CSCE 121
Introduction to Program Design & Concepts
I/O Streams

Dr. Tim McGuire

Grateful acknowledgment to Dr. Michael Moore and Dr. Paul Taele for some of the material on which these slides are based.

Standard Input, Output, and Error

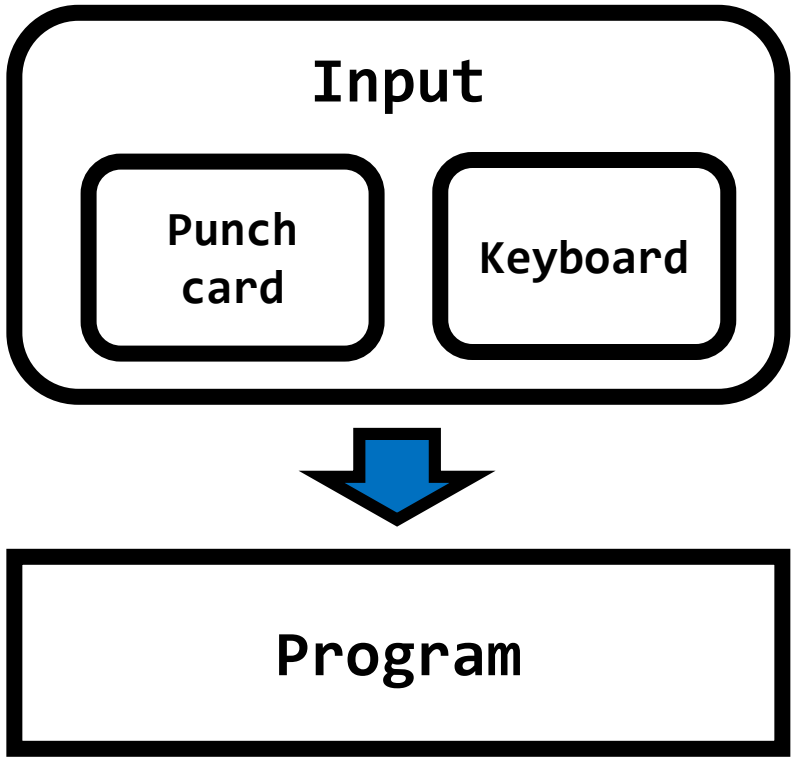
Standard Input (`stdin`)

About

- ***standard input*** – the default source for receiving input
- **`stdin`** – shorthand for standard input such as in C++

Details

- **Before:** `stdin` came from input such as punch cards.
- **Now:** normally refers to keyboard input.
- In C++, `stdin` is instantiated as `cin` object.



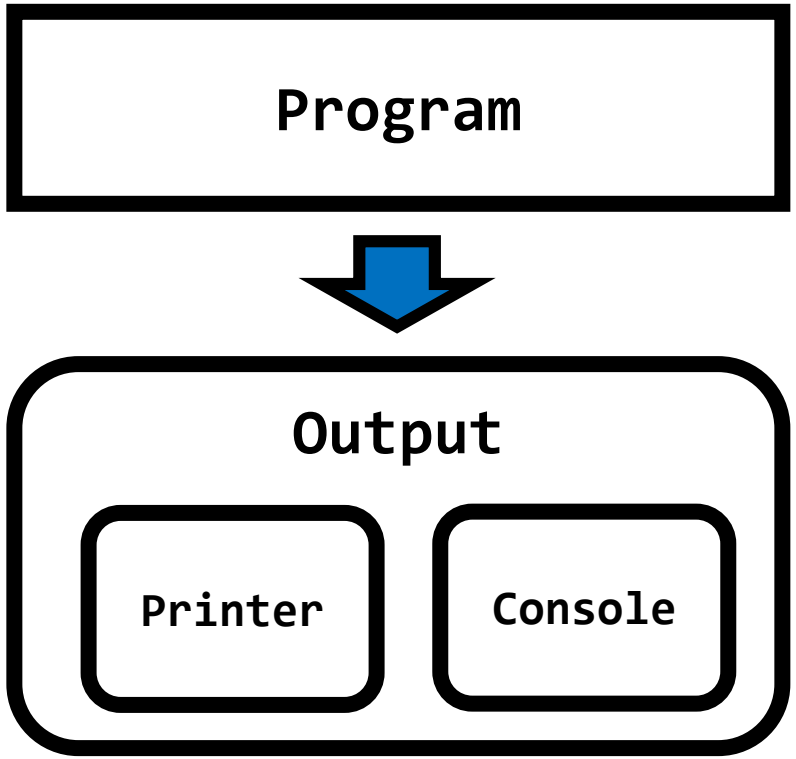
Standard Output (`stdout`)

About

- ***standard output*** – the default source for sending output
- **`stdout`** – shorthand for standard output such as in C++

Details

- **Before:** `output` came from output such as printers.
- **Now:** normally refers to console output.
- In C++, `stdout` is instantiated as `cout` object.



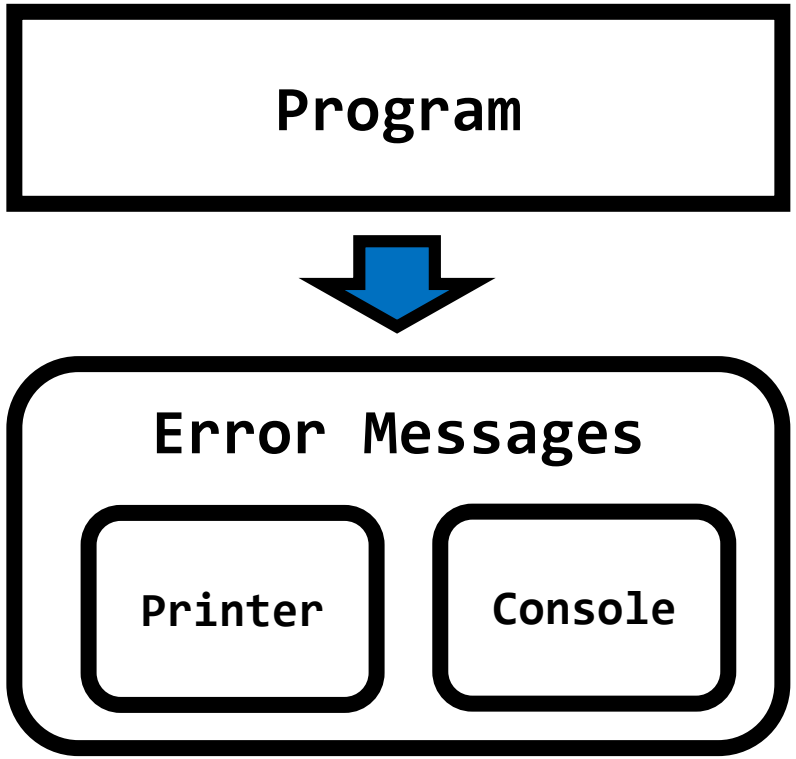
Standard Output (`stderr`)

About

- ***standard error*** – the default source for sending error messages
- **`stderr`** – shorthand for standard error such as in C++

Details

- Normally sent to the console.
- In C++, `stderr` is instantiated as `cerr` object.



