Students:
Section 3.9 is a part of 1 assignment: **CSC108 CH03.1-3.10 P3A**

This assignment's due date has passed. Activity will still be recorded, but will not count towards this assignment (unless the due date is changed). See this article for more info.

Includes: PA
Due: 02/20/2025, 11:59 PM EST

3.9 Order of evaluation

Precedence rules

The order in which operators are evaluated in an expression are known as ***precedence rules***. Arithmetic, logical, and relational operators are evaluated in the order shown below.

Table 3.9.1: Precedence rules for arithmetic, logical, and relational operators.

Operator/Convention	Description	Explanation
( )	Items within parentheses are evaluated first	In `(a * (b + c)) - d`, the + is evaluated first, then *, then -.
!	! (logical NOT) is next	`! x || y` is evaluated as `(!x) || y`
* / % + -	Arithmetic operators (using their precedence rules; see earlier section)	`z - 45 * y < 53` evaluates * first, then -, then <.
< <= > >=	Relational operators	`x < 2 || x >= 10` is evaluated as `(x < 2) || (x >= 10)` because < and >= have precedence over ||.
== !=	Equality and inequality operators	`x == 0 && x != 10` is evaluated as `(x == 0) && (x != 10)` because == and != have precedence over &&. == and != have the same precedence and are evaluated left to right.
&&	Logical AND	`x == 5 || y == 10 && z != 10` is evaluated as `(x == 5) || ((y == 10) && (z != 10))` because && has precedence over ||.
||	Logical OR	|| has the lowest precedence of the listed arithmetic, logical, and relational operators.

Feedback?

PARTICIPATION ACTIVITY 3.9.1: Applying the precedence rules to an expression can be thought of as a 'tree'.

Start 2x speed

`x + 1 > y * z || z == 3`

Captions

1. Expressions like x + 1 > y * z || z == 3 are evaluated using precedence rules. Among +, >, *, ||, and ==, the * comes first.
2. Next comes +, then >, then ==, and finally ||.
3. The expression is actually treated like a 'tree', evaluated from the bottom upwards.
4. If x is 7, y is 6, and z is 3, then y * z is 18. Next, x + 1 is 8. Next, 8 > 18 is false. Next, z == 3 is true. Finally, false || true is true.

Feedback?

PARTICIPATION ACTIVITY 3.9.2: Order of evaluation.

To teach precedence rules, these questions intentionally omit parentheses; good style would use parentheses to make order of evaluation explicit.

1) Which operator is evaluated first?
`! y && x`
   - &&
   - !

2) Which operator has precedence?
`w + 3 > x - y * z`
   - +
   - -
   - >
   - *

3) In what order are the operators evaluated?
`w + 3 != y - 1 && x`
   - +, !=, -, &&
   - +, -, &&, !=
   - +, -, !=, &&

4) To what does this expression evaluate, given int x = 4, int y = 7.
`x == 3 || x + 1 > y`
   - true
   - false

Feedback?

Common error: Missing parentheses

A common error is to write an expression that is evaluated in a different order than expected. Good practice is to use parentheses in expressions to make the intended order of evaluation explicit. Several examples are below.

PARTICIPATION ACTIVITY 3.9.3: Common errors in expressions.

1) Does `! x == 3` evaluate as `! (x == 3)`?
   - Yes
   - No

2) Does `w + x == y + z` evaluate as `(w + x) == (y + z)`?
   - Yes
   - No

3) Does `w && x == y && z` evaluate as `(w && x) == (y && z)`?
   - Yes
   - No

Feedback?

PARTICIPATION ACTIVITY 3.9.4: Order of evaluation.

Which illustrates the actual order of evaluation via parentheses?

1) `! green == red`
   - (!green) == red
   - !(green == red)
   - (!green =)= red

2) `bats < birds || birds < insects`
   - ((bats < birds) || birds) < insects
   - bats < (birds || birds) < insects
   - (bats < birds) || (birds < insects)

3) `! (bats < birds) || (birds < insects)`
   - ! ((bats < birds) || (birds < insects))
   - (! (bats < birds)) || (birds < insects)
   - ((bats) < birds) || (birds < insects)

4) `(num1 == 9) || (num2 == 0) && (num3 == 0)`
   - (num1 == 9) || ((num2 == 0) && (num3 == 0))
   - ((num1 == 9) || (num2 == 0)) && (num3 == 0)
   - (num1 == 9) || (num2 == (0 && num3) == 0)

Feedback?

Common error: Math expression for range

A common error often made by new programmers is to write expressions like `16 < age < 25`, as one might see in mathematics.

The meaning, however, almost certainly is not what the programmer intended. Suppose age is presently 28. The expression is evaluated left-to-right, so evaluation of `16 < age` yields true. Next, the expression `true < 25` is evaluated; clearly not the programmer's intent. However, true is actually 1, and evaluating `1 < 25` will yield true. Thus, the above expression evaluates to true, even for ages greater than 25.

Thus, `16 < age < 25` is actually the same as `(16 < age) < 25`, which evaluates to `(true) < 25` for any age over 16, which is the same as `(1) < 25`, which evaluates to true. The correct way to do such a comparison is: `(age > 16) && (age < 25)`.

PARTICIPATION ACTIVITY 3.9.5: Expression for detecting a range.

1) A programmer erroneously wrote an expression as: 0 < x < 10.
Rewrite the expression using logical AND. Use parentheses.

`(0 < x)`

Check Show answer

Feedback?

Common error: Bitwise rather than logical operators

Logical AND is && and not just &, and logical OR is || and not just |. & and | represent ***bitwise operators***, which perform AND or OR on corresponding individual bits of the operands.

A common error is to use a bitwise operator instead of a logical operator, typing & instead of && or typing | instead of ||. A bitwise operator may yield different behavior than expected.

PARTICIPATION ACTIVITY 3.9.6: Bitwise vs. logical operators.

Indicate if the expression correctly uses logical operators.

1) (x > 5) & (y > 3) & (z != 0)
   - Yes
   - No

2) (x == 0) || (y == 0) | (z == 0)
   - Yes
   - No

3) ((x == y) && (y == z)) || (w == 0)
   - Yes
   - No

Feedback?

How was this section? | Provide section feedback

Activity summary for assignment: CSC108 CH03.1-3.10 P3A

Due: 02/20/2025, 11:59 PM EST

163 / 163 points

163 / 163 points submitted to BlackboardLearn

This assignment's due date has passed. Activity will still be recorded, but will not count towards this assignment (unless the due date is changed). See this article for more info.

Completion details