IEEE STANDARDS ASSOCIATION IEEE

IEEE 浮点运算标准

电气和电子工程师学会计算机协会

由
微处理器标准委员会

电气和电子工程师学会
公园大道 3 号
美国纽约州纽约市 10016-
5997

IEEE 标准 754™-2019

IEEE 标准 754™-2019
(修订 IEEE 标准 754-2008）

IEEE 浮点运算标准

赞助商

微处理器标准委员会
的
电气和电子工程师学会计算机协会

2019 年 6 月 13 日批准

IEEE-SA 标准委员会

摘要：本标准规定了计算机编程环境中二进制和十进制浮点运算的交换和运算格式及方法。本标准规定了异常情况及其默认处理方法。符合本标准的浮点系统的实现可完全由软件、硬件或软件与硬件的任何组合来实现。对于本标准规范部分规定的运算，数值结果和异常情况由输入数据值、运算顺序和目标格式唯一决定，全部由用户控制。

关键词： 算术, 二进制, 计算机, 十进制, 指数, 浮点数, 格式, IEEE 754™, 交换, NaN, 数字, 四舍五入, 有效数字, 次正常值。

电气与电子工程师学会（The Institute of Electrical and
Electronics Engineers, Inc.美国纽约州纽约市公园大道 3
号 10016-5997

PDF： ISBN 978-1-5044-5924-2 STD23738
打印： ISBN 978-1-5044-5925-9 STDPD23738

IEEE 禁止歧视、骚扰和欺凌。

有关 IEEE 标准文件的重要通知和免责声明

IEEE 文件的使用受重要声明和法律免责声明的约束。这些通知和免责声明或对本页面的引用出现在所有标准中，可在 "关于 IEEE 标准文件的重要通知和免责声明 "标题下找到。也可向 IEEE 索取或在 http://standards.ieee.org/ipr/disclaimers.html 上查看。

关于使用 IEEE 标准文件的通知和免责声明

IEEE 标准文件（标准、推荐实践和指南）（包括全面使用和试用）由 IEEE 协会和 IEEE 标准协会（"IEEE-SA"）标准委员会的标准协调委员会制定。IEEE （"协会"）通过美国国家标准协会（"ANSI"）批准的共识开发流程制定标准，该流程汇集了代表不同观点和利益的志愿者，以实现最终产品。IEEE 标准是通过科学、学术和行业技术工作组制定的文件。IEEE 工作组中的志愿者并不一定是该协会的成员，他们的参与没有 IEEE 的报酬。虽然 IEEE 负责管理过程并制定规则以促进共识制定过程的公平性，但 IEEE 不会独立评估、测试或验证其标准中包含的任何信息或任何判断的准确性。

IEEE 标准不保证或确保安全、安保、健康或环境保护，也不确保不会干扰其他设备或网络。IEEE 标准文件的实施者和用户有责任确定并遵守所有适当的安全、保安、环境、健康和干扰保护措施以及所有适用的法律和法规。

IEEE 不保证或表示其标准中包含的材料的准确性或内容，并明确否认本文件或与标准相关的任何其他文件中未包含的所有保证（明示、暗示和法定），包括但不限于以下保证：适销性；特定用途的适用性；非侵权性；以及材料的质量、准确性、有效性、时效性或完整性。此外，IEEE 拒绝承认与以下方面有关的任何及所有条件：结果；勤奋努力。IEEE 标准文件 "按原样 "和 "不带任何故障 "提供。

使用 IEEE 标准完全出于自愿。IEEE 标准的存在并不意味着没有其他方法来生产、测试、测量、购买、销售或提供与 IEEE 标准范围相关的其他产品和服务。此外，在批准和发布标准时所表达的观点可能会因技术发展和标准用户的意见而发生变化。

在出版和提供其标准时，IEEE 并非为任何个人或实体或代表任何个人或实体建议或提供专业或其他服务，也不承诺履行任 何 其他个人或实体对他人承担的任何责任。任何人在使用任何 IEEE 标准文件时，都应依靠自己的独立判断，在任何特定情况下行使合理的谨慎，或酌情寻求有能力的专业人士的建议，以确定特定 IEEE 标准的适当性。

在任何情况下，IEEE 均不对任何直接、间接、附带、特殊、惩戒性或后果性损害（包括但不限于采购替代产品或服务；使用、数据或利润损失；或业务中断），无论其原因如何，也无论其责任理论如何，是否是合同、严格责任或侵权行为（包括疏忽或其他），以任何方式产生于任何标准的发布、使用或依赖，即使已被告知此类损害的可能性，也无论此类损害是否可预见。

翻译

IEEE 共识制定过程只涉及审查英文文件。如果 IEEE 标准被翻译，则只有 IEEE 出版的英文版本才应被视为已获批准的 IEEE 标准。

正式声明

未按照《IEEE-SA 标准委员会操作手册》处理的书面或口头声明不应被视为或推断为 IEEE 或其任何委员会的正式立场，也不应被视为或依赖为 IEEE 的正式立场。在讲座、座谈会、研讨会或教育课程中，介绍 IEEE 标准信息的个人应明确其观点应视为个人观点，而非 IEEE 的正式立场。