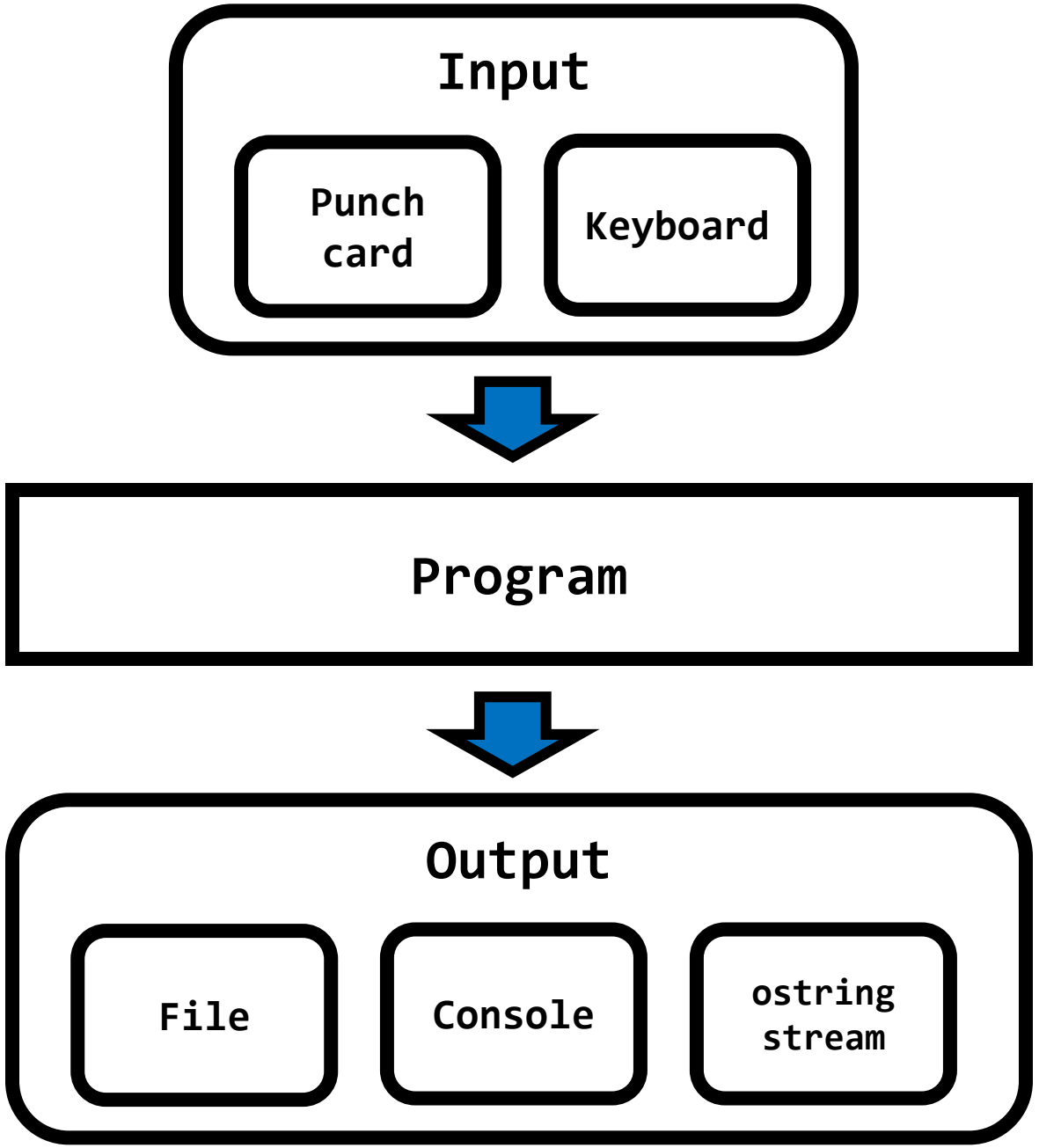
Standard Stream Destinations

About

- Destinations of standard streams can also be modified.
- E.g., standard error is frequently redirected to a file instead of to the console.

Details

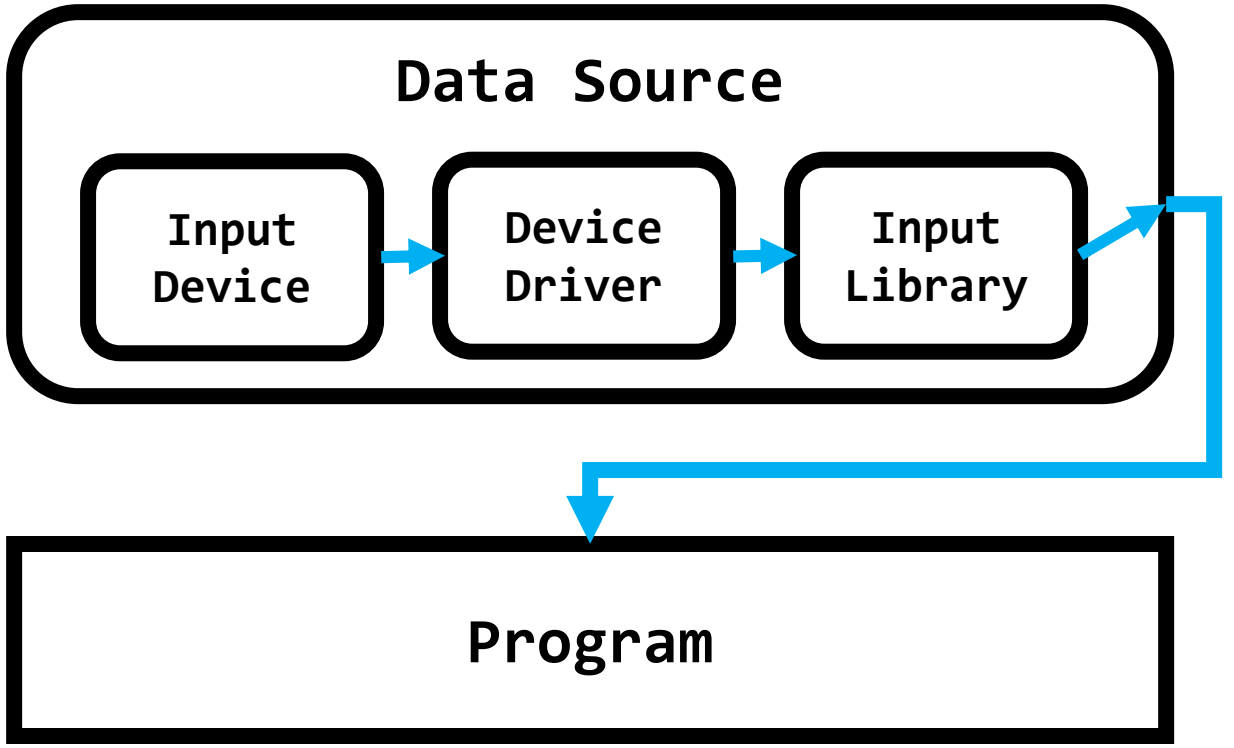
- `cout` output can be directed to `ostringstream`, so output can be read in a string instead.



High-level Input and Output Model

Input Stream Model

1. **Input Device.** The user provides input with an input device (e.g., keyboard).
2. **Device Driver.** The input device's computer program communicates the user's input to the computer.
3. **Input Library.** The programming language's input library will accept the device driver's data.



