

The C++ Language

Input and Output

Output

- *Output* is information generated by a program.
- Frequently sent the screen or a file.
- An *output stream* is used to send information.
 - Another type supplied by C++
 - Very complex, made up of several simple types.

Output to The Screen

- Syntax:

```
cout << Expression << Expression ... ;
```

- The << is called the *insertion operator*
- The value of each expression is printed to the screen from left to right.
- Must include basic input/output support:

```
#include <iostream>
```

3

Input and Output - Struble

Output to The Screen (Examples)

```
int cost;  
cost = 259;  
  
cout << cost;  
  
cout << "The cost is " << cost;  
  
cout << "You need "  
    << cost / 100  
    << " dollars and "  
    << cost % 100 << " cents.";
```

Output
(for each statement)

```
259  
  
The cost is 259  
  
You need 2 dollars and  
59 cents.
```

4

Input and Output - Struble

Output to The Screen

- The output on the previous slide would really be on one line

```
259The cost is 259You need 2 dollars and 59 cents.
```

- How do we force a new line to be used?

5

Input and Output - Struble

Output to Separate Lines

- Use the `endl` *manipulator* to start new lines.
 - Manipulators are special identifiers.
- By using `endl`, the following output statements generate 3 lines of output

```
cout << cost << endl;  
cout << "The cost is " << cost << endl;  
cout << "You need " << cost / 100 << " dollars and "  
    << cost % 100 << " cents." << endl;
```

6

Input and Output - Struble

Debugging Tip!

- When writing your programs, it's often useful to include extra output statements to print out the value of variables or other kinds of information.
- Use output statements generously, but remove them before submitting your assignment for grading!

7

Input and Output - Struble

Formatting Output

- Care is needed when displaying numbers

```
int i,j,k;  
i = 15; j = 2; k = 6;  
cout << "Results:" << i << j << k << endl;
```

displays

```
Results:1526
```

- Remember to put in spaces

```
cout << "Results: " << i << ' ' << j << ' ' << k  
<< endl;  
Results: 15 2 6
```

8

Input and Output - Struble

Manipulators

- *Manipulators* are special identifiers that change the behavior of input and output.
- Used frequently for formatted output
- Except for `endl`, you must include support for manipulators

```
#include <iomanip>
```

```
cout << ExpressionOrManipulator << ExpressionOrManipulator ... ;
```

9

Input and Output - Struble

Displaying Decimal Points

- C++ does not show a decimal point for floating point values with only 0 after the decimal.
- Use the `showpoint` manipulator to always display the decimal point.
- Use the `noshowpoint` manipulator to hide the decimal point again.

10

Input and Output - Struble

Formatting Numeric Output

- C++ uses scientific notation to display floating point values unless told otherwise.
- Use the `fixed` manipulator to display a fixed number of digits after the decimal.
 - This should generally be combined with `showpoint`
- Use the `scientific` manipulator to return to scientific notation

11

Input and Output - Struble

Formatting Numeric Output

- Example

```
const double SMALL = 0.000012345678;
// using scientific notation
cout << SMALL << endl;
// Now use fixed point notation
cout << showpoint << fixed;
cout << SMALL;
cout << scientific << noshowpoint; // back to default
```
- Displays

```
1.23457e-005
0.000012
```

12

Input and Output - Struble

Formatting Numeric Output

- To control
 - The number of digits of precision in scientific notation
 - The number of digits after the decimal in fixed point notationuse the `setprecision` manipulator.

```
setprecision(IntegerExpression)
```

13

Input and Output - Struble

Precision Example

- Example

```
const double SMALL = 0.000012345678;
cout << setprecision(8);
// using scientific notation
cout << SMALL << endl;
// Now use fixed point notation
cout << showpoint << fixed;
cout << SMALL;
```
- Displays (notice the rounding on the second output)

```
1.2345678e-005
0.00001235
```

14

Input and Output - Struble

Fixed Width Output

- You can force a value to take up a certain amount of space by using the `setw` manipulator.
- Syntax:

```
setw( IntegerExpression )
```
- Warnings
 - `setw` applies only to the next value printed
 - `setw` does not work with string variables

15

Input and Output - Struble

Fixed Width Output (Examples)

```
int a, b, c;  
a = 10; b = 3247; c = -123;
```

```
cout << setw(5) << a  
      << b  
      << setw(5) << c << endl;
```

```
cout << setw(6) << "Year"  
      << setw(3) << b  
      << setw(4) << a << endl;
```

Output

```
    103247  -123  
    {      } {  
    5      5
```

```
  Year3247  10  
  {        } {  
  6        4
```

3247 is too wide, so C++ prints it out as is.