对标准的评论

欢迎任何相关方对 IEEE 标准文件的修订提出意见，无论其是否隶属于 IEEE。但是，IEEE 不提供与 IEEE 标准文件有关的咨询信息或建议。对文件的修改建议应以文本修改建议的形式提出，并附上适当的支持意见。由于 IEEE 标准代表了相关利益方的共识，因此对意见和问题的任何回复也必须得到利益平衡方的同意。因此，IEEE 及其学会和标准协调委员会的成员无法立即对意见或问题做出回复，除非相关问题之前已经得到解决。出于同样的原因，IEEE 也不对口译请求作出回应。欢迎任何希望参与 IEEE 标准修订的人员加入相关的 IEEE 工作组。

有关标准的意见应提交至以下地址：秘书，IEEE-SA 标准委员会
445 Hoes Lane
美国新泽西州皮斯卡塔韦 08854

法律法规

IEEE 标准文件的用户应查阅所有适用的法律和法规。遵守任何 IEEE 标准文件的规定并不意味着遵守任何适用的法规要求。标准的实施者有责任遵守或参考适用的法规要求。IEEE 并不打算通过发布其标准来敦促不符合适用法律的行为，这些文件也不得被解释为不符合适用法律的行为。

版权

根据美国和国际版权法，IEEE 起草和批准的标准版权归 IEEE 所有。这些标准由 IEEE 提供，并被广泛用于公共和私人用途。这些用途包括在法律法规中参考使用，以及在私人自律、标准化和推广工程实践与方法中使用。IEEE 将这些文件提供给公共机构和私人用户使用和采用，并不意味着放弃这些文件的任何版权权利。

复印件

在支付相应费用的前提下，IEEE 将授予用户有限的、非排他性的许可，允许其复印任何单个标准的部分内容，仅供公司或组织内部使用，或个人非商业用途。如需安排支付许可费，请联系版权结算中心（Copyright Clearance Center）客户服务部，地址：222 Rosewood Drive, Danvers, MA 01923 USA；+1 978 750 8400。也可通过版权结算中心获得许可，复印任何单个标准的部分内容供教育课堂使用。

更新 IEEE 标准文件

IEEE 标准文件的用户应了解，这些文件可能随时被新版本取代，也可能随时通过发布修订、勘误或勘误表进行修改。当前的 IEEE 文档在任何时候都由当前版本的文档以及当时有效的任何修正案、勘误表或勘误表组成。

每个 IEEE 标准至少每十年审查一次。如果一份文件已经有十年以上的历史，却没有经过修订过程，那么我们有理由认为，其内容虽然仍有一定的价值，但已不能完全反映当前的技术水平。用户应注意检查自己是否拥有任何 IEEE 标准的最新版本。

要确定给定文件是否为当前版本，以及是否通过发布修正案、勘误表或勘误表进行过修订，请访问 IEEE Xplore：http://ieeexplore.ieee.org/，或按前面列出的地址联系 IEEE。有关 IEEE-SA 或 IEEE 标准制定过程的更多信息，请访问 IEEE-SA 网站 http://standards.ieee.org。

勘误表

所有 IEEE 标准的勘误表（如有）均可在 IEEE-SA 网站上查阅，网址如下：http://standards.ieee.org/findstds/errata/index.html。我们鼓励用户定期查看此 URL 中的勘误表。

专利

请注意，实施本标准可能需要使用受专利权保护的主题。通过发布本标准，IEEE 不会对与之相关的任何专利权的存在或有效性采取任何立场。如果专利持有人或专利申请人已通过 "已接受保证函" 提交了保证声明，则该声明将在 IEEE-SA 网站 http://standards.ieee.org/about/sasb/patcom/patents.html 上列出。保证书可说明提交者是否愿意无偿或以合理的费率、合理的条款和条件授予专利权许可，这些条款和条件明显不存在对希望获得此类许可的申请人的任何不公平歧视。

可能存在尚未收到保证书的必要专利权利要求。IEEE 不负责确定可能需要许可的基本专利权利要求，不负责对专利权利要求的法律有效性或范围进行调查，也不负责确定在提交保证书（如有）时或在任何许可协议中提供的任何许可条款或条件是否合理或不具歧视性。明确告知本标准的用户，

确定任何专利权的有效性以及侵犯此类权利的风险完全由他们自己负责。更多信息可向 IEEE 标准协会索取。

与会者

浮点工作组的下列成员为本标准的制定做出了贡献：

戴维-霍夫，*主席*
迈克-考利肖，*编辑*

Jonathan Bradbury
Neil Burgess
David H. C. Chen
Marius Cornea
John H. Crawford
Joe Darcy
James Demmel
Florent de Dinechin
Ken Dockser
Hossam A. H. Fahmy
Warren E. Ferguson
David M. Gay
Ivan Godard
Roger A. Golliver
Mrudula Gore
Trenton Grale
米歇尔-哈
克 约翰-豪
瑟
彼得-亨德森
威廉-卡汉
R.Christoph
Lauter Vincent
Lefèvre David
Lutz
泰耶-马蒂森
戴维-马图拉
伊恩-麦金托
什
Richard A. Painter
Bogdan Pasca
Nathalie Revol
Jason Riedy
Eric M. Schwarz
James W. Thomas
Leonard Tsai
Fred J. Tydeman
Liang-Kai Wang
Lee Winter