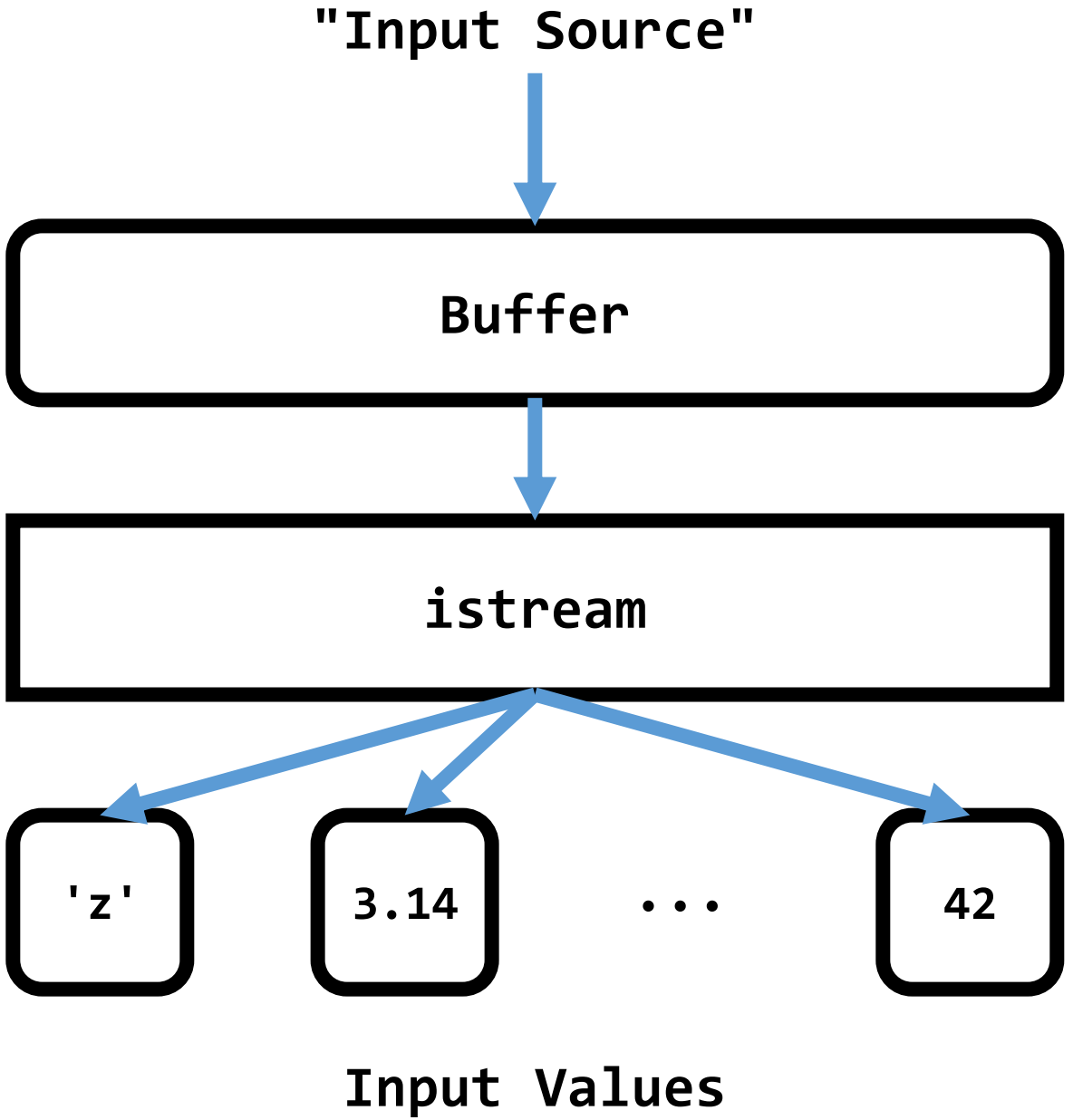
Input Stream Model: `istream`

About

- C++'s input stream model is `istream`.

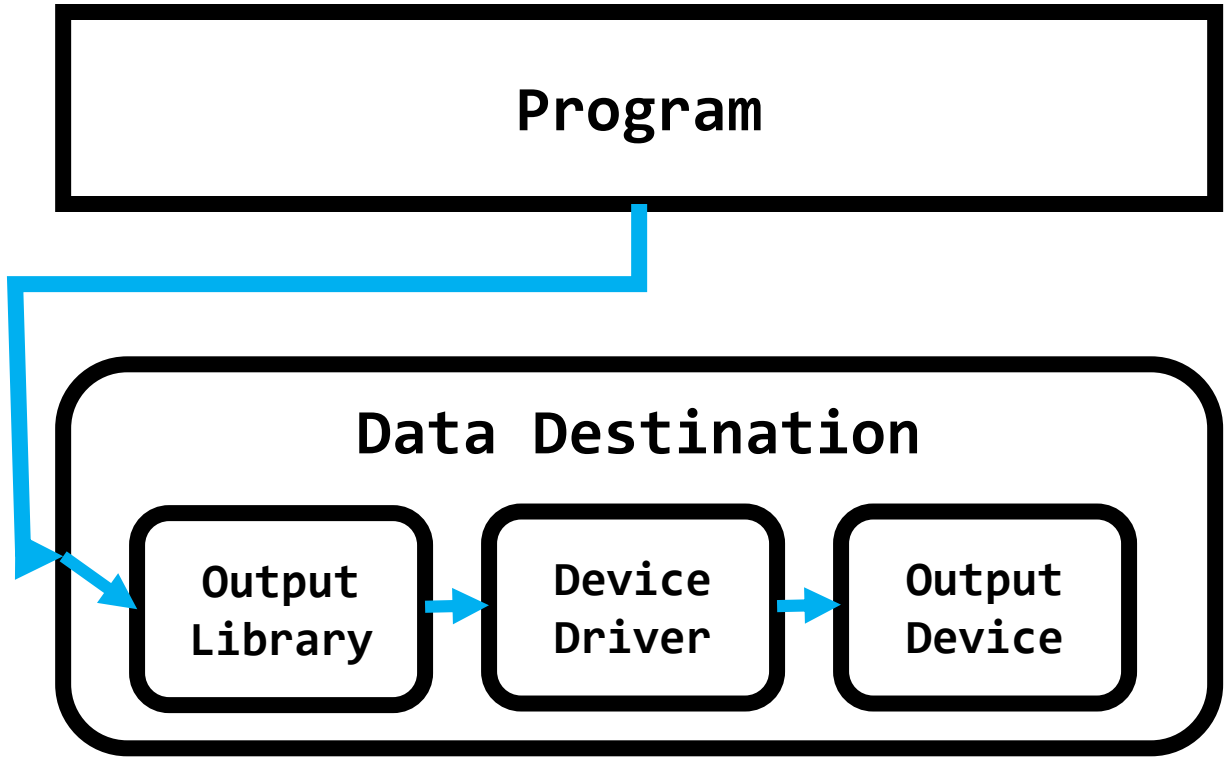
Details: An `istream`…

- …gets characters from some input device (e.g., keyboard, string, file).
- …converts sequences of characters into data types (e.g., `int`, `string`).



Output Stream Model

1. **Output Library.** The programming language's output library will transmit the data to the device driver.
2. **Device Driver.** The device driver will receive the data from the output library.
3. **Output Device.** The output device (e.g., console) will receive the data from the device driver.



