

Intermediate Programming

Day 25

Outline

- Exercise 24
- File I/O
- `std::stringstream`
- Object Oriented Programming
- Review Questions

Exercise 24

Populate **counters** so that each entry has:

- Key: collected words
- Value: the number of times that word appears in the file..

main.cpp

```
...  
void main( void )  
{  
    typedef map< string , int > s2i;  
    typedef s2i::const_iterator s2i_citer;  
    s2i counters;  
  
    ...  
    string word;  
    while( cin >> word ) counters[word]++;  
    ...  
}
```

Exercise 24

Rearrange so that each entry in the new map has an integer key, and an entire vector of strings as its value.

main.cpp

```
...  
void main( void )  
{  
    typedef map< string , int > s2i;  
    typedef s2i::const_iterator s2i_citer;  
    s2i counters;  
  
    ...  
    string word;  
    while( cin >> word ) counters[word]++;  
  
    ...  
    typedef map< int , vector< string > > i2v;  
    typedef i2v::const_iterator i2v_citer;  
  
    i2v words_by_freq;  
    for( s2i_citer it=counters.cbegin() ; it!=counters.cend() ; it++ )  
        words_by_freq[ it->second ].push_back( it->first );  
  
    ...  
}
```

Exercise 24

Output the new map's contents.

main.cpp

```
...
void main( void )
{
    typedef map< string , int > s2i;
    typedef s2i::const_iterator s2i_citer;
    s2i counters;

    ...
    string word;
    while( cin >> word ) counters[word]++;

    ...
    typedef map< int , vector< string > > i2v;
    typedef i2v::const_iterator i2v_citer;

    i2v words_by_freq;
    for( s2i_citer it=counters.cbegin() ; it!=counters.cend() ; it++ )
        words_by_freq[ it->second ].push_back( it->first );

    typedef vector< string >::const_iterator v_citer;
    for( i2v_citer it=words_by_freq.cbegin() ; it!=words_by_freq.cend() ; it++ )
    {
        std::cout << "Frequency: " << it->first << std::endl;
        for( v_citer _it=it->second.cbegin() ; _it!=it->second.cend() ; _it ++ )
            std::cout << *_it << std::endl;
    }

    ...
}
```

Exercise 24

Invoke `std::sort` from the STL to sort the contents of `vec2` and compare the implementations.

sort.cpp

```
...  
#include <algorithm>  
...  
void main( void )  
{  
    ...  
    std::sort( vec2.begin() , vec2.end() );  
    ...  
}  
...
```

```
>> ./sort  
Enter the count: 100000  
Your sort time = 223(ms)  
STL's sort time = 57(ms)  
>>
```

Outline

- Exercise 24
- File I/O
- `std::stringstream`
- Object Oriented Programming
- Review Questions

File I/O

Recall that in C++ we write/read to/from the command with handles:

- `std::cout`
- `std::cin`

using the (overloaded) insertion and extraction operators:

- `<<`
- `>>`

File I/O

- In C, `printf` wrote to `stdout` and `scanf` read from `stdin`
 - `fprintf` and `fscanf` were their counterparts for files
- In C++, we have `std::cout` and `std::cin`
 - `std::ofstream` and `std::ifstream` are their counterparts for files
 - These are declared in the file-stream header

```
#include <fstream>
```

which declares classes:
 - `ofstream`: for writing to a file (inherits* from `ostream`)
 - `ifstream`: for reading from a file (inherits* from `istream`)
 - `fstream`: for reading **and** writing to/from a file (inherits* from `ostream` and `istream`)
 - The class `ostream` (resp. `istream`) defines the extraction (resp. insertion) operator `<<` (resp. `>>`) so `ofstream` (resp. `ifstream`) inherits* it.
 - Since `fstream` derives* from both `ostream` and `istream`, it inherits* both.

File I/O (`std::ofstream`)

main.cpp

```
#include <iostream>
#include <fstream>
int main( void )
{
    std::ofstream ofile( "hello.txt" );
    ofile << "Hello, World!" << std::endl;
    return 0;
}
```

```
>> ./a.out
>> cat hello.txt
Hello, World!
>>
```

File I/O (`std::ofstream`)

- `ofstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "w" flag

```
main.cpp
#include <iostream>
#include <fstream>
int main( void )
{
    std::ofstream ofile( "hello.txt" );
    ofile << "Hello, World!" << std::endl;
    return 0;
}
```

```
>> ./a.out
>> cat hello.txt
Hello, World!
>>
```

File I/O (`std::ofstream`)