下列投票委员会成员对本标准进行了投票。投票者可以投赞成票、反对票或弃权票。

Robert Aiello
Amelia Andersdotter
Israel Barrientos
Demetrio Bucaneg Jr.
David H. C. Chen
James Cloos
Marius Cornea
Mike Cowlishaw
James Demmel
Ken Dockser
Hossam A. H. Fahmy
Andrew Fieldsend
David M. Gay
H.格利肯斯坦
罗杰-A-戈利弗
Randall Groves
Michel Hack
Peter Harrod
Chris N. Hinds
Werner Hoelzl
David G. Hough
Piotr Karocki
R.Baker Kearfott
Jim Kulchisky
Christoph Lauter
Vincent Lefèvre
Edward Mccall
Jean-Michel Muller
Bruce Muschlitz
内德-内达尔科夫
Nick S. A. Nikjoo
Richard A. Painter
John Pryce
Nathalie Revol
Jason Riedy
Randy Saunders
Eric M. Schwarz
James Stine
Walter Struppler
James W. Thomas
Michael Thompson
Leonard Tsai
Forrest Wright
Jian Yu
奥伦-袁

当 IEEE-SA 标准委员会于 2019 年 6 月 13 日批准本标准时，其成员如下：

加里-霍夫曼，*主席*
Ted Burse，*副主席*
让-菲利普-福尔，*前任主席*
Konstantinos Karachalios，*秘书*

Masayuki Ariyoshi
Stephen D. Dukes
J.特拉维斯-格里
菲斯 Guido Hiertz
Christel Hunter
约瑟夫-科普芬格* 托马斯-科希
约翰-D-库利克
David J. Law
Joseph Levy
Howard Li
Xiaohui Liu
Kevin Lu
Daleep Mohla
Andrew Myles
Annette Reilly
Dorothy Stanley
Sha Wei
Phil Wennblom
Philip Winston
Howard Wolfman
Feng Wu
Jingyi Zhou

* 名誉委员

导言

本简介并非 IEEE Std 754-2019（IEEE 浮点运算标准）的一部分。

该标准是 IEEE 计算机协会微处理器标准委员会浮点工作组的产品，由该委员会赞助。

本标准为执行浮点运算提供了一套规范，其结果与硬件、软件或二者结合的处理方式无关。对于本标准规范部分规定的运算，数值结果和异常情况由输入数据、运算和目的地的值唯一决定，全部由用户控制。

该标准为系统执行二进制和十进制浮点运算定义了一系列商业上可行的方法。制定本标准的主要考虑因素包括

a) 促进现有程序从不同的计算机转移到遵守本标准的计算机，以及遵守本标准的计算机之间的转移。

b) 提高用户和程序员的能力和安全性，虽然他们并不精通数值方法，但很可能会尝试编制复杂的数值程序。

c) 鼓励专家们开发并发布强大而高效的数字程序，这些程序只需稍加编辑和重新编译，就可移植到任何符合本标准并具有足够容量的计算机上。加上语言控制，应该可以编写出在所有符合标准的系统上产生相同结果的程序。

d) 直接支持

- 执行时诊断异常
- 更顺畅地处理异常
- 以合理的成本实现区间运算。

e) 规定发展

- 常见的基本函数，如 *exp* 或 *cos*
- 高精度（多字）算术
- 耦合数值计算和符号代数计算。

f) 使进一步完善和扩展成为可能，而不是被排除在外。

在编程环境中，该标准还旨在为数值社区与编程语言设计者之间的对话奠定基础。我们希望在未来几年内，能定义出控制表达式评估和异常情况的语言定义方法，这样就有可能编写出在所有符合标准的系统上产生相同结果的程序。不过，我们也认识到，语言的实用性和安全性有时是对立的，效率和可移植性也是对立的。

因此，我们希望语言设计者能将此处描述的全套操作、精度和异常控制作为指南，为程序员提供可

移植地控制表达式和异常的能力。同时，我们也希望设计者能在本标准的指导下，以完全可移植的方式提供扩展功能。

资料性附件提供了更多信息--附件 A 列出了书目资源，附件 B 建议了调试支持的编程环境功能，附件 C 列出了标准操作的所有参考资料。

目录

1. 概述 11
1.1 范围 11
1.2 目的 11
1.3 夹杂物 11
1.4 不适用情况 11
1.5 编程环境考虑因素 12
1.6 用词 12

2. 定义、缩略语和首字母缩略词 13
2.1 定义 13
2.2 缩略语 15

3. 浮点格式 16
3.1 概述 16
3.2 规格等级 17
3.3 浮点数据集 17
3.4 二进制交换格式编码 19
3.5 十进制交换格式编码 20
3.6 交换格式参数 23
3.7 可扩展的精确度 25