16

Input and Output - Struble

Fixed Width Output

- C++ normally right-justifies fixed width output
- Use the `left` manipulator to switch to left-justified output.
- Use the `right` manipulator to switch back.

```
cout << left << setw(4)      1   23   4
    << 1 << 2;                {   {   {
                                4   3   3
```

17

Input and Output - Struble

Fixed Width Output

- Occasionally, a character other than a space is needed to fill in extra spaces in fixed width output.
- Use the `setfill` manipulator to use a different character.
- Syntax

```
setfill(CharacterExpression)
```

18

Input and Output - Struble

Fixed Width Output (Example)

- Example: Print out an integer with leading zeroes.

```
cout << setfill('0')           0010
      << setw(4)
      << 10 << endl;
// reset fill character to a space
cout << setfill(' ');
```

19

Input and Output - Struble

Escape Sequences

- C++ uses special sequence of characters to print out some information.
- Example: Print out "Hello" including the quotes

```
cout << "\"Hello\" " << endl;
```
- The backslash `\` followed by a character is called an *escape sequence*.

<code>\"</code>	Double quote
<code>\\</code>	Single backslash
<code>\t</code>	A tab character
<code>\n</code>	A new line

20

Input and Output - Struble

Input

- *Input* is information that is provided to the program
- Frequently from the keyboard or a file
- An *input stream* is used to receive information
 - Another type supplied by C++

21

Input and Output - Struble

Input From the Keyboard

- Syntax

```
cin >> Variable >> Variable ... ;
```

- The >> is called the *extraction operator*
- Information is read from left to right
 - Skips leading *whitespace*
 - Spaces, tabs, new lines, etc.
- Information is stored into variables from left to right.
- Must include `iostream` just like output to screen.

22

Input and Output - Struble

Input From the Keyboard (Example)

```
10.0 ab 9 100 c
D 98.6
```

```
int i;
string s;
double d;
char c;

cin >> d >> s >> i;    // d is 10.0, s is "ab", i is 9
cin >> s >> c;          // s is "100", c is 'c'

cin >> c >> i >> s;    // c is 'd', i is 98, s is ".6"
```

23

Input and Output - Struble

Tracing Input

- Need to track the *reading marker*
 - Location of next character to read
- Extraction operator skips leading whitespace
 - Stops at last valid character for type or whitespace

```
int i;

 98 98.6          98 98.6          98 98.6
↑                ↑                ↑
cin >> i;         cin >> i;         cin >> i;
```

24

Input and Output - Struble

Exercise (in class)

- Trace the reading marker and determine what is stored in each variable after each statement.

```
Hello World! 212.0 95.7 93
```

```
int i;  
double d;  
string s;  
char c;  
  
cin >> c >> s;  
cin >> s >> d >> i >> c;  
cin >> d >> c >> i;
```

25

Input and Output - Struble

Ignoring Input

- Sometimes you do not want to store information from the keyboard.
- Use the `cin.ignore` function to ignore input.
- Syntax

```
cin.ignore( IntegerExpr, CharacterExpr );
```

- Semantics
 - Ignore up to `IntegerExpr` input characters or until `CharExpr` is read and discarded from the keyboard.

26

Input and Output - Struble

Ignoring Input (Examples)

```
Hello World! 98.6
My name is Johnny.

cin.ignore(100, '\n');
```

```
Hello World! 98.6
My name is Johnny.

cin.ignore(10, '\n');
```

27

Input and Output - Struble

Ignoring Input

- Problem: Ignore input up to and including a specified character.

```
#include <climits> // upper and lower bounds,
                  // defines INT_MAX, INT_MIN
const char DELIMITER = '['; // what to ignore up to

int main() {
    // Code, code, code
    cin.ignore(INT_MAX, DELIMITER); // skip past DELIMITER
    // next character to read follows DELIMITER
    cin >> intgr;
    // more code
}
```