- `ofstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "w" flag
- Since `ofstream` inherits* from `ostream`, anything we can "<<" to an `ostream`, we can "<<" to the `ofstream`

```
main.cpp
#include <iostream>
#include <fstream>
int main( void )
{
    std::ofstream ofile( "hello.txt" );
    ofile << "Hello, World!" << std::endl;
    return 0;
}
```

```
>> ./a.out
>> cat hello.txt
Hello, World!
>>
```

File I/O (`std::ofstream`)

- `ofstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "w" flag
- Since `ofstream` inherits* from `ostream`, anything we can "<<" to an `ostream`, we can "<<" to the `ofstream`
- `ofstream` has a destructor* that closes the file
 - When an `ofstream` object goes out of scope (or is deleted), it automatically closes itself

```
main.cpp
#include <iostream>
#include <fstream>
int main( void )
{
    std::ofstream ofile( "hello.txt" );
    ofile << "Hello, World!" << std::endl;
    return 0;
}
```

```
>> ./a.out
>> cat hello.txt
Hello, World!
>>
```

File I/O (`std::ifstream`)

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
int main( void )
{
    std::ifstream ifile( "hello.txt" );
    std::string word;
    while( ifile>>word ) std::cout << word << ' ';
    std::cout << std::endl;
    return 0;
}
```

```
>> ./a.out
Hello, World!
>>
```

File I/O (`std::ifstream`)

- `ifstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "r" flag

```
main.cpp
#include <iostream>
#include <fstream>
#include <string>
int main( void )
{
    std::ifstream ifile( "hello.txt" );
    std::string word;
    while( ifile>>word ) std::cout << word << ' ';
    std::cout << std::endl;
    return 0;
}
```

```
>> ./a.out
Hello, World!
>>
```

File I/O (`std::ifstream`)

- `ifstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "r" flag
- Since `ifstream` inherits* from `istream`, anything we can ">>" to an `istream`, we can ">>" to the `ifstream`

main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
int main( void )
{
    std::ifstream ifile( "hello.txt" );
    std::string word;
    while( ifile>>word ) std::cout << word << ' ';
    std::cout << std::endl;
    return 0;
}
```

```
>> ./a.out
Hello, World!
>>
```


File I/O (`std::ifstream`)

- `ifstream` has a constructor* taking a `string` specifying the filename
 - Calling the constructor with a filename string is like calling `fopen` with the filename using a "r" flag
- Since `ifstream` inherits* from `istream`, anything we can ">>" to an `istream`, we can ">>" to the `ifstream`
- `ifstream` has a destructor* that closes the file
 - When an `ifstream` object goes out of scope (or is deleted), it automatically closes itself

```
main.cpp
#include <iostream>
#include <fstream>
#include <string>
int main( void )
{
    std::ifstream ifile( "hello.txt" );
    std::string word;
    while( ifile>>word ) std::cout << word << ' ';
    std::cout << std::endl;
    return 0;
}
```

```
>> ./a.out
Hello, World!
>>
```

Outline

- Exercise 24
- File I/O
- **`std::stringstream`**
- Object Oriented Programming
- Review Questions

std::stringstream

- Instead of reading or writing to console, it reads and writes to a temporary string (“buffer”) stored inside

```
main.cpp
#include <iostream>
#include <sstream>
int main( void )
{
    std::stringstream ss;
    ss << "Hello, world!" << std::endl;
    std::cout << ss.str();
    return 0;
}
```

std::stringstream

- Instead of reading or writing to console, it reads and writes to a temporary string (“buffer”) stored inside
 - The string buffer can be accessed with the member function:
`string stringstream::str(void)`

```
main.cpp
#include <iostream>
#include <sstream>
int main( void )
{
    std::stringstream ss;
    ss << "Hello, world!" << std::endl;
    std::cout << ss.str();
    return 0;
}
```

```
>> ./a.out
Hello, world!
>>
```

std::stringstream

Since it inherits from both **istream** and **ostream**

- we can insert and extract data from a **stringstream**

main.cpp

```
#include <string>
#include <iostream>
#include <sstream>
int main( void )
{
    std::stringstream ss;
    ss << "Hello" << ' ' << 35 << " world";
    std::string word1, word2;
    int num;
    ss >> word1 >> num >> word2;
    std::cout << word1 << ", " << word2 << "!" << std::endl;
    return 0;
}
```

```
>> ./a.out
Hello, world!
>>
```

std::stringstream

Since it inherits from both **istream** and **ostream**

- we can insert and extract data from a **stringstream**
- If we define an operator "<<" (resp. ">>") for inserting (resp. extracting) a new class* to an **ostream** (resp. **istream**) these can be used to insert into (resp. extract from) a **stringstream**.

