


CS481 - Object-Oriented Programming

I/O Streams

- ❖ C++ can use C-style I/O
 - That's what we've done so far
- ❖ Could wrap C I/O in a class
 - But `printf` arg list not type-safe
- ❖ So, new C++ I/O
 - Known as `iostreams` (in library)

1

I/O Cautions



- ❖ Easy things are easy
 - Simple input, output
- ❖ Hard things can be complex
 - Underlying buffer objects
 - Manipulator implementation
- ❖ May not always be used (GUI)

2

Stream Classes

```
#include <iostream.h>
```

- ❖ `istream` - source of bytes
 - `ifstream` - byte source (from file)
 - Other inputs (char array, string, etc.)
- ❖ `ostream` - sink of bytes
 - `ofstream` - byte sink (to file)
 - Other outputs (char array, string, etc.)

3

Standard Streams

```
using namespace std;
```

- ❖ Standard input: `cin`
 - Like `stdin`
- ❖ Standard output: `cout`
 - Like `stdout`
- ❖ Standard error: `cerr`
 - Like `stderr`

4

I/O Operators

- ❖ Inserter: `<<` (like `printf`, ...)
 - Formats data types into bytes
 - Puts bytes into stream
- ❖ Extractor: `>>` (like `scanf`, ...)
 - Takes bytes from stream
 - Converts representation to data types

5

Operator Overload

- ❖ A preview (more later)
- ❖ C++ operator implementation
 - Can be implemented as functions
 - But, functions can be overloaded
 - ◆ By signature - arg data type(s)
- ❖ Operators can be overloaded

6

CS481 - Object-Oriented Programming

Why << and >> ?

- ❖ Not heavily used
 - Shift left and shift right
- ❖ Reasonable precedence
 - Expression parsing not affected by overloading
- ❖ Normal meaning unaffected

7

I/O Example

```
int main(void)
{
    cout << "Hello there\n";
    cout << "Enter two numbers: ";

    int    n1, n2;

    cin >> n1;
    cin >> n2;

    ...
}
```

8

I/O Example

```
...
cout << "The sum of ";
cout << n1;
cout << " and ";
cout << n2;
cout << " is ";
cout << (n1 + n2);
cout << '\n';

return 0; /* Return success */
}
```

9

I/O Example

```
Hello there
Enter two numbers: 5 18
The sum of 5 and 18 is 23
```

```
Hello there
Enter two numbers: 14.5 6.8
The sum of 14 and 2695 is 2709
```

Why?

10

I/O Example

```
cout << "Enter two real numbers: ";
double d1, d2;
cin >> d1 >> d2;
cout << "The sum of " << d1
    << " and " << d2;
cout << " is " << (d1 + d2)
    << endl;
```

```
Enter two numbers: 14.5 6.8
The sum of 14.5 and 6.8 is 21.3
```

11

I/O Example

I/O operation chaining

```
cout << "Enter two real numbers: ";
double d1, d2;
cin >> d1 >> d2;
cout << "The sum of " << d1
    << " and " << d2;
cout << " is " << (d1 + d2)
    << endl;
```

```
Enter two numbers: 14.5 6.8
The sum of 14.5 and 6.8 is 21.3
```

12

CS481 - Object-Oriented Programming

I/O Example

Manipulator: EOL

```

cout << "Enter two real numbers: ";
double d1, d2;
cin >> d1 >> d2;
cout << "The sum of " << d1
    << " and " << d2;
cout << " is " << (d1 + d2)
    << endl;
    
```

Enter two numbers: 14.5 6.8
 The sum of 14.5 and 6.8 is 21.3

13

Manipulators

- ❖ endl
 - Inserts '\n'
 - Flushes output
- ❖ flush
- ❖ Base control
 - oct, dec, hex

14

Manipulators

<ul style="list-style-type: none"> ❖ endl <ul style="list-style-type: none"> - Inserts '\n' - Flushes output ❖ flush ❖ Base control <ul style="list-style-type: none"> - oct, dec, hex 	<ul style="list-style-type: none"> ❖ With args <ul style="list-style-type: none"> - setw() <ul style="list-style-type: none"> ◆ Field width - setprecision() - setfill() <ul style="list-style-type: none"> ◆ Fill character
--	---


15

Manipulators

<ul style="list-style-type: none"> ❖ endl <ul style="list-style-type: none"> - Inserts '\n' - Flushes output ❖ flush ❖ Base control <ul style="list-style-type: none"> - oct, dec, hex 	<ul style="list-style-type: none"> ❖ With args <ul style="list-style-type: none"> - setw() <ul style="list-style-type: none"> ◆ Field width - setprecision() - setfill() <ul style="list-style-type: none"> ◆ Fill character ❖ More ...
--	---

16

File I/O



- ❖ File streams
 - Input: ifstream
 - Output: ofstream
- ❖ Opened in constructor
 - Closed in destructor (or explicitly)
- ❖ Same operators (<<, >>, etc.)

17

Open Modes

- ❖ Second constructor argument
 - Replaces default value
- ❖ Values from ios class
 - ios::app, ios::nocreate,
 - ios::noreplace, ios::binary, ...
 - Combine with bitwise-OR (|)

18

CS481 - Object-Oriented Programming

Input Options


- ❖ Extractor (>>) is like scanf
 - Whitespace termination (can overflow!)
 - Reads across lines (past '\n')

19

Input Options

- ❖ Extractor (>>) is like scanf
 - Whitespace termination (can overflow!)
 - Reads across lines (past '\n')

Extractor for character array is very dangerous, like gets() in C!



20

Input Options

- ❖ Extractor (>>) is like scanf
 - Whitespace termination (can overflow!)
 - Reads across lines (past '\n')
- ❖ Line-at-a-time input (like fgets)
 - Stream member functions
 - ◆ get(), getline()

21

Line Input Functions

- ❖ get(s, n, delim= '\n')
 - Leaves delimiter in stream
 - Other overloaded forms exist
- ❖ getline(s, n, delim= '\n')
 - Removes delimiter from stream
- ❖ Subtle status differences (??)

22

Line Input Example

```
#include <fstream.h>

char buff[30];

ifstream infile("iotest2.cpp");
assert(infile);
ofstream outfile("junk.txt");
assert(outfile);
```

23

Line Input Example

```
while(infile.get(buff, sizeof buff))
{
    if (infile.peek() == '\n')
        infile.get(); //dispose of nl.
    else
        outfile <<
            "**** Buffer overflow ****"
            << endl;
    outfile << buff << endl;
}
```

24

CS481 - Object-Oriented Programming

Line Input Example

```

...
int main (void)
{
  char  buff[30];
  *** Buffer overflow ***
  ifstream infile("iotest2.cp
p");
  assert(infile);
  *** Buffer overflow ***
  ofstream outfile("junk.txt"
);

```

Output from program

25

In-Memory "I/O"

- ❖ Like `sscanf()`, `sprintf()`
 - But safer and more flexible
 - ◆ Type-safe conversions
 - ◆ Buffer overflow protection

26

In-Memory "I/O"

- ❖ Like `sscanf()`, `sprintf()`
- ❖ Two variations
 - Character array
 - ◆ `istrstream`, `ostrstream`
 - C++ string class
 - ◆ `istringstream`, `ostringstream`

27