4. 属性和四舍五入 26
4.1 属性规范 26
4.2 属性的动态模式 26
4.3 舍入方向属性 27

5. 业务 29
5.1 概述 29
5.2 小数指数计算 30
5.3 同构通用计算运算 31
5.4 一般计算操作的格式 33
5.5 静音计算操作 35
5.6 信号--计算操作 37
5.7 非计算业务 37
5.8 从浮点格式转换为整数格式的详情 39
5.9 浮点数据四舍五入为积分值的操作详情 41
5.10 totalOrder 谓词的详细信息 42
5.11 比较谓词的细节 43
5.12 浮点数据与外部字符序列之间的转换详情 44

6. 无穷大、NaNs 和符号位 48
6.1 无穷大运算 48
6.2 对 NaN 的运算 48
6.3 符号位 50

7. 异常和默认异常处理 51
7.1 概述：异常和标记 51
7.2 无效操作 52
7.3 除以零 53
7.4 溢出 53
7.5 下溢 53
7.6 不精确 54

8. 备用异常处理属性 55
8.1 概述 55
8.2 恢复备用异常处理属性 55

8.3 立即和延迟交替异常处理属性 56

9. 建议的业务 58
9.1 符合语言和实施定义的操作 58
9.2 其他数学运算 58
9.3 动态模式操作 65
9.4 削减行动 66
9.5 增强算术运算 68
9.6 最小和最大操作数 69
9.7 NaN 有效载荷操作 71

10. 表达评估 72
10.1 表达式评估规则 72
10.2 分配、参数和函数值 72
10.3 用于表达式评估的首选宽度属性 73
10.4 字面意义和改变价值的优化 74

11. 可重复的浮点运算结果 75

附件 A（资料性）参考书目 77

附件 B（资料性）程序调试支持 79

附件 C（资料性）业务清单 81

IEEE 浮点运算标准

1. 概述

1.1 范围

本标准规定了计算机系统中浮点运算的格式和操作。定义了异常条件，并规定了这些条件的处理方法。

1.2 目的

本标准提供了一种浮点数计算方法，无论采用硬件、软件或两者结合的方式进行处理，计 算 结 果 都 是 相同的。在输入数据相同的情况下，计算结果将完全相同，与执行方式无关。数学处理过程中的错误和错误条件将以一致的方式报告，与执行方式无关。

1.3 内含物

本标准规定

- 二进制和十进制浮点数据格式，用于计算和数据交换。
- 加法、减法、乘法、除法、融合乘加、平方根、比较和其他运算。
- 整数和浮点格式之间的转换。
- 不同浮点格式之间的转换。
- 浮点格式和字符序列外部表示之间的转换。
- 浮点异常及其处理，包括非数字（NaN）数据。

1.4 不适用情况

本标准未作规定：

— 整数的格式

— NaNs 的符号字段和意义字段的解释。

1.5 编程环境考虑因素

该标准规定了 2 和 10 两种弧度的浮点运算。编程环境可以在一个弧度或两个弧度上符合该标准。

本标准并未定义符合标准的编程环境的所有方面。这些行为应由支持本标准的编程语言定义（如果有的话）来定义，否则应由特定的实现来定义。某些编程语言规范可能允许由实施方案来定义某些行为。

语言定义的行为应由支持本标准的编程语言标准来定义。这样，所有同时符合本浮点标准和该语言标准的实现，在这些语言定义的行为方面的表现都是相同的。对于那些旨在在所有平台上精确再现结果的语言标准，其对行为的规定应比那些旨在在每个平台上最大限度地提高性能的语言标准更为严格。

由于本标准所要求的设施是目前常用编程语言所不具备的，因此，如果这些语言的标准不再进行修订，它们就可能无法完全符合本标准。如果语言可以通过函数库或类或软件包进行扩展，以提供一个符合标准的环境，那么该扩展应定义语言标准通常定义的所有语言行为。

实现定义的行为是由符合本标准的特定编程环境的具体实现所定义的。实现定义的行为既不是本标准规定的，也不是任何相关编程语言标准或编程语言扩展标准规定的。

是否符合本标准是特定编程环境的具体实施的属性，而不是语言规范的属性。

但是，如果一种语言标准的构造能使该语言的每一种符合标准的实现都自动符合该标准，那么也可以说该语言标准符合该标准。

1.6 词语用法

该标准使用了以下三个词来区分不同层次的要求和可选性：

- **可"**（may）表示在标准范围内允许采取的行动，并不暗示优先选择（"可 "表示 "允许"）。
- **应 "**表示为符合标准而必须严格遵守的强制性要求，不允许偏离（"应 "表示 "必须"）。
- **should** "表示在几种可能性中，推荐一种特别合适，而不提及或排除其他可能性；或表示某种行动方案是首选，但不一定是必需的；或表示（在否定形式中）某种行动方案是不可取的，但不禁止（"should "表示 "建议"）。

更多

- **可能 "**表示一种情况可能发生，但并不暗示这种情况发生的可能性（"可能 "表示 "有可能"）。
- 数字后的 "**见 "**是本标准中由数字标识的条款或子条款的对照索引
- **注释**引入了信息性文本（即不是本标准的要求）。

