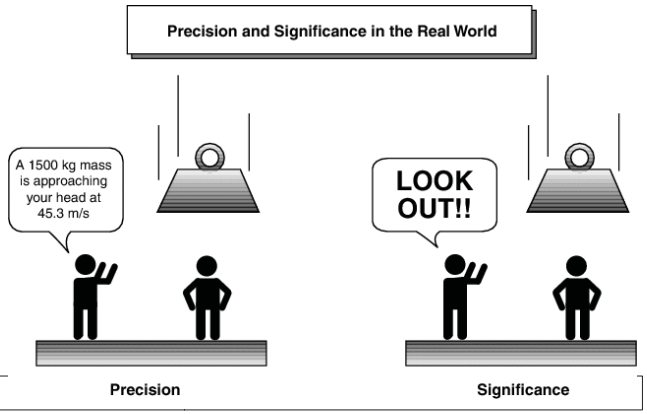
Slides #5

Formatting & Strings

Sections 3.8-3.12



CMPT 125/128

Topics

1) How can we format the output on the screen?
2) How can we input strings?
3) How can we analyze C++ code without a computer?

Formatting Output

```
Random Shopping List:
    SKU Item                          #  $ per     Price
 ------ -----------------           --- ------  --------
  10041 Hammer                        8   1.00      8.00
  16334 Book on C++                   1   2.25      2.25
  29169 Cowboy hat                    5  15.35     76.75
  21478 USB Memory Stick (8Gig)      19   0.10      1.90
  36962 Cellular phone                5   0.90      4.50
  15705 Pens (assorted)               6   1.11      6.66
  33281 Pizzas (anchovies & cheese)   8  22.00    176.00
  19961 Gum                          12  34.45    413.40
  12995 Hybrid vehicles               3  88.11    264.33
  14827 Staples                      17  64.73   1100.41
 ------ -----------------           --- ------  --------
        Total                                    2054.20
```


Formatting Overview

- We can control how cout formats output.
  - setw() Set width of output.
  - setprecision() Set number of decimal places.
  - fixed Prevent scientific notation & forces display of decimal point.
  - left or right Control left vs right justification.
- These are all called manipulators:
  -
  - Must #include <iomanip>

setw()

- setw() is a manipulator:
  -
    - Great for lining up data on the screen.
  - setw() only affects the one next element.
- Example:

```
cout << "[" << 12 << "]";
cout << "[" << setw(5) << 12 << "]";
```

Pads with spaces when item is fewer characters than the setw()'s width.

if it's larger than width.

Making a table

```
#include <iostream>
#include <iomanip>
using namespace std;
int main() {

    const int WIDTH1 = 15;
    const int WIDTH2 = 18;
    const int WIDTH3 = 12;

    cout << setw(WIDTH1) << "Name:"
         << setw(WIDTH2) << "Fav Food"
         << setw(WIDTH3) << "Fav Number"<<endl;

    cout << setw(WIDTH1) << "Dr. Evil"
         << setw(WIDTH2) << "Cupcakes"
         << setw(WIDTH3) << "100000000"<<endl;

    cout << setw(WIDTH1) << "I.L.B. Bach"
         << setw(WIDTH2) << "Anchovies"
         << setw(WIDTH3) << "1997"<<endl;
    // . . . . .  omitted to fit on slide.
    return 0;
}
```

```
          Name:          Fav Food  Fav Number
       Dr. Evil          Cupcakes   100000000
    I.L.B. Bach          Anchovies        1997
            Me    Pizza and Cake             0
```


setprecision()

- setprecision() displays a floating-point value...

```
const double PI = 3.14159265;
cout << PI << endl;                          3.14159
cout << setprecision(4)  << PI << endl;      3.142
cout << setprecision(3)  << PI << endl;      3.14
cout << setprecision(2)  << PI << endl;      3.1
cout << setprecision(1)  << PI << endl;      3
cout << setprecision(15) << PI << endl;      3.14159265
```

Value is rounded.

