UNIT II - Managing simple Input and Output operations - Operators and Expressions - Decision Making: Branching statements, looping statements - Function: Declaration, Definition - Passing arguments by value - Recursion - Storage classes.

2.2 OPERATORS AND EXPRESSIONS

(I) OPERATORS

Operator is a symbol that performs the operation on one or more operands. C Language provides the following operators:

i. Arithmetic operators

ii. Relational operators

iii. Logical operators

iv. Assignment operators

v. Increment and Decrement operators

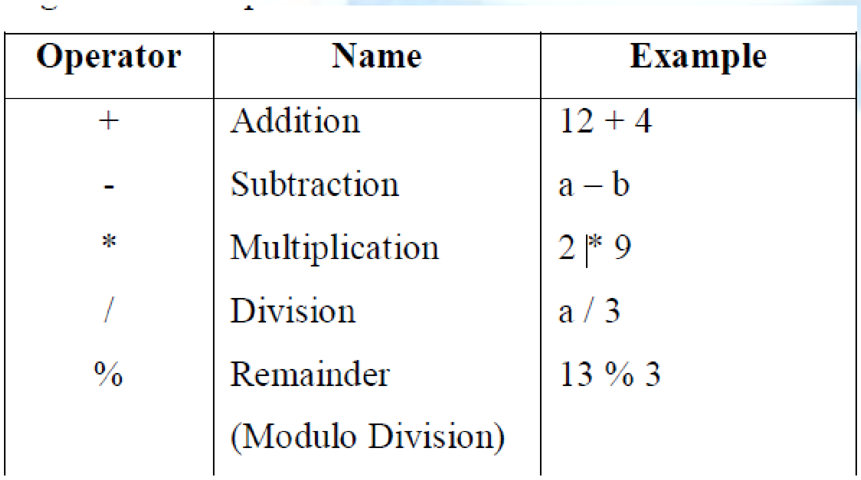
vi. Conditional operators

vii. Bitwise operators

viii. Special operators

i) Arithmetic Operators:

Arithmetic operations like addition, subtraction, multiplication, division etc can be performed by using arithmetic operators.



Operator	Name	Example
+	Addition	12 + 4
-	Subtraction	a – b
*	Multiplication	2 * 9
/	Division	a / 3
%	Remainder (Modulo Division)	13 % 3

Program

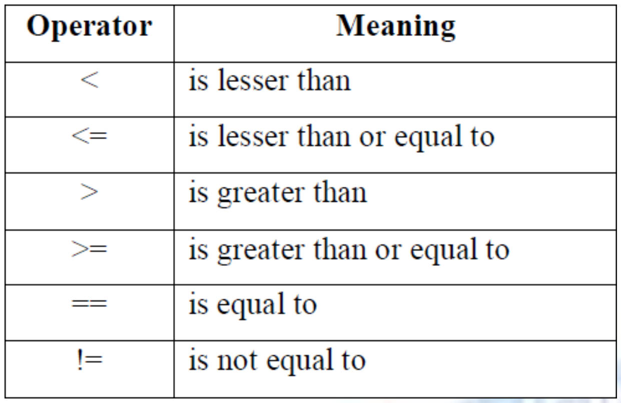
```
#include<stdio.h>
#include<conio.h>
void main()
{
int a =15;
int b=10;
int add,sub,mul,div,mod;
add = a+b;
sub = a-b;
mul=a*b;
div= a/b;
mod= a%b;
printf( “addition=%d”,add);
prtintf(“subtraction=%d”,sub);
printf”multiplication=%d”,mul);
printf(“division=%d”,div);
printf(“modulo=%d”,mod);
getch();
}
```

Output:

```
Addition= 25
Subtraction=5
Multiplication=150
Division=1
Modulo=5
```

ii) <u>Relational Operators</u>:

- Relational operators are used to compare two or more operands.
- We use relational expression in if, for and while statements.
- Relational expressions return either **True (1) or False (0).**