2. 定义、缩写和首字母缩略词

2.1 定义

本标准适用以下术语和定义。

适用属性：管理本标准计算操作特定执行实例的属性值。语言规定了如何确定适用属性。

算术格式：浮点运算格式：一种浮点运算格式，可用于表示本标准运算中的浮点操作数或结果。

属性：本标准操作的隐式参数，用户可在编程语言中通过指定一个常量值来静态设置该参数。属性一词可以指参数（如 "舍入方向属性"）或其值（如 "向零舍入属性"）。

基本格式：五种浮点表示法之一，三种二进制，两种十进制，其编码由本标准规定，可用于算术运算。任何符合标准的实施方案都能实现一种或多种基本格式。

偏置指数：指数与常数（偏置）之和，偏置指数的取值范围为非负。

二进制浮点数：弧度为 2 的浮点数。

块：语块：一种语言定义的语法单位，用户可为其指定属性。语言标准可为用户提供为不同范围的块指定属性的方法，甚至大到整个程序，小到单个操作。

典型编码：浮点表示格式的首选编码。"典型编码 "也适用于小数、有限数的符号、无穷大和 NaN，尤其是十进制格式。

群组：以给定浮点格式表示给定浮点数的所有浮点表示的集合。在这种情况下，-0 和 +0 被认为是不同的，属于不同的群组。

计算操作：产生浮点结果或可能发出浮点异常信号的操作。计算操作产生浮点或其他目标格式的结果，必要时通过四舍五入使其适合。

正确四舍五入：本标准将无限精确结果转换为浮点数的方法，由适用的四舍五入方向决定。这样得到的浮点数称为正确四舍五入数。

十进制浮点数：弧度为十的浮点数。

十进制：使用密集十进制编码方案将三位十进制数字编码为十位。计算运算接受操作数中所有 1024 个可能的小数。大多数计算运算只产生 1000 个标准小数。

去规范化数：*见：***次正态数。**

目的地：目的地：对一个或多个操作数进行操作的结果位置。目的地可以由用户明确指定，也可以由系统隐式提供（例如，子表达式的中间结果或程序的参数）。尽管有些语言将中间计算结果放在用户无法控制的目的地，但本标准还是根据目的地的格式和操作数的值来定义操作结果。

动态模式：一种可选的动态设置属性的方法，通过本标准的操作来设置、测试、保存和恢复属性。

异常：异常：当对某些特定操作数的操作结果不适 合 所有合理应用时发生的事件。该操作可能通过调用默认异常处理或备用异常处理发出异常信号。异常处理可能会发出更多异常信号。认识到在不同的编程环境中，*事件*、*异常*和*信号的*定义方式各不相同。

指数：指数：有限浮点表示法的组成部分，表示在确定浮点表示法的值时弧度所提升的整数幂。当符号被视为整数位数和分数域时，使用指数 e；当符号被视为整数时，使用指数 q；$e = q + p - 1$，其中 p 是以位数为单位的格式精度。

可扩展精度格式：由用户控制定义精度和范围的格式。

扩展精度格式：通过提供更宽的精度和范围来扩展受支持的基本格式的格式。

外部字符序列：浮点数据的字符序列表示法，包括程序文本中浮点字面量的字符序列。

标志：*参见：***状态标志。**

浮点数据：可以用浮点格式表示的浮点数或非数字（NaN）。在本标准中，浮点数据并不总是与其表示或编码区分开来。

浮点数：可用浮点格式表示的有限或无限数字。非 NaN 的浮点数据。所有浮点数，包括零和无穷小，都是有符号的。

浮点运算：操作数或结果为浮点数据的操作。

浮点表示：浮点表示法：浮点格式中未编码的一种，表示有限数值、带符号的无穷大、静态 NaN 或信号 NaN。有限数值的表示有三个部分：符号、指数和有效数字；其数值是有效数字的符号乘积和指数的幂级数。

格式：格式：一组数值和符号的表示法，可能还附有编码。

fusedMultiplyAdd：操作 fusedMultiplyAdd(x, y, z) 将 $(x \times y) + z$ 的计算结果视作无限制的范围和精度，只对目标格式进行一次四舍五入。

通用操作：本标准中的一种操作，可接受各种格式的操作数，其结果格式可能取决于操作数的格式。

同质运算：本标准中的一种操作，其操作数和返回结果的格式都相同。

实施定义：由符合本标准的特定编程环境的具体实现所定义的行为。

整数格式：一种未在本标准中定义的格式，表示整数的一个子集，也可能表示代表无穷大、NaN 或负零的附加值。

交换格式：一种在本标准中定义了特定固定宽度编码的格式。

语言定义：由支持本标准的编程语言标准所定义的行为。

NaN：不是数字--一种符号浮点数据。有两种 NaN 表示：静态 NaN 和信号 NaN。大多数运算在传播**静态 NaN** 时不会发出异常信号，而在给定**信号 NaN 操作数**时会发出无效运算异常信号。

较窄/较宽的格式：如果一种格式的浮点数集是另一种格式的适当子集，则前者称为较窄格式，后者称为较宽格式。较宽的格式可能具有更高的精度、范围或（通常）两者都有。

非计算操作：非运算操作：非运算操作。

正常数：对于特定格式而言，是一个非零的有限浮点数，其大小大于或等于最小 b^{emin} 值，其中 b 是弧度。正常数可以使用格式中的全部精度。在本标准中，零既不是正则数，也不是次正则数。

