Substitution vs. Conversion

C++ allows automatic conversions between the built-in numeric types; with numeric conversion, the compiler runs a built-in algorithm and tries to calculate the closest value that you desire. That's **not what happens** with objects in a class hierarchy.

When you pass an `ofstream` object to a function that expects an `ostreams&`, **no conversion takes place at all!** Instead, the `ofstream` object is automatically treated as if it **were** an `ostream` object, because the `ostream` and `ofstream` classes are related as in a special way through inheritance. Because the `ofstream` class is derived from the `ostream` class we can **substitute it** for the expected `ostream` parameter.

We can do that because the derived class inherits all of the characteristics of its base class, so that anything an `ostream` object can do, an `ofstream` object can do as well, by definition. This ability to allow a derived or subclass object to be used in any context that expects a base-class object is known as the **Liskov Substitution Principle**, after computer scientist Barbara Liskov.