

ISO/IEC JTC 1/SC 22/WG14 N1182

Date: 2006-08-02

Reference number of document: **ISO/IEC TR 24747**

Committee identification: ISO/IEC JTC 1/SC 22/WG 14

Information Technology —

Programming languages, environments and system software interfaces —

Extensions to the C Library, to Support Mathematical Special Functions —

warning

This document is an ISO/IEC draft Technical Report. It is not an ISO/IEC International Technical Report. It is distributed for review and comment. It is subject to change without notice and shall not be referred to as an International Technical Report or International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: Technical Report Type 2

Document subtype: n/a

Document stage: (2) Working Draft Technical Report

Document language: E

Copyright notice

This ISO document is copyright-protected by ISO. Except as permitted under the applicable laws of the user's country, neither this ISO draft nor any extract from it may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, photocopying, recording or otherwise, without prior written permission being secured.

Requests for permission to reproduce should be addressed to either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56, CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Reproduction may be subject to royalty payments or a licensing agreement.

Front matter

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2. The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote. In exceptional circumstances, the joint technical committee may propose the publication of a Technical Report of one of the following types:

- type 1, when the required support cannot be obtained for the publication of an International Standard, despite repeated efforts;
- type 2, when the subject is still under technical development or where for any other reason there is the future but not immediate possibility of an agreement on an International Standard;
- type 3, when the joint technical committee has collected data of a different kind from that which is normally published as an International Standard ("state of the art," for example).

Technical Reports of types 1 and 2 are subject to review within three years of publication, to decide whether they can be transformed into International Standards. Technical Reports of type 3 do not necessarily have to be reviewed until the data they provide are considered to be no longer valid or useful.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC TR 24747 is a Technical Report of type 2.

Contents

Contents	iii
List of Tables	v
1 General	1
1.1 Relation to C Standard Library Introduction	1
1.2 Categories of extensions	1
1.3 Headers	1
2 Mathematical special functions	3
2.1 Additions to header <code><math.h></code>	3
2.1.1 associated Laguerre polynomials	6
2.1.2 associated Legendre functions	6
2.1.3 beta function	6
2.1.4 (complete) elliptic integral of the first kind	6
2.1.5 (complete) elliptic integral of the second kind	7
2.1.6 (complete) elliptic integral of the third kind	7
2.1.7 regular modified cylindrical Bessel functions	7
2.1.8 cylindrical Bessel functions (of the first kind)	8
2.1.9 irregular modified cylindrical Bessel functions	8
2.1.10 cylindrical Neumann functions	8
2.1.11 (incomplete) elliptic integral of the first kind	9
2.1.12 (incomplete) elliptic integral of the second kind	9
2.1.13 (incomplete) elliptic integral of the third kind	9
2.1.14 exponential integral	9
2.1.15 Hermite polynomials	10
2.1.16 Laguerre polynomials	10
2.1.17 Legendre polynomials	10
2.1.18 Riemann zeta function	11
2.1.19 spherical Bessel functions (of the first kind)	11
2.1.20 spherical associated Legendre functions	11
2.1.21 spherical Neumann functions	12
2.2 Additions to header <code><tgmath.h></code>	12

Bibliography

15

Index

17

List of Tables

1	Numerical library summary	1
2	Additions to header <code><math.h></code> synopsis	5
3	Additions to header <code><tgmath.h></code> synopsis	12

1 General

[tr.intro]

- 1 This technical report describes extensions to the *C standard library* that is described in the International Standard for the C programming language [6].
- 2 This technical report is non-normative. Some of the library components in this technical report may be considered for standardization in a future version of C, but they are not currently part of any C standard. Some of the components in this technical report may never be standardized, and others may be standardized in a substantially changed form.
- 3 The goal of this technical report is to build more widespread existing practice for an expanded C standard library. It gives advice on extensions to those vendors who wish to provide them.

1.1 Relation to C Standard Library Introduction

[tr.description]

- 1 Unless otherwise specified, the whole of the ISO C Standard Library introduction [lib.library] is included into this Technical Report by reference.

