Students:
Section 2.4 is a part of 1 assignment: **CSC108 CH02.1-2.10 P2A**
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: 02/04/2025, 11:59 PM EST

2.4 Arithmetic expressions (general)

Basics

An ***expression*** is any individual item or combination of items, like variables, literals, operators, and parentheses, that evaluates to a value, like 2 * (x + 1). A common place where expressions are used is on the right side of an assignment statement, as in y = 2 * (x + 1).

A ***literal*** is a specific value in code like 2. An ***operator*** is a symbol that performs a built-in calculation, like +, which performs addition. Common programming operators are shown below.

Table 2.4.1: Arithmetic operators.

Arithmetic operator	Description
+	The **addition** operator is **+**, as in x + y.
-	The **subtraction** operator is **-**, as in x - y. Also, the - operator is for **negation**, as in -x + y, or x + -y.
*	The **multiplication** operator is *****, as in x * y.
/	The **division** operator is **/**, as in x / y.

Feedback?

PARTICIPATION ACTIVITY 2.4.1: Expressions.

Indicate which are valid expressions. x and y are variables, and are the only available variables.

1) `x + 1`
- Valid
- Not valid

2) `2 * (x - y)`
- Valid
- Not valid

3) `x`
- Valid
- Not valid

4) `2`
- Valid
- Not valid

5) `2x`
- Valid
- Not valid

6) `2 + (xy)`
- Valid
- Not valid

7) `x - -2`
- Valid
- Not valid

Feedback?

PARTICIPATION ACTIVITY 2.4.2: Capturing behavior with an expression.

Does the expression correctly capture the intended behavior?

1) 6 plus numItems:
```
6 + numItems
```
- Yes
- No

2) 6 times numItems:
```
6 x numItems
```
- Yes
- No

3) totDays divided by 12:
```
totDays / 12
```
- Yes
- No

4) 5 times i:
```
5i
```
- Yes
- No

5) The negative of userVal:
```
-userVal
```
- Yes
- No

6) n factorial
```
n!
```
- Yes
- No

Feedback?

Evaluation of expressions

An expression **evaluates** to a value, which replaces the expression. Ex: if x is 5, then x + 1 evaluates to 6, and y = x + 1 assigns y with 6.

An expression is evaluated using the order of standard mathematics, such order known in programming as ***precedence rules***, listed below.

Table 2.4.2: Precedence rules for arithmetic operators.

Operator/Convention	Description	Explanation
( )	Items within parentheses are evaluated first	In 2 * (x + 1), the x + 1 is evaluated first, with the result then multiplied by 2.
unary -	- used for negation (unary minus) is next	In 2 * -x, the -x is computed first, with the result then multiplied by 2.
*** / %**	Next to be evaluated are *, /, and %, having equal precedence.	(% is discussed elsewhere)
+ -	Finally come + and - with equal precedence.	In y = 3 + 2 * x, the 2 * x is evaluated first, with the result then added to 3, because * has higher precedence than +. Spacing doesn't matter: y = 3+2 * x would still evaluate 2 * x first.
left-to-right	If more than one operator of equal precedence could be evaluated, evaluation occurs left to right.	In y = x * 2 / 3, the x * 2 is first evaluated, with the result then divided by 3.

Feedback?

PARTICIPATION ACTIVITY 2.4.3: Evaluating expressions.

Captions

1. An expression like 3 * (x + 10 / w) evaluates to a value, using *precedence rules*. Items within parentheses come first, and / comes before +, yielding 3 * (x + 5).
2. Evaluation finishes inside the parentheses: 3 * (x + 5) becomes 3 * 9.
3. Thus, the original expression evaluates to 3 * 9 or 27. That value replaces the expression. So y = 3 * (x + 10 / w) becomes y = 27, so y is assigned with 27.
4. Many programmers prefer to use parentheses to make order of evaluation more clear when such order is not obvious.

Feedback?

PARTICIPATION ACTIVITY 2.4.4: Evaluating expressions and precedence rules.

Select the expression whose parentheses match the evaluation order of the original expression.

1) `y + 2 * z`
- (y + 2) * z
- y + (2 * z)

2) `z / 2 x`
- (z / 2) - x
- z / (2 - x)

3) `x * y * z`
- (x * y) * z
- x * (y * z)

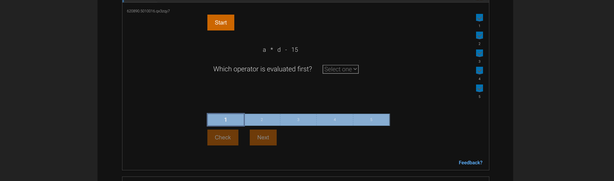
4) `x + 1 * y / 2`
- ((x + 1) * y) / 2
- x + ((1 * y) / 2)
- x+ (1 * (y / 2))

5) `x / 2 + y / 2`
- ((x / 2) + y) / 2
- (x / 2) + (y / 2)

6) What is totCount after executing the following?
```
numItems = 3;
totCount = 1 + i2 * numItems;  * 4;
```
- 44
- 41

Feedback?

CHALLENGE ACTIVITY 2.4.1: Precedence rules for arithmetic operators.



Feedback?

Using parentheses to make the order of evaluation explicit

*A common error is to omit parentheses and assume a different order of evaluation than actually occurs, leading to a bug. Ex: if x is 3, then 5 * x+1 might appear to evaluate as 5 * (3+1) or 20, but actually evaluates as (5 * 3) + 1 or 16 (spacing doesn't matter). Good practice is to use parentheses to make order of evaluation explicit, rather than relying on precedence rules, as in: y = (m * x) + b, unless order doesn't matter as in x + y + z.*

Example: Calorie expenditure

A website lists the calories expended by men and women during exercise as follows (source):

Men: Calories = [(Age × 0.2017) + (Weight × 0.09036) + (Heart Rate × 0.6309) – 55.0969] × Time / 4.184

Women: Calories = [(Age × 0.074) – (Weight × 0.05741) + (Heart Rate × 0.4472) – 20.4022] × Time / 4.184

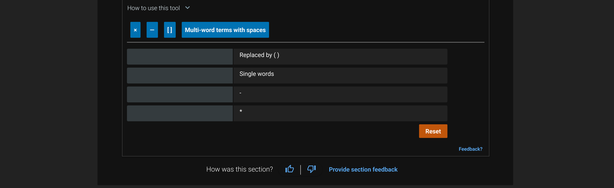
Below are those expressions written using programming notation:

```
caloriesMan = ( (ageYears * 0.2017) + (weightPounds * 0.09036) + (heartBPM * 0.6309) - 55.0969 ) * timeMinutes / 4.184
caloriesWoman = ( (ageYears * 0.074) - (weightPounds * 0.05741) + (heartBPM * 0.4472) - 20.4022 ) * timeMinutes / 4.184
```

PARTICIPATION ACTIVITY 2.4.5: Converting a formatted expression to a program expression.

Consider the example above. Match the changes that were made.

How to use this tool



Feedback?

How was this section? | Provide section feedback

Activity summary for assignment: CSC108 CH02.1-2.10 P2A
Due: 02/04/2025, 11:59 PM EST
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.

136 / 136 points
136 / 136 points submitted to Blackboard.Learn

Completion details