```
/* The following code example is taken from the book
 * "The C++ Standard Library - A Tutorial and Reference, 2nd Edition"
 * by Nicolai M. Josuttis, Addison-Wesley, 2012
 *
 * (C) Copyright Nicolai M. Josuttis 2012.
 * Permission to copy, use, modify, sell and distribute this software
 * is granted provided this copyright notice appears in all copies.
 * This software is provided "as is" without express or implied
 * warranty, and with no claim as to its suitability for any purpose.
 */
#include <iostream>
#include <locale>
#include <string>
#include <cstdlib>
#include <exception>
using namespace std;

int main()
{
    try {
        // create the default locale from the user's environment
        locale langLocale("");

        // and assign it to the standard output channel
        cout.imbue(langLocale);

        // process the name of the locale to find out whether German is used
        cout << langLocale.name() << endl;
        bool isGerman = (langLocale.name().substr(0,2) == "de" ||
                         langLocale.name().substr(0,3) == "ger" ||
                         langLocale.name().substr(0,3) == "Ger");

        // read locale for the input
        cout << (isGerman ? "Sprachumgebung fuer Eingaben: "
                          : "Locale for input: ") << endl;
        string s;
        cin >> s;
        if (!cin) {
            if (isGerman) {
                cerr << "FEHLER beim Einlesen der Sprachumgebung"
                     << endl;
            }
            else {
                cerr << "ERROR while reading the locale" << endl;
            }
            return EXIT_FAILURE;
        }
        locale cinLocale(s);  // create locale by string (since C++11)

        // and assign it to the standard input channel
        cin.imbue(cinLocale);

        // read and output floating-point values in a loop
        cout << (isGerman ? "Gleitkommawerte: "
                          : "Floating-point values: ") << endl;
        double value;
```

```
        while (cin >> value) {
            cout << value << endl;
        }
    }
    catch (const std::exception& e) {
        cerr << "Exception: " << e.what() << endl;
        return EXIT_FAILURE;
    }
}
```