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. Good practice 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;
}
```

```
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

double personHeight;

Check Show answer

double personHeight;

The compiler will allocate a particular memory location for personHeight.

2) Declare a double variable named packageWeight and initialize the variable to 7.1.

double packageWeight = 7.1;

Check Show answer

Correct

double packageWeight = 7.1;

The compiler will allocate a particular memory location for packageWeight and store 7.1 in that memory location.

3) Assign ballRadius with ballHeight divided by 2.0. Do not use the fraction 1.0 / 2.0; instead, divide ballHeight directly by 2.0.

ballRadius = ballHeight / 2.0;

Check Show answer

Correct

ballRadius = ballHeight / 2.0;

Note that 2.0, not 2, should be used when dealing with floating-point numbers.

4) Assign ballRadius with ballHeight multiplied by one half, namely (1.0 / 2.0). Use the parentheses around the fraction.

ballRadius = ballHeight * (1.0 / 2.0);

Check Show answer

Correct

```
ballRadius = ballHeight * (1.0 /
2.0);
```

or

```
ballRadius = (1.0 / 2.0) *
ballHeight;
```

The statement first evaluates (1.0 / 2.0), which is 0.5, and then evaluates ballHeight * 0.5. Ex: If ballHeight is 12.0, the expression evaluates to 12.0 * (1.0 / 2.0), which is 12.0 * 0.5, or 6.0.

Feedback?

PARTICIPATION ACTIVITY | 2.7.2: Floating-point literals.

1) Which statement best declares and initializes the double variable?

○ double

Correct

A floating-point literal should have a fractional part, even if 0.

currHumidity = 99%;

- (selected) double currHumidity = 99.0;
- double currHumidity = 99;

2) Which statement best assigns the variable? Both variables are of type double.

- cityRainfall = measuredRain - 5;
- (selected) cityRainfall = measuredRain - 5.0;

Correct

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

3) Which statement best assigns the variable? cityRainfall is of type double.

- cityRainfall = .97;
- (selected) cityRainfall = 0.97;

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).

PARTICIPATION ACTIVITY 2.7.3: Floating-point versus integer.

Choose the best type for a variable to represent each item.

1) The number of cars in a parking lot.
- ○ double
- ◉ int

Correct

Countable: 1 car, 2 cars, 3 cars, etc.

2) The current temperature in Celsius.
- ◉ double
- ○ int

Correct

A measurement.

3) A person's height in centimeters.
- ◉ double
- ○ int

Correct

A measurement.

4) The number of hairs on a person's head.
- ○ double
- ◉ int

Correct

The number of hairs may be large, but is still countable: 1 hair, 2 hairs, etc.

5) The average number of kids per household.
- ◉ double
- ○ int

Correct

Nobody has exactly 2.2 kids, but the average almost certainly should involve a fraction.

Feedback?

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: ";
   cin  >> gasVolume;

   cout << "Enter oil volume: ";
   cin  >> oilVolume;

   mixRatio = gasVolume / oilVolume;

   cout << "Gas to oil mix ratio is " << mixRatio << ":1" << endl;

   return 0;
}
```

```
Enter gas volume: 10.5
Enter oil volume: 0.0
Gas to oil mix ratio is inf:1
```

Feedback?

PARTICIPATION ACTIVITY 2.7.4: Floating-point division.

Determine the result.

1) 13.0 / 3.0

- ○ 4
- ◉ 4.333333

Correct

Floating-point division retains the fractional value.

- Positive infinity

2) 0.0 / 5.0

- 0.0
- Positive infinity
- Negative infinity

Correct

0.0 divided by 5.0 is 0.0.

3) 12.0 / 0.0

- 12.0
- Positive infinity
- Negative infinity

Correct

Dividing by 0.0 results in infinity. The operations results in positive infinity.

4) 0.0 / 0.0

- 0.0
- Infinity
- Not a number

Correct

Floating-point division of zero by zero is a special case that results in not a number, or NaN.

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) π r^3 (Hint: r^3 can be computed using *)

(Notes)

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

int main() {
   double piVal = 3.14159;
   double sphereVolume;
   double sphereRadius;

   cin >> sphereRadius;

   /* Your solution goes here  */
sphereVolume  =  (4.0 / 3.0)*piVal*sphereRadius*sphereRadius*sphereRadius;
   cout << sphereVolume << endl;

   return 0;
```

```
}
```

Run ✓ All tests passed

✓ Testing with radius 1.0

Your value `4.188786666666666`

✓ Testing with radius 0.0

Your value `0`

✓ Testing with radius 5.5

Your value `696.9093816666666`

Feedback?