Document number: P0631R0
Date: 2017-03-19
Project: Programming Language C++
Audience: Library Evolution Working Group, SG6 (Numerics)
Reply-to: Lev Minkovsky lminkovsky@outlook.com

Math Constants

Introduction

C++ inherited from C a rich library of mathematical functions, which continues to grow with every release. Amid all this abundance, there is a strange gap: none of the major mathematical constants is defined in the standard. This proposal is aimed to rectify this omission.

Motivation

Mathematical constants such as π and *e* frequently appear in mathematical algorithms. A software engineer can easily define them, but from their perspective, this is akin to making a reservation at a restaurant and being asked to bring their own salt. The C++ implementers appreciate this need and attempt to fulfil it with non-standard extensions.

The IEEE Standard 1003.1™-2008 a.k.a POSIX.1-2008 stipulates that on all systems supporting the X/Open System Interface Extension, "the <math.h> header shall define the following symbolic constants. The values shall have type double and shall be accurate to at least the precision of the double type."

M_E - value of e
M_LOG2E - value of $\log_2 e$
M_LOG10E - value of $\log_{10} e$
M_LN2 - value of $\ln 2$
M_LN10 - value of $\ln 10$
M_PI - value of π
M_PI_2 - value of $\frac{\pi}{2}$
M_PI_4 - value of $\frac{\pi}{4}$
M_1_PI - value of $\frac{1}{\pi}$
M_2_PI - value of $\frac{2}{\pi}$
M_2_SQRTPI - value of $\frac{2}{\sqrt{\pi}}$
M_SQRT2 - value of $\sqrt{2}$

M_SQRT1_2 - value of $\frac{\sqrt{2}}{2}$

POSIX.1-2008 explicitly states that these constants are outside of the ISO C standard and should be hidden behind an appropriate feature test macro. On some POSIX-compliant systems, this macro is defined as _USE_MATH_DEFINES, which led to a common assumption that defining this macro prior to the inclusion of math.h makes these constants accessible. In reality, this is true only in the following scenario:

1) The implementation defines these constants, and
2) It uses _USE_MATH_DEFINES as a feature test macro, and
3) This macro is defined prior to the first inclusion of math.h or any header file that directly or indirectly includes math.h.

These makes the availability of these constants extremely fragile when the code base is ported from one implementation to another or to a newer version of the same implementation. In fact, something as benign as including a new header file may cause them to disappear.

The OpenCL standard by the Kronos Group offers the same set of preprocessor macros in three variants: with a suffix _H, with a suffix _F and without a suffix, to be used in fp16, fp32 and fp64 calculations respectively. The first and the last sets are macro-protected. It also defines in the cl namespace the following variable templates:

e_v, log2e_v, log10e_v, ln2_v, ln10_v , pi_v, pi_2_v, pi_4_v, one_pi_v, two_pi_v, two_sqrtpi_v, sqrt2_v, sqrt1_2_v,

as well as their instantiations based on a variety of floating point types and abovementioned macros. An OpenCL developer can therefore utilize a value of cl::pi_v<float>; they can also access cl::pi_v<double>, but only if the cl_khr_fp64 macro is defined.

The GNU C++ library offers an alternative approach. It includes an implementation-specific file ext\cmath that defines in the __gnu_cxx namespace the templated definitions of the following constants:

__pi,__pi_half,__pi_third,__pi_quarter,__root_pi_div_2,__one_div_pi,__two_div_pi,__two_div_root_pi ,__e,__one_div_e, __log2_e, __log10_e, __ln_2, __ln_3, __ln_10, __gamma_e, __phi, __root_2, __root_3,__root_5, __root_7

The access to these constants is quite awkward. For example, to use a double value of π, a programmer would have to write __gnu_cxx::__math_constants::__pi<double>.

All these efforts, although helpful, clearly indicate the need for standard C++ to provide a set of math constants that would be both easy to use and appropriately accurate.

Design Considerations and Proposed Definitions

The ISO C++ set of math constants should be comprised of the same mathematical values as in the IEEE Standard 1003.1™-2008. They should be available as both an ordinary variable and a variable template. Many developers that could potentially benefit from these constants come from C or even Fortran

background. They should feel free to use as much or as little of C++ as they prefer. It would be awkward if we expect from them to use something like std::pi<double> as the only template instantiation in their code base.

The ordinary constants should be defined as follows:

```
constexpr long double pi
constexpr long double e
constexpr long double log2e
constexpr long double log10e
constexpr long double ln2
constexpr long double ln10
constexpr long double sqrt2
constexpr long double pi_2
constexpr long double pi_4
constexpr long double one_pi
constexpr long double two_pi
constexpr long double two_sqrtpi
constexpr long double one_sqrt2
```

The long double type is more accurate than double on some platforms, and this extra accuracy can potentially be beneficial. The initialization part of these definitions should be implementation-specific.

The variable templates should be defined as follows:

```
template<typename T> constexpr T pi_v
template<typename T> constexpr T e_v
template<typename T> constexpr T log2e_v
template<typename T> constexpr T log10e_v
template<typename T> constexpr T ln2_v
template<typename T> constexpr T ln10_v
template<typename T> constexpr T sqrt2_v
template<typename T> constexpr T pi_2_v
template<typename T> constexpr T pi_4_v
template<typename T> constexpr T one_pi_v
template<typename T> constexpr T two_pi_v
template<typename T> constexpr T two_sqrtpi_v
template<typename T> constexpr T one_sqrt2_v
```

The initialization part of these definitions should also be implementation-specific and possibly different for different types because of explicit specializations.