main.cpp

```
#include <iostream>
#include <sstream>
#include <vector>
using namespace std;
ostream& operator<<( ostream& os , const vector<int>& vec)
{
    for( size_t i=0 ; i<vec.size() ; i++ ) os << vec[i] << ' ';
    return os;
}
istream& operator>>( istream& is , vector<int>& vec )
{
    int i;
    while( is>>i ) vec.push_back(i);
    return is;
}
int main( void )
{
    stringstream ss( "1 2 3 4 5" );
    vector<int> vec;
    ss >> vec;
    cout << vec << endl;
    return 0;
}
```

```
>> ./a.out
1 2 3 4 5
>>
```

std::stringstream

- Like the file-stream, the string-stream also comes in flavors that only do reading or writing:
 - istreamstringstream ↔ ifstream
 - ostreamstringstream ↔ ostream

Outline

- Exercise 24
- File I/O
- `std::stringstream`
- Object Oriented Programming
- Review Questions

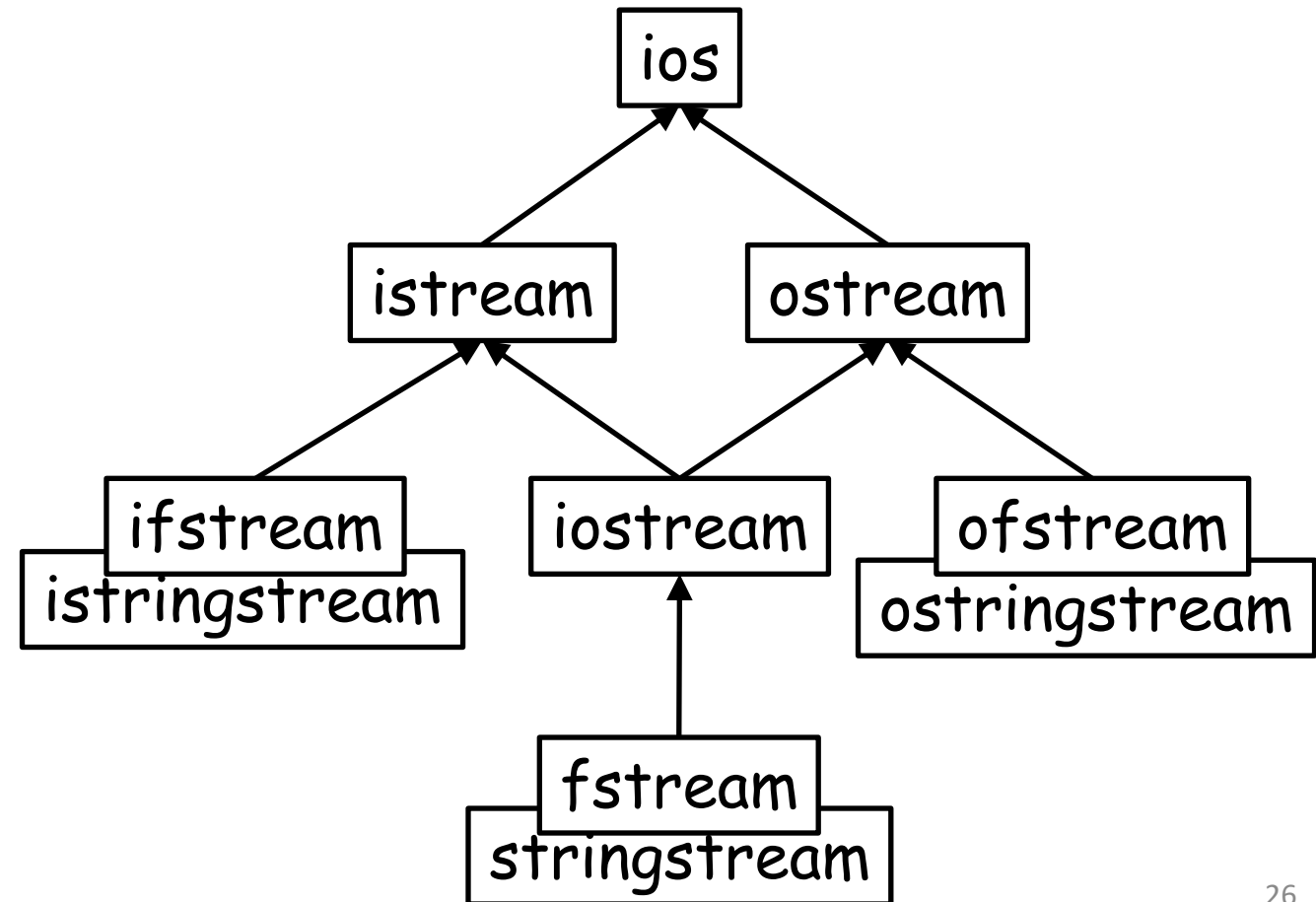
Object Oriented Programming

In C++ **classes** are similar to **structs** in C, but additionally support:

- Functionality for acting on the **class's** data
 - E.g. An **ofstream** object not only stores information about the output file stream but also provides functionality for opening/closing the file handle
- Field protection for controlling who has access to a **class's** data.
(By default, only the class itself has access.)
- Special functions called *constructors* which are invoked when an object of a particular **class** is created.
- Special functions called *destructors* which are invoked when an object of a particular **class** goes out of scope or is destroyed.
- Inheritance.

C++ stream class hierarchy

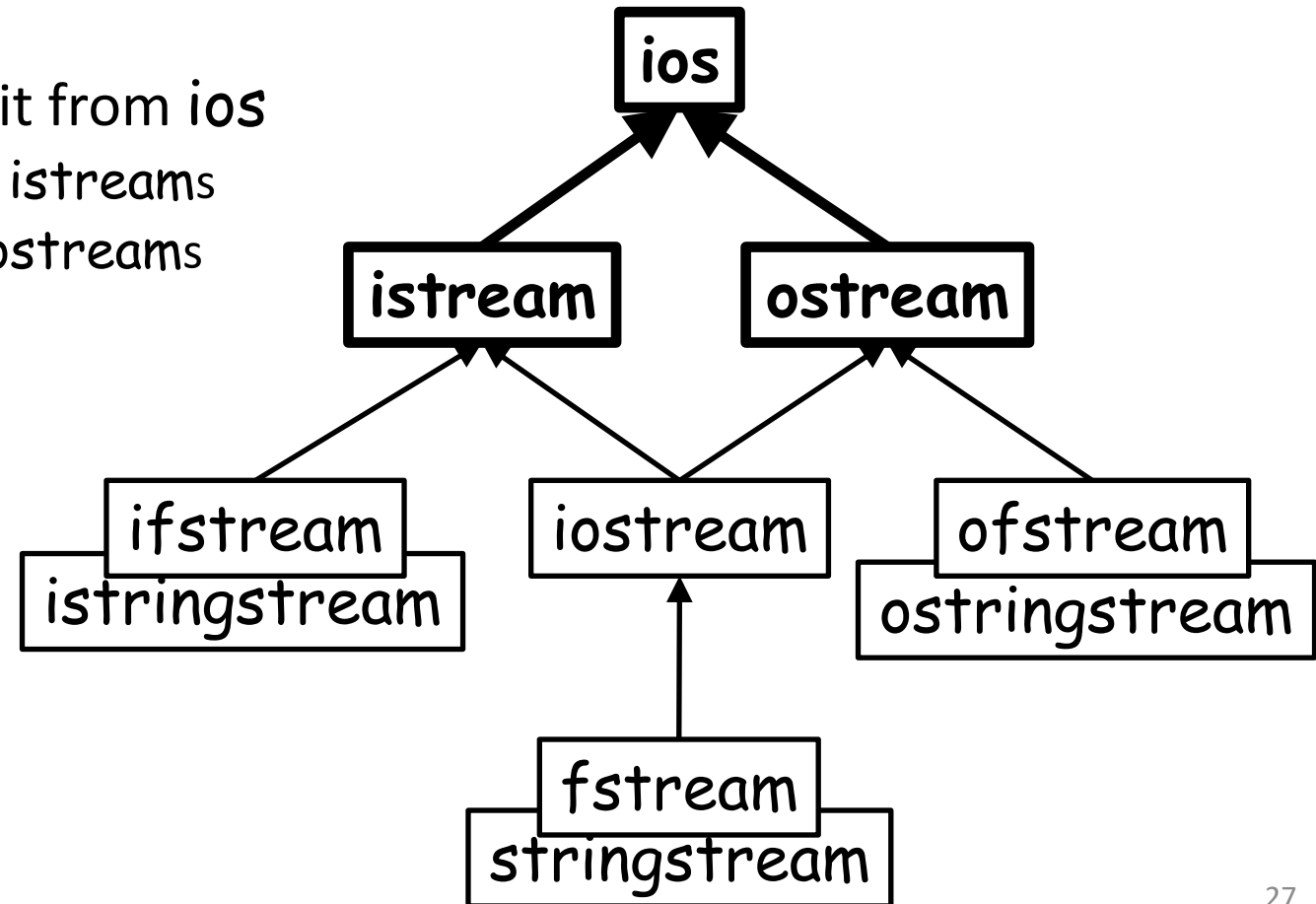
Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).



