HIGH-LEVEL PROGRAMMING 2

String Streams

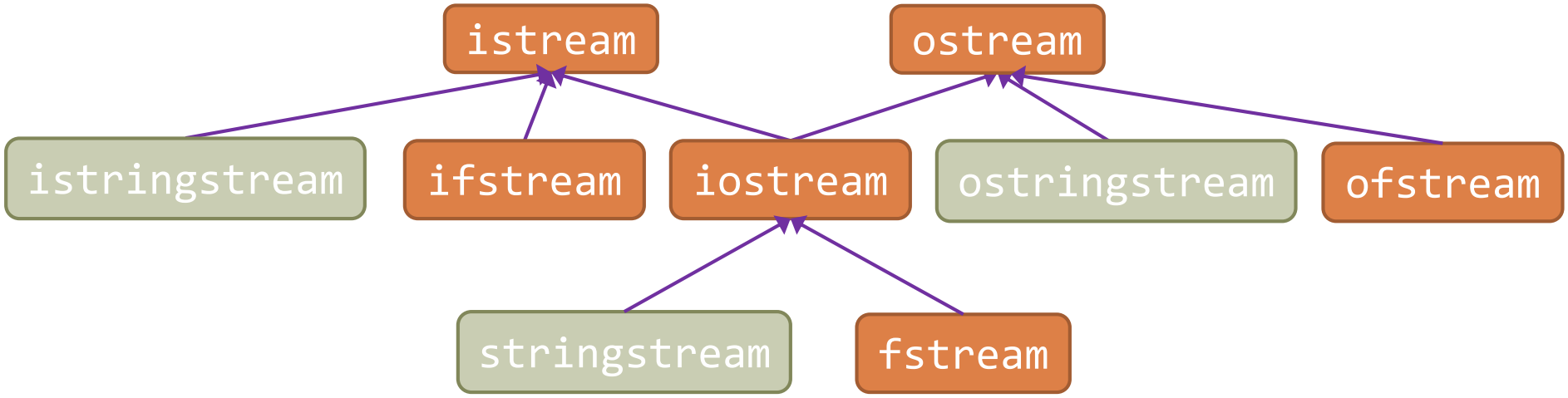
by Prasanna Ghali

I/O Streams

- *Stream* represents basic unit of communication between C++ program and its I/O environment
- Stream is channel between a *source* and a *destination* which allows source to push formatted data to destination

I/O Streams Hierarchy

- `std::istream` connects [source] input device, file, or `std::string` to [destination] program
- `std::ostream` connects [source] program to [destination] output device, file, or `std::string`



std::ostream

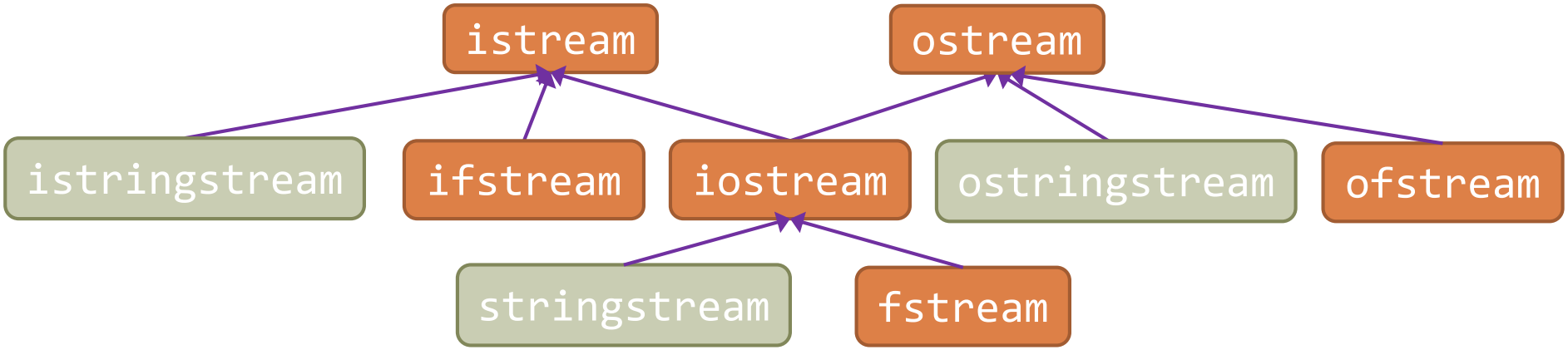

```
template <typename T>
std::ostream& operator<<(std::ostream& os,
                         std::complex<T> const& rhs) {
  os << "(" << rhs.real() << ", " << rhs.imag() << ")";
  return os;
}

std::complex<double> cd{1.1, 2.2};
std::cout << cd << "\n";

std::ofstream ofs{"file.txt"};
ofs << cd << "\n";
ofs.close();
```

String Streams

- You can use `string` as source of `istream` or target for `ostream`
  - `istream` that reads from `string` called `istringstream`
  - `ostream` that stores characters written to it in a `string` is called `ostringstream`
- `stringstream` is *adapter class* that allows you to access `string`s as streams



Uses (1/2)

- An `istringstream` is useful for extracting numeric values from `string`

```
std::string date{"March 22, 2021"};
std::string month = ???
int day = ???
int year = ???
```

Uses (2/2)

- An `istringstream` is useful for extracting numeric values from `string`
- Conversely, an `ostringstream` can be useful for formatting output for system that requires `string` argument

```
std::string month {"March"};
int day {2};
int year {2021};
std::string date = ???;
```

Example (1/2)


```
struct Date {
  std::string month;
  int day, year;
};

Date str_to_date(std::string const& str) {
  std::istringstream iss{str};
  Date d;
  std::string comma;
  iss >> d.month >> d.day >> comma >> d.year;
  return d;
}

Date today = str_to_date("March 1, 2021");
std::cout << today.month << " " << std::setw(2)
          << std::setfill('0') << today.day
          << ", " << today.year << "\n";
```

Example (2/2)


```
struct Date {
  std::string month;
  int day, year;
};

std::string date_to_str(Date const& d) {
  std::ostringstream oss;
  oss << d.month << " " << std::setw(2) << std::setfill('0')
      << d.day << ", " << d.year;
  return oss.str();
}

Date today = str_to_date("March 1, 2021");
// write to standard output stream: March 01, 2021
std::cout << date_to_str(today) << "\n";
```

std::strings: Numeric Conversions

Function	Effect
`stoi(str, idx=nullptr, base=10)`	Converts *str* to an int
`stol(str, idx=nullptr, base=10)`	Converts *str* to a long
`stoul(str, idx=nullptr, base=10)`	Converts *str* to an unsigned long
`stoll(str, idx=nullptr, base=10)`	Converts *str* to a long long
`stoull(str, idx=nullptr, base=10)`	Converts *str* to an unsigned long long
`stof(str, idx=nullptr)`	Converts *str* to a float
`stod(str, idx=nullptr)`	Converts *str* to a double
`stold(str, idx=nullptr)`	Converts *str* to a long double
`to_string(val)`	Converts *val* to a std::string

Example: Numeric Conversions (1/2)

Function	Effect
`stoi(str, idx=nullptr, base=10)`	Converts *str* to an `int`

```
int string_to_int(std::string const& s) {
  std::istringstream iss{s};
  int ival;
  iss >> ival;
  return ival;
}
```

Example: Numeric Conversions (2/2)

Function	Effect
`to_string(val)`	Converts *val* to a `std::string`

```
std::string int_to_string(int val) {
  std::ostringstream oss;
  oss << val;
  return oss.str();
}
```