I/O and File (Stream)

Seonghyun Park

dd December, 2018

Outline

Today's Goal

- Persist data or communicate with outside the program using I/O and file
- Get familiar with standard library functions related to I/O and stream, i.e. `iostream`, `sstream`, `fstream`

Motivation

- Unless you use I/O, your data is collapsed when your program terminates
- your program can communicate with outside world via various types of i/o (display, keyboard inputs, fies to disk, network packets)
- We focus on C++ I/O stream library, which provides universal I/O interfaces
- `ostream` converts C++ objects to output byte streams, and `iostream` vice versa.

Syntax

- Two most important operators
  - << operator converts C++ objects into byte streams; `std::ostream& operator<<(std::ostream& os, const T& t)`
  - >> operator converts byte streams into C++ objects; `std::istream& operator>>(std::istream& is, T& t)`
- You've already used this a lot, right?

```
// Similar to <<(std::cout, "Hello, World!")
std::cout << "Hello, World!\n";

// Similar to >>(std::cout, c)
std::cin >> c;
```

Operator Overloading

- I/O stream operators can also be overloaded for user-defined classes
- `operator<<` and `operator>>` are overloaded as non-member functions

```
std::ostream&
operator<<(std::ostream& os, const Point& p)
{
  os << '(' << p.x << ',' << p.y << ')';

  // DON't forget to return os!
  return os;
}

std::cout << p << '\n'; // (3,4)
```

String Stream

- We can redirect streams to or from `std::strings`. (found in `<sstream>`)
  - `istringstream` for reading from a `string`
  - `ostringstream` for writing to a `string`
  - `stringstream` for reading from and writing to a `string`

Example

```
std::ostringstream oss;

oss << "{name: " << name << ", age: " << age << '}';

// Results from ostringstream can be read using
// `.str()` method
std::cout << oss.get() << '\n';
```

File stream

- It is useful to persist large data to persistent storage e.g. databases
- We focus on reading from and writing to files with `<fstream>`
  - `ofstream` for reading from a file
  - `ifstream` for writing to a file
  - `fstream` for reading from and writing to a file

Example

```
ofstream ofs{"output"};
if (!ofs)
  error("couldn't open output file")

ofs << "Hello, file\n";
```

Tips

- operator¿¿ skips whitespaces. If you want to read inputs including whitespaces, use `get` or `getline` functions.

```
istream& is = std::cin;
char c; std::string name;

if (is.get(c) && c != '"')
  error("wrong format");
while (is.get(c) && c != '"') {
  name += c;
}

std::cout << name << '\n';
std::string dst;
getline(std::cin, dst);
```