C++ stream class hierarchy

Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).

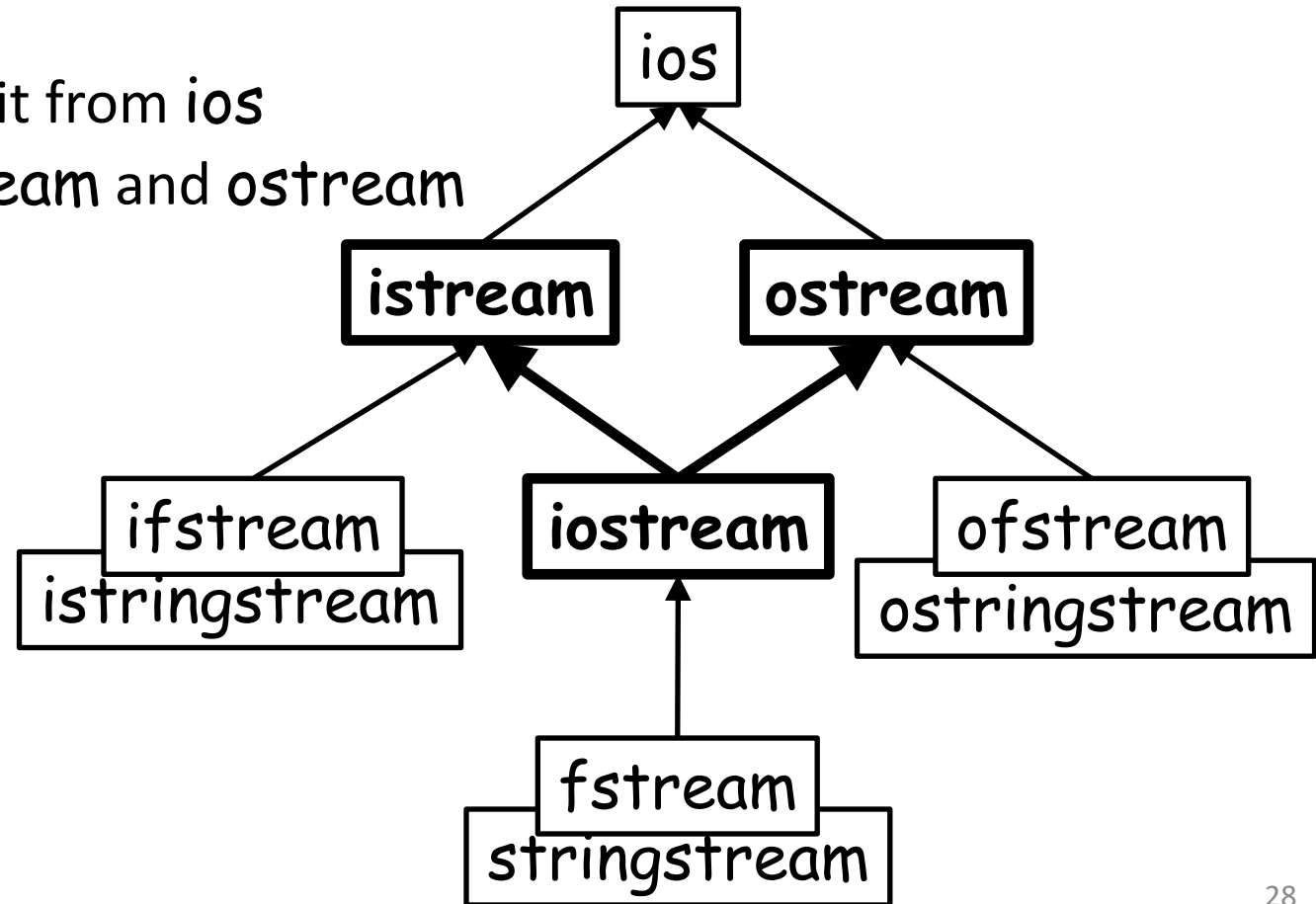
- `istream` and `ostream` both inherit from `ios`
 - Stream extraction (`>>`) defined for all `istream`s
 - Stream insertion (`<<`) defined for all `ostream`s



C++ stream class hierarchy

Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).

