have 0 at that position otherwise 1(true).	a | b
~	bitwise complement	performs complement operation on an operand and bitwise complement changes 1 to 0 and 0 to 1	~a
^	bitwise exclusive OR	returns 1 if the corresponding bits of two operands are opposite else 0	a^b
<<	shift left	shifts the number of bits to the left side	a << 1
>>	shift right	shifts the number of bits to the right side	a >> 1

Ternary or Conditional Operators

The ternary or conditional operators are used to construct the conditional expression. A conditional operator pair "?:"

Syntax:

```
exp1 ? exp2 : exp3
```

Here exp1, exp2, exp3 are expressions.

exp1 is evaluated first. If it is true, then the expression exp2 is evaluated and becomes the value of the expression. If exp1 is false, then exp3 is evaluated and its value becomes the value of the expression.

Example:

```
a = 100;
b = 200;
x = (a > b) ? a : b;
```

Misc Operators

Misc Operators are also called Miscellaneous Operators. Some of the Misc operators are (sizeof() ?: , & *)

Operator	**Name of Operator**	**What it does**	**How it is used**
sizeof()	sizeof	It returns the size of variable	if variable a is an integer variable the sizeof(a) will return 4
?:	conditional or ternary operator	if the condition is true then it returns the value of x else value of y	condition?x:y
cast	type cast	it converts one datatype to another datatype	int(5.260) would return 5
,	comma operator	Used to link the related expressions together	a = (1,2,3) would return 3
&	Address Operator	returns the address of the variable.	&a
*	pointer operator	pointer to a variable	*a

Assignment Operators

An assignment operator is used to assign values to the operands. Some of the assignment operators are (=, +=, -=, *=, /=, %=)

Operator	Name of Operator	What it does	How it is used
=	assignment	assign value of variable b to variable a	a = b
+=	plus assign	a = a+b (adds values of a to b and assign this value to a)	a += b
-=	minus assign	a = a-b (subtracts values of b from a and assign this value to a)	a -= b
*=	times assign	a = a*b (Multiplies a with b and assign the value to a)	a *= b
/=	div assign	a = a/b (divides a by b and assigns the value to a)	a /= b
%=	Mod assign	a = a%b (divides a by b and assigns the value of the remainder to a)	a %= b

Special Operators

C supports some special operators some of the special operators are (comma operator, address operator, size of operator, pointer operator)

Example:

```
m = sizeof(a)
```

Operator	Name of Operator	What it does	How it is used
,	Comma	Used to link the related expressions together	value = (x=10, y=5)
&	Address Operator	returns the address of the variable.	&a
sizeof()	sizeof	returns the size of a variable	m = sizeof(a)

Precedence(or priority) and Associativity of Operators in C

Precedence determines which operator is performed first in an expression if there are more than one operator of different precedence(lower precedence means higher priority). Associativity determines in which direction we should start computing the operators having the same precedence.

Token	Operator	Precedence	Associativity
()	function call/ Expression Grouping	1	left-to-right
[]	array element	1	left-to-right
++	postfix increment	1	left-to-right
- -	postfix decrement	1	left-to-right
++	prefix increment	2	right-to-left
- -	prefix decrement	2	right-to-left
+	unary plus	2	right-to-left
-	unary minus	2	right-to-left
!	Logical negation	2	right-to-left
~	one's complement	2	right-to-left
*	indirection	2	right-to-left
&	address	2	right-to-left
sizeof	size(in bytes)	2	right-to-left
(type)	type cast	2	right-to-left
*	multiplication	3	left-to-right
/	division	3	left-to-right
%	modulus	3	left-to-right
+	addition	4	left-to-right

-	subtraction	4	left-to-right
<<	left shift	5	left-to-right
>>	right shift	5	left-to-right
<	less than	6	left-to-right
<=	less than or equal to	6	left-to-right
>	greater than	6	left-to-right
>=	greater than or equal to	6	left-to-right
==	equality	7	left-to-right
!=	inequality	7	left-to-right
&	bitwise AND	8	left-to-right
^	bitwise XOR	9	left-to-right
|	bitwise OR	10	left-to-right
&&	Logical AND	11	left-to-right
||	Logical OR	12	left-to-right
?:	conditional expression	13	right-to-left
= *= /= %= += -= &= ^= |= <<= >>=	assignment operators	14	right-to-left
,	comma operator	15	left-to-right

C Library - <limits.h>

The limits.h header determines various properties of the various variable types. The macros defined in this header, limit the values of various variable types like char, int and long.

These limits specify that a variable cannot store any value beyond these limits, for example an unsigned character can store up to a maximum value of 255.

Example

The following example shows the usage of few of the constants defined in limits.h file.

```
#include <stdio.h>
#include <limits.h>

int main() {

   printf("The number of bits in a byte %d\n", CHAR_BIT);

   printf("The minimum value of SIGNED CHAR = %d\n", SCHAR_MIN);
   printf("The maximum value of SIGNED CHAR = %d\n", SCHAR_MAX);

   return(0);
}
```

Basic C Programs - II

A. Write down the steps to perform operations of following programs. Draw appropriate flowchart for the same.
B. Perform the following programs using GCC. Write down the commands to compile and run the program.

1. Provided three numbers in input, Find the greatest number using a C program.
2. Write a simple C program to check if the number is negative or positive.
3. Write a simple C program to find the simple interest I = p*r*n/100.
4. Write a simple C program to find the average of three numbers.
5. Write a simple C program to check the year is a leap year.
6. Write a simple C program to check if the number is Odd or Even.
7. Write a C program to swap two int numbers using a temporary variable.
8. Write a C Program to Find the Size of primitive data types. [Use sizeof operator] [Write only C code]
9. Write a C Program to demonstrate the limits of data types using limits.h [Write only C code]

Name	Meaning
CHAR_BIT	Bits in a char
CHAR_MAX	Maximum value of char
CHAR_MIN	Minimum value of char
INT_MAX	Maximum value of int
INT_MIN	Minimum value of int
LONG_MAX	Maximum value of long
LONG_MIN	Minimum value of long
SCHAR_MAX	Maximum value of signed char
SCHAR_MIN	Minimum value of signed char
SHRT_MAX	Maximum value of short
SHRT_MIN	Minimum value of short
UCHAR_MAX	Maximum value of unsigned char
UINT_MAX	Maximum value of unsigned int
ULONG_MAX	Maximum value of unsigned long
USHRT_MAX	Maximum value of unsigned short