Object-oriented Programming

Week 10 | Lecture 1

Defining Aliases

- The keyword **typedef** can be used to declare synonyms (aliases) for previously defined data types

- Creating a name using *typedef* does not create a new data type; typedef creates only an alternate name for the existing data type

Example

```
int main()
{
    typedef int i;
    i var1 = 5;
    cout << var1;     // outputs 5

    typedef float f;
    f var2 = 2.8;
    cout << var2;     // outputs 2.8
}
```

Stream I/O

- C++ is a type-safe language
- I/O in C++ occurs in **streams**
- Streams are simply sequence of bytes
- In input operations, data is transferred from input device (keyboard) to main memory
- In output operations, data is transferred from main memory to output device (display screen)

Stream I/O

- C++ provides both low-level (unformatted) and high-level (formatted) I/O capabilities
- Unformatted I/O is efficient for high-volume data processing
- C++ includes the **standard stream libraries** for I/O operations

Stream I/O

- The C++ **iostream** library provides a lot of I/O capabilities
- The <iostream> header file defines the *cin*, *cout*, *cerr* and *clog* objects
- The <iomanip> header declares services useful for performing formatted I/O with so-called **parameterized stream manipulators**

Stream Operators

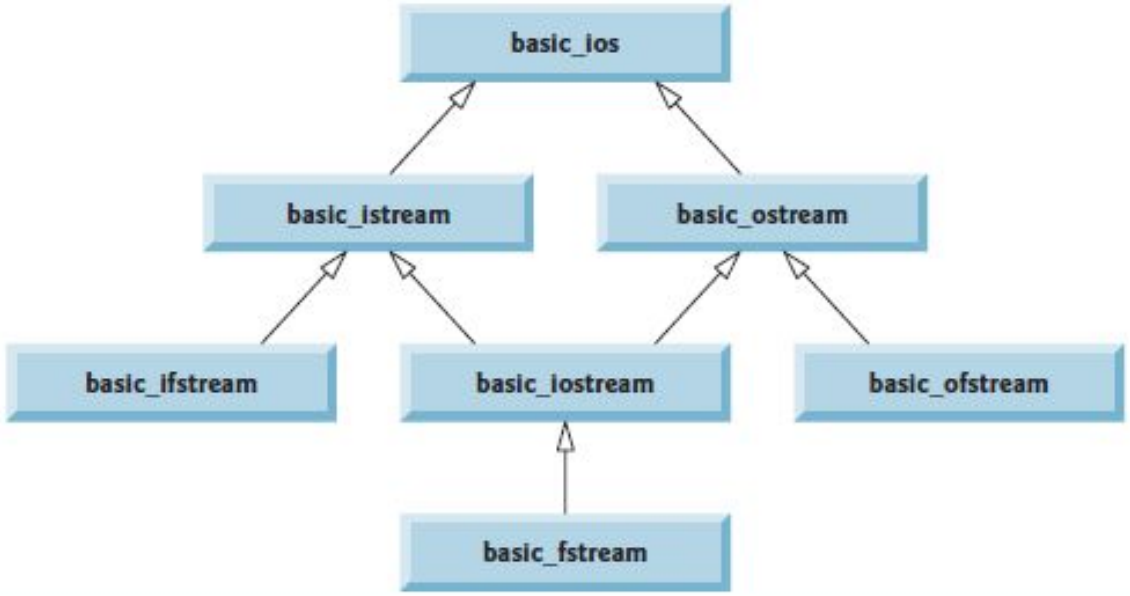
- The left-shift operator (**<<**) is overloaded to serve as stream insertion operator
- The right-shift operator (**>>**) is overloaded to serve as stream extraction operator
- These operators are used with the standard stream objects cin, cout, cerr and clog

Standard Stream Objects

- Predefined object **cin** is an istream instance
- The object **cout** is an ostream instance

- The predefined object **cerr** is an ostream instance and is said to be “connected to” the standard error device, normally the screen

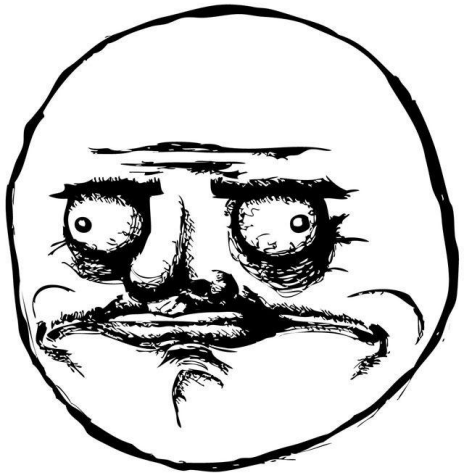
- The predefined object **clog** is an instance of the ostream class and is said to be “connected to” the standard error device. Outputs to clog are **buffered**



Stream-I/O template hierarchy portion showing the main file-processing templates.

Output Stream

- C++ determines data types automatically
- But this feature can sometimes *“gets in the way”*



Example

```
int main()
{
    char * word = “Hello”;
    cout << “Address is ” << word;
}
```

// prints Hello as output instead of address

Solution

- Cast the char * to a void *
-

```
int main()
{
    char * word = “Hello”;
   cout << “Address is ”;
   cout << static_cast< void *>(word);
} // prints address as output
```