Practice with inheritance

William Hendrix

Lecture 20

Outline

- Polymorphism
- Inheritance hierarchies
- Minilab
- **Correction:** you should only declare functions `virtual` in class definition, not prototype

Polymorphism

- From Greek for “many forms”
- Refers to functions or operators with the same name but different actions depending on context
- Example

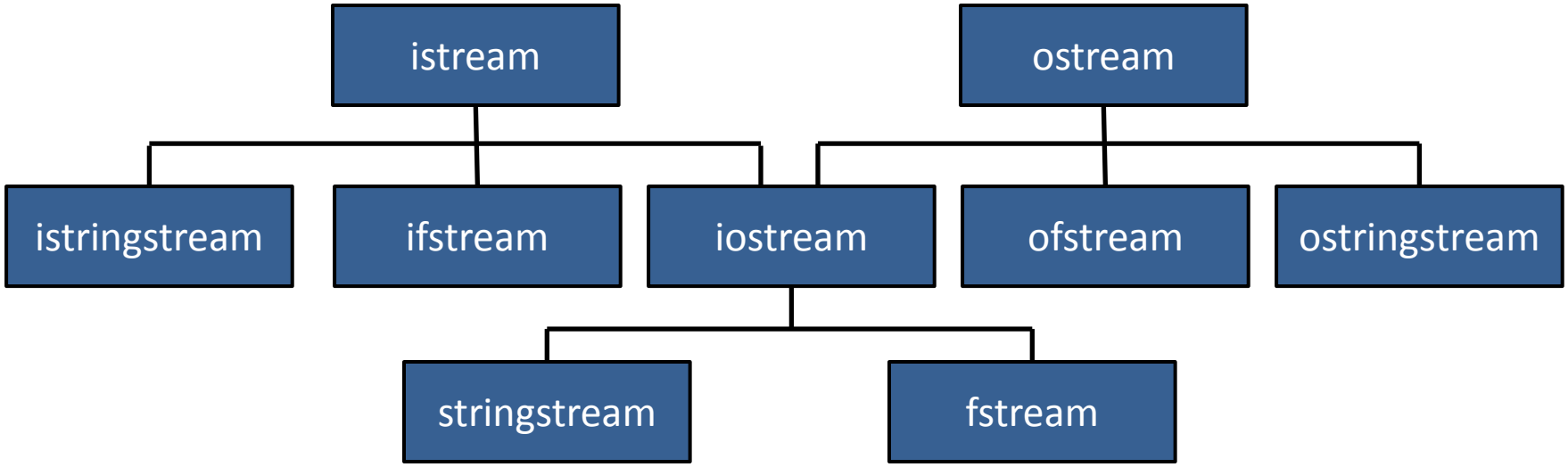
```
int x = 2 + 3;
string full_name = first + ‘ ’ + last;
```

- Overloaded operators are polymorphic
- Overloaded member functions are polymorphic
- C++ allows multiple functions with same name provided that parameters or return type are different

```
int add(int x, int y)
string add(string x, string y)
ostream& operator<<(ostream& out, CMatrix mat)
```

Inheritance hierarchies

- Class inheritance can be used to create hierarchies of inheritance



- Multiple inheritance is possible but highly discouraged
- Defining common ancestors with specializations allows for specialized objects to be treated in the same way
  - `cin`, `ifstream`, and `istringstream` can all be passed in place of `istream&` function parameter
- Classes with all `virtual` functions can be used to define common interface for subclasses

Minilab

- Write a collection of classes to represent a Menagerie and all of the Birds it contains. You should write the following classes
  - Bird: name and a birdCall() method (should never be called)
  - Design specialized types of Birds that call in different ways
  - Menagerie
    - Collection (array/vector) of birds
    - addBird(Bird& bird): adds a Bird to the Menagerie
    - experience(): makes all of the Birds call
- Example output

  You are in a menagerie:

  A duck named Bernard quacks.

  A parrot named Polly says, "How do you do?"

  A toucan named Sam caws from its perch.

  A goose named Gertrude honks with abandon.

Tonight

Lab 5 will be Monday, Feb 24