# 2.9 Constant variables

A good practice is to minimize the use of literal numbers in code. One reason is to improve code readability. newPrice = origPrice - 5 is less clear than newPrice = origPrice - priceDiscount. When a variable represents a literal, the variable's value should not be changed in the code. If the programmer precedes the variable declaration with the keyword const, then the compiler will report an error if a later statement tries to change that variable's value. An initialized variable whose value cannot change is called a ***constant variable***. A common convention, or good practice, is to name constant variables using upper case letters with words separated by underscores, to make constant variables clearly visible in code.

## Figure 2.9.1: Constant variable example: Lightning distance.

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

/*
 * Estimates distance of lightning based on seconds
 * between lightning and thunder
 */

int main() {
   const double SPEED_OF_SOUND    = 761.207; //
Miles/hour (sea level)
   const double SECONDS_PER_HOUR = 3600.0;   //
Secs/hour
   double secondsBetween;
   double timeInHours;
   double distInMiles;

   cout << "Enter seconds between lightning and
thunder: ";
   cin  >> secondsBetween;

   timeInHours = secondsBetween / SECONDS_PER_HOUR;
   distInMiles = SPEED_OF_SOUND * timeInHours;

   cout << "Lightning strike was approximately" <<
endl;
   cout << distInMiles << " miles away." << endl;

   return 0;
}
```

```
Enter seconds between lightning and
thunder: 7
Lightning strike was approximately
1.48012 miles away.

...