28

Input and Output - Struble

Reading Whitespace (Single Character)

- Whitespace is normally ignored by the extraction operator.
- Use the `cin.get` function to read in a single character, whether it's a space or not.
- Syntax

```
cin.get(CharVariable);
```

29

Input and Output - Struble

Reading Whitespace (Example)

```
char ch1, ch2, ch3;
```

```
  A M           A M           A M           A M  
  ↑             ↑             ↑             ↑  
cin >> ch1;   cin.get(ch2);   cin >> ch3;  
  
// ch1 contains 'A', ch2 contains ' ', ch3 contains 'M'
```

30

Input and Output - Struble

Reading Whitespace (Strings)

- Strings containing white space can be read as long as there is a delimiting character.
- To read from the marker to the end of line into a string, use the `getline` function
 - Delimiting character is `'\n'`, the newline character
- Syntax

```
getline(cin, StringVariable);
```

31

Input and Output - Struble

Reading Whitespace (Example)

```
Fred Flintstone      Laborer      13301
Barney Rubble        Laborer      43583

string aLine;

getline(cin, aLine);
// aLine now contains "Fred Flintstone\tLaborer\t13301"

Fred Flintstone      Laborer      13301
Barney Rubble        Laborer      43583
↑
```

32

Input and Output - Struble

Reading Whitespace (Strings)

- Notice that the delimiting character is read, but **NOT** stored in the string.
- A different delimiting character can be specified by using a third parameter
- Syntax

```
getline(cin, StringVariable, CharExpr);
```

33

Input and Output - Struble

Reading Whitespace (Example)

```
Fred Flintstone      Laborer      13301
Barney Rubble        Laborer      43583
```

```
const char DELIMITER = '\\t';

getline(cin, Name, DELIMITER); // "Fred Flintstone"
getline(cin, Title, DELIMITER); // "Laborer"
cin >> id;                      // 13301

getline(cin, Name2, DELIMITER); // What is this?
```

34

Input and Output - Struble

Files

- A *file* is used to store data on a disk
- Programs often need to read files for input or send output to a file
- C++ provides *file streams*
 - *Input file streams*
 - *Output file streams*
- Must include support for file streams

```
#include <fstream>
```

35

Input and Output - Struble

Input File Streams

- To read input from a file, first declare a variable with the type `ifstream`
 - `ifstream` denotes input file stream
- Example

```
ifstream inStream;
```

36

Input and Output - Struble

Input File Streams

- The input file stream must be associated with a file on the disk.
- To associate it with a file, it must be opened, and the name of the file specified.
- Example

```
inStream.open("InputData.txt");
```

- The file named InputData.txt must exist and be in the same directory as your `.cpp` file.

37

Input and Output - Struble

Input File Streams

- To read input from the file, use the extraction operator and other input functions we described.
 - Replace `cin` with the variable name of your input stream, e.g., `inStream`

- Examples

```
inStream >> id;  
inStream.ignore(INT_MAX, DELIMITER);  
getline(inStream, aLine);
```

38

Input and Output - Struble

Input File Streams

- When you are finished with the input file stream, it must be *closed*.
 - Tells Windows the file is no longer in use
- Example

```
inStream.close();
```

39

Input and Output - Struble

Output File Streams

- To send output to a file, first declare a variable with the type `ofstream`
 - `ofstream` denotes output file stream
- Example

```
ofstream outStream;
```

40

Input and Output - Struble

Output File Streams

- Just like input file streams, output file streams need to be associated with a file on your disk
- To associate it with a file, it must be opened and the name of the file specified.
- Example

```
ostream.open("Results.txt");
```
- The output file is created in the same directory as your .cpp file
 - If the output file already exists, it is replaced with new contents.

41

Input and Output - Struble

Output File Streams

- To send output to a file, use the insertion operator and other output manipulators we described.
 - Replace `cout` with the variable name of your output file stream.
- Examples

```
ostream << "Hello World!" << endl;
ostream << fixed << showpoint << 0.00001 << endl;
```

42

Input and Output - Struble

Output File Streams

- In order to guarantee that all of your output is written the file must be *closed*
 - Tells Windows your program is done with the file.
 - Just like with input file streams.
- Example

```
outStream.close();
```