Operator	Meaning
<	is lesser than
<=	is lesser than or equal to
>	is greater than
>=	is greater than or equal to
==	is equal to
!=	is not equal to

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a =15;
int b=10;
printf( “a<b:%d”,a<b);
prtintf(“a<=b:%d”,a<=b);
printf”a>b:%d”,a>b);
printf(“a>=b:%d”,a>=b);
printf(“a==b:%d”,a==b);
printf(“a!=b:%d”,a!=b);
getch();
}
```

Output

```
a<b: 0
a<=b:0
a>b:1
a>=b:1
a==b:0
a!=b:1
```

iii) Logical Operators:

- ✓ Logical operators are used to combine the results of two or more relational expressions.

Operator	Meaning
!	Logical NOT
&&	Logical AND
||	Logical OR

- ✓ Logical NOT is a unary operator that negates the logical value of its single operand.
- ✓ Logical NOT convert a 1 to 0, and 0 to 1.
- ✓ Logical AND produces 1 if both operands are 1, otherwise produce 0.
- ✓ Logical OR produces 0 if both operands are 0, otherwise it produces 1.

iv) Assignment Operator:

Assignment operator '=' is used to assign a constant or a value of an expression or a value of a variable to other variable.

Syntax

Variable = expression (or) value

Operators		Example	Explanation
Simple assignment operator	=	sum=10	10 is assigned to variable sum
Compound assignment operators	+=	sum+=10	sum=sum+10
	-=	sum-=10	sum = sum-10
	*=	sum*=10	sum = sum*10
	/+	sum/=10	sum = sum/10
	%=	sum%=10	sum = sum%10
	&=	sum&=10	sum = sum&10

(v) Increment and Decrement Operators (unary):

- ➢ increment (++) - Adds one to the variables

- decrement (--) - Subtract one from the variable

Operator	Meaning
++ x	Pre increment (Increment then display)
-- x	Pre decrement (decrement then display)
x ++	Post increment (display then increment)
x --	Post decrement (display then decrement)

Example:

a=10;

++a =11

a++ =10

--a =9

a-- =10

vi) <u>Conditional Operator (or) Ternary Operator:</u>

?: is known as conditional operator. It is equivalent to simple if then else statement. It checks the condition and executes the exp1 if condition is true otherwise it execute exp2.

Syntax

condition ? exp1 : exp 2;

Program:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a = 5
int b = 3
int max;
max = a > b ? a : b ;
printf("Maximum is %d", max);
getch();
}
```

Output:

```
Maximum is 5
```

In this example, it checks the condition „a > b", if it is true, then the value of „a" is assigned to „max", otherwise the value of „b" is assigned to „max".

vii) <u>Bitwise Operators:</u>

- Bitwise operators are used to calculate the data at bit level.
- It operates on integers only.

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
^	Bitwise XOR
<<	Shift left
>>	Shift right
~	Bitwise NOT (or) One's complement

Bitwise AND (&)

This operator compare the operands in corresponding bits position and produces 1 if both operand bits are 1, otherwise produces 0.

Bitwise OR (|)

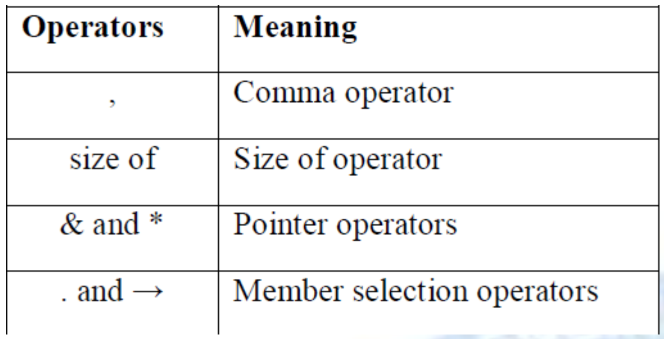
This operator compare the operands in corresponding bits position and produces 0 if both operand bits are 0, otherwise produces 1.