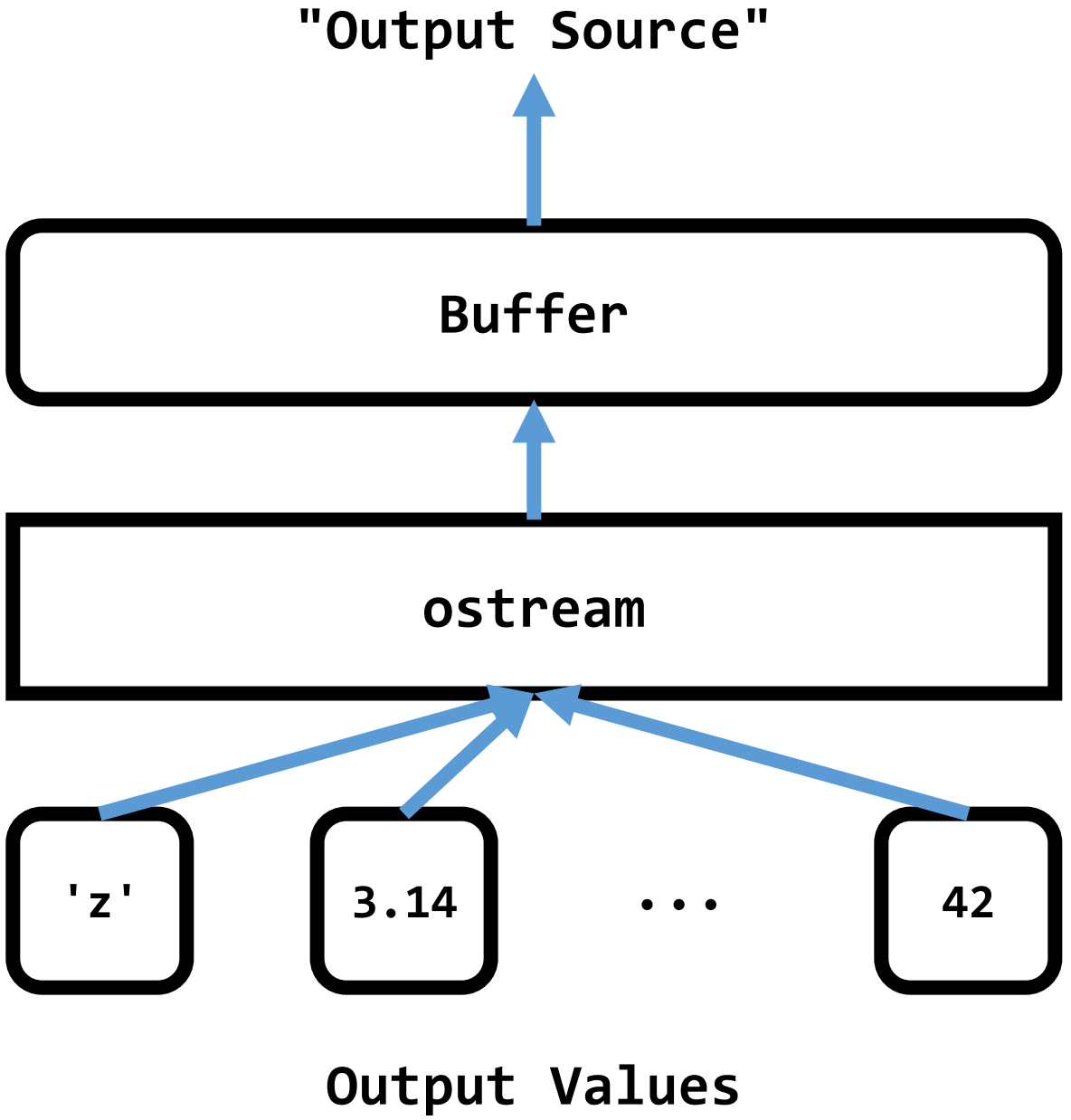
Output Stream Model: `ostream`

About

- C++'s onput stream model is `ostream`.

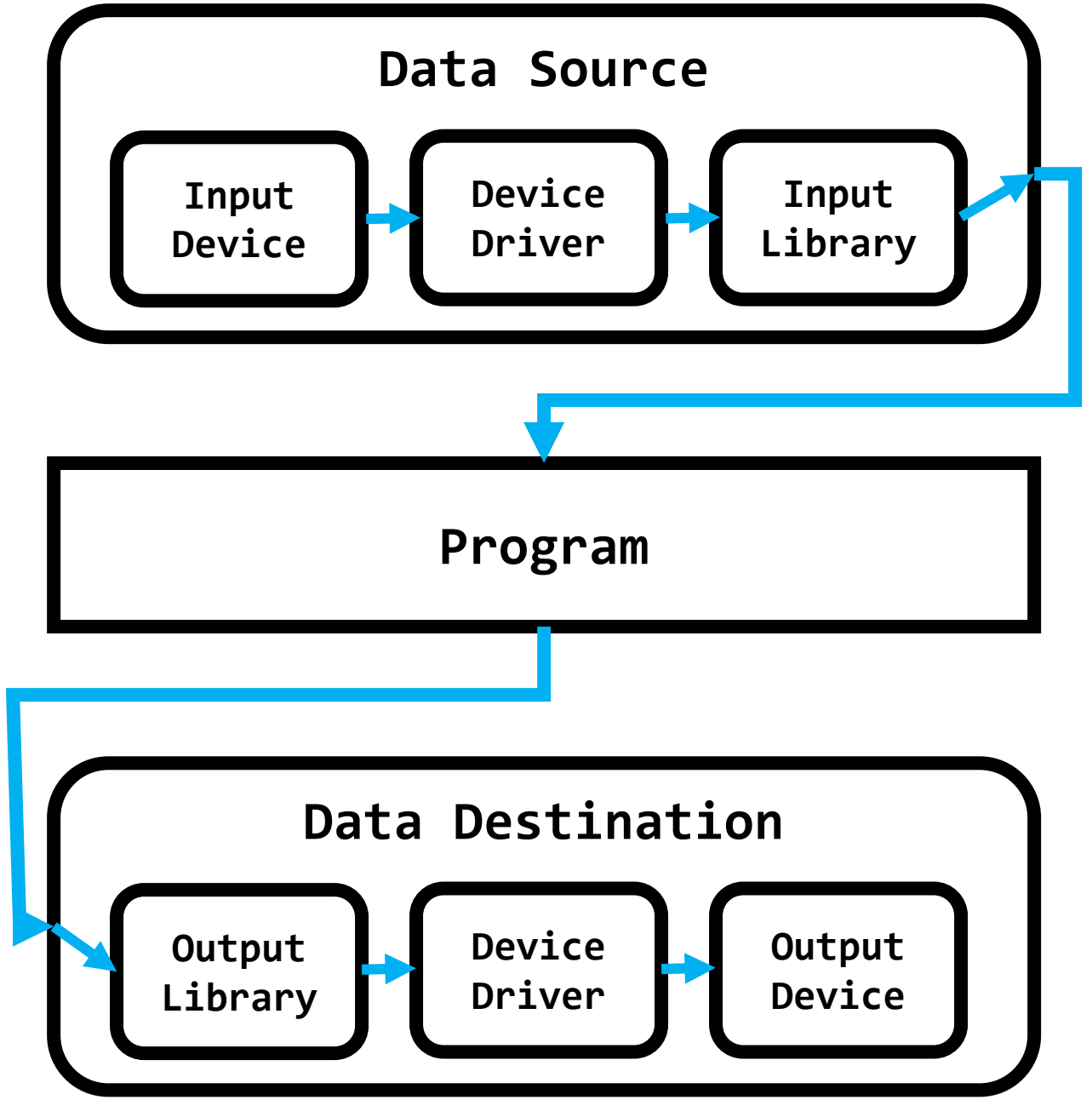
Details: An `ostream`…

- …converts the data types (e.g., `int`, `string`) into sequences of characters.
- …send the characters to some output device (e.g., console, string, file).



Stream Model

- **Support:** Both input and output models.
- **Stored Format**
  - Typically in text (i.e., characters).
  - Also in other formats (e.g., binary).
- **OOP Paradigm**
  - **Inheritance.** New streams can be supported with the stream model.
  - **Polymorphism.** New streams can be used the same as old streams.

