Programming Fundamentals (CT-175)
Lab 03

Unary and Binary Operators, Operators Precedence, and Associativity

Objectives

The objective of this lab is to familiarize students with different types of operators supported by C, operator precedence, and association. By the end of this lab students will be able to write simple sequential programs by using different types of operators.

Tools Required

DevC++ IDE

Course Coordinator –
Course Instructor –
Lab Instructor –
Prepared By Department of Computer Science and Information Technology
NED University of Engineering and Technology

Operators in C

An operator is a character or symbol that represents a specific mathematical, relational, or logical operation. There are many operators in C language which are mainly classified into three categories, that is, Unary, Binary, and Ternary.

Unary Operators

Unary operators take only one operand as input. The details of different unary operators used in C language are given in the following Table.

Operator	Description	Example
+	Unary plus. Changes the sign of an operand to positive	+A
–	Unary minus. Changes the sign of an operand to negative	-A
&	Address of operator. Returns the memory address of a variable	&A
!	Not logical operator. Reverse logical state of a binary operand.	!A
~	Bitwise Complement Operator. It is a unary operator that has an effect to 'flip' the bits. Meaning, all the 0s become 1s and vice-versa.	~A
sizeof( )	Returns the number of bytes occupied by a variable in memory	sizeof(A)
++	Increment operator. Increments value of an operand by unity	A++/++A
--	Decrement operator. Decrements value of an operand by unity	A-- / --A
,	Comma operator. Used to link similar expressions together.	int a, b, c;

Binary Operators

Binary operators take two operands as input. A number of binary operators are supported by C language as described below.

Arithmetic Operators

These operators are used for performing mathematical operations. The details are given in the following Table.

Operator	Description	Example
+	Adds two operands.	10+20 = 30
–	Subtracts second operand from the first.	10–20 = -10
*	Multiplies both operands.	10*20 = 200
/	Divides numerator by denominator.	20/10 = 2
%	Modulus Operator and remainder of after an integer division.	20%10 = 0

Relational Operators

A relational operator checks the relationship between two operands. The operands are preferred to be whole numbers. If the relation is true, it returns 1; if the relation is false, it returns value 0. It means the output of applying relational operator is always binary. Relational operators are used in conditional and iterative structures.

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

Logical Operators

Logical operators are used only when both the operands are binary. An expression containing a logical operator also returns binary output, that is, either 0 or 1. Logical operators are commonly used in decision making in C programming.

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

Bitwise Operators

Bitwise operators are used in C programming to perform bit-level operations. During computation, mathematical operations like: addition, subtraction, multiplication, division, etc are converted to bit-level which makes processing faster and saves power. Bitwise operator can be applied only on integers. The result of applying binary operators is also an integer.

Operators	Meaning of operators	Example – let P=60, Q=13
&	Bitwise AND. It copies a bit to the result when it exists in both the operands.	(P & Q) = 12, which is, 0000 1100
|	Bitwise OR. It copies a bit when it exists in either of the operands.	(P | Q) = 61, which is, 0011 1101
^	Bitwise exclusive OR. It copies the bit when it is set in one of the operands, but not in both.	(P | Q) = 61, which is, 0011 1101
>>	Shift right. It moves the value of the left operand to the right by the number of bits that the right operand specifies.	P >> 2 = 15 which is, 0000 1111
<<	Shift left. It moves the value of the right operand to the left by the number of bits that the right operand specifies.	P << 2 = 240 which is, 1111 0000

Assignment Operators

An assignment operator is the operator used to assign a new value to a variable. Following Table provides details of all the assignment operators supported by C language.

Operator	Description	Equivalent	Example
=	Basic assignment. p becomes equal to q	N/A	p = q
+=	Addition assignment .The addition of p and q becomes equal to p	p = p + q	p += q
-=	Subtraction assignment .The subtraction of q from p becomes equal to p	p = p – q	p -= q
*=	Multiplication assignment. The product of p and q becomes equal to p	p = p * q	p *= q
/=	Division assignment. The division of p by q becomes equal to p	p = p / q	p /= q
%=	Modulo assignment. The remainder of p divided by q becomes equal to p	p = p % q	p %= q
&=	Bitwise AND assignment. The bitwise AND of p and q becomes equal to p	p = p & q	p &= q
|=	Bitwise OR assignment. The bitwise OR of p and q becomes equal to p	p = p | q	p |= q
^=	Bitwise XOR assignment. The bitwise XOR of p and q becomes equal to p	p = p ^ q	p ^= q

<<=	Bitwise left shift assignment. p left shifted by q becomes equal to p	p = p << q	p <<= q
>>=	Bitwise right shift assignment. p right shifted by q becomes equal to p	p = p >> q	p >>= q

Ternary Operators

Ternary operator takes three operands as input (?:). There is only one ternary operator in C language which will be discussed in the next lab.

Operators Precedence in C

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. We can alter the default precedence of operators by making use of parenthesis ( ). Parenthesis has higher precedence level. For example, x=(7+3)*2, in this expression 7+3 will be evaluated first and then the result will be multiplied with 2.

Operators Associativity in C

The associativity of operators determines the direction in which an expression is evaluated. For example, consider the expression b=a. Here, the value of "a" is assigned to "b", and not the other way around. It's because the associativity of the = operator is from right to left. Also, if two operators of the same precedence (priority) are present, associativity determines the direction in which they execute. Consider an example 1 == 2 != 3. Here, operators == and != have the same precedence. And, their associativity is from left to right. Hence, 1 == 2 is executed first.

In the following Table all the operators are listed along with their associativity. The operators with the highest precedence appear at the top of the Table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

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
Ternary	?:	Right to left
Assignment	=, +=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=	Right to left
Comma	,	Left to right

Exercise

1. Write a program that asks the user to enter two numbers, obtains them from the user and prints their sum, product, difference, quotient and remainder.
2. State the order of evaluation of the operators in each of the following C statements and show the value of x after each statement is performed.

a) x = 7 + 3 * 6 / 2 - 1;
b) x = 2 % 2 + 2 * 2 - 2 / 2;
c) x = (3 * 9 * (3 + (9 * 3 / (3))));

3. Write c program that find the result of the following operations: you are allowed to initialize any values. Observe and write in comment how that operation produce results.

a) 5 + 4
b) 10/2
c) True OR False
d) 20 MOD 3
e) 5< 8
f) 25 MOD 70
g) "A" > "H"
h) NOT True
i) 25/70
j) False AND True
k) 20 * 0.5
l) 35 <= 35

4. Write C program to create a BMI calculator application that reads the user's weight in pounds and height in inches (or, if you prefer, the user's weight in kilograms and height in meters), then calculates and displays the user's body mass index.
5. Write a program for swapping two numbers with and without using a third variable. Ask user to enter the two numbers.

Lab 03 Evaluation		
Student Name:	**Student ID:**	**Date:**
Task No.	**Marks**	**Remarks by teacher in accordance with the rubrics**
1		
2		
3		
4		
5		