1.2 Categories of extensions

[tr.intro.ext]

- 1 This technical report describes library extensions to the C Standard Library to support Mathematical Special functions to be added to `<math.h>` and `<tgmath.h>`.

1.3 Headers

[tr.intro.namespaces]

- 1 Vendors should not simply add declarations to standard headers in a way that would be visible to users by default. [*Note:* That would fail to be standard conforming, because the new names could conflict with user macros. —*end note*] Users should be required to take explicit action to have access to library extensions.
- 2 It is recommended either that additional declarations in standard headers be protected with a macro that is not defined by default, or else that all extended headers be placed in a separate directory that is not part of the default search path.

Table 1: Numerical library summary

Subclause	Header(s)
2.1 Additions to	<code><math.h></code>
2.2 Additions to	<code><tgmath.h></code>

2 Mathematical special functions

[tr.num.sf]

2.1 Additions to header <math.h>

[tr.num.sh.math]

- 1 Table 2 summarizes the functions that are added to header <math.h>. The detailed signatures are given in the synopsis.
- 2 Each of these functions is provided for arguments of type float, double, and long double. The signatures added to header <math.h> are:

// [2.1.1] associated Laguerre polynomials:

```
double    assoc_laguerre(unsigned n, unsigned m, double x);
float     assoc_laguerref(unsigned n, unsigned m, float x);
long double assoc_laguerrel(unsigned n, unsigned m, long double x);
```

// [2.1.2] associated Legendre functions:

```
double    assoc_legendre(unsigned l, unsigned m, double x);
float     assoc_legendref(unsigned l, unsigned m, float x);
long double assoc_legendrel(unsigned l, unsigned m, long double x);
```

// [2.1.3] beta function:

```
double    beta(double x, double y);
float     betaf(float x, float y);
long double betal(long double x, long double y);
```

// [2.1.4] (complete) elliptic integral of the first kind:

```
double    comp_ellint_1(double k);
float     comp_ellint_1f(float k);
long double comp_ellint_1l(long double k);
```

// [2.1.5] (complete) elliptic integral of the second kind:

```
double    comp_ellint_2(double k);
float     comp_ellint_2f(float k);
long double comp_ellint_2l(long double k);
```

// [2.1.6] (complete) elliptic integral of the third kind:

```
double    comp_ellint_3(double k, double nu);
float     comp_ellint_3f(float k, float nu);
long double comp_ellint_3l(long double k, long double nu);
```