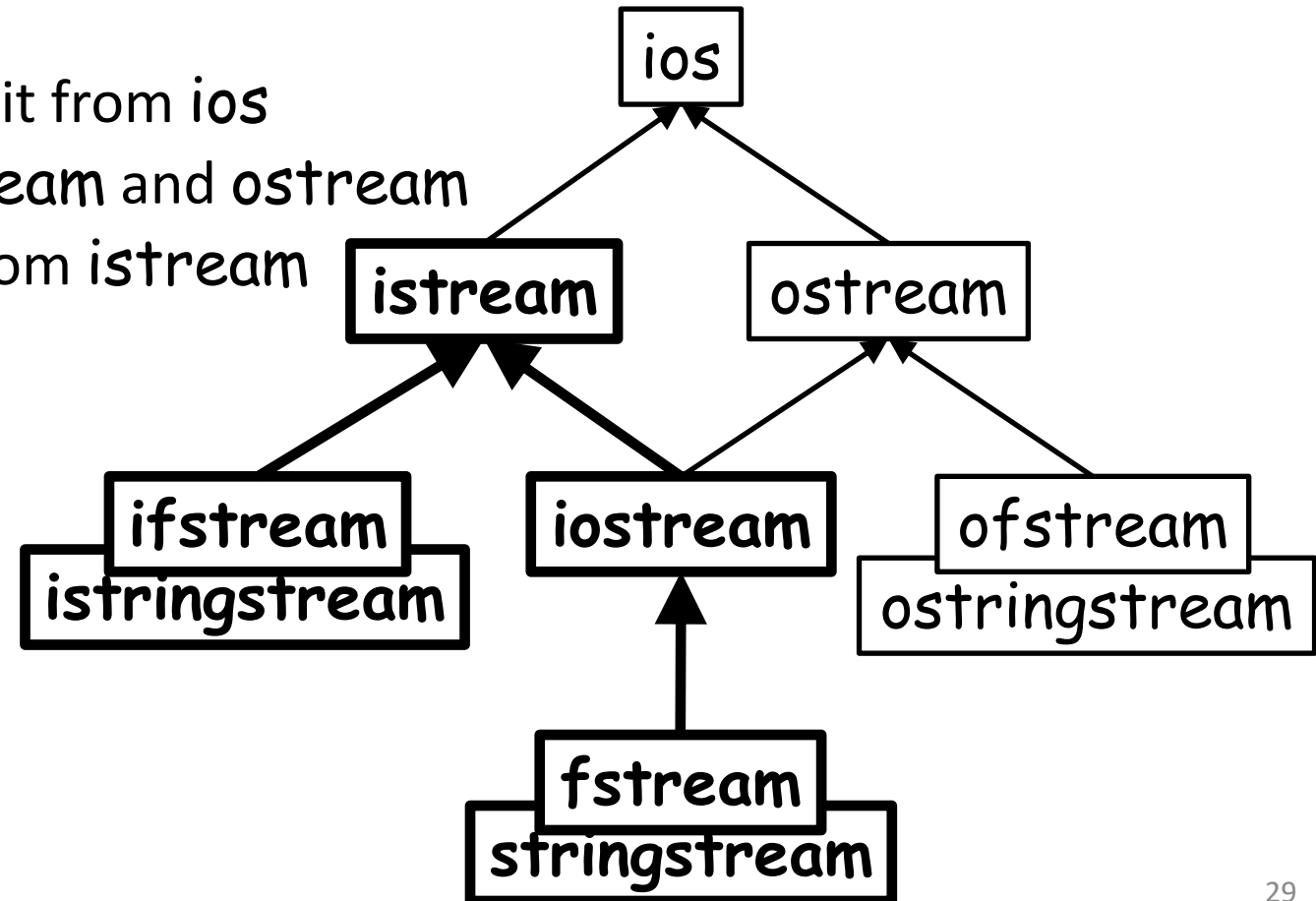
- `istream` and `ostream` both inherit from `ios`
- `iostream` inherits from both `istream` and `ostream`
 - multiple inheritance is allowed



C++ stream class hierarchy

Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).

