Universiti Malaysia PAHANG
Engineering • Technology • Creativity

BTE2313

Chapter 5: Arithmetic & Logical expression

by
Sulastri Abdul Manap
Faculty of Engineering Technology
sulastri@ump.edu.my

Objectives

- In this chapter, you will learn about:
  1) Learn about arithmetic operators to perform calculations in C++
  2) Discover how a program evaluates arithmetic expressions
  3) Explain how to solve and create a logical expression
  4) Explain how to use operator AND (&&) and OR (||)

Arithmetic Operators

Symbol	Meaning	Example	Answer
=	Assign	a=5	5 is assigned to a, now a is equal to 5
+	Add	6+9	15
-	Subtract	9-6	3
*	Multiply	6*9	54
/	Divide	9/6	1 (for int) 1.5 (for float)
%	Modulus	9%6	3

Arithmetic Operations

- **Arithmetic Expressions** are made of *operands* and *operators*
- *Operands* are actually the values used in the operations
- C++ arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division
  - % modulus (or remainder) operator
- +, -, *, and / can be the operators for integer and floating-point numbers
- % operator only can be used for integer types

Arithmetic Operations: Division

- When division operations is implemented on integer operands, the result will be an integer. The fractional portion of the result is thrown away.
- Examples : 14 / 5 = 2

  5 / 2 = 2

  25 / 8 = 3
- If the division has at least one operand is a floating point number, the result will be in floating point.
- Examples : 14.0 / 5 = 2.8

  5.0 / 2 = 2.5

  25 / 8.0 = 3.125

  In above examples, the integers are temporarily transformed into a floating numbers, then the division is done.

Arithmetic Operations: Modulus

- The operation of **x % y** resulting the remainder after **x** is divided by **y**.
- It is an operation that requires **both** operands are integers
- Examples : 12 % 7 = 5

  4 % 2 = 0

  90 % 8 = 2

  3 % 4 = 3
- This operations can be used to decide whether an integer number is even or odd, as example 7 % 2 = 1 is odd while

  12 % 2 = 0 even.

Order of Precedence

- The basic of precedence in programming is similar to mathematical precedence.
- The highest priority is (), followed by *, /, and % (multiplication, division and modulus) which are at the same level of precedence
- Addition and subtraction is next to be evaluated, and both have the same level of precedence
- When there are operators have the same level of precedence, operations will be performed from left to right
- `3 * 7 - 6 + 2 * 5 / 4 + 6` means

  `(((3 * 7) - 6) + ((2 * 5) / 4 )) + 6`

- If you have more than two operations, use parenthesis to avoid confusion on their precedence.

Order of Precedence (cont.)

Priority	Operator Type	Operator(s)	Associativity
Highest	Primary	`() [] . ->`	Left to right
	Unary	`++ -- & * !`	Right to left
	Arithmetic	`* / %`	Left to right
	Arithmetic	`+ -`	Left to right
	Relational	`< <= > >=`	Left to right
	Relational	`== !=`	Left to right
	Logical	`&&`	Left to right
	Logical	`||`	Left to right
Lowest	Assignment	`=`	Right to left

Example: Order of Precedence

Example 1

$5 + 8 < 14 - 2 \,||\, 6 > 3$

The + and − have highest precedence.
The + goes first (it's on the left).

$13 < 14 - 2 \,||\, 6 > 3$

Now the −

$13 < 12 \,||\, 6 > 3$

Next the <

$0 \,||\, 6 > 3$

Now the >

$0 \,||\, 1$

Last, the OR operator

1

Example 2

$6 + 7 >= 12 \,\&\&\, (3+4) > 2 * 4$

The ( ) is primary and will be performed first. 3 and 4 are added.

$6 + 7 >= 12 \,\&\&\, 7 > 2 * 4$

Multiplication now has the highest precedence.

$6 + 7 >= 12 \,\&\&\, 7 > 8$

The addition is performed next.

$13 >= 12 \,\&\&\, 7 > 8$

Now the >=

And then the >

$1 \,\&\&\, 0$

Last, but not least, the AND

0

Logical Relationship

- To make any decision, conditional expressions are needed so that we can make comparison.
- A conditional expressions in C++ is made up of logical (Boolean) expressions, which will produce Boolean values (TRUE or FALSE)
- RELATIONAL OPERATORS: allows you to make comparison in a program
- Relational operators:
  - → Allow comparison
  - → Need two operands
  - → Resulting to Boolean values

Relational operators in C++

Operator	Description	Example	True when value of num is	False when value of num is
==	Equal to	num==10	10	Other than 10
!=	Not equal to	num!=10	Other than 10	10
>	Bigger than	num>10	Bigger than 10	10 or smaller
>=	Bigger than or equal	num>=10	10 or bigger	Smaller than 10
<	Less than	num<10	Smaller than 10	10 or bigger
<=	Less than or equal	num<=10	10 or smaller	Bigger than 10
!	Not	!(num==10)	Other than 10	10

Logical Operators

- Logical or Boolean operators allows programmers to combine and compare logical expressions when there are two or more relationships.

Symbol	Meaning	Description
!	NOT	The opposite value
&&	AND	Both values must be true for the value itself to be true
||	OR	Either one value is true makes both true

Truth table for AND (&&) OR (||)

- Operator && and || are used in the expression when several logical conditions to be checked.
- The expression's value is determined using the truth table below.

Operator	LE1	LE2	LE1 && LE2
&&	0 0 1 1	0 1 0 1	0 0 0 1

Operator	LE1	LE2	LE1 || LE2
||	0 0 1 1	0 1 0 1	0 1 1 1

Examples: Logical Expressions

- `(Salary <3000) && (child > 3)`
- `(CGPA > 3.5) || (merit_activity>3500)`
- To check whether a price is bigger than RM 100.00 or code discount = 'R'

  → `(price>100.00)||(codeDisc == 'R')`
- To check whether a mark is bigger than 50 and lower than 60

  → `(mark >50) && (mark <60)`
- To test whether a number is an even number

  → `num%2 ==0`

Exercises

1. Convert a temperature from degrees Fahrenheit to degrees Celsius using the formula

$$°C = \frac{5}{9} \times (°F - 32)$$

2. Given number of seconds, convert them to hours, minutes, and seconds.