Bitwise XOR (^)

This operator compare the operands in corresponding bits position and produces 1 if both operand bits are same, otherwise produces 0.

viii) Special Operators: (Miscellanous Operator)

C language supports some of the special operators.



Operators	Meaning
,	Comma operator
size of	Size of operator
& and *	Pointer operators
. and →	Member selection operators

Comma Operator: It is used to separate elements.

Example :

```
int X, Y;
```

sizeof Operator: It is used to return the size of the data type or variable.

Example :

```
sizeof(Y);
```

Member Selection Operators: It is used to refer structure or union member element.

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
int Y;
printf (“Size of Y is %d”, sizeof(Y));
getch();
}
```

Output:

```
Size of Y is 2
```

(II) <u>OPERATORS: PRECEDENCE AND ASSOCIATIVITY</u>

An operator is a special symbol that is used to perform particular mathematical or logical computations like addition, multiplication, comparison and so on. The value of operator is applied to be called operands.

Precedence and Associativity are two characteristics of operators that determine the evaluation order of subexpressions in absence of brackets.

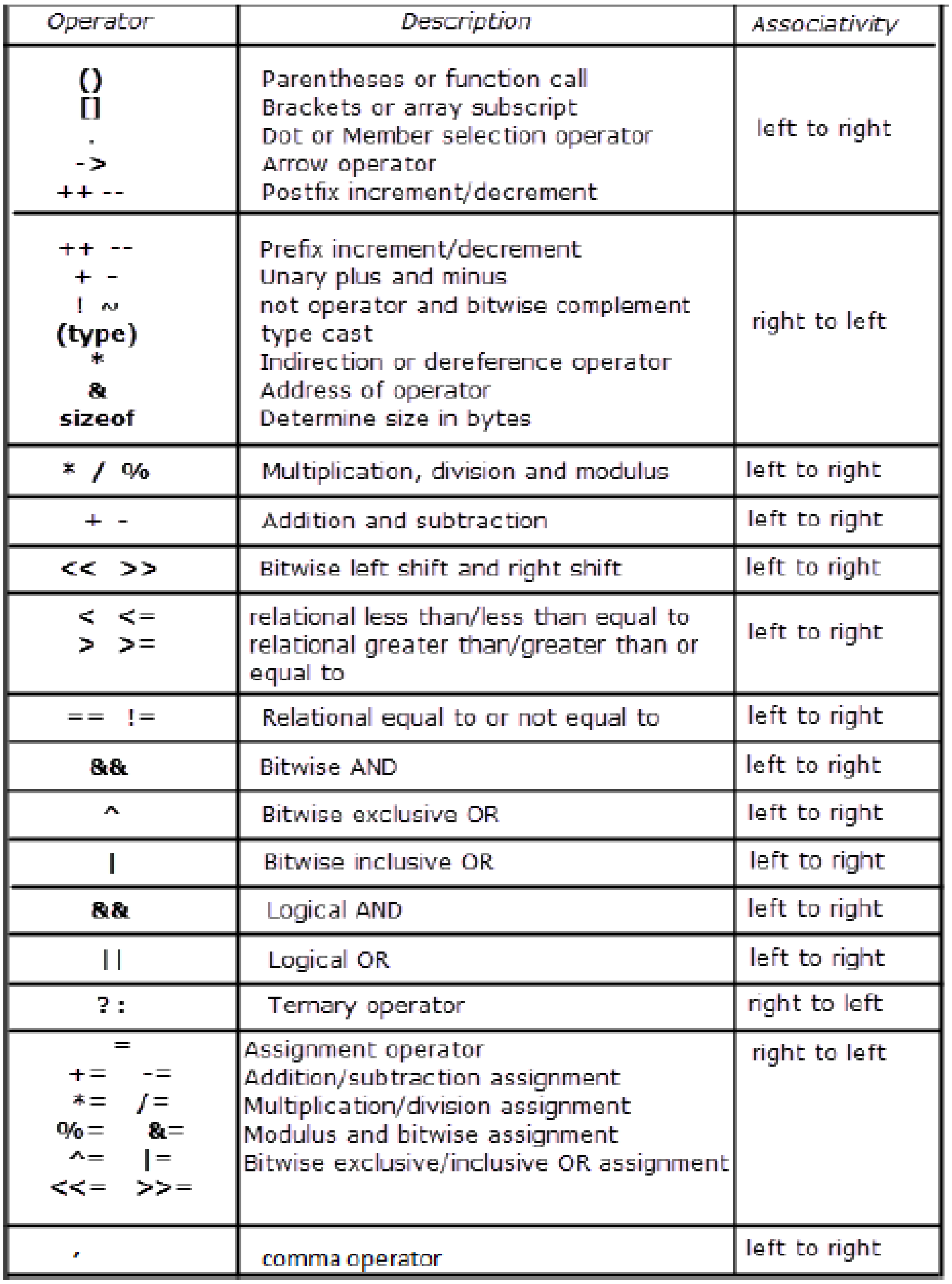
Precedence of operators

The precedence rule is used to determine the order of application of operators in evaluating sub expressions. The operator with the highest precedence is operated first. Parenthesis operator has the highest priority.

Associativity of operators

The associativity rule is applied when two or more operators are having same precedence in the sub expression.

- An operator can be ***left-to-right associative or right-to-left*** associative.
- All operators with same precedence have same associativity



Operator	Description	Associativity
()	Parentheses or function call	left to right
[]	Brackets or array subscript	
.	Dot or Member selection operator	
->	Arrow operator	
++ --	Postfix increment/decrement	
++ --	Prefix increment/decrement	right to left
+ -	Unary plus and minus	
! ~	not operator and bitwise complement	
(type)	type cast	
*	Indirection or dereference operator	
&	Address of operator	
sizeof	Determine size in bytes	
