```
#include <type_traits>  // std::is_floating_point
#include <cassert>      // standard C assert macro

template <typename T>
const bool sane_for_pi = std::is_floating_point<T>::value;  // same type

template <typename T> const T pi(3.1415926535897932385);    // distinct types

void testPi()
{
    assert(!sane_for_pi<bool>);
    assert(!sane_for_pi<int>);

    assert( sane_for_pi<float>);
    assert( sane_for_pi<double>);
    assert( sane_for_pi<long double>);

    const float       pi_as_float       = 3.1415927;
    const double      pi_as_double      = 3.141592653589793;
    const long double pi_as_long_double = 3.1415926535897932385;

    assert(pi<float>       == pi_as_float);
    assert(pi<double>      == pi_as_double);
    assert(pi<long double> == pi_as_long_double);
}
```

Variable templates may be **declared** at namespace-scope or as **`static` members** of a **`class`**, **`struct`**, or **`union`** but are not permitted as non**`static`** members nor at all in **function scope**:

```
template <typename T> T vt1;            // OK, external linkage
template <typename T> static T vt2;     // OK, internal linkage

namespace N
{
    template <typename T> T vt3;        // OK, external linkage
    template <typename T> static T vt4; // OK, internal linkage
}

struct S
{
    template <typename T> T vt5;        // Error, not static
    template <typename T> static T vt6; // OK, external linkage
};
```