

C++ Programming - Expressions

Outline

- Arithmetic expressions
- Relational expressions (conditions)
- Logical expressions (decisions)
- Increment operators
- Bitwise operators

Expressions

- An **expression** is a sequence of **operands** (constants or variables) and **operators**
 - e.g., $a * b - c$, $(m + n) * (x + y)$, $2 * 3 + 4$
- An expression is evaluated from left to right using the rule of **precedence** of operators
 - Precedence
 - Highest priority: **()**
 - High priority: *****, **/**, **%**
 - Low priority: **+**, **-**
 - What are the results of **x** and **y**
 - $x = 9 - 12 / 3 + 3 * 2 - 1 ;$
 - $y = 9 - 10 / (3 + 3) * (2 - 1) ;$

Arithmetic Expressions

- Arithmetic operators
 - Binary operators
 - `+, -, *`: for all integer and float
 - e.g., `a+b`, `10-4`, `-5`, `2.0*10`
 - `/`:
 - integer: give the `int` quotient. E.g., `10 / 3 = 3`
 - Float: give the `float` quotient.
 - e.g., `10.0/3 = 3.333333`
 - `%`(modulus): only for integers, give the remainder
 - e.g., `5%3 = 2`

Relational Expressions (Conditions)

- Expressions with relational operators
 - `<`, `<=`, `>`, `>=`, `==`, `!=`
- Compare variables and values
- The result of a relational expression is either `0` (`false`) or `1` (`true`)
 - `4 < 5: true`
 - `4 > 5: false`
 - `4 != 5: true.`
- `x > 10: unknown`, depending on the value of `x`.

Relational Expressions

Operator	Description	Example
>	greater than	5 > 4
>=	greater than or equal to	mark >= score
<	less than	height < 75
<=	less than or equal to	height <= input
==	equal to	score == mark
!=	not equal to	5 != 4

Logical Expressions (Decisions)

- Comprise relational expressions (conditions) and logical operators
 - Logical operators
 - `&&` (two ampersands): means *and*.
 - e.g., `(GPA >= 1.7 && GPA <= 1.99)`
 - `||` (two vertical bars): means *or*.
 - e.g., `(spellingErrors > 5 || grammarErrors > 3)`
 - `!` (an exclamation point): means *not*.
 - e.g., `!(score < 80)`
- Allows you to give more than one condition
- A logical expression is also called a **decision**

Logical Expressions

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true, then Logical NOT operator will make false.	!(A && B) is true.

Class Exercises

- Translate the following English questions into C/C++ decisions.
 - The height is not equal to zero
 - The temperature (variable: temp) is greater than 32 . 0 and less than 212 . 0
 - The absolute value of pos is greater than 5 . 0

Increment and decrement expressions

- **++** (increment), **--** (decrement)
 - Has only one operand. Only applicable to integer
 - e.g.,
 - » **x++** is equivalent to **x=x+1**
 - » **++x** is equivalent to **x=x+1**
 - » **x--** is equivalent to **x=x-1**
 - » **--x** is equivalent to **x=x-1**

sizeof

This operator accepts one parameter, which can be either a type or a variable, and returns the size in bytes of that type or object:

```
x = sizeof (char)
```

x is assigned the value 1, because char is a type with a size of one byte.

Rules for ++ and --

- **postfix** ++ (or --)
 - the **expression is evaluated first** using the original value of the variable
 - then the variable is incremented (**or decremented**) by one
- **prefix** ++ (or --)
 - the **variable is incremented (or decremented) by one first**
 - then the expression is evaluated using the new value of the variable

Class Exercises

```
m    =    5;  
y    =    ++m;  
x    =    m++;
```

What are the final values of `x`, `y`, and `m`?

Conditional ternary expression

- Format
 - `exp1 ? exp2 : exp3`
 - If `exp1` **is true**,
 - The result of `exp1 ? exp2 : exp3` is `exp2` (`exp3` is not evaluated)
 - If `exp1` **is false**,
 - The result of `exp1 ? exp2 : exp3` is `exp3` (`exp2` is not evaluated)

Examples

Example 1

```
x = 4;  
y = 5;  
z = (x > y) ? x : y;
```

Example 2

```
x = 5;  
y = 4;  
z = (x > y) ? x : y;
```

The value of **z** is ?

Bitwise Operators

- Work on binary system of all integer types

• Operator	Meaning
&	bitwise AND
	bitwise OR
^	bitwise Exclusive OR
~	bitwise complement
<<	shift left
>>	shift right

Example

$a = 5$ (00000101)

$b = 9$ (00001001)

$a = 5, b = 9$

$a \& b = 1$

$a | b = 13$

$a \wedge b = 12$

$\sim a = 250$

$b \ll 1 = 18$

$b \gg 1 = 4$

Shift, Multiplication and Division

- $14: 0000\ 1110\ (2^3+2^2+2^1)$
- $14 \ll 1$ (shift one bit left: $0001\ 1100$) $(2^4+2^3+2^2=28)$
- $14 \gg 1$ (shift one bit right: $0000\ 0111$) $(2^2+2^1+2^0=7)$

Shift, Multiplication and Division

- Multiplication and division are often slower than shift.
- Multiplying 2 can be replaced by shifting 1 bit to the left.

```
n = 10;  
m = n << 1;  
cout << n*2 << ' = ' << m;  
m = n << 2;  
cout << n*4 << ' = ' << m;
```

- Division by 2 can be replaced by shifting 1 bit to the right.

```
n = 10  
m = n >> 1;  
cout << n/2 << ' = ' << m;
```

Comma Operator

- An expression can be composed of multiple sub-expressions separated by commas.
 - Sub-expressions are evaluated left to right.
 - The entire expression evaluates to the value of the rightmost sub-expression.

An Example

```
x = (a++, b++);
```

Evaluation steps:

1. **a** is incremented
2. **b** is assigned to **x**
3. **b** is incremented

What if the parentheses are missing?

Operator Precedence

	Operator	Precedence level
—	()	1
—	~, ++, --, unary -	2
—	*, /, %	3
—	+, -	4
—	<<, >>	5
—	<, <=, >, >=	6
—	==, !=	7
—	&	8
—	^	9
—		10
—	&&	11
—		12
—	=, +=, -=, etc.	14
—	,	15