Enter seconds between lightning and
thunder: 1
Lightning strike was approximately
0.211446 miles away.
```

Feedback?

**PARTICIPATION ACTIVITY** 2.9.1: Constant variables.

Which of the following statements are valid declarations and uses of a constant integer variable named STEP_SIZE?

1) `int STEP_SIZE = 5;`

○ True

● False

**Correct**

Declares and initializes an int variable, but the variable is not a constant.

✓

2) `const int STEP_SIZE = 14;`

● True

○ False

**Correct**

Declares a constant int variable STEP_SIZE and initializes the constant with the value 14.

✓

3) `totalStepHeight = numSteps * STEP_SIZE;`

● True

○ False

**Correct**

Constant variables can be used in expressions just like other variables.

✓

4) `STEP_SIZE = STEP_SIZE + 1;`

○ True

● False

**Correct**

Results in a compilation error. Constant variables cannot be changed within assignment statements.

✓

**Feedback?**

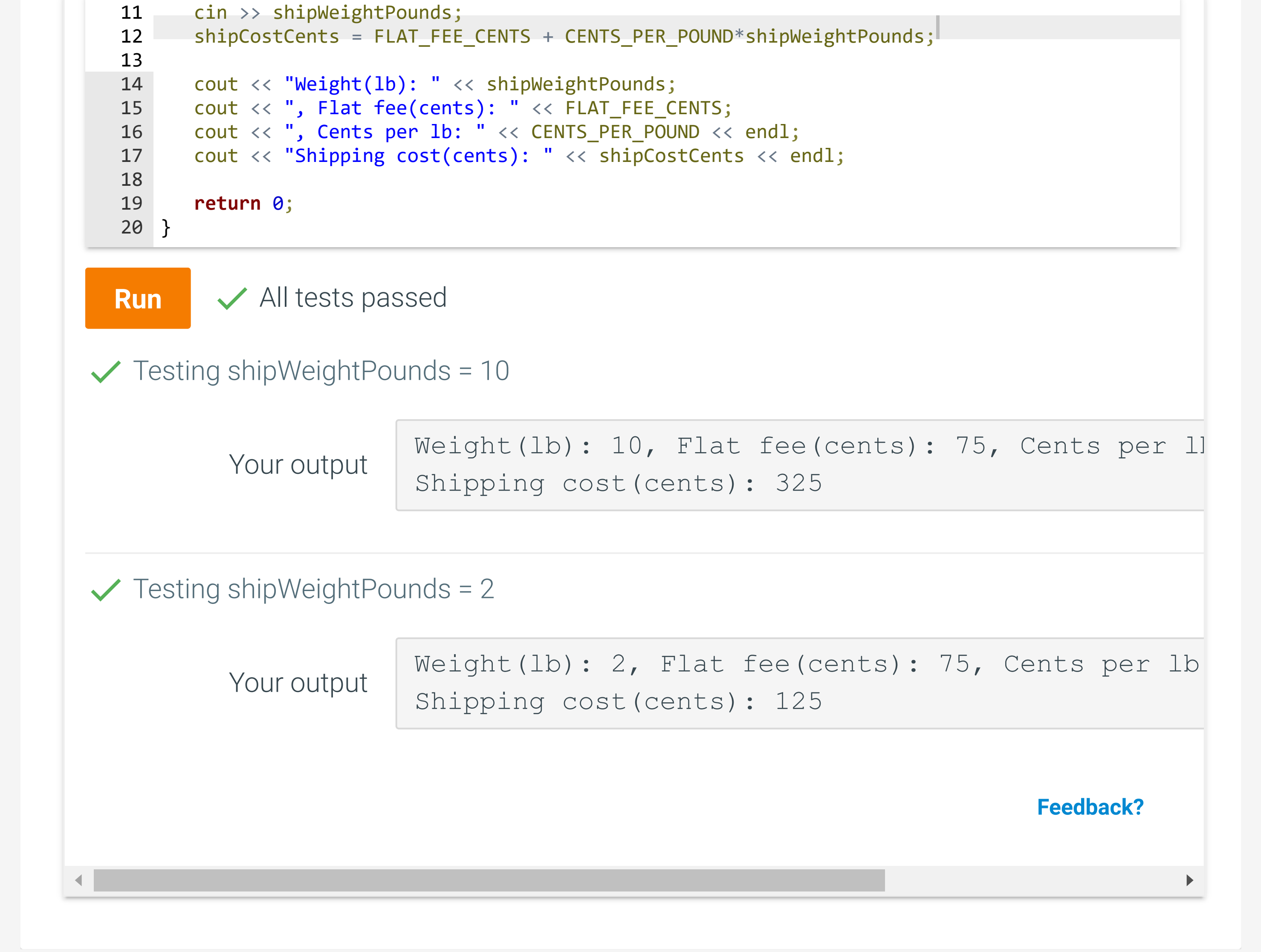
**CHALLENGE ACTIVITY** 2.9.1: Using constants in expressions. ✓

The cost to ship a package is a flat fee of 75 cents plus 25 cents per pound.
1. Declare a const named CENTS_PER_POUND and initialize with 25.
2. Get the shipping weight from user input storing the weight into shipWeightPounds.
3. Using FLAT_FEE_CENTS and CENTS_PER_POUND constants, assign shipCostCents with the cost of shipping a package weighing shipWeightPounds.

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

int main() {
   int shipWeightPounds;
   int shipCostCents = 0;
   const int FLAT_FEE_CENTS = 75;

   /* Your solution goes here  */
   const int CENTS_PER_POUND = 25;
```



```
   cin >> shipWeightPounds;
   shipCostCents = FLAT_FEE_CENTS + CENTS_PER_POUND*shipWeightPounds;

   cout << "Weight(lb): " << shipWeightPounds;
   cout << ", Flat fee(cents): " << FLAT_FEE_CENTS;
   cout << ", Cents per lb: " << CENTS_PER_POUND << endl;
   cout << "Shipping cost(cents): " << shipCostCents << endl;

   return 0;
}
```

✓ All tests passed

✓ Testing shipWeightPounds = 10

Your output

```
Weight(lb): 10, Flat fee(cents): 75, Cents per lb
Shipping cost(cents): 325
```

✓ Testing shipWeightPounds = 2

Your output

```
Weight(lb): 2, Flat fee(cents): 75, Cents per lb
Shipping cost(cents): 125
```