Topic 3

1. Variables
2. Arithmetic
3. Input and output
4. Problem solving: first do it by hand
5. Strings
6. Chapter summary

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Input

- Sometimes the programmer does not know what should be stored in a variable – but the user does.

- The programmer must get the input value from the user
  - Users need to be prompted -- *how else would they know they need to type something?*
  - Prompts are done in output statements

- The keyboard needs to be read from
  - This is done with an input statement

Copyright © 2018 by John Wiley & Sons. All rights reserved

Input with `cin >>`

The input statement

- To read values from the keyboard, you input them from an object called **`cin`**.
- The "double greater than" operator **`>>`** denotes the "send to" command.

```
cin >> bottles;
```

is an *input statement*.

Of course, the variable **`bottles`** must be defined earlier.

Copyright © 2018 by John Wiley & Sons. All rights reserved

Input with `cin >>` to multiple variables

You can read more than one value in a single input statement:

```
cout << "Enter the number of bottles and cans: ";
cin >> bottles >> cans;
```

The user can supply both inputs on the same line:

```
Enter the number of bottles and cans: 2 6
```

Alternatively, the user can press the Enter key or tab key after each input, as `cin` treats all blank spaces the same

Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output

- When you print an amount in dollars and cents, you want it to be *rounded* to two significant digits.
- You learned earlier how to round off and store a value but, for output, we want to round off *only* for display.
- A ***manipulator*** is something that is sent to **`cout`** to specify how values should be formatted.
- To use manipulators, you must include the **`iomanip`** header in your program:

```
#include <iomanip>
```

and of course

```
using namespace std;
```

is also needed

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output for Dollars and Cents: `setprecision()`

Which do you think the user prefers to see on her gas bill?

```
Price per liter: $1.22
```

or

```
Price per liter: $1.21997
```


Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output Examples: Table 7

Output Statement	Output	Comment
`cout << 12.345678;`	`12.3457`	By default, a number is printed with 6 significant digits.
`cout << fixed << setprecision(2) << 12.3;`	`12.30`	The `fixed` and `setprecision` manipulators control the number of digits after the decimal point.
`cout << ":" << setw(6) << 12;`	`:    12`	Four spaces are printed before the number, for a total width of 6 characters.
`cout << ":" << setw(2) << 123;`	`:123`	If the width not sufficient, it is ignored.
`cout << setw(6) << ":" << 12;`	`    :12`	The width only refers to the next item. Here, the : is preceded by five spaces.

Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output, Dollars and Cents

You can combine manipulators and values to be displayed into a single statement:

```
price_per_liter = 1.21997;
cout << fixed << setprecision(2)
   << "Price per liter: $"
   << price_per_liter << endl;
```

This code produces this output:

```
Price per liter: $1.22
```


Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output with `setw()` to Align Columns

Use the **`setw`** manipulator to set the *width* of the next output field.

The width is the total number of characters, including digits, the decimal point, and spaces.

If you want aligned columns of certain widths, use the **`setw()`** manipulator.

For example, if you want a number to be printed, right justified, in a column that is eight characters wide, you use

```
<< setw(8)
```

before EVERY COLUMN's DATA.

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Exercise: Formatting Examples

- Given

```
int quantity = 10;
double price = 19.95;
```

What do the following statements print? *(show leading spaces as underscores _)*

```
cout << "Quantity:" << setw(4) << quantity;

cout << "Price:" << fixed << setw(8) <<
   setprecision(2) << price;

cout << "Price:" << fixed << setprecision(2) << price;

cout << fixed << setprecision(3) << price;

cout << fixed << setprecision(1) << price;
```

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Formatted Output, Another Example

This code:

```
price_per_ounce_1 = 10.2372;
price_per_ounce_2 = 117.2;
price_per_ounce_3 = 6.9923435;
cout << setprecision(2);
cout << setw(8) << price_per_ounce_1;
cout << setw(8) << price_per_ounce_2;
cout << setw(8) << price_per_ounce_3;
cout << "--------" << endl;
```

produces this output:

```
   10.24
  117.20
    6.99
--------
```

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

`setprecision` versus `setw`: Persistence

There is a notable difference between the **`setprecision`** and **`setw`** manipulators.

Once you set the precision, that precision is used for all floating-point numbers until the next time you set the precision.

But **`setw`** affects only the *next* value.

Subsequent values are formatted without added spaces.

Copyright © 2018 by John Wiley & Sons. All rights reserved

A Complete Program for Volumes

ch02/volume2.cpp

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

int main()
{
   // Read price per pack

   cout << "Please enter the price for a six-pack: ";
   double pack_price;
   cin >> pack_price;

   // Read can volume

   cout << "Please enter the volume for each can (in ounces): ";
   double can_volume;
   cin >> can_volume;
```


Copyright © 2018 by John Wiley & Sons. All rights reserved

A Complete Program for Volumes (continued)

```
   // Compute pack volume

   const double CANS_PER_PACK = 6;
   double pack_volume = can_volume * CANS_PER_PACK;

   // Compute and print price per ounce

   double price_per_ounce = pack_price / pack_volume;

   cout << fixed << setprecision(2);
   cout << "Price per ounce: " << price_per_ounce << endl;

   return 0;
}
```

Sample Program Run:

```
Please enter the price for a six-pack: 2.95
Please enter the volume for each can (in ounces): 12
Price per ounce: 0.04
```

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Topic 4

1. Variables
2. Arithmetic
3. Input and output
4. Problem solving: first do it by hand
5. Strings
6. Chapter summary

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved

Problem Solving: Before you write C++, do it by hand

- Write the algorithm (steps) in English, and carry out hand calculations to verify it
  - before typing your C++ code.
  - Pick simple, concrete values to test your algorithm
  - Comments at the top of the program are a good place to write the algorithm first

- For example (Ch. 2.4, Self Check 2&3), write the pseudocode for the following problem, and test with 2 sets of values:
  - model inflating a spherical balloon. First the balloon is inflated to a diameter (which is provided as an input). Then inflate the balloon by an inch, and display the amount the volume has grown. Repeat that step twice. The volume of a sphere is $\frac{4}{3}\pi r^3$

Big C++ by Cay Horstmann
Copyright © 2018 by John Wiley & Sons. All rights reserved