2.5 Arithmetic expressions (int)

Below is a simple program that includes an expression involving integers.

Figure 2.5.1: Expressions examples: Leasing cost.

```
#include <iostream>
using namespace std;

/* Computes the total cost of leasing a car given the down payment,
   monthly rate, and number of months
*/

int main() {
   int downPayment;
   int paymentPerMonth;
   int numMonths;
   int totalCost;  // Computed total cost to be output

   cout << "Enter down payment: ";
   cin  >> downPayment;

   cout << "Enter monthly payment: ";
   cin  >> paymentPerMonth;

   cout << "Enter number of months: ";
   cin  >> numMonths;

   totalCost = downPayment + (paymentPerMonth * numMonths);

   cout << "Total cost: " << totalCost << endl;

   return 0;
}
```

```
Enter down payment: 500
Enter monthly payment: 300
Enter number of months: 60
Total cost: 18500
```

Feedback?

PARTICIPATION ACTIVITY 2.5.1: Simple program with an arithmetic expression.

Consider the example above.

1) Would removing the parentheses as below have yielded the same result?

```
downPayment +
paymentPerMonth *
numMonths
```

Correct

The * would still be evaluated first, because * has higher precedence than +. But using parentheses makes the programmer's intent more clear to people reading the program.

- (•) Yes
- ( ) No

2) Would using two assignment statements as below have yielded the same result? Assume this declaration exists: int totalMonthly

```
totalMonthly =
paymentPerMonth *
numMonths;
totalCost = downPayment +
totalMonthly;
```

- (•) Yes
- ( ) No

Correct

The evaluation is ultimately the same, yielding the same result. Breaking larger expressions into multiple assignments with smaller expressions can sometimes improve code readability.

Feedback?

Style: Single space around operators

A good practice is to include a single space around operators for readability, as in numItems + 2, rather than numItems+2. An exception is minus used as negative, as in: xCoord = -yCoord. Minus (-) used as negative is known as ***unary minus***.

PARTICIPATION ACTIVITY 2.5.2: Single space around operators.

Retype each statement to follow the good practice of a single space around operators.

Note: If an answer is marked wrong, something differs in the spacing, spelling, capitalization, etc. This activity emphasizes the importance of such details.

1) `housesCity = housesBlock *10;`

housesCity = housesBlock * 10;

Check Show answer

Correct

housesCity = housesBlock * 10;

The spaces ensure the operator is clearly visible. Note that a semicolon (;) is not an operator and should not be preceded by a space.

2) `tot = num1+num2+2;`

tot = num1 + num2 + 2;

Check Show answer

Correct

tot = num1 + num2 + 2;

The spaces reduce human reader confusion with num2 and 2.

3) `numBalls=numBalls+1;`

numBalls = numBalls + 1;

Check Show answer

Correct

numBalls = numBalls + 1;

The spaces around the = ensure the assignment is clearly visible.

4) `numEntries = (userVal+1)*2;`

numEntries = (userVal + 1) * 2;

Check Show answer

Correct

numEntries = (userVal + 1) * 2;

The spaces may lead to more readable expressions.

Feedback?

Compound operators

Special operators called ***compound operators*** provide a shorthand way to update a variable, such as userAge ***+=*** 1 being shorthand for userAge = userAge + 1. Other compound operators include ***-=***, ****=***, ***/=***, and ***%=***.

PARTICIPATION ACTIVITY 2.5.3: Compound operators.

1) numAtoms is initially 7. What is numAtoms after: numAtoms += 5?

12

Check Show answer

Correct

12

Same as numAtoms = numAtoms + 5, so 7 + 5 is 12.

2) numAtoms is initially 7. What is numAtoms after: numAtoms *= 2?

14

Check Show answer

Correct

14

Same as numAtoms = numAtoms * 2, so 7 * 2 is 14.

3) Rewrite the statement using a compound operator. If the

Correct

statement can't be rewritten using a compound operator, type: Not possible
carCount = carCount / 2;

carCount /= 2;

Check Show answer

carCount /= 2;

carCount /= 2 is same as carCount = carCount / 2.

4) Rewrite the statement using a compound operator. If the statement can't be rewritten using a compound operator, type: Not possible
numItems = boxCount + 1;

Not possible

Check Show answer

Correct

Not possible

A compound operator is shorthand for when a variable is being updated, not for a more general assignment. Both sides must thus have the same variable, but here the variables are different (numItems and boxCount).

Feedback?

No commas allowed

Commas are not allowed in an integer literal. So 1,333,555 is written as 1333555.

PARTICIPATION ACTIVITY 2.5.4: Expression in statements.

1) Is the following an error? Suppose an int's maximum value is 2,147,483,647.

```
numYears = 1,999,999,999;
```

- (•) Yes
- ( ) No

Correct

Commas aren't allowed in an integer literal.

Feedback?

CHALLENGE ACTIVITY 2.5.1: Compute an expression.

Write a statement that assigns finalResult with the sum of num1 and num2, divided by 3. Ex: If num1 is 4 and num2 is 5, finalResult is 3.

```
#include <iostream>
using namespace std;

int main() {
   int num1;
   int num2;
   int finalResult;

   cin >> num1;
   cin >> num2;

   /* Your solution goes here  */
   finalResult = (num1 + num2)/3;

   cout << "Final result: " << finalResult << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with 4, 5

Your output `Final result: 3`

✓ Testing with 4, 11

Your output `Final result: 5`

Feedback?

CHALLENGE ACTIVITY 2.5.2: Total cost.

A drink costs 2 dollars. A taco costs 4 dollars. Given the number of each, compute total cost and assign totalCost with the result. Ex: 4 drinks and 6 tacos yields totalCost of 32.

```
#include <iostream>
using namespace std;

int main() {
```

```
   int numDrinks;
   int numTacos;
   int totalCost;

   cin >> numDrinks;
   cin >> numTacos;

   /* Your solution goes here  */
   totalCost = numDrinks* 2 + numTacos*4;

   cout << "Total cost: " << totalCost << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with 4 drinks and 6 tacos

Your output `Total cost: 32`

✓ Testing with 2 drinks and 8 tacos

Your output `Total cost: 36`

Feedback?