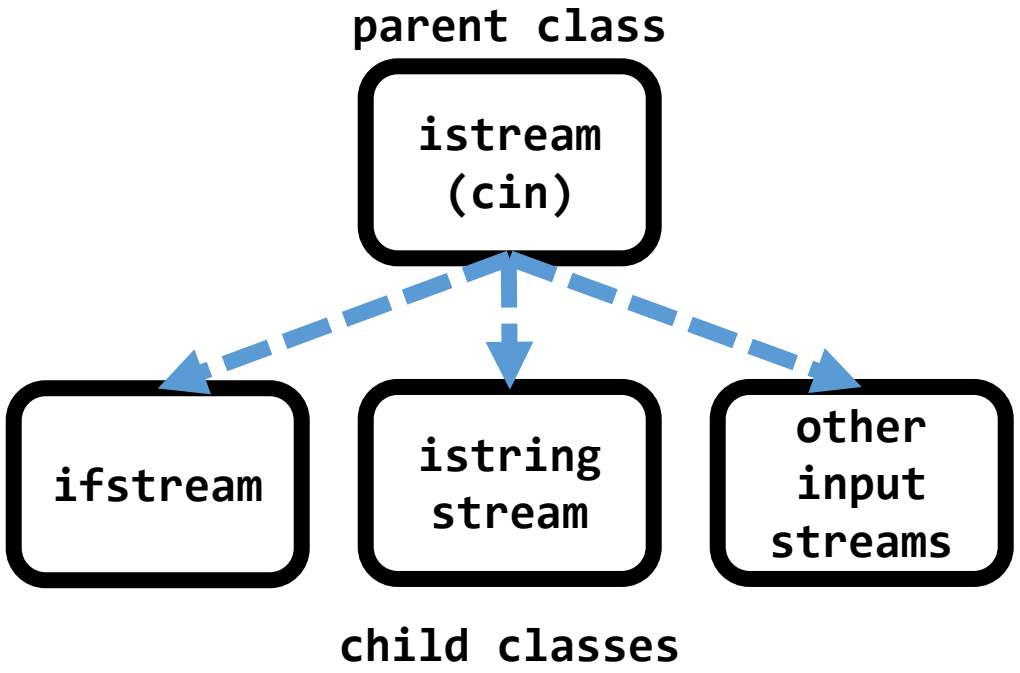


Input and Output Model with OOP Paradigm

Inheritance with Input and Output Model

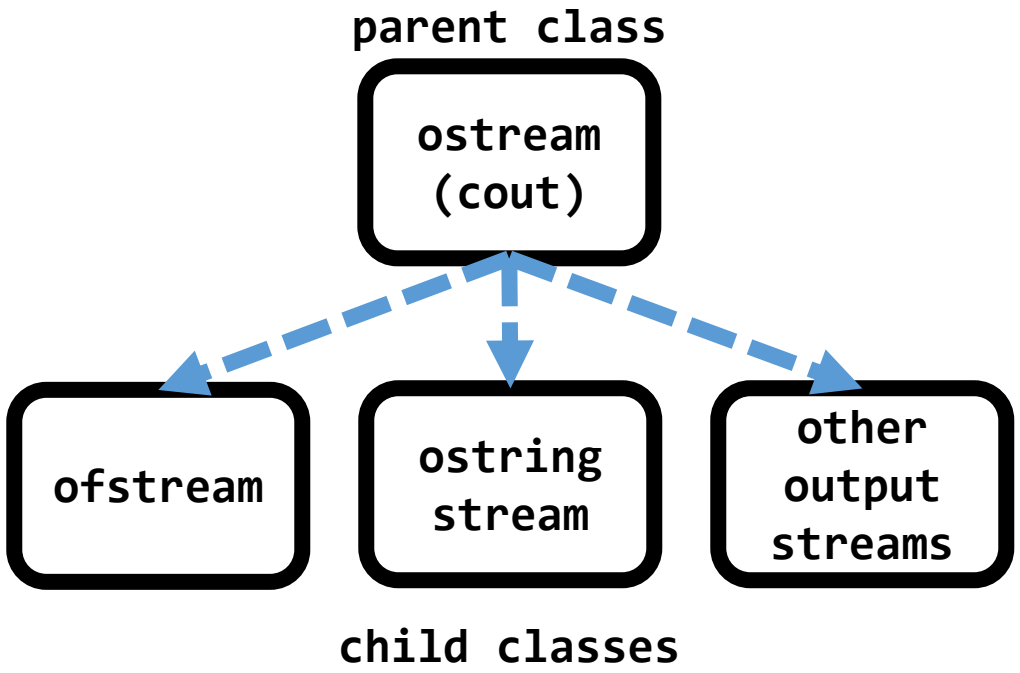
Input Model

- Can use the same functions.
- Can use the same stream states.



Output Model

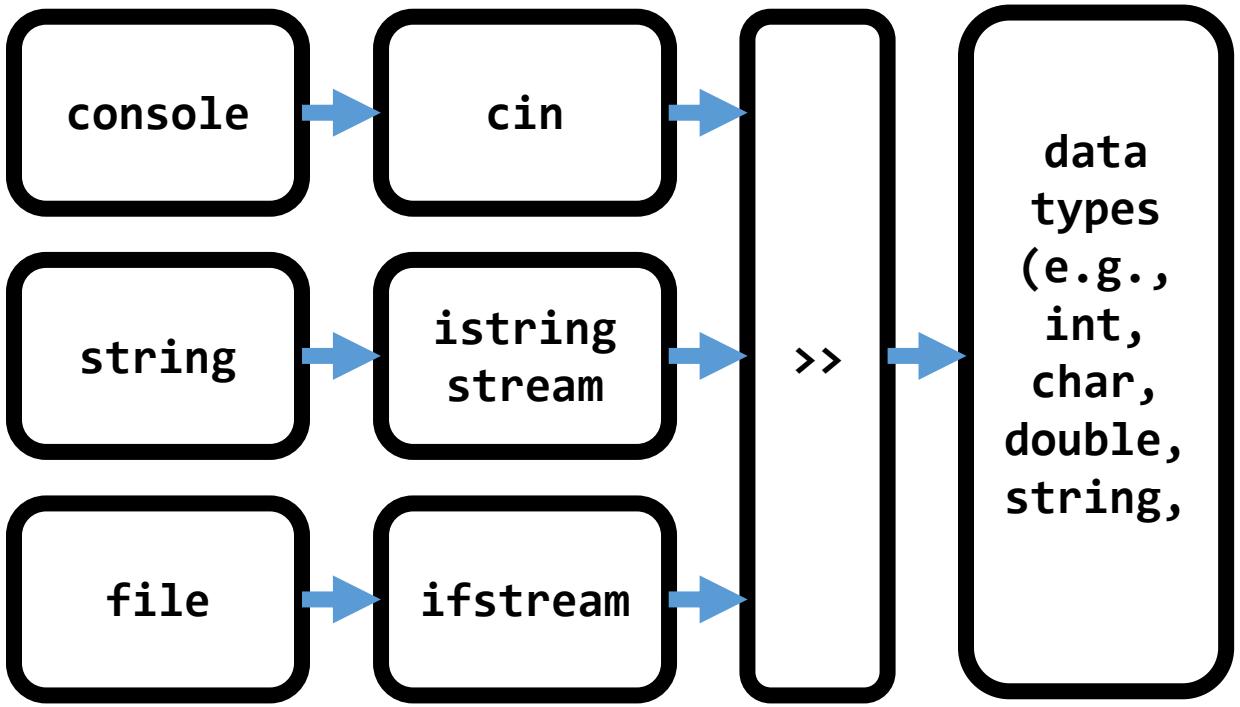
- Can use the same functions.
- Can use the same stream states.



Polymorphism with Input Model

About

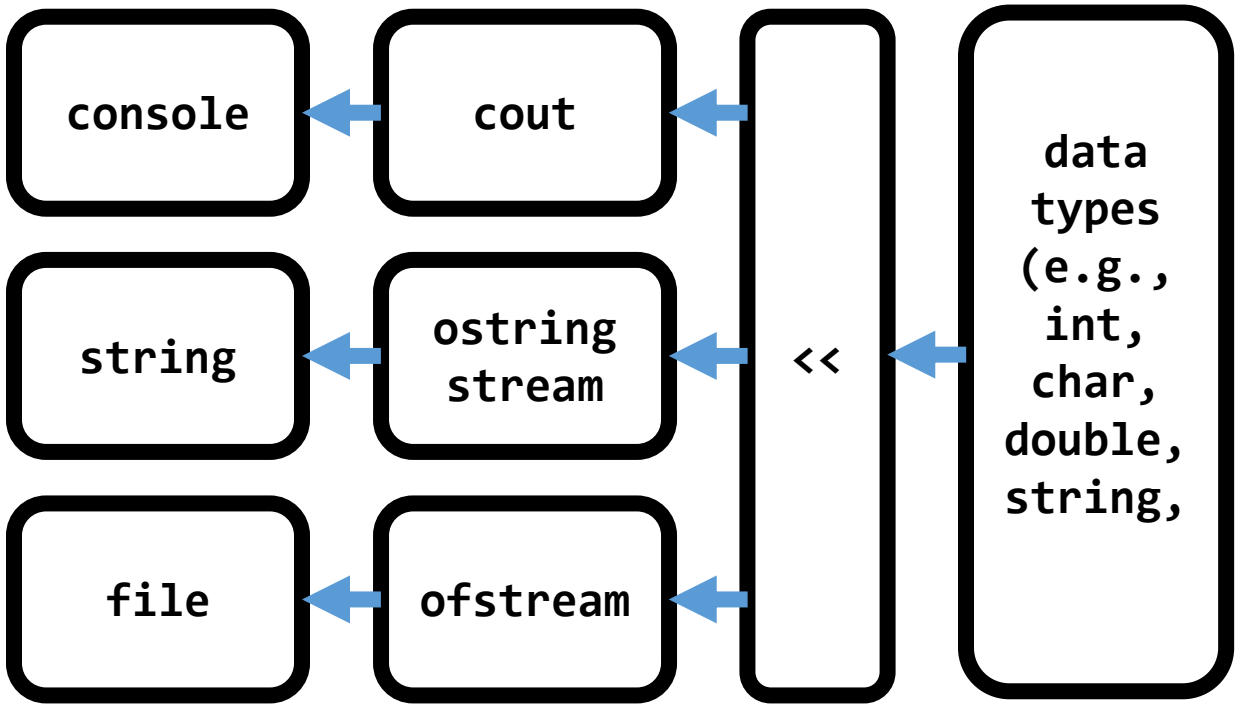
- The extraction operator >> can be used identically with the different input streams.
- While the usage is identical, its functionality will be adapted according to its type.
- Can also be applied for functions with `istream` parameters, such as for `getline()`.