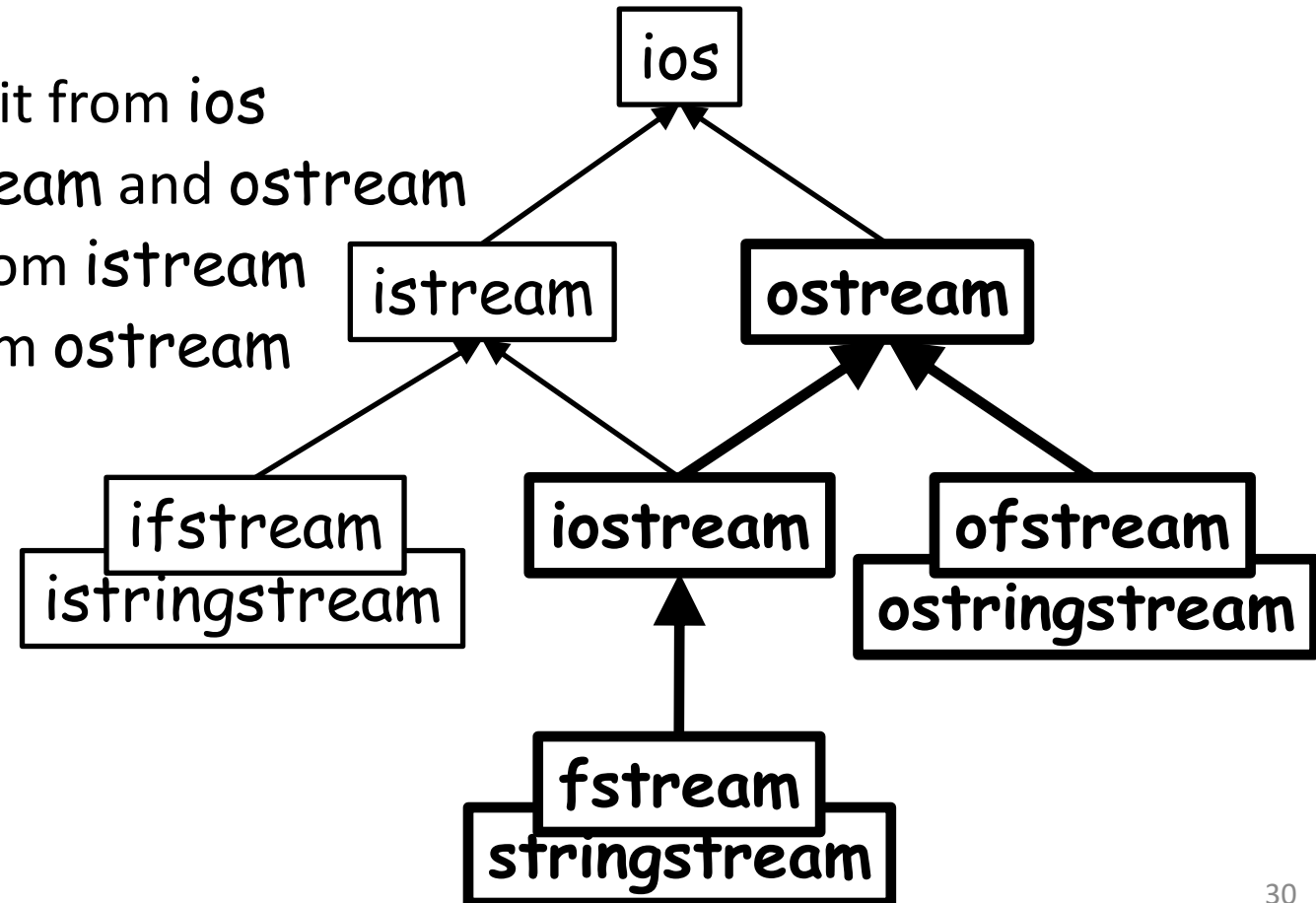
- `istream` and `ostream` both inherit from `ios`
- `iostream` inherits from both `istream` and `ostream`
- Stream extraction (`>>`) inherited from `istream`



C++ stream class hierarchy

Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).