不是数字：*参见：***NaN.**

操作：本标准定义了对零个或多个操作数进行操作并产生结果或副作用（如动态模式或标志或控制流的变化，或两者兼而有之）的必备和推荐操作。在本标准中，操作以命名函数的形式编写；在特定的编程环境中

它们可能由操作符表示，或由特定格式函数族表示，或由名称可能与本标准不同的操作或函数表示。

有效载荷：有效载荷：NaN 中包含的信息，可能是诊断信息。

精度：格式中可表示的最大有效位数 p，或结果四舍五入的位数。

首选指数：对于十进制运算的结果，当结果精确时，最能反映操作数量纲的指数 q 值。

preferredWidth 方法：编程语言用于确定通用操作和函数的目标格式的方法。一些 **preferredWidth** 方法可利用宽格式的额外范围和精度，而不需要在编写程序时进行显式转换。

量子：有限浮点表示法的量子是其符号最后一个位置的单位值。它等于弧度上升到指数 q，当符号被视为整数时使用指数 q。

静音操作：从不发出浮点异常信号的操作。

弧度：二进制或十进制浮点数的表示基数，二或十。

结果：传送到目的地的浮点表示或编码。

信号当对某些特定操作数的操作没有适合所有合理应用的结果时，该操作可能会通过调用默认处理方式或（如果明确要求）用户选择的语言定义的替代处理方式，发出一个或多个异常信号。

有效数字：有限浮点数中包含有效数字的部分。通过选择适当的指数偏移量，可将有效数字视为整数、分数或其他定点形式。十进制或次正常二进制的有效数字也可以包含不重要的前导零。

状态标志：状态标志：一种变量，可以有两种状态，升高或降低。当状态标志升起时，可能会传递与系统相关的额外信息，某些用户可能无法获取这些信息。本标准的运算在出现异常时，可能会引发以下状态标志：不精确、下溢、溢出、除以零和无效运算。

次正常数：在特定格式中，大小小于该格式最小正常数大小的非零浮点数。次正常数不使用同一格式正常数的全部精度。

支持的格式：浮点格式：编程环境中提供的浮点格式，其实现符合本标准的要求。因此，编程环境提供的格式可能多于其支持的格式，因为只有那些按照本标准实施的格式才被称为支持的格式。此外，如果一种整数格式与所支持的浮点格式之间的转换是按照本标准提供的，那么这种整数格式就是受支持的。

尾部符号字段：编码二进制或十进制浮点格式的一个组成部分，包含除前导数字外的所有意义数字。在这些格式中，偏置指数或组合字段编码或暗示前导符号位。

用户：用户： 任何非本标准规定的个人、硬件或程序，有权访问和控制本标准规定的编程环境的操作。

操作宽度：本标准规定的操作目的地格式；它将是符合本标准的实现所提供的支持格式之一。

2.2 缩略语

LSB 最小有效位

MSB 最大有效位

NaN 非 数字

qNaN 安静 NaN

sNaN 信号 NaN

3. 浮点格式

3.1 概述

3.1.1 格式

本条款定义浮点数 或格式，用于表示实数的有限子集（见 3.2）。格式的特点是弧度、精度和指数范围，每种格式可以表示一组独特的浮点数据（见 3.3）。

所有格式都可作为**算术格式**支持；也就是说，它们可用于表示浮点操作数或本标准后续条款所述操作的结果。

本条款为二进制和十进制格式的子集（见 3.4 和 3.5）定义了特定的固定宽度编码。这些**交换格式**根据其大小进行标识（见 3.6），可用于实现之间的浮点数据交换。

本条款定义了五种**基本格式：**

— 三种二进制格式，编码长度分别为 32、64 和 128 位。

— 两种十进制格式，编码长度分别为 64 位和 128 位。

建议使用其他算术格式来扩展这些基本格式（见 3.7）。

选择支持本标准中的哪种格式是由语言决定的，或者，如果相关语言标准未作规定，或由实现决定，则是由实现决定的。本标准中使用的格式名称不一定是编程环境中使用的名称。

3.1.2 一致性

任何受支持格式的符合性实现都应提供初始化该格式的方法，并应提供该格式与所有其他受支持格式之间的转换。

受支持的算术格式的符合性实现应为该格式提供第 5 条中定义的本标准的所有操作。

受支持的交换格式的符合性实现应为该格式提供使用本条款定义的特定编码读写该格式的方法。

编程环境在特定弧度下，通过实施该弧度的一种或多种基本格式作为支持的算术格式和支持的交换格式，来符合本标准。

3.2 规格等级

浮点运算是实数运算的系统近似，如表 3.1 所示。浮点运算只能表示实数连续体的有限子集。因此，实数运算的某些性质，如加法的联立性，并不总是适用于浮点运算。

表 3.1--特定格式不同规范级别之间的关系

第 1 级	**{-∞ ... 0 ... +∞}**	扩展实数
多对一↓	*四舍五入*	↑ 投影（除 NaN 外)
二级	**{-∞ ... -0}○ {+0 ... +∞}○ NaN**	浮点数据 - 一个代数封闭系统。
一对多 ↓	*表示规格*	↑ 多对一
第 3 级	***(符号、指数、意义符)*** ○ **{-∞, +∞}qNaN ○ sNaN**	浮点数据的表示方法
一对多 ↓	*浮点数据表示编码*	↑ 多对一
第 4 级	**0111000...**	位字符串。

