```
//
// This is a standard library support code to the chapters of the book
// "Programming -- Principles and Practice Using C++" by Bjarne Stroustrup
//

#ifndef STD_LIB_FACILITIES_GUARD
#define STD_LIB_FACILITIES_GUARD 1

#include <algorithm> // 标准算法库，主要应用在容器上
#include <cmath> // 数学计算的库
#include <iomanip> // 用来对输入输出操作的格式进行更加方便的控制的库
#include <iostream>  // 输入输出流
#include <fstream> // 对文件操作的库
#include <stdexcept> // 预定义异常类
#include <string> // 字符串类
#include <sstream> // string 流
#include <vector> // 向量

using namespace std; // 使用名为 std 的命名空间

//------------------------------------------------------------------------------
-

// The call to keep_window_open() is needed on some Windows machines to prevent
// them from closing the window before you have a chance to read the output.
inline void keep_window_open() // 定义一个内联函数，有利于频繁调用
{
    cin.get(); // 等待输入
}

//------------------------------------------------------------------------------
-

inline void keep_window_open(const string& str) // 定义一个内联函数 参数为 string 类型
{
    static int attempts = 10; // Maximum number of attempts before forceful exit

    while (--attempts) // attempts 递减 只要不为 0 就循环继续
    {
        cout << "Please enter " << str << " to exit" << endl; // 打印 please enter 字符串 to exit

        bool exit = true; // exit 布尔标识为 true
```

```
        for(string::const_iterator p = str.begin(); p != str.end(); ++p) // 从字符串开始遍历到结束
            if (*p != cin.get()) // 如果当前字母的值与输入的不一样
            {
                exit = false; // exit 布尔标识值为 false
                break; // 跳出循环
            }

        if (exit) // 如果 exit 为 true
            break; // 跳出当前主循环
    }
}

//------------------------------------------------------------------------------

// Helper function to show an error message
inline void error(const string& errormessage)  // 定义一个内联函数 参数为表示错误信息的 string
{
    cerr << errormessage << endl; // 输出包含 errormessage 的标准错误流的信息
    throw runtime_error(errormessage); // 抛出一个 runtime_error
}

//------------------------------------------------------------------------------

inline void error(string s1, string s2) // 定义一个内联函数 参数为 s1 和 s2
{
    error(s1+s2); // 调用 error 方法 参数为 s1+s2
}

//------------------------------------------------------------------------------

template <typename Target, typename Source> // 模板函数
Target narrow_cast(Source src) // 一个名为 narrow_cast 的模板函数
{
    Target tgt = src; // 赋值

    if ((Source)tgt != src) // 如果强制类型转换的 tgt 不等于 src
        error("Invalid narrowing conversion"); // 调用 error 方法 抛出错误

    return tgt; // 返回 tgt
```

```
}

//------------------------------------------------------------------------------
-

inline ios_base& general(ios_base& b)    // to complement fixed and scientific
{
    b.setf(ios_base::fmtflags(0), ios_base::floatfield);
    return b;
}

//------------------------------------------------------------------------------
-

#endif // STD_LIB_FACILITIES_GUARD
```