- Notes:
  - setprecision() stays in effect until changed.
  - must include <iomanip>

fixed

- fixed does 2 things:
  1) Prevents scientific notation (like 1.23E8)
  2) Changes setprecision() to... (not significant digits)

```
const double BIG = 12345678.9;
cout << BIG << endl;                 1.23457e+007
cout << fixed;
cout << BIG << endl;                 12345678.900000
cout << setprecision(2);
cout << BIG << endl;                 12345678.90
```

Big number defaults to scientific notation.

fixed avoids sci. notn. Default precision...

fixed + setprecision(2) gives 2 digits after .

```
cout << fixed << setprecision(2);
cout << "Cost 1: " << 1.9876 << endl;  Cost 1: 1.99
cout << "Cost 2: " << 3.55 << endl;    Cost 2: 3.55
cout << "Cost 3: " << 2.1 << endl;     Cost 3: 2.10
cout << "Cost 4: " << 1.0 << endl;     Cost 4: 1.00
cout << "Cost 5: " << 1 << endl;       Cost 5: 1
```

fixed and setprecision() no effect on integers.

left / right

- Control...
  within setw()'s width.
  - Default is right.

```
cout << '[' << left   << setw(8) << "Hi!" << ']'<<endl;
cout << '[' << right << setw(8) << "Hi!" << ']'<<endl;
```

```
[Hi!      ]
[      Hi!]
```

- Notes:
  - setprecision() stays in effect until changed.
  - must include <iomanip>

Modifier summary

Stream Manipulator	Effect	Persistent	Affects
setw(*n*)	Set width of next output to be *n* characters wide. Pad with spaces.		Everything
setprecision(*n*)	Display *n* digits after dec. point. Display *n* significant digits.	Persistent	Floating
fixed	Prevents scientific notation (plus changes setprecision())	Persistent	Floating
left	Left justified	Persistent	Everything
right	Right justified	Persistent	Everything

Review

1. Complete the statements with stream modifiers (if any) so they generate the output:

a) `cout << "L" << _____ << _____ << "R" << endl;`

```
L       R
```

b) `cout << _____`
`<< 0.0 << "\t" << 1000000000.0 << endl;`

```
0       1e+009
```

c) `cout << _____`
`<< 5.5469 << "\t" << 123456789.0 << endl;`

```
5.55    1.23e+008
```

d) `cout << _____ << _____`
`<< 5.5469 << "\t" << 123456789.0<< endl;`

```
5.55    123456789.00
```


Strings

Section 3.9

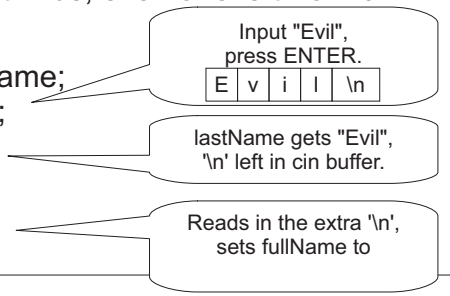
String input

- Read in single words with:
  ```
  string firstName;
  cin >> firstName;
  ```
  - Input "Dr Evil" will only set firstName to "Dr":
    -
- Read whole line of text with:
  ```
  string fullName;
  ```
  - Input "Dr Evil" will set fullName to "Dr Evil".
    - It stops on end of line ('\n').