* / %	Multiplication, division and modulus	left to right
+ -	Addition and subtraction	left to right
<< >>	Bitwise left shift and right shift	left to right
< <=	relational less than/less than equal to	left to right
> >=	relational greater than/greater than or equal to	
== !=	Relational equal to or not equal to	left to right
&&	Bitwise AND	left to right
^	Bitwise exclusive OR	left to right
|	Bitwise inclusive OR	left to right
&&	Logical AND	left to right
||	Logical OR	left to right
? :	Ternary operator	right to left
=	Assignment operator	right to left
+= -=	Addition/subtraction assignment	
*= /=	Multiplication/division assignment	
%= &=	Modulus and bitwise assignment	
^= |=	Bitwise exclusive/inclusive OR assignment	
<<= >>=		
,	comma operator	left to right

Arithmetic Operator Precedance

Arithmetic evaluation is carried out using two phases from left to right.

- During the first phase, highest priority operators are evaluated.

- In the second phase, lowest priority operators are evaluated.

Precedance	Operator
High	* / %
Low	+ -

Example:

Result= 6+4/ 2.

Phase1: (4/2 operation is evaluated first)

Result= 6+2

Phase2: (6+2 operation is evaluated next)

Result= 8

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
Result= 6+4/ 2;
printf("Result=%d",Result);
getch();
}
```

Output

```
Result= 8
```

(III) EXPRESSIONS

An expression is a sequence of operators and operands that specifies the computation. An operand can be a variable, constant or a function call. An operator is a symbol that is used to write a mathematical, logical or relational expression.

Simple Expression

An expression that has only one operator is known as simple expression.

Example:

X=a+b;

X=++a;

Compound Expression

An expression that has more than one operator is known as compound expression.

Example:

X=a+b*c/f;

Arithmetic Expression

An expression consisting of arithmetic operators is known as arithmetic expression.

Example:

X=a+b;

Logical Expression

An expression consisting of logical operators is known as Logical expression.

Example:

X=a>b;

Program

```
#include<stdio.h>
#include<conio.h>
void main()
{
Result= 6+4/ 2;
printf("Result=%d",Result);
getch();
}
```

Output

```
Result= 8
```