// [2.1.7] regular modified cylindrical Bessel functions:

```
double    cyl_bessel_i(double nu, double x);
float     cyl_bessel_if(float nu, float x);
```

```
long double   cyl_bessel_il(long double nu, long double x);

// [2.1.8] cylindrical Bessel functions (of the first kind):
double        cyl_bessel_j(double nu, double x);
float         cyl_bessel_jf(float nu, float x);
long double   cyl_bessel_jl(long double nu, long double x);

// [2.1.9] irregular modified cylindrical Bessel functions:
double        cyl_bessel_k(double nu, double x);
float         cyl_bessel_kf(float nu, float x);
long double   cyl_bessel_kl(long double nu, long double x);

// [2.1.10] cylindrical Neumann functions;
// cylindrical Bessel functions (of the second kind):
double        cyl_neumann(double nu, double x);
float         cyl_neumannf(float nu, float x);
long double   cyl_neumannl(long double nu, long double x);

// [2.1.11] (incomplete) elliptic integral of the first kind:
double        ellint_1(double k, double phi);
float         ellint_1f(float k, float phi);
long double   ellint_1l(long double k, long double phi);

// [2.1.12] (incomplete) elliptic integral of the second kind:
double        ellint_2(double k, double phi);
float         ellint_2f(float k, float phi);
long double   ellint_2l(long double k, long double phi);

// [2.1.13] (incomplete) elliptic integral of the third kind:
double        ellint_3(double k, double nu, double phi);
float         ellint_3f(float k, float nu, float phi);
long double   ellint_3l(long double k, long double nu, long double phi);

// [2.1.14] exponential integral:
double        expint(double x);
float         expintf(float x);
long double   expintl(long double x);

// [2.1.15] Hermite polynomials:
double        hermite(unsigned n, double x);
float         hermitef(unsigned n, float x);
long double   hermitel(unsigned n, long double x);

// [2.1.16] Laguerre polynomials:
double        laguerre(unsigned n, double x);
float         laguerref(unsigned n, float x);
long double   laguerrel(unsigned n, long double x);

// [2.1.17] Legendre polynomials:
double        legendre(unsigned l, double x);
```

```

float      legendref(unsigned l, float x);
long double legendrel(unsigned l, long double x);

// [2.1.18] Riemann zeta function:
double     riemann_zeta(double);
float      riemann_zetaf(float);
long double riemann_zetal(long double);

// [2.1.19] spherical Bessel functions (of the first kind):
double     sph_bessel(unsigned n, double x);
float      sph_besself(unsigned n, float x);
long double sph_bessell(unsigned n, long double x);

// [2.1.20] spherical associated Legendre functions:
double     sph_legendre(unsigned l, unsigned m, double theta);
float      sph_legendref(unsigned l, unsigned m, float theta);
long double sph_legendrel(unsigned l, unsigned m, long double theta);

// [2.1.21] spherical Neumann functions;
// spherical Bessel functions (of the second kind):
double     sph_neumann(unsigned n, double x);
float      sph_neumannf(unsigned n, float x);
long double sph_neumannl(unsigned n, long double x);

```

Table 2: Additions to header <math.h> synopsis

Functions:		
assoc_laguerre	cyl_bessel_j	hermite
assoc_legendre	cyl_bessel_k	legendre
beta	cyl_neumann	laguerre
comp_ellint_1	ellint_1	riemann_zeta
comp_ellint_2	ellint_2	sph_bessel
comp_ellint_3	ellint_3	sph_legendre
cyl_bessel_i	expint	sph_neumann

- 3 Each of the functions declared above shall return a NaN (Not a Number) if any argument value is a NaN, but it shall not report a domain error. Otherwise, each of the functions declared above shall report a domain error for just those argument values for which:
- the function description's Returns clause explicitly specifies a domain, and those arguments fall outside the specified domain; or
 - the corresponding mathematical function value has a non-zero imaginary component; or
 - the corresponding mathematical function is not mathematically defined.¹⁾
- 4 Unless otherwise specified, a function is defined for all finite values, for negative infinity, and for positive infinity.

¹⁾A mathematical function is mathematically defined for a given set of argument values if it is explicitly defined for that set of argument values or if its limiting value exists and does not depend on the direction of approach.

2.1.1 associated Laguerre polynomials**[tr.num.sf.Lnm]**

```
double    assoc_laguerre(unsigned n, unsigned m, double x);
float     assoc_laguerref(unsigned n, unsigned m, float x);
long double assoc_laguerrel(unsigned n, unsigned m, long double x);
```

1 *Effects:* These functions compute the associated Laguerre polynomials of their respective arguments n, m, and x.

2 *Returns:* The assoc_laguerre functions return

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x), \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if n >= 128.

2.1.2 associated Legendre functions**[tr.num.sf.Plm]**

```
double    assoc_legendre(unsigned l, unsigned m, double x);
float     assoc_legendref(unsigned l, unsigned m, float x);
long double assoc_legendrel(unsigned l, unsigned m, long double x);
```

1 *Effects:* These functions compute the associated Legendre functions of their respective arguments l, m, and x.

2 *Returns:* The assoc_legendre functions return

$$P_\ell^m(x) = (1-x^2)^{m/2} \frac{d^m}{dx^m} P_\ell(x), \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if l >= 128.

2.1.3 beta function**[tr.num.sf.beta]**

```
double    beta(double x, double y);
float     betaf(float x, float y);
long double betal(long double x, long double y);
```

1 *Effects:* These functions compute the beta function of their respective arguments x and y.