\n detail

- Problem with extra '\n':
  - cin >> myString;
    - Stops at a '\n', and then...
  - getline(cin, myString);
    - Stops on first '\n' it finds, even one left from cin!
- Example:
  - ```
    string lastName, fullName;
    cout << "Last name: ";
    cin >> lastName;

    cout << "Full name: ";
    getline(cin, fullName);
    ```



Fruit Eating: cin vs getline

- Recap:
  - cin:
    - skips any leading whitespace & reads data;
    - stops on a space or a newline and...
  - getline stops on just a newline, but...
- Imagine cin and getline are at a buffet.
  - Represent each key-presesd with a fruit:

 (Carrot)  (Watermelon)  (Nectarine)

Fruit Eating: cin vs getline

Secret Agent Program:

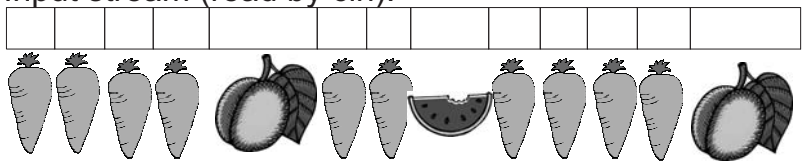
```
string lastName, fullName;
cout << "Last name: ";
cin >> lastName;

cout << "Full name: ";
getline(cin, fullName);
```

Desired Operation:

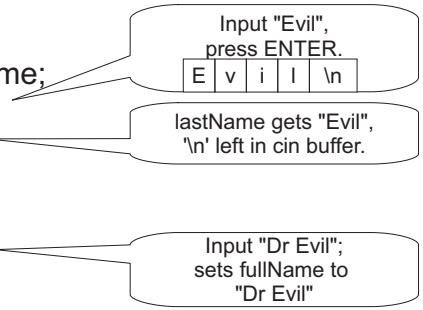
Last name: **Evil**
Full name: **Dr Evil**

Input stream (read by cin):


```

Ignore the '\n'

- Use cin.ignore()
  - cin.ignore() will wait until cin has at least 1 character, and then ignore it!
  -
- Example:
  - string lastName, fullName;
    cout << "Last name: ";
    cin >> lastName;
    cin.ignore();
    cout << "Full name: ";
    getline(cin, fullName);



Fruit Eating: cin vs getline – Fixed!

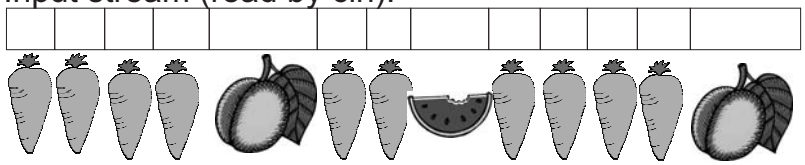
Secret Agent Program:

```
string lastName, fullName;
cout << "Last name: ";
cin >> lastName;
cin.ignore();
cout << "Full name: ";
getline(cin, fullName);
```

Desired Operation:

Last name: **Evil**
Full name: **Dr Evil**

Input stream (read by cin):



Hand Debugging

Section 3.13

Hand tracing

- Analyzing Code
  - Programmers must be able to trace through source code by hand.
  - Used to:
    - Understand some code.
    - Find & fix bugs!
- Hand tracing:
  -

Hand trace example

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

int main() {                                                    zombies aliens  bunnies  score
      const int POINTS_BAD_GUY = 5;
      const int POINTS_GOOD_GUY = 3;

      int zombies, aliens, bunnies;
      cout << "Intergalactic Ranger Score Calculator: "<<endl;
      cout << "# zombies & aliens you locked-up "
            << "(sep. by a space):"<<endl;
      cin >> zombies >> aliens;
      cout << "# bunnies you saved: "<<endl;
      cin >> bunnies;

      int score = zombies + aliens * POINTS_BAD_GUY;
      score += bunnies * POINTS_GOOD_GUY;

      cout << "Your score is: "<<score<<endl;
      return 0;
}
```


Summary

- Use manipulators to control output to screen:
  - setw() Set width of output.
  - setprecision() Set number of decimal places.
  - fixed Prevent scientific notation.
  - showpoint Force display of decimal point.
  - left or right Control left vs right justification.
- Line of text input with getline(cin, varNameHere)
  - cin.ignore() to skip a lingering line-feed ('\n').
- Code analysis skills like hand tracing.