Polymorphism with Output Model

About

- The insertion operator << can be used identically with the different output streams.
- While the usage is identical, its functionality will be adapted according to its type.
- Can also be applied for functions with `ostream` parameters.



Example #1: OOP Paradigm with Input Model

About

- The following program creates input streams for console, string, and file.
- Each extracted value from the input streams are then output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a word: ";
11   string input = "";
12   cin >> input;
13   cout << "cin: " << input << endl;
14
15   string text = "second";
16   istringstream sin(text);
17   sin >> input;
18   cout << "sin: " << input << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   fin >> input;
23   cout << "fin: " << input << endl;
24
25   return 0;
26 }
```

Example #1: OOP Paradigm with Input Model

Lines 10 to 13. Input stream code for inputting from the console.

Lines 15 to 18. Input stream code for inputting from a string variable.

Lines 20 to 23. Input stream code for inputting from an input file.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a word: ";
11   string input = "";
12   cin >> input;
13   cout << "cin: " << input << endl;
14
15   string text = "second";
16   istringstream sin(text);
17   sin >> input;
18   cout << "sin: " << input << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   fin >> input;
23   cout << "fin: " << input << endl;
24
25   return 0;
26 }
```

Example #1: OOP Paradigm with Input Model

Console Output

```
user@computer:/mnt/c/code$ ./a.out
Enter a word: first
cin: first
sin: second
fin: third
user@computer:/mnt/c/code$
```

Text File

```
third
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a word: ";
11   string input = "";
12   cin >> input;
13   cout << "cin: " << input << endl;
14
15   string text = "second";
16   istringstream sin(text);
17   sin >> input;
18   cout << "sin: " << input << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   fin >> input;
23   cout << "fin: " << input << endl;
24
25   return 0;
26 }
```

Example #2: OOP Paradigm with Input Model

About

- The following program creates input streams for console, string, and file.
- Each extracted line from the input streams are then output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a sentence: ";
11   string line = "";
12   getline(cin, line);
13   cout << "getline(cin, line): " << line << endl;
14
15   string text = "This is the second line.";
16   istringstream sin(text);
17   getline(sin, line);
18   cout << "getline(sin, line): " << line << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   getline(fin, line);
23   cout << "getline(fin, line): " << line << endl;
24
25   return 0;
26 }
```

Example #2: OOP Paradigm with Input Model

Lines 10 to 13. Input stream code for inputting a line from the console.

Lines 15 to 18. Input stream code for inputting a line from a string variable.

Lines 20 to 23. Input stream code for inputting a line from an input file.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a sentence: ";
11   string line = "";
12   getline(cin, line);
13   cout << "getline(cin, line): " << line << endl;
14
15   string text = "This is the second line.";
16   istringstream sin(text);
17   getline(sin, line);
18   cout << "getline(sin, line): " << line << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   getline(fin, line);
23   cout << "getline(fin, line): " << line << endl;
24
25   return 0;
26 }
```

Example #2: OOP Paradigm with Input Model

Console Output

```
user@computer:/mnt/c/code$ ./a.out
Enter a sentence: This is the first line.
getline(cin, line): This is the first line.
getline(sin, line): This is the second line.
getline(fin, line): This is the third line.
user@computer:/mnt/c/code$
```

Text File

```
This is the third line.
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cin, std::cout;
6  using std::endl, std::string;
7  using std::istringstream, std::ifstream;
8
9  int main() {
10   cout << "Enter a sentence: ";
11   string line = "";
12   getline(cin, line);
13   cout << "getline(cin, line): " << line << endl;
14
15   string text = "This is the second line.";
16   istringstream sin(text);
17   getline(sin, line);
18   cout << "getline(sin, line): " << line << endl;
19
20   string file = "input.txt";
21   ifstream fin(file);
22   getline(fin, line);
23   cout << "getline(fin, line): " << line << endl;
24
25   return 0;
26 }
```

Example #3: OOP Paradigm with Output

About

- The following program creates outputs streams for console, string, and file.
- Each output stream accepts a `Text` object, then output to the corresponding output stream.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cout, std::endl, std::string;
6  using std::ostream, std::ostringstream, std::ofstream;
7
8  struct Text {
9    string value = "";
10 };
11
12 ostream& operator<<(std::ostream& out, const Text& text) {
13   out << text.value;
14   return out;
15 }
16
17 int main() {
18   Text text = {"Hello, world!"};
19   cout << "cout: " << text << endl;
20
21   string output = "";
22   ostringstream sout(output);
23   sout << text << endl;
24   cout << "sout: " << sout.str() << endl;
25
26   string file = "output.txt";
27   ofstream fout(file);
28   fout << "fout: " << text << endl;
29
30   return 0;
31 }
```

Example #3: OOP Paradigm with Output

Lines 8 to 10. A struct class definition for the Text data type.

Lines 12 to 15. A function for the insertion operator that takes in an output stream parameter.

Lines 17 to 28. The usage of output streams for console, string, and file.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cout, std::endl, std::string;
6  using std::ostream, std::ostringstream, std::ofstream;
7
8  struct Text {
9    string value = "";
10 };
11
12 ostream& operator<<(std::ostream& out, const Text& text) {
13   out << text.value;
14   return out;
15 }
16
17 int main() {
18   Text text = {"Hello, world!"};
19   cout << "cout: " << text << endl;
20
21   string output = "";
22   ostringstream sout(output);
23   sout << text << endl;
24   cout << "sout: " << sout.str() << endl;
25
26   string file = "output.txt";
27   ofstream fout(file);
28   fout << "fout: " << text << endl;
29
30   return 0;
31 }
```

Example #3: OOP Paradigm with Output

Console Output

```
user@computer:/mnt/c/code$ ./a.out
cout: Hello, world!
sout: Hello, world!
user@computer:/mnt/c/code$
```

Text File

```
fout: Hello, world!
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  #include <fstream>
5  using std::cout, std::endl, std::string;
6  using std::ostream, std::ostringstream, std::ofstream;
7
8  struct Text {
9    string value = "";
10 };
11
12 ostream& operator<<(std::ostream& out, const Text& text) {
13   out << text.value;
14   return out;
15 }
16
17 int main() {
18   Text text = {"Hello, world!"};
19   cout << "cout: " << text << endl;
20
21   string output = "";
22   ostringstream sout(output);
23   sout << text;
24   cout << "sout: " << sout.str() << endl;
25
26   string file = "output.txt";
27   ofstream fout(file);
28   fout << "fout: " << text;
29
30   return 0;
31 }
```

Formatting Output

Formatting Output

- Can control how output displays for numeric, string data:
  - size
  - position
  - number of digits
- Requires **`iomanip`** header file

Manipulators

- Modify the state of a stream
- Most manipulators are “sticky.”
  - They are set and are permanent until changed again.

- http://www.cplusplus.com/reference/library/manipulators/

Stream Manipulators

- Used to control how an output field is displayed

- Some affect just the next value displayed (not sticky):
  - `setw(x)`: print in a field at least `x` spaces wide. Use more spaces if field is not wide enough
- All this is the kind of stuff you look up when you need it

Stream Manipulators

- Some affect values until changed again (sticky):
  - **`fixed`**: use decimal notation for floating-point values
  - **`setprecision(x)`** : when used with **`fixed`**, print floating-point value using **`x`** digits after the decimal. Without **`fixed`**, print floating-point value using **`x`** significant digits
  - **`showpoint`**: always print decimal for floating-point values

Stream Manipulators

Stream Manipulator	Description
`setw(n)`	Establishes a print field of *n* spaces.
`fixed`	Displays floating-point numbers in fixed point notation.
`showpoint`	Causes a decimal point and trailing zeroes to be displayed, even if there is no fractional part.
`setprecision(n)`	Sets the precision of floating-point numbers.
`left`	Causes subsequent output to be left justified.
`right`	Causes subsequent output to be right justified.

