SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

COURSE NAME : 23ITT101- PROBLEM SOLVING & C PROGRAMMING

I YEAR /I SEMESTER

Unit II – C PROGRAMMING BASICS

Topic : Operators

Topics Covered

- Operators
  - Introduction
  - Definitions
  - Types of Operators

Introduction

- Operators are special symbols that perform operations on variables and values.
- Operators are symbols which take one or more operands or expressions and perform arithmetic or logical computations.
- Operands are variables or expressions which are used in conjunction with operators to evaluate the expression.

- The number of operands of an operator is called its **arity**.
- Based on arity, operators are classified as

    nullary (no operands)

    unary (1 operand),

    binary (2 operands)

    ternary (3 operands).

Types of Operators

- ➢ Arithmetic operators
- ➢ Relational operators
- ➢ Logical operators
- ➢ Assignment operators
- ➢ Increment and decrement operators
- ➢ Conditional operators
- ➢ Bitwise operators
- ➢ Special operator – Comma operator and sizeof operator

ARITHMETIC OPERATORS

- Arithmetic operators are used to perform numerical calculations among the values.

OPERATOR	MEANING
+	Addition
-	Subtraction
*	Multiplication
/	Division
%	Modulo Division

- Arithmetic Operators: These are the operators used to perform arithmetic/mathematical operations on operands.

  Examples: (+, -, *, /, %,++,–).

  Arithmetic operator are of two types:

- Unary Operators: Operators that operates or works with a single operand are unary operators. For example: (++ , –)
- Binary Operators: Operators that operates or works with two operands are binary operators. For example: (+ , – , * , /)

Example:

```
int a = 10, b = 3;

int sum = a + b;        // 13
int diff = a - b;       // 7
int prod = a * b;       // 30
int quot = a / b;       // 3
int mod = a % b;        // 1
```

RELATIONAL OPERATOR

- Relational Operators are used to compare two quantities and take certain decision depending on their relation.
- If the specified relation is true it returns one. If the specified relation is false it returns zero.

OPERATOR	MEANING
<	Is less than
<=	Is less than or equal to
>	Is greater than
>=	Is greater than or equal to
==	Is equal to
!=	Is not equal to

Example:

```
if (a > b) {
    // True if a is greater than b
}
```

LOGICAL OPERATORS

These operators are used for testing more than one condition and making decisions. 'c' has three logical operators they are:

OPERATOR	MEANING	
&&	Logical AND	Returns true if both operands are true.
||	Logical OR	Returns true if at least one operand is true.
!	Logical NOT	Reverses the boolean value.

Example:

```
if (a > 5 && b < 5) {
   // True if a is greater than 5 AND b is less than 5
}
```

ASSIGNMENT OPERATORS

These operators are used for assigning the result of an expression to a variable.

OPERATOR	MEANING
=	Simple Assignment
+=	Add and Assign
-=	Subtract and Assign
*=	Multiply and Assign
/=	Divide and Assign
%=	Modulus and Assign

Example:

```
int z = 10;
z += 5; // z is now 15
```

INCREMENT & DECREMENT OPERATORS

- Two most useful operators which are present in 'c' are increment and decrement operators.
- Operators: ++ and --
- ++ adds one to the operand
- -- subtracts one from the operand.
- Both are unary operators and can be used as pre or post increment/decrement.

Example:

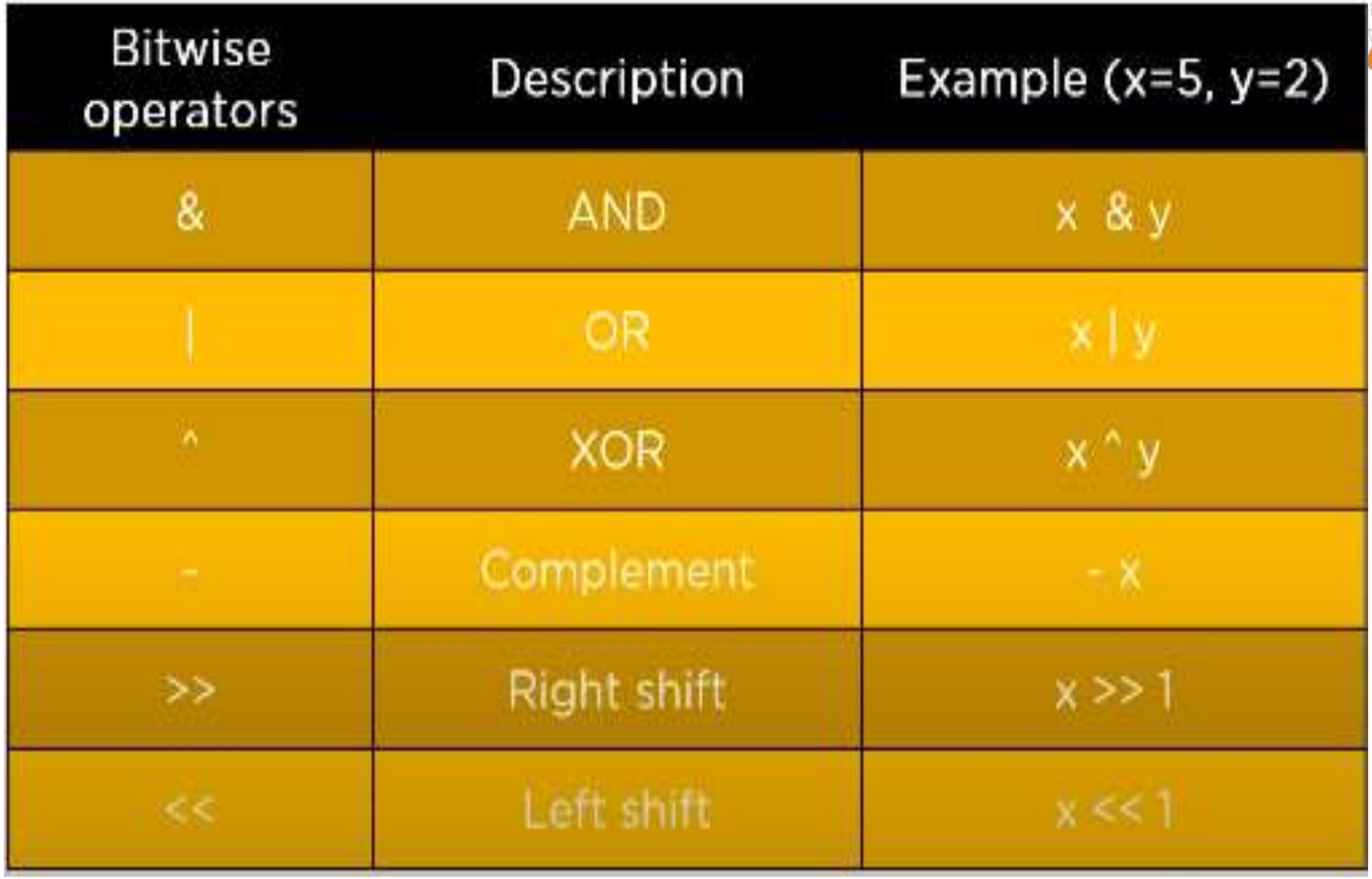
```
int count = 10;

count++;        // count is now 11
count--;        // count is now 10
```

BITWISE OPERATORS

These operators works on bit level. Applied to Integers only

OPERATOR	MEANING
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
<<	Shift Left
>>	Shift Right
^	Bitwise Exclusive OR



Bitwise operators	Description	Example (x=5, y=2)
&	AND	x & y
|	OR	x | y
^	XOR	x ^ y
-	Complement	- x
>>	Right shift	x >> 1
<<	Left shift	x << 1

a	b	a & b	a | b	a ^ b
0	0	0	0	1
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1

- ➢The & (bitwise AND) in C takes two numbers as operands and does AND on every bit of two numbers. The result of AND is 1 only if both bits are 1.
- ➢The | (bitwise OR) in C takes two numbers as operands and does OR on every bit of two numbers. The result of OR is 1 if any of the two bits is 1.
- ➢The ^ (bitwise XOR) in C takes two numbers as operands and does XOR on every bit of two numbers. The result of XOR is 1 if the two bits are different.

- The << (left shift) in C takes two numbers, left shifts the bits of the first operand, the second operand decides the number of places to shift.
- The >> (right shift) in C takes two numbers, right shifts the bits of the first operand, the second operand decides the number of places to shift.
- The **left-shift** and right-shift operators are equivalent to **multiplication** and division by 2 respectively
- The ~ (bitwise NOT) in C takes one number and inverts all bits of it

```
#include <stdio.h>
int main()
{
    // a = 5(00000101), b = 9(00001001)
    unsigned char a = 5, b = 9;
      // The result is 00000001

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

      // The result is 00001101
    printf("a|b = %d\n", a | b);
// The result is 00001100
    printf("a^b = %d\n", a ^ b);
 // The result is 11111010
    printf("~a = %d\n", a = ~a);
// The result is 00010010
    printf("b<<1 = %d\n", b << 1);
// The result is 00000100
    printf("b>>1 = %d\n", b >> 1);
      return 0;
}
```

Output

```
a = 5, b = 9
a&b = 1
a|b = 13
a^b = 12
~a = 250
b<<1 = 18
b>>1 = 4
```

CONDITIONAL OPERATORS ?:

Conditional operator or ternary operator are used to construct conditional expressions of the form.

Syntax:

condition ? expression1 : expression2;

Example:

int max = (a > b) ? a : b;

// max will be a if a is greater than b, otherwise b

SPECIAL OPERATORS

'C' supports some special operators such as comma operator, sizeof operator and pointer operators.

Comma operator:

- Comma operator is used to combine related expressions.
- A comma linked list of expressions are evaluated left to right and the value of right most expression is the value of combined expression..

Example: value=(x=10, y=5, x+y);

Sizeof Operator:

- It is a compile time unary operator which can be used to compute the size of its operand.
- The result of sizeof is of unsigned integral type which is usually denoted by size_t.
- Basically, sizeof operator is used to compute the size of the variable.
- Sizeof is an operator used to return the number of bytes the operand occupies.

Syntax:

```
m=sizeof(sum);  k=sizeof(2351);
```

Summary

- An operator is **a symbol which operates on a variable or value.** There are types of operators like arithmetic, logical, conditional, relational, bitwise, assignment operators etc. Some special types of operators are also present in C like sizeof(), Pointer operator, Reference operator etc.

Thank you