本标准运算所依据的数学结构是扩展实数，即实数集合加上正无穷和负无穷。对于给定的格式，*四舍五入*过程（见第 4 条）将扩展实数映射为该格式中的*浮点数*。*浮点数据*可以是带符号的零、有限非零数、带符号的无穷大或 NaN（非零数），可以映射到格式中的一个或多个*浮点数据表示*。

浮点数据的格式表示包括

— 三元组（*符号*、*指数*、*有效数字*）；在弧度 *b 中*，三元组表示的浮点数为 $(-1)^{sign} \times b^{exponent} \times significand$

- +∞, -∞

— qNaN（安静），sNaN（信号）。

*编码*将浮点数据的表示映射到位字符串。编码可能会将浮点数据的某些表示映射到一个以上的位字符串。NaN 编码应用于存储回溯诊断信息（见 6.2）。

3.3 浮点数据集

本子条款规定了可在所有浮点格式中表示的浮 点数据集；交换格式中浮点数据特定表示的编码在 3.4 和 3.5 中定义，交换格式的参数在 3.6 中定义。

特定格式下可表示的 有限浮点数集合由以下整数参数决定：

— *b* = 半径，2 或 10

— *p* = 有效数字的位数（精度）

— *emax* = 最大指数 *e*

— *emin* = 所有格式的最小指数 *e emin*

应为 1 - *emax*。

表 3.2 列出了每种基本格式的这些参数值，其中每种格式都用其弧度和编码位数来标识。扩展和可扩展精度格式对这些参数的限制见 3.7。

在每种格式中，应表示以下浮点数据：

— 形式为 $(-1)^s \times b^e \times m$ *的*有符号零和非零浮点数，其中

  — s 为 0 或 1。

  — e 是任意整数 $emin \le e \le emax$。

  — m 是一个数字，由一个数字字符串表示，其形式为

  $_{d0}$. $_{d1\ d2 \ldots dp-1}$ 其中 $_{di}$ 为整数位 $0 \le {}_{di} < b$（因此 $0 \le m < b$）。

— 两个无穷大，+∞ 和 -∞。

— 两个 NaN，qNaN（安静）和 sNaN（信号）。

这些是唯一的浮点数据表示。

在上述描述中，符号 m 是以科学形式表示的，弧度点紧跟在第一位数字之后。为了某些目的，将符号视为整数也很方便；在这种情况下，有限浮点数是这样描述的：

— 形式为 $(-1)^s \times b^q \times c$ 的有符号零和非零浮点数，其中

  — s 为 0 或 1。

  — q 是任意整数 $emin \le q + p - 1 \le emax$。

  — c 是一个数字，由一个数字字符串表示，其形式为
  $_{d0\ d1\ d2 \ldots dp-1}$ 其中 $_{di}$ 是一个整数位 $0 \le {}_{di} < b$（因此 c 是一个整数位 $0 \le c < b^p$）。

这种将符号视为整数 c 及其相应指数 q *的*观点，与科学形式的观点描述了完全相同的零和非零浮点数集（对于有限浮点数，e = q + p - 1，m = c × b）。(对于有限浮点数，$e = q + p - 1$，$m = c \times b^{1-p}$。)

最小的正浮点数是 b^{emin}，最大的*正浮点数*是 $b^{emax} \times (b - b^{1-p})$。幅度小于 b^{emin} 的格式的非零浮点数称为*次正常值，*因为它们的幅度介于零和最小正常值幅度之间。它们的有效数字总是少于 p。每个有限浮点数都是最小次正常量级 $b^{emin} \times b^{1-p}$ 的整数倍。

对于数值为 0 的浮点数，符号 s 提供了额外的信息。虽然所有格式都有不同的 +0 和 -0 表示法，但 0 的符号在某些情况下（如除以 0）是重要的，而在其他情况下则不重要（见 6.3）。二进制交换格式对 +0 和 -0 只有一种表示法，但十进制格式有多种表示法。在本标准中，当符号不重要时，0 和 ∞ 不带符号。

表 3.2- 定义基本格式浮点数的参数

	二进制格式（*b=2*）			**十进制格式**（*b=10*）	
规范	**二进制 32**	**二进制 64**	**二进制 128**	**小数64**	**小数 128**
p, 位数	24	53	113	16	34
最大值	+127	+1023	+16383	+384	+6144

3.4 二进制交换格式编码

每个浮点数在 一种二进制交换格式中只有一种编码。为了使编码具有唯一性，根据 3.3 中的参数，通过减小 e 来最大化符号 m 的值，直到 $e = emin$ 或 $m \geq 1$。完成这一过程后，如果 $e = emin$ 且 $0 < m < 1$，则浮点数为次正态数。次正常数（和零）使用保留的偏指数值进行编码。

