|  |  |  |
| --- | --- | --- |
| **CL-1002 Programming Fundamentals** | **LAB - 04**  **Introduction of operators and math.h library functions** | |
| **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**  **Fall 2023**  **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**  **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES**  **NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES** | |  |

OPERATORS:

**C Arithmetic Operators**

There are many operators in C for manipulating data which include arithmetic Operators, Relational Operators, Logical operators and many more which will be discussed   
accordingly. Some of the fundamental operators are:

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| + | Adds two operands. | A + B = 30 |
| − | Subtracts second operand from the first. | A − B = -10 |
| \* | Multiplies both operands. | A \* B = 200 |
| / | Divides 1st operand by 2nd operand. | B / A = 2 |
| % | Modulus Operator and remainder of after an integer division. | B % A = 0 |
| ++ | Increment operator increases the integer value by one. | A++ = 11 |
| -- | Decrement operator decreases the integer value by one. | A-- = 9 |

**C Relational Operators**

A relational operator checks the relationship between two operands. If the relation is true, it returns 1 (true); if the relation is false, it returns value 0 (false).

Relational operators are used in decision making and loops.

|  |  |  |
| --- | --- | --- |
| Operator | Meaning of Operator | Example |
| == | Is Equal to | 5 == 3 returns 0 |
| > | Is Greater than | 5 > 3 returns 1 |
| < | Is Less than | 5 < 3 returns 0 |
| != | Is Not equal to | 5 != 3 returns 1 |
| >= | Is Greater than or equal to | 5 >= 3 returns 1 |
| <= | Is Less than or equal to | 5 <= 3 returns 0 |

**C Logical Operators**

An expression containing a logical operator returns either 0 (false) or 1 (true) depending upon whether the expression results true or false. Logical operators are commonly used in decision making in C programming.

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

**Bitwise Operators C**

The computer runs on binary, thus on the lowest level all operations are performed on bits. In C language, you can perform these operations using bitwise operators.

|  |  |
| --- | --- |
| Operator | Meaning of operators |
| & | Bitwise AND |
| | | Bitwise OR |
| ^ | Bitwise exclusive OR |
| ~ | Bitwise complement |
| << | Shift left |
| >> | Shift right |

**Comma Operator**

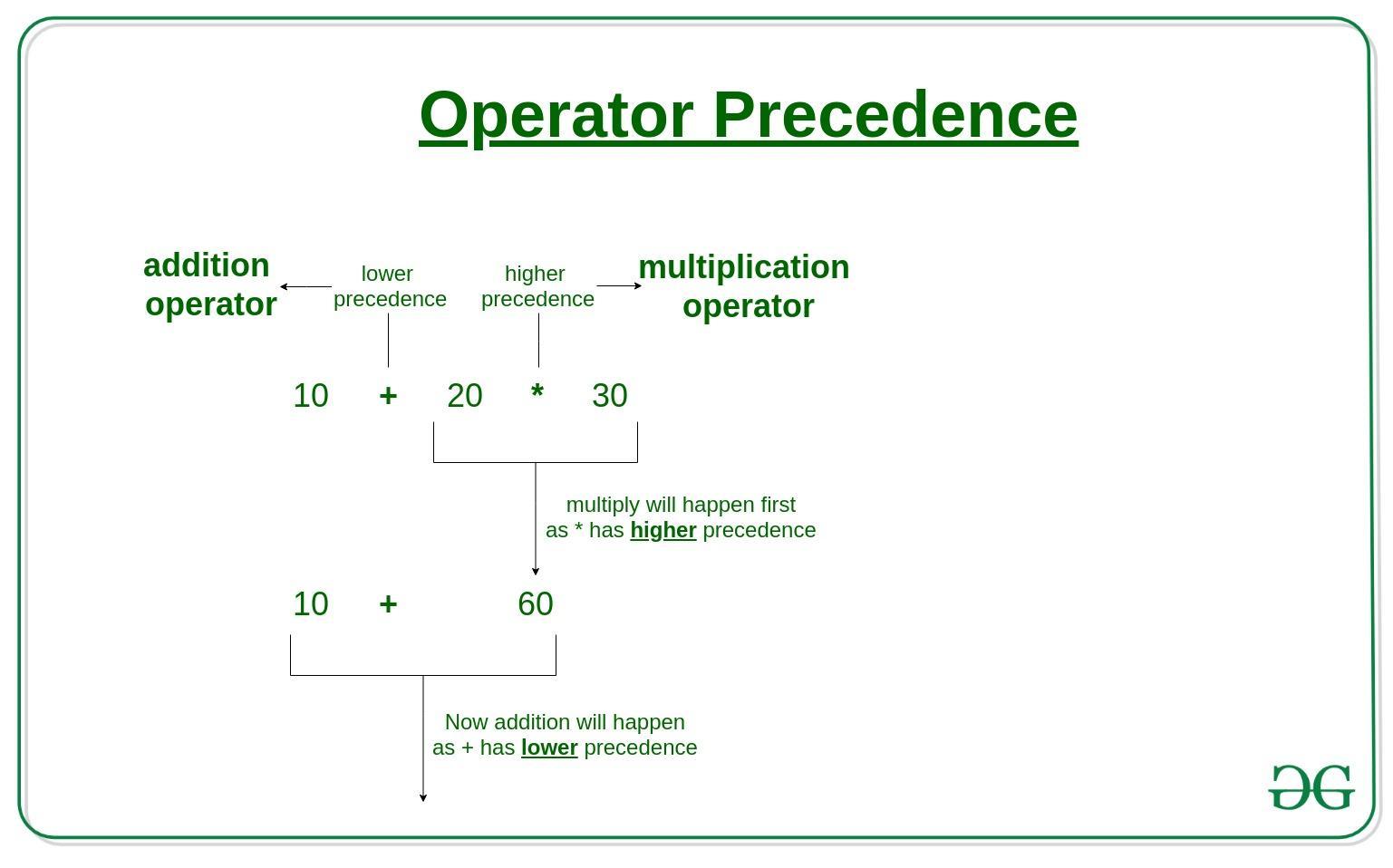
Comma operators are used to link related expressions together. For example:

**int a, c = 5, d;**

**The sizeof operator**

The sizeof is a unary operator that returns the size of data (constants, variables, array, structure, etc).

**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.

In the table below, 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 |
| Conditional | ?: | Right to left |
| Assignment | = += -= \*= /= %=>>= <<= &= ^= |= | Right to left |
| Comma | , | Left to right |

**Math library functions**

*#include <math.h>*

Math library functions allow you to perform certain common mathematical calculations.