2 *Returns:* The beta functions return

$$B(x,y) = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}.$$

2.1.4 (complete) elliptic integral of the first kind**[tr.num.sf.elIK]**

```
double    comp_ellint_1(double k);
float     comp_ellint_1f(float k);
long double comp_ellint_1l(long double k);
```

- 1 *Effects:* These functions compute the complete elliptic integral of the first kind of their respective arguments k .
- 2 *Returns:* The `comp_ellint_1` functions return

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}.$$

2.1.5 (complete) elliptic integral of the second kind

[tr.num.sf.ellEx]

```
double    comp_ellint_2(double k);
float     comp_ellint_2f(float k);
long double comp_ellint_2l(long double k);
```

- 1 *Effects:* These functions compute the complete elliptic integral of the second kind of their respective arguments k .
- 2 *Returns:* The `comp_ellint_2` functions return

$$E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 \sin^2 \theta} d\theta.$$

2.1.6 (complete) elliptic integral of the third kind

[tr.num.sf.ellPx]

```
double    comp_ellint_3(double k, double nu);
float     comp_ellint_3f(float k, float nu);
long double comp_ellint_3l(long double k, long double nu);
```

- 1 *Effects:* These functions compute the complete elliptic integral of the third kind of their respective arguments k and n .
- 2 *Returns:* The `comp_ellint_3` functions return

$$\Pi(\nu, k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}.$$

2.1.7 regular modified cylindrical Bessel functions

[tr.num.sf.I]

```
double    cyl_bessel_i(double nu, double x);
float     cyl_bessel_if(float nu, float x);
long double cyl_bessel_il(long double nu, long double x);
```

- 1 *Effects:* These functions compute the regular modified cylindrical Bessel functions of their respective arguments ν and x .
- 2 *Returns:* The `cyl_bessel_i` functions return

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k! \Gamma(\nu + k + 1)}, \quad \text{for } x \geq 0.$$

- 3 *Note:* The effect of calling each of these functions is implementation-defined if $\nu \geq 128$.

2.1.8 cylindrical Bessel functions (of the first kind)**[tr.num.sf.J]**

```
double    cyl_bessel_j(double nu, double x);
float     cyl_bessel_jf(float nu, float x);
long double cyl_bessel_jl(long double nu, long double x);
```

1 *Effects:* These functions compute the cylindrical Bessel functions of the first kind of their respective arguments nu and x.

2 *Returns:* The cyl_bessel_j functions return

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k! \Gamma(\nu+k+1)}, \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if nu >= 128.

2.1.9 irregular modified cylindrical Bessel functions**[tr.num.sf.K]**

```
double    cyl_bessel_k(double nu, double x);
float     cyl_bessel_kf(float nu, float x);
long double cyl_bessel_kl(long double nu, long double x);
```

1 *Effects:* These functions compute the irregular modified cylindrical Bessel functions of their respective arguments nu and x.

2 *Returns:* The cyl_bessel_k functions return

$$K_\nu(x) = (\pi/2)i^{\nu+1}(J_\nu(ix) + iN_\nu(ix)) = \begin{cases} \frac{\pi}{2} \frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}, & \text{for } x \geq 0 \text{ and non-integral } \nu \\ \frac{\pi}{2} \lim_{\mu \rightarrow \nu} \frac{I_{-\mu}(x) - I_\mu(x)}{\sin \mu\pi}, & \text{for } x \geq 0 \text{ and integral } \nu \end{cases}$$

3 *Note:* The effect of calling each of these functions is implementation-defined if nu >= 128.

2.1.10 cylindrical Neumann functions**[tr.num.sf.N]**

```
double    cyl_neumann(double nu, double x);
float     cyl_neumannf(float nu, float x);
long double cyl_neumannl(long double nu, long double x);
```

1 *Effects:* These functions compute the cylindrical Neumann functions, also known as the cylindrical Bessel functions of the second kind, of their respective arguments nu and x.