String Streams

Input String Stream

Input String Stream

Motivation: Programmer wants to read from `string` data instead of standard input (e.g., keyboard).

Solution: Use an input string stream to read from associated `string` data.

Setup: Use `istringstream` variable from `sstream` class.

Usage: Similar to `cin`.

Example Syntax

```
// minimal example using input string stream
#include <sstream>
#include <string>
using std::string, std::istringstream;

int main() {
    string word = "";
    istringstream sin(<some string value>);

    sin >> word;

    return 0;
}
```

Example, Reading Words from a String

About

- The following program reads a string as a stream.
- String values from the stream are then extracted and output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string word = "";
9    string sentence = "Welcome to C++ programming!";
10
11   istringstream sin(sentence);
12   for (int i = 0; i < 4; ++i) {
13     sin >> word;
14     cout << "word: " << word << endl;
15   }
16   return 0;
17 }
```

Example, Reading Words from a String

Lines 8 to 9. String variables for reading and streaming.

Lines 11 to 15. The string is read into a stream, and then extracted and output to the console.

```
1   #include <iostream>
2   #include <string>
3   #include <sstream>
4   using std::cout, std::endl;
5   using std::string, std::istringstream;
6
7   int main() {
8     string word = "";
9     string sentence = "Welcome to C++ programming!";
10
11    istringstream sin(sentence);
12    for (int i = 0; i < 4; ++i) {
13      sin >> word;
14      cout << "word: " << word << endl;
15    }
16    return 0;
17  }
```

Example, Reading Words from a String

Console Output

```
user@computer:/mnt/c/code$ ./a.out
word: Welcome
word: to
word: C++
word: programming!
user@computer:/mnt/c/code$
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string word = "";
9    string sentence = "Welcome to C++ programming!";
10
11   istringstream sin(sentence);
12   for (int i = 0; i < 4; ++i) {
13     sin >> word;
14     cout << "word: " << word << endl;
15   }
16   return 0;
17 }
```

Example, Reading Names and Ages from Console

About

- The following program reads lines from the console containing a name and age.
- The user is prompted to enter the information, which is formatted and output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string lastName = "";
9    string firstName = "";
10   int age = 0;
11   string line = "";
12
13   cout << "Enter first name, last name, and age." << endl;
14   cout << "(Type \"quit\" to exit program.)" << endl;
15   while (true) {
16     cout << "> ";
17     getline(cin, line);
18     istringstream sin(line);
19
20     sin >> firstName;
21     if ("quit" == firstName) { break; }
22     sin >> lastName;
23     sin >> age;
24
25     cout << lastName << ", ";
26     cout << firstName << " | ";
27     cout << age << " years old" << endl;
28   }
29   return 0;
30 }
```

Example, Reading Names and Ages from Console

Lines 8 to 11. Variables are initialized.

Lines 13 to 15. The user is prompted to enter name and age, or quit when done.

Lines 16 to 18. The user input is received and streamed.

Lines 20 to 23. The data is inserted into variables, or the program quits.

Lines 25 to 28. The curret name and age is output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string lastName = "";
9    string firstName = "";
10   int age = 0;
11   string line = "";
12
13   cout << "Enter first name, last name, and age." << endl;
14   cout << "(Type \"quit\" to exit program.)" << endl;
15   while (true) {
16     cout << "> ";
17     getline(cin, line);
18     istringstream sin(line);
19
20     sin >> firstName;
21     if ("quit" == firstName) { break; }
22     sin >> lastName;
23     sin >> age;
24
25     cout << lastName << ", ";
26     cout << firstName << " | ";
27     cout << age << " years old" << endl;
28   }
29   return 0;
30 }
```

Example, Reading Names and Ages from Console

Console Output

```
user@computer:/mnt/c/code$ ./a.out
Enter first name, last name, and age.
(Type "quit" to exit program.)
> Bjarne Stroustrup 70
Stroustrup, Bjarne | 70 years old
> Vint Cerf 77
Cerf, Vint | 77 years old
> Linus Torvalds 51
Torvalds, Linus | 51 years old
> quit
user@computer:/mnt/c/code$
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string lastName = "";
9    string firstName = "";
10   int age = 0;
11   string line = "";
12
13   cout << "Enter first name, last name, and age." << endl;
14   cout << "(Type \"quit\" to exit program.)" << endl;
15   while (true) {
16     cout << "> ";
17     getline(cin, line);
18     istringstream sin(line);
19
20     sin >> firstName;
21     if ("quit" == firstName) { break; }
22     sin >> lastName;
23     sin >> age;
24
25     cout << lastName << ", ";
26     cout << firstName << " | ";
27     cout << age << " years old" << endl;
28   }
29   return 0;
30 }
```

Tokenizing Strings

Tokenizing Strings

About

- ***delimiter*** – a type of character that indicates the end of one string and the start of another
- ***token*** – a type of substring between delimiters
- ***tokenize*** – the parsing process of extracting tokens from a string

Example Syntax

```
getline(stream, token, delimiter);
```

- **`getline()`:** function with three-arguments to handle tokenization
- ***`stream`*:** the input stream object that stores the text to tokenize
- ***`token`*:** the variable to store the token
- ***`delimiter`*:** the character to indicate the delimiter

Example, Tokening a Single String

About

- The following program takes a date that is stored as a string value, then parses the string into a date in a different format.
- At the end of the program, the two date formats are output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string date = "02/14/2021";
9    istringstream sin(date);
10
11   string month = "";
12   getline(sin, month, '/');
13   string day = "";
14   getline(sin, day, '/');
15   string year = "";
16   getline(sin, year);
17
18   cout << "US Date:     " << date << endl;
19   cout << "Global Date: ";
20   cout << year << "-" << month;
21   cout << "-" << day << endl;
22   return 0;
23 }
```

Example, Tokening a Single String

Lines 8 to 9. Initializing the date string and input string stream.

Lines 11 to 16. Tokenizing the date string into into month, day, and year strings from the slash delimiter.

Lines 18 to 22. Outputting the two date formats to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string date = "02/14/2021";
9    istringstream sin(date);
10
11   string month = "";
12   getline(sin, month, '/');
13   string day = "";
14   getline(sin, day, '/');
15   string year = "";
16   getline(sin, year);
17
18   cout << "US Date:     " << date << endl;
19   cout << "Global Date: ";
20   cout << year << "-" << month;
21   cout << "-" << day << endl;
22   return 0;
23 }
```

Example, Tokening a Single String

Console Output

```
user@computer:/mnt/c/code$ ./a.out
US Date:     02/14/2021
Global Date: 2021-02-14
user@computer:/mnt/c/code$
```

```cpp
#include <iostream>
#include <string>
#include <sstream>
using std::cout, std::endl;
using std::string, std::istringstream;

int main() {
  string date = "02/14/2021";
  istringstream sin(date);

  string month = "";
  getline(sin, month, '/');
  string day = "";
  getline(sin, day, '/');
  string year = "";
  getline(sin, year);

  cout << "US Date:     " << date << endl;
  cout << "Global Date: ";
  cout << year << "-" << month;
  cout << "-" << day << endl;
  return 0;
}
```

Example, Tokening List of Strings

About

- The following program loads a string of phone numbers into a string stream, reads each line in the string, and then parses each line for specific values.
- The lines are parsed into tokens, then output to the console.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string home = "1-979-555-1234\n";
9    string cell = "1-512-888-4321\n";
10   string work = "1-210-444-9876";
11
12   string text = home + cell + work;
13   istringstream lineSin(text);
14   string line = "";
15   while (getline(lineSin, line)) {
16     cout << "phone number: " << line << endl;
17
18     string country = "";
19     string area    = "";
20     string local   = "";
21     istringstream tokenSin(line);
22     getline(tokenSin, country, '-');
23     getline(tokenSin, area, '-');
24     getline(tokenSin, local);
25
26     cout << "   country: " << country << endl;
27     cout << "   area:    " << area << endl;
28     cout << "   local:   " << local << endl;
29   }
30   return 0;
31 }
```

Example, Tokening List of Strings

Lines 8 to 12. Different string values are concatenated and then initialized into a string stream.