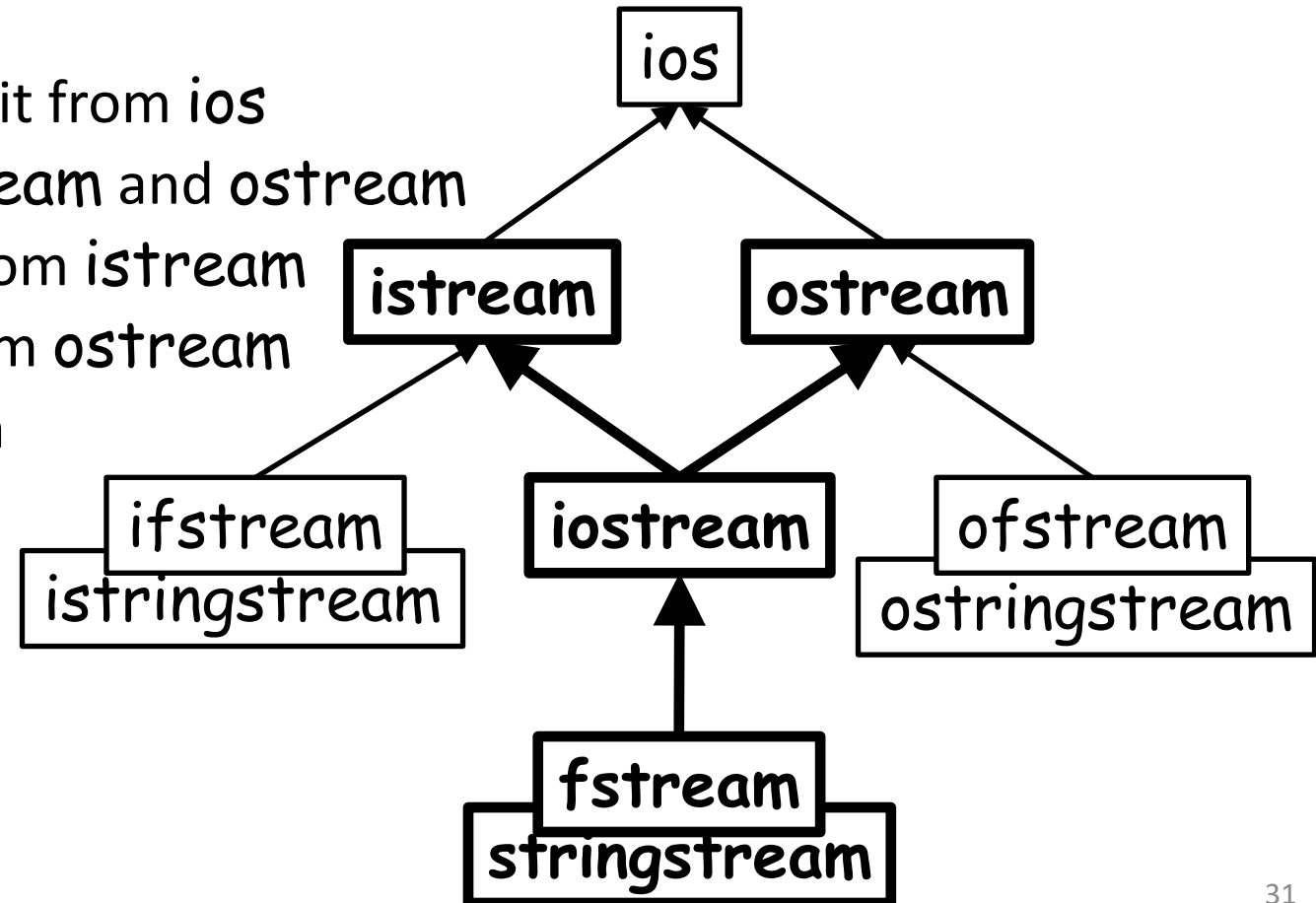
- `istream` and `ostream` both inherit from `ios`
- `iostream` inherits from both `istream` and `ostream`
- Stream extraction (`>>`) inherited from `istream`
- Stream insertion (`<<`) inherited from `ostream`



C++ stream class hierarchy

Inheritance diagram for streams – arrows indicate who inherits from whom (“is-a” relationship).

- `istream` and `ostream` both inherit from `ios`
- `iostream` inherits from both `istream` and `ostream`
- Stream extraction (`>>`) inherited from `istream`
- Stream insertion (`<<`) inherited from `ostream`
- `fstream` and `stringstream` both inherit from `iostream`



Outline

- Exercise 24
- File I/O
- `std::stringstream`
- Object Oriented Programming
- Review Questions

Review questions

1. How do you read/write files in C++?

Create input / output filestreams

```
std::ifstream ifile( "hello.txt" );  
std::ofstream ofile( "hello.txt" );
```

and insert into / extract from the files

```
ifile >> str1 >> str2;  
ofile << "Hello, World!" << std::endl;
```

Review questions

2. What is a `stringstream` in C++?

A stream supporting insertion/extraction, which keeps its data buffered in a `std::string`.

Review questions

3. How do you extract the contents of a `stringstream`?

Either use the stream extraction operator ">>", or use the `str(void)` member function.

Review questions

4. What does a constructor do?

Initializes the resources associated with a class

Review questions

5. What does a destructor do?

Releases/deallocates the resources associated with a class

Exercise 25

- Website -> Course Materials -> Exercise 25