Math constants should be defined in the same place as the rest of common mathematical functions such as sqrt. If we continue to maintain the existing set of C++ headers, this would mean that they should be present in the std namespace or one of its inline namespaces and be accessible via the <cmath> header. If however we encourage the C++ community to transition from header files to modules, they can be defined in the std.numeric module.

A “Hello world” program for math constants

```
#include <cmath>

using namespace std;

template<typename T> constexpr  T circle_area(T r) { return pi_v<T> * r * r; }

int main()
{
	static_assert(!!pi);
	static_assert(!!circle_area(1.0));
	return 0;
}
```

Proposed Changes in the Standard

26.9.1 Header <cmath> synopsis

After

long double sph_neumannl(unsigned n, long double x);

the following definitions should be inserted:

// 26.9.7, mathematical constants

// 26.9.7.1, mathematical constant variables
constexpr long double pi = see below
constexpr long double e = see below
constexpr long double log2e = see below
constexpr long double log10e = see below
constexpr long double ln2 = see below
constexpr long double ln10 = see below
constexpr long double sqrt2 = see below
constexpr long double pi_2 = see below
constexpr long double pi_4 = see below
constexpr long double one_pi = see below
constexpr long double two_pi = see below
constexpr long double two_sqrtpi = see below
constexpr long double one_sqrt2 = see below

// 26.9.7.2, mathematical constant variable templates
template<typename T> constexpr T pi_v = see below
template<typename T> constexpr T e_v = see below

template<typename T> constexpr T log2e_v = see below
template<typename T> constexpr T log10e_v = see below
template<typename T> constexpr T ln2_v = see below
template<typename T> constexpr T ln10_v = see below
template<typename T> constexpr T sqrt2_v = see below
template<typename T> constexpr T pi_2_v = see below
template<typename T> constexpr T pi_4_v = see below
template<typename T> constexpr T one_pi_v = see below
template<typename T> constexpr T two_pi_v = see below
template<typename T> constexpr T two_sqrtpi_v = see below
template<typename T> constexpr T one_sqrt2_v = see below

In the § 26.9.1, footnote 1, the sentence "The contents and meaning of the header <cmath> are the same as the C standard library header <math.h>, with the addition of a three-dimensional hypotenuse function (26.9.3) and the mathematical special functions described in 26.9.5" should be rewritten as "The contents and meaning of the header <cmath> are the same as the C standard library header <math.h>, with the addition of a three-dimensional hypotenuse function (26.9.3), the mathematical special functions described in 26.9.5 and the mathematical constants described in 26.9.7. "

After § 26.9.6, a new section § 26.9.7 should be inserted:

26.9.7 Mathematical constants

26.9.7.1 Mathematical constant variables

constexpr long double pi
constexpr long double e
constexpr long double log2e
constexpr long double log10e
constexpr long double ln2
constexpr long double ln10
constexpr long double sqrt2
constexpr long double pi_2
constexpr long double pi_4
constexpr long double one_pi
constexpr long double two_pi
constexpr long double two_sqrtpi
constexpr long double one_sqrt2

[1] *Remarks:* These variables shall be initialized with implementation-defined values of π, e, $\log_2 e$, $\log_{10} e$, ln2, ln10, $\sqrt{2}$, $\frac{\pi}{2}$, $\frac{\pi}{4}$, $\frac{1}{\pi}$, $\frac{2}{\pi}$, $\frac{2}{\sqrt{\pi}}$, and $\frac{\sqrt{2}}{2}$, respectively.

26.9.7.1 Mathematical constant variable templates

template<typename T> constexpr T pi_v
template<typename T> constexpr T e_v
template<typename T> constexpr T log2e_v

template<typename T> constexpr T log10e_v
template<typename T> constexpr T ln2_v
template<typename T> constexpr T ln10_v
template<typename T> constexpr T sqrt2_v
template<typename T> constexpr T pi_2_v
template<typename T> constexpr T pi_4_v
template<typename T> constexpr T one_pi_v
template<typename T> constexpr T two_pi_v
template<typename T> constexpr T two_sqrtpi_v
template<typename T> constexpr T one_sqrt2_v

[1] *Remarks:* These variable templates shall be initialized with implementation-defined possibly type-dependent values of π, e, $\log_2 e$, $\log_{10} e$, ln2, ln10, $\sqrt{2}$, $\frac{\pi}{2}$, $\frac{\pi}{4}$, $\frac{1}{\pi}$, $\frac{2}{\pi}$, $\frac{2}{\sqrt{\pi}}$, and $\frac{\sqrt{2}}{2}$, respectively. Their specializations should be available for all floating-point types, see **3.9.1.**

References

The POSIX version of math.h is described at http://pubs.opengroup.org/onlinepubs/9699919799/basedefs/math.h.html.

The OpenCL mathematical constants are defined in a file opencl_math_constants, see https://raw.githubusercontent.com/KhronosGroup/libclcxx/master/include/opencl_math_constants.

The GNU math extensions: https://gcc.gnu.org/onlinedocs/gcc-6.1.0/libstdc++/api/a01120_source.html

Acknowledgments

The author would like to thank Edward Smith-Rowland for his review of the draft proposal and Vishal Oza, Daniel Krügler and Matthew Woehlke for their participation in the related thread at the std-proposals user group.