2 *Returns:* The cyl_neumann functions return

$$N_\nu(x) = \begin{cases} \frac{J_\nu(x) \cos \nu\pi - J_{-\nu}(x)}{\sin \nu\pi}, & \text{for } x \geq 0 \text{ and non-integral } \nu \\ \lim_{\mu \rightarrow \nu} \frac{J_\mu(x) \cos \mu\pi - J_{-\mu}(x)}{\sin \mu\pi}, & \text{for } x \geq 0 \text{ and integral } \nu \end{cases}$$

3 *Note:* The effect of calling each of these functions is implementation-defined if `nu >= 128`.

2.1.11 (incomplete) elliptic integral of the first kind

[tr.num.sf.ellF]

```
double    ellint_1(double k, double phi);
float     ellint_1f(float k, float phi);
long double ellint_1l(long double k, long double phi);
```

1 *Effects:* These functions compute the incomplete elliptic integral of the first kind of their respective arguments `k` and `phi` (`phi` measured in radians).

2 *Returns:* The `ellint_1` functions return

$$F(k, \phi) = \int_0^{\phi} \frac{d\theta}{\sqrt{1 - k^2 \sin^2 \theta}}, \quad \text{for } |k| \leq 1.$$

2.1.12 (incomplete) elliptic integral of the second kind

[tr.num.sf.ellE]

```
double    ellint_2(double k, double phi);
float     ellint_2f(float k, float phi);
long double ellint_2l(long double k, long double phi);
```

1 *Effects:* These functions compute the incomplete elliptic integral of the second kind of their respective arguments `k` and `phi` (`phi` measured in radians).

2 *Returns:* The `ellint_2` functions return

$$E(k, \phi) = \int_0^{\phi} \sqrt{1 - k^2 \sin^2 \theta} d\theta, \quad \text{for } |k| \leq 1.$$

2.1.13 (incomplete) elliptic integral of the third kind

[tr.num.sf.ellP]

```
double    ellint_3(double k, double nu, double phi);
float     ellint_3f(float k, float nu, float phi);
long double ellint_3l(long double k, long double nu, long double phi);
```

1 *Effects:* These functions compute the incomplete elliptic integral of the third kind of their respective arguments `k`, `nu`, and `phi` (`phi` measured in radians).

2 *Returns:* The `ellint_3` functions return

$$\Pi(\nu, k, \phi) = \int_0^{\phi} \frac{d\theta}{(1 - \nu \sin^2 \theta) \sqrt{1 - k^2 \sin^2 \theta}}, \quad \text{for } |k| \leq 1.$$

2.1.14 exponential integral

[tr.num.sf.ei]

```
double    expint(double x);
float     expintf(float x);
long double expintl(long double x);
```

1 *Effects:* These functions compute the exponential integral of their respective arguments x .

2 *Returns:* The expint functions return

$$\text{Ei}(x) = - \int_{-x}^{\infty} \frac{e^{-t}}{t} dt .$$

2.1.15 Hermite polynomials

[tr.num.sf.Hn]

```
double    hermite(unsigned n, double x);
float     hermitef(unsigned n, float x);
long double hermitel(unsigned n, long double x);
```

1 *Effects:* These functions compute the Hermite polynomials of their respective arguments n and x .

2 *Returns:* The hermite functions return

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2} .$$

3 *Note:* The effect of calling each of these functions is implementation-defined if $n \geq 128$.

2.1.16 Laguerre polynomials

[tr.num.sf.Ln]

```
double    laguerre(unsigned n, double x);
float     laguerref(unsigned n, float x);
long double laguerrel(unsigned n, long double x);
```

1 *Effects:* These functions compute the Laguerre polynomials of their respective arguments n and x .

2 *Returns:* The laguerre functions return

$$L_n(x) = \frac{e^x}{n!} \frac{d^n}{dx^n} (x^n e^{-x}), \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if $n \geq 128$.