如图 3.1 所示，二进制交换格式中浮点数据的表示以 k 位唯一编码，在以下三个字段中排序：

a) 1 位符号 S

b) w 位偏置指数 $E = e + 偏置$

c) $(t = p - 1)$-位尾部有效数字字段数字字符串 $T = d_1 d_2 \ldots d_{p-1}$; 有效数字的前导位、$d_0$，隐含在偏置指数 E 中。

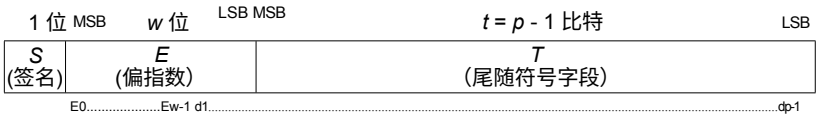

图 3.1- 二进制交换浮点格式

表 3.5 列出了二进制交换格式的 k、p、t、w 和*偏置*值（见 3.6）。编码的偏置指数 E 的

范围应包括

— 1 和 $2^w - 2$ 之间的每一个整数（含），以编码正常数

— 保留值 0，用于编码±0 和次正常数

— 保留值 $2^w - 1$，用于编码 ±∞ 和 NaN。

浮点数据的表示形式 r 和所表示的浮点数据的值 v 是根据组成字段推断出来的，具体如下：

a) 如果 $E = 2^w - 1$ 且 $T \neq 0$，那么 r 是 qNaN 或 sNaN，v 是 NaN，与 S 无关，那么 d_1 就只能区分 qNaN 和 sNaN（见 6.2.1）。

b) 如果 $E = 2^w - 1$，$T = 0$，那么 r 和 $v = (-1)^S \times (+\infty)$。

c) 如果 $1 \leq E \leq 2^w - 2$，那么 r 就是（S，（E-*偏置*），$(1 + 2^{1-p} \times T)$）；
相应浮点数的值为 $v = (-1)^S \times 2^{E-bias} \times (1 + 2^{1-p} \times T)$；因此，正常数的隐含前导符号位为 1。

d) 如果 $E = 0$，$T \neq 0$，那么 r 就是（S，$emin$，$(0 + 2^{1-p} \times T)$）；
相应浮点数的值为 $v = (-1)^S \times 2^{emin} \times (0 + 2^{1-p} \times T)$；因此，亚正态数的隐含前导符号位为 0。

e) 如果 $E = 0$，$T = 0$，则 r 为（S，$emin$，0），$v = (-1)^S \times (+0)$（有符号零，见 6.3）。在二进制交换格式中，所有数字和 NaN 编码都是规范编码。

注意 - 当 k 为 64 或 32 的倍数且 ≥ 128 时，对于这些编码，以下所有情况均为真（其中 round( ) 舍入到最接近的整数）：

$$
\begin{aligned}
k &= 1 + w + t &&= w + p &&= 32 \times \text{ceiling}((p + \text{round}(4 \times \log_2(p + \text{round}(4 \times \log_2(p))))- 13)- 13)/32 \\
w &= k - t - 1 &&= k - p &&= \text{round}(4 \times \log_2(k))- 13 \\
t &= k - w - 1 &&= p - 1 &&= k - \text{round}(4 \times \log_2(k)) + 12 \\
p &= k - w &&= t + 1 &&= k - \text{round}(4 \times \log_2(k)) + 13 \\
\text{最大值} &= \text{偏见} &&= 2^{(w-1)} - 1 \\
\text{伊明} &= 1 - \text{最大值} &&= 2 - 2^{(w-1)}.
\end{aligned}
$$

3.5 十进制交换格式编码

3.5.1 组群

与二进制浮点格式不同，在十进制浮 点格式中，一个数字可能有多种表示形式。浮点数所映射的表示集合称为浮点数的*同类数*；同类数的成员是同一浮点数的不同*表示*。例如，如果 c 是 10 的倍数，而 q 小于允许的最大值，那么（s, q, c）和（$s, q + 1, c / 10$）就是同一个浮点数的两种表示形式，属于同一个群组。

虽然在数值上相等，但可以通过十进制特定运算（见 5.3.2、5.5.2 和 5.7.3）来区分同组中的不同成员。不同浮点数的群组可能有不同的成员数。如果一个有限非零数的表示从其最重要的非零位到其最不重要的非零位有 n *个*小数位，那么该表示的群组最多有 $p - n + 1$ 个成员，其中 p 是格式中的精度位数。

例如，一位数的浮点数可能有多达 p *种*不同的表示方法，而没有尾零的 p *位数浮*点数只有一种表示方法。(如果一个 n *位*浮点数接近格式指数范围的极值，那么它的同群成员可能少于 $p - n + 1$）。一个零的同类数要大得多：+0 的同类数和 -0 的同类数一样，每个指数都有一个表示。

对于十进制运算，除了指定一个数值结果外，运算操作还根据 5.2 选择结果的同族成员。十进制应用可以利用同族传递的额外信息。

3.5.2 编码

十进制交换格式中浮点数据的表示以 k 位编码在以下三个字段中，其详细布局和规范（首选）编码说明如下。

a) 1 位符号 S。

b) 一个 $w + 5$ 位的组合字段 G 编码分类，如果编码数据是一个有限数字，还有指数 q 和四个符号位（其中 1 或 3 位是隐含的）。偏置指数 E 是一个 $w