# 2.7 Floating-point numbers (double)

## Floating-point (double) variables

A **floating-point number** is a real number, like 98.6, 0.0001, or -666.667. The term "floating-point" refers to the decimal point being able to appear anywhere ("float") in the number. A variable declared as type **double** stores a floating-point number. Ex: **double milesTravel** declares a double variable.

A **floating-point literal** is a number with a fractional part, even if that fraction is 0, as in 1.0, 0.0, or 99.573. <u>Good practice</u> is to always have a digit before the decimal point, as in 0.5, since .5 might mistakenly be viewed as 5.

Figure 2.7.1: Variables of type double: Travel time example.

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

int main() {
    double milesTravel; // User input of miles to travel
    double hoursFly; // Travel hours if flying those miles
    double hoursDrive; // Travel hours if driving those miles

cout << "Enter miles to travel: ";
    cin >> milesTravel;

hoursFly = milesTravel / 500.0; // Plane flies 500 mph
    hoursDrive = milesTravel / 60.0; // Car drives 60 mph

cout << milesTravel << " miles would take:" << endl;
    cout << " " << hoursFly << " hours to fly" << endl;
    cout << " " << hoursDrive << " hours to drive" << endl;
    return 0;
}</pre>
```

```
Enter miles to travel: 1800
1800 miles would take:
3.6 hours to fly
30 hours to drive
...