2.1.17 Legendre polynomials

[tr.num.sf.Pl]

```
double    legendre(unsigned l, double x);
float     legendref(unsigned l, float x);
long double legendrel(unsigned l, long double x);
```

1 *Effects:* These functions compute the Legendre polynomials of their respective arguments l and x .

2 *Returns:* The legendre functions return

$$P_\ell(x) = \frac{1}{2^\ell \ell!} \frac{d^\ell}{dx^\ell} (x^2 - 1)^\ell, \quad \text{for } |x| \leq 1.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if $l \geq 128$.

2.1.18 Riemann zeta function**[tr.num.sf.riemannzeta]**

```
double    riemann_zeta(double x);
float     riemann_zetaf(float x);
long double riemann_zetal(long double x);
```

1 *Effects:* These functions compute the Riemann zeta function of their respective arguments x.

2 *Returns:* The riemann_zeta functions return

$$\zeta(x) = \begin{cases} \sum_{k=1}^{\infty} k^{-x}, & \text{for } x > 1 \\ 2^x \pi^{x-1} \sin\left(\frac{\pi x}{2}\right) \Gamma(1-x) \zeta(1-x), & \text{for } x < 1 \end{cases}.$$

2.1.19 spherical Bessel functions (of the first kind)**[tr.num.sf.j]**

```
double    sph_bessel(unsigned n, double x);
float     sph_besself(unsigned n, float x);
long double sph_bessell(unsigned n, long double x);
```

1 *Effects:* These functions compute the spherical Bessel functions of the first kind of their respective arguments n and x.

2 *Returns:* The sph_bessel functions return

$$j_n(x) = (\pi/2x)^{1/2} J_{n+1/2}(x), \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if n >= 128.

2.1.20 spherical associated Legendre functions**[tr.num.sf.Ylm]**

```
double    sph_legendre(unsigned l, unsigned m, double theta);
float     sph_legendref(unsigned l, unsigned m, float theta);
long double sph_legendrel(unsigned l, unsigned m, long double theta);
```

1 *Effects:* These functions compute the spherical associated Legendre functions of their respective arguments l, m, and theta (theta measured in radians).

2 *Returns:* The sph_legendre functions return

$$Y_{\ell}^m(\theta, 0)$$

where

$$Y_{\ell}^m(\theta, \phi) = (-1)^m \left[\frac{(2\ell+1)}{4\pi} \frac{(\ell-m)!}{(\ell+m)!} \right]^{1/2} P_{\ell}^m(\cos \theta) e^{im\phi}, \quad \text{for } |m| \leq \ell.$$

[*Note:* This formulation avoids any need to return non-real numbers. —end note]

3 *Note:* The effect of calling each of these functions is implementation-defined if l >= 128.

2.1.21 spherical Neumann functions

[tr.num.sf.n]

```
double    sph_neumann(unsigned n, double x);
float     sph_neumannf(unsigned n, float x);
long double sph_neumannl(unsigned n, long double x);
```

1 *Effects:* These functions compute the spherical Neumann functions, also known as the spherical Bessel functions of the second kind, of their respective arguments `n` and `x`.

2 *Returns:* The `sph_neumann` functions return

$$n_n(x) = (\pi/2x)^{1/2} N_{n+1/2}(x), \quad \text{for } x \geq 0.$$

3 *Note:* The effect of calling each of these functions is implementation-defined if `n >= 128`.

2.2 Additions to header <tgmath.h>

[tr.sf.tgmath]

1 The header <tgmath.h> includes the header <math.h> and defines several type-generic macros.

2 Of the functions added by this TR to <math.h> without an `f` (float) or `l` (long double) suffix, several have one or more parameters whose corresponding real type is `double`. For each such function there is a corresponding type-generic macro. ²⁾ The parameters whose corresponding real type is `double` in the function synopsis are generic parameters. Use of the macro invokes a function whose corresponding real type and type domain are determined by the arguments for the generic parameters. ³⁾

3 Use of the macro invokes a function whose generic parameters have the corresponding real type determined as follows:

- First, if any argument for generic parameters has type `long double`, the type determined is `long double`.
- Otherwise, if any argument for generic parameters has type `double` or is of integer type, the type determined is `double`.
- Otherwise, the type determined is `float`.

