Streamsl

C++'s stream library is the means by which a C++ program can **interact** with its environment,

- **the user and**
- **the file system**

```
#include <iostream>

int main() {
    std::cout << "Hellow World!" << std::endl;
    return 0;
}
```

The program above can **get annoying** to write for common names like

- cout
- endl

```
#include <iostream>
using std::cout;
using std::endl;
int main() {
    cout << "Hellow World!" << endl;
    return 0;
}
```

- Whenever you use cout , the compiler will assume you mean std::cout

Stream

An abstraction for I/O

```
Illustration:
                        std cout
            ----------------------------
screen      |                          |  << "Hello world!"
            ----------------------------

                        std cout
            ----------------------------
screen      |       "Hello world!"     |
            ----------------------------

                                std cout
                       ---------------------------
                "Hello world!"     |              |
                       ---------------------------
```

You can write data of multiple types to stream objects

```
cout << "Strings work!" << endl;//you can push data of string type into stream
cout << 1729 << endl;//integer type
cout << 3.14 << endl;//double type
cout << "Mixed types - " << 1123 << endl;//mix
```

In particular, any primitive type can be inserted into a stream!
For other types, you need to **explicitly** tell C++ how to do this.

Output Stream

- Can only receive data
- Operator Insertion: <<
- Insertion converts data to string and sends to stream
- You can send the data to a file using a std::ofstream , which is a special type of std::ostream

```cpp
#include <iostream>
#include <fstream>

using std::cout;
using std::endl;
using std::string;
using std::ofstream;//Output file stream
int main() {
    //Output to console
    string s = "Why is 1 + 1 equal to ";
    cout << s <<  2 << endl;//insert different primitive types of data into output stream "cout"
    //Output to file
    ofstream file("output.txt");
    file << s << 2 << endl;
}
```

Input Stream

- Can only give you data
- The std::cin is an example of an input stream
- Operator Extraction: >>
- Extraction gets data from stream as a string and converts it into the appropriate type

```cpp
#include <iostream>
#include <fstream>

using std::cout;    using std::endl;
using std::string;

void readNumbers() {
    // Create our ifstream and make it open the file
    std::ifstream input("numbers.txt");

    // This will store the values as we get them form the stream
    int value;
    while(input >> value) {// Extract next number from input
        cout << "Value read: " << value << endl;
    }
}

void readHaikuWord() {
    // Create our ifstream and make it open the file
    std::ifstream input("haiku.txt");

    // This will store the values as we get them form the stream
    string word;
    while(input >> word) {//Extract next string from input
        cout << "Word read: " << word << endl;
    }
}
```

```cpp
void readHaikuLine() {
    // Create our ifstream and make it open the file
    std::ifstream input("haiku.txt");

    // This will store the lines as we get them from the stream
    string line;
    while(std::getline(input, line)) {
        cout << line << endl;
    }
}

int main() {
    readNumbers();
    cout << "==========================" << endl;
    readHaikuWord();
    cout << "==========================" << endl;
    readHaikuLine();
    return 0;
}
```

- When << , >> or getline() cannot read any data from stream, they will set fall bit as true and return true