Introduction to C Programming (22ECS145/245)

Module 2

- **Operators in C, Type conversion and typecasting**
- **Decision control and Looping statements:** Introduction to decision control, Conditional branching statements, iterative statements, nested loops, break and continue statements, goto statement.

- **Textbook: Chapter 9.15-9.16, 10.1-10.6**

Operators in C

An operator is a symbol that tells the computer to perform particular mathematical and logical functions.

To manipulate variables and data in the program Operators are used.

Operators supported in ‘C’ are:

1) Arithmetic Operators
2) Relational Operators
3) Equality Operators
4) Logical Operators
5) Unary Operators
6) Conditional Operators
7) Bitwise Operators
8) Assignment Operators
9) Comma Operators

Arithmetic Operators

- These operators are used to perform basic arithmetic operations

Operator	Name	Result	Syntax	Example (b=5, c=2)
+	Addition	Sum	a = b + c	a = 7
-	Subtraction	Difference	a = b - c	a = 3
*	Multiplication	Product	a = b * c	a = 10
/	Division	Quotient	a = b / c	a = 2
%	Modulus	Remainder	a = b % c	a = 1

Relational Operators

- This operator compares two operands in order to find out the relation between them.
- The output will be either 0 (False) or 1 (True).

Operator	Name	Syntax	Example (b=5, c=2)
<	Lesser than	a = b < c	a = 0 (False)
>	Greater than	a = b > c	a = 1 (True)
<=	Lesser than or Equal to	a = b <= c	a = 0 (False)
>=	Greater than or Equal to	a = b >= c	a = 1 (True)
= =	Equal to	a = b = = c	a = 0 (False)
!=	Not equal to	a = b!= c	a = 1 (True)

Equality Operators

1. Equal to(==)
2. Not equal to (!=)

Operator	Meaning
==	Returns 1 if both operands are equal, 0 otherwise
!=	Returns 1 if operands do not have the same value, 0 otherwise

Logical Operators

- These are used to test more than one condition and make decision. The different logical operators are:
- Logical NOT
- Logical AND
- Logical OR

Logical NOT (!) The *output is true* when *input is false* and vice versa. It accepts only one input.

Input	Output
X	!X
0	1
1	0

Logical Operators

- **Logical AND (&&)** The *output is true* only if *both inputs are true*. It accepts two or more inputs.

Input		Output
X	Y	X && Y
0	0	0
0	1	0
1	0	0
1	1	1

- **Logical OR (||)** The *output is true* only if *any of its input is true*. It accepts two or more inputs.

Input		Output
X	Y	X || Y
0	0	0
0	1	1
1	0	1
1	1	1

Unary Operators

- ➢Unary Minus Operator

Example: int a, b =100; a=-(b); **Result**: a=-10

- ➢Increment (++)
  - ✓Pre increment:

  Example: If a=8, b= ++a what will be the value of a and b?

  - ✓Post increment

  Example: If q=6, p= q++ what will be the value of p and q?

- ➢Decrement (--)
  - ✓Pre decrement **Example:** If y=4, x= --y what will be the value of x and y?
  - ✓Post decrement **Example:** If n=5, m= n-- what will be the value of m and n?

Conditional Operator

It takes three arguments

Expression1 ? Expression2 : Expression3

Where,

Expression1→ Condition

Expression2→ Statement followed if condition is true

Expression3→ Statement followed if condition is false

Example:

large = (4 > 2) ? 4: 2 →large = 4

Bitwise Operators

These works on bits and performs bit by bit operations.

The different types of bitwise operators are:

- Bitwise NOT (**~**)
- Bitwise AND (**&**)
- Bitwise OR (**|**)
- Bitwise XOR (**^**)
- Bitwise left shift (**<<**)
- Bitwise right shift (**>>**)

Bitwise Operators

Bitwise NOT (~) X	~X
0	1
1	0

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	0

X	0	1	0	0	0	1	1	0
X<<2	0	0	0	1	1	0	0	0

X	0	1	0	0	0	1	1	0
X>>2	0	0	0	1	0	0	0	1

Assignment Operators

The assignment operator is used to assign the values to the variables on the left hand side. The symbol “**=**” is used as an assignment operator.

x = 10, **c = a+b**

Shorthand Assignment:

x = x+2 → **x+=2**

Multiple Assignment

a=10,b=10,c=10 → **a=b=c=10**

Comma Operators & sizeof operators

- **Comma Operator:** It can be used as operator in expression and as separator in declaring variables.
- **sizeof() operator:** It is used to determine the size of variable or value in bytes.

Type Conversion and Typecasting

Type conversion is done when the expression has variables of different data types.

The data type is promoted from lower to higher level where the hierarchy of data types are

Short/char→ int→ unsigned int→ long int→ unsigned long int→ float →double→ long double

Type Conversion and Typecasting

- Typecasting is also known as forced conversion.
- Typecasting an arithmetic expression tells the compiler to represent the value of the expression.
- Higher data type has to be converted into the value of a lower data type.

Example:

Float salary=10000.00;

Int sal;

Sal=(int) salary;

Output:

10000