Enter miles to travel: 400.5
400.5 miles would take:
0.801 hours to fly
6.675 hours to drive
```

Feedback?

PARTICIPATION ACTIVITY

2.7.1: Declaring and assigning double variables.

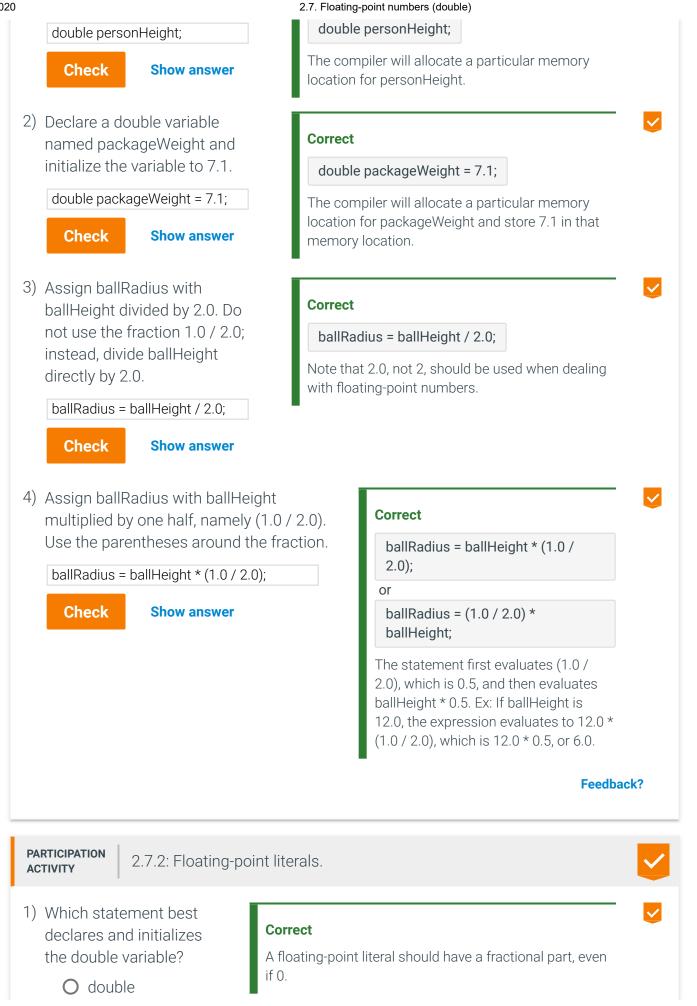


All variables are of type double and already-declared unless otherwise noted.

1) Declare a double variable named personHeight.

Correct





currHumidity = 99%;

- double currHumidity = 99.0;
- O double currHumidity = 99;
- Which statement best assigns the variable?Both variables are of type double.
  - O cityRainfall = measuredRain 5;
  - cityRainfall = measuredRain 5.0;
- 3) Which statement best assigns the variable? cityRainfall is of type double.
  - O cityRainfall = .97;
  - ocityRainfall = 0.97;

#### Correct

Best to use a floating-point literal like 5.0, rather than an integer literal like 5, when dealing with floating-point variables.

#### **Correct**

Best to have the 0 before the decimal point so that the decimal point isn't overlooked. Just .97 might be seen as 97 by a person reading the code.

Feedback?

## Scientific notation

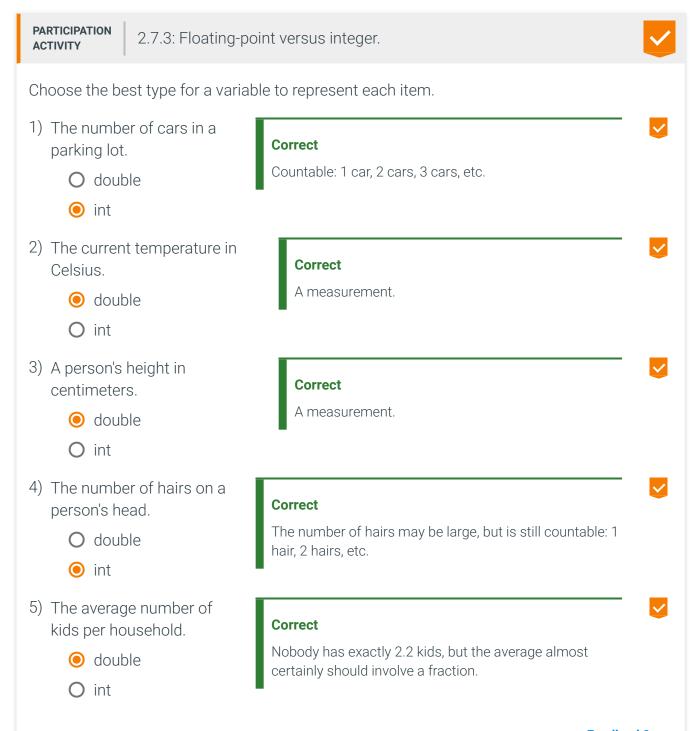
Very large and very small floating-point values may be printed using scientific notation. Ex: If a floating variable holds the value 299792458.0 (the speed of light in m/s), the value will be printed as 2.99792e+08.

## Choosing a variable type (double vs. int)

A programmer should choose a variable's type based on the type of value held.

- Integer variables are typically used for values that are counted, like 42 cars, 10 pizzas, or -95 days.
- Floating-point variables are typically used for values that are measured, like 98.6 degrees, 0.00001 meters, or -666.667 grams.
- Floating-point variables are also used when dealing with fractions of countable items, such as the average number of cars per household.

Note: Some programmers warn against using floating-point for money, as in 14.53 representing 14 dollars and 53 cents, because money is a countable item (reasons are discussed further in another section). int may be used to represent cents, or to represent dollars when cents are not included as for an annual salary (e.g., 40000 dollars, which are countable).



## Floating-point divide by zero

Dividing a nonzero floating-point number by zero results in *infinity* or *-infinity*, depending on the signs of the operands. Printing a floating-point variable that holds infinity or -infinity outputs inf or -inf.

If the dividend and divisor in floating-point division are both 0, the division results in a "not a number". **Not a number** (**NaN**) indicates an unrepresentable or undefined value. Printing a floating-point variable that is not a number outputs **nan**.

Figure 2.7.2: Floating-point division by zero example. #include <iostream> using namespace std; int main() { double gasVolume; double oilVolume; double mixRatio; cout << "Enter gas volume: ";</pre> cin >> gasVolume; cout << "Enter oil volume: ";</pre> cin >> oilVolume; mixRatio = gasVolume / oilVolume; cout << "Gas to oil mix ratio is " << mixRatio << ":1" << endl;</pre> return 0; Enter gas volume: 10.5 Enter oil volume: 0.0 Gas to oil mix ratio is inf:1 Feedback? **PARTICIPATION** 2.7.4: Floating-point division. **ACTIVITY** Determine the result. 1) 13.0 / 3.0 Correct Floating-point division retains the fractional value. 4.333333

O Positive infinity 2) 0.0 / 5.0 **Correct** 0.0 0.0 divided by 5.0 is 0.0. O Positive infinity O Negative infinity 3) 12.0 / 0.0 Correct **O** 12.0 Dividing by 0.0 results in infinity. The operations results in Positive infinity positive infinity. O Negative infinity 4) 0.0 / 0.0 Correct 0.0 Floating-point division of zero by zero is a special case O Infinity that results in not a number, or NaN. Not a number Feedback?

**CHALLENGE ACTIVITY** 

2.7.1: Sphere volume.



Given sphereRadius and piVal, compute the volume of a sphere and assign sphereVolume with the result. Use (4.0 / 3.0) to perform floating-point division, instead of (4/3) which performs integer division.

Volume of sphere =  $(4.0 / 3.0) \pi r^3$  (Hint:  $r^3$  can be computed using \*)

(Notes)

```
1 #include <iostream>
2 using namespace std;
4 int main() {
      double piVal = 3.14159;
      double sphereVolume;
6
      double sphereRadius;
8
9
      cin >> sphereRadius;
10
      /* Your solution goes here */
11
12 sphereVolume = (4.0 / 3.0)*piVal*sphereRadius*sphereRadius*sphereRadius;
13
      cout << sphereVolume << endl;</pre>
14
15
      return 0;
```