4 For each unsuffixed function added to <math.h> the corresponding type-generic macro has the same name as the function. These type-generic macros are:

Table 3: Additions to header <tgmath.h> synopsis

Macros:		
<code>assoc_laguerre</code>	<code>cyl_bessel_j</code>	<code>hermite</code>
<code>assoc_legendre</code>	<code>cyl_bessel_k</code>	<code>legendre</code>
<code>beta</code>	<code>cyl_neumann</code>	<code>laguerre</code>
<code>comp_ellint_1</code>	<code>ellint_1</code>	<code>riemann_zeta</code>
<code>comp_ellint_2</code>	<code>ellint_2</code>	<code>sph_bessel</code>
<code>comp_ellint_3</code>	<code>ellint_3</code>	<code>sph_legendre</code>
<code>cyl_bessel_i</code>	<code>expint</code>	<code>sph_neumann</code>

²⁾Like other function-like macros in Standard libraries, each *type-generic macro* can be suppressed to make available the corresponding ordinary function.

³⁾If the type of the argument is not compatible with the type of the parameter for the selected function, the behavior is undefined.

If all arguments for generic parameters are real, then use of the macro invokes a real function; otherwise, use of the macro results in undefined behavior.

Bibliography

- [1] International Standards Organization: *Technical Report 1 on C++ Library Extensions*. International Standard ISO/IEC TR 19768:2006.
- [2] Milton Abramowitz and Irene A. Stegun (eds.): *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, volume 55 of National Bureau of Standards Applied Mathematics Series. U. S. Government Printing Office, Washington, DC: 1964. Reprinted with corrections, Dover Publications: 1972.
- [3] Walter Brown, "A Proposal to Add Mathematical Special Functions to the C++ Standard Library," WG21 Document N1422 = 03-0004, 2003.
- [4] P. J. Plauger, "Proposed Special Math functions." WG14 Document N1023, 2003.
- [5] International Standards Organization: *Quantities and units, Third edition*. International Standard ISO 31-11:1992. ISBN 92-67-10185-4.
- [6] International Standards Organization: *Programming Languages – C, Second edition*. International Standard ISO/IEC 9899:1999.
- [7] International Standards Organization: *Programming Languages – C++*. International Standard ISO/IEC 14882:2003.
- [8] William H. Press, et al.; *Numerical Recipes in C : The Art of Scientific Computing*. Second Edition ISBN 0-521-43108-5

Index

assoc_laguerre, 6
assoc_legendre, 6

Bessel functions, 7, 8, 11, 12
beta, 6

comp_ellint_1, 6
comp_ellint_2, 7
comp_ellint_3, 7
cyl_bessel_i, 7
cyl_bessel_j, 8
cyl_bessel_k, 8
cyl_neumann, 8

ellint_1, 9
ellint_2, 9
ellint_3, 9
elliptic integrals, 6, 7, 9
 complete, 6, 7
 incomplete, 9

Eulerian integral of the first kind, *see* beta
expint, 9
exponential integral, 9

hermite, 10
 H_n polynomials, 10

I_ν , 7

J_ν , 8
 j_n , 11

K_ν , 8

laguerre, 10
Laguerre polynomials, 6
legendre, 10

Legendre functions, 6, 10, 11

`<math.h>`, 3

N_ν , 8
NaN, 5
Neumann functions, 8, 12
 n_n function, 12

P_l polynomials, 10
 P_m , 6

riemann_zeta, 11

special functions, 3–13
sph_bessel, 11
sph_legendre, 11
sph_neumann, 12
spherical harmonics, 11

`<tgmath.h>`, 12
type-generic macro, 12

undefined behavior, 13

$Y_l^m(\theta, \phi)$, 11

ζ function, 11