Line 13. An input string stream is initialized on the string of concatenated phone numbers.

Lines 14 to 15. The while loop stores each line into a string variable until there are no more lines.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string home = "1-979-555-1234\n";
9    string cell = "1-512-888-4321\n";
10   string work = "1-210-444-9876";
11
12   string text = home + cell + work;
13   istringstream lineSin(text);
14   string line = "";
15   while (getline(lineSin, line)) {
16     cout << "phone number: " << line << endl;
17
18     string country = "";
19     string area    = "";
20     string local   = "";
21     istringstream tokenSin(line);
22     getline(tokenSin, country, '-');
23     getline(tokenSin, area, '-');
24     getline(tokenSin, local);
25
26     cout << "   country: " << country << endl;
27     cout << "   area:    " << area << endl;
28     cout << "   local:   " << local << endl;
29   }
30   return 0;
31 }
```

Example, Tokening List of Strings

Line 16. The entire line is output to the console.

Lines 18 to 20. The string variables are initialized into empty strings.

Line 21. Another input string stream is initialized, but for the current line of a single phone number.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string home = "1-979-555-1234\n";
9    string cell = "1-512-888-4321\n";
10   string work = "1-210-444-9876";
11
12   string text = home + cell + work;
13   istringstream lineSin(text);
14   string line = "";
15   while (getline(lineSin, line)) {
16     cout << "phone number: " << line << endl;
17
18     string country = "";
19     string area    = "";
20     string local   = "";
21     istringstream tokenSin(line);
22     getline(tokenSin, country, '-');
23     getline(tokenSin, area, '-');
24     getline(tokenSin, local);
25
26     cout << "   country: " << country << endl;
27     cout << "   area:    " << area << endl;
28     cout << "   local:   " << local << endl;
29   }
30   return 0;
31 }
```

Example, Tokening List of Strings

Lines 22 to 24. For each `getline()` function:

- 1st argument: handles input string stream for current phone number
- 2nd argument: stores output to the variable
- 3rd argument: if any, stops the tokenizing at the delimiter value

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string home = "1-979-555-1234\n";
9    string cell = "1-512-888-4321\n";
10   string work = "1-210-444-9876";
11
12   string text = home + cell + work;
13   istringstream lineSin(text);
14   string line = "";
15   while (getline(lineSin, line)) {
16     cout << "phone number: " << line << endl;
17
18     string country = "";
19     string area    = "";
20     string local   = "";
21     istringstream tokenSin(line);
22     getline(tokenSin, country, '-');
23     getline(tokenSin, area, '-');
24     getline(tokenSin, local);
25
26     cout << "   country: " << country << endl;
27     cout << "   area:    " << area << endl;
28     cout << "   local:   " << local << endl;
29   }
30   return 0;
31 }
```

Example, Tokening List of Strings

Lines 26 to 28. The tokens from the phone number are output to the console.

- **Country.** The phone number's country code.
- **Area.** The phone number's area code.
- **Local.** The phone number's local number.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cout, std::endl;
5  using std::string, std::istringstream;
6
7  int main() {
8    string home = "1-979-555-1234\n";
9    string cell = "1-512-888-4321\n";
10   string work = "1-210-444-9876";
11
12   string text = home + cell + work;
13   istringstream lineSin(text);
14   string line = "";
15   while (getline(lineSin, line)) {
16     cout << "phone number: " << line << endl;
17
18     string country = "";
19     string area    = "";
20     string local   = "";
21     istringstream tokenSin(line);
22     getline(tokenSin, country, '-');
23     getline(tokenSin, area, '-');
24     getline(tokenSin, local);
25
26     cout << "   country: " << country << endl;
27     cout << "   area:    " << area << endl;
28     cout << "   local:   " << local << endl;
29   }
30   return 0;
31 }
```

Example, Tokening List of Strings

Console Output

```
user@computer:/mnt/c/code$ ./a.out
phone number: 1-979-555-1234
   country: 1
   area:    979
   local:   555-1234
phone number: 1-512-888-4321
   country: 1
   area:    512
   local:   888-4321
phone number: 1-210-444-9876
   country: 1
   area:    210
   local:   444-9876
user@computer:/mnt/c/code$
```

```cpp
#include <iostream>
#include <string>
#include <sstream>
using std::cout, std::endl;
using std::string, std::istringstream;

int main() {
  string home = "1-979-555-1234\n";
  string cell = "1-512-888-4321\n";
  string work = "1-210-444-9876";

  string text = home + cell + work;
  istringstream lineSin(text);
  string line = "";
  while (getline(lineSin, line)) {
    cout << "phone number: " << line << endl;

    string country = "";
    string area    = "";
    string local   = "";
    istringstream tokenSin(line);
    getline(tokenSin, country, '-');
    getline(tokenSin, area, '-');
    getline(tokenSin, local);

    cout << "   country: " << country << endl;
    cout << "   area:    " << area << endl;
    cout << "   local:   " << local << endl;
  }
  return 0;
}
```

Output String Stream

Output String Stream

Motivation: Programmer wants to output `string` data into a string buffer instead of to the screen.

Solution: Write to an output string stream from associated `string` data.

Setup: Use `ostringstream` variable from `sstream` class.

Usage: Similar to `cout`.

Example Syntax

```
// minimal example using output string stream
#include <iostream>
#include <string>
#include <sstream>
using std::cout, std::endl;
using std::string, std::ostringstream;

int main() {
  ostringstream sout;
  sout << "hello ";
  sout << "world";

  cout << sout.str() << endl;
  return 0;
}
```

Example, Inputting Words and Outputting Questions

About

- The following program prompts the user to input words, then outputs the words to the console as a question.
- The user continues until quitting the program.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::ostringstream;
6
7  int main() {
8    cout << "Enter: subject verb object" << endl;
9    cout << "(Type \"quit\" to quit.)" << endl;
10
11   string subject = "";
12   string verb = "";
13   string object = "";
14   while (true) {
15     ostringstream sout;
16     cout << "> ";
17     cin >> subject;
18     if ("quit" == subject) { break; }
19     cin >> verb;
20     cin >> object;
21     sout << "Do " << subject << " ";
22     sout << verb << " ";
23     sout << object << "?";
24     cout << sout.str() << endl;
25   }
26   return 0;
27 }
```

Example, Inputting Words and Outputting Questions

Lines 8 to 9. User is prompted to enter words.

Lines 11 to 13. Variables are initialized.

Lines 15 to 20. User input is stored in stream, or program quits.

Lines 21 to 24. User input is formatted into a question.

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::ostringstream;
6
7  int main() {
8    cout << "Enter: subject verb object" << endl;
9    cout << "(Type \"quit\" to quit.)" << endl;
10
11   string subject = "";
12   string verb = "";
13   string object = "";
14   while (true) {
15     ostringstream sout;
16     cout << "> ";
17     cin >> subject;
18     if ("quit" == subject) { break; }
19     cin >> verb;
20     cin >> object;
21     sout << "Do " << subject << " ";
22     sout << verb << " ";
23     sout << object << "?";
24     cout << sout.str() << endl;
25   }
26   return 0;
27 }
```

Example, Inputting Words and Outputting Questions

Console Output

```
user@computer:/mnt/c/code$ ./a.out
Enter: subject verb object
(Type "quit" to quit.)
> cats chase mice
Do cats chase mice?
> birds sing songs
Do birds sing songs?
> people eat food
Do people eat food?
> quit
user@computer:/mnt/c/code$
```

```
1  #include <iostream>
2  #include <string>
3  #include <sstream>
4  using std::cin, std::cout, std::endl;
5  using std::string, std::ostringstream;
6
7  int main() {
8    cout << "Enter: subject verb object" << endl;
9    cout << "(Type \"quit\" to quit.)" << endl;
10
11   string subject = "";
12   string verb = "";
13   string object = "";
14   while (true) {
15     ostringstream sout;
16     cout << "> ";
17     cin >> subject;
18     if ("quit" == subject) { break; }
19     cin >> verb;
20     cin >> object;
21     sout << "Do " << subject << " ";
22     sout << verb << " ";
23     sout << object << "?";
24     cout << sout.str() << endl;
25   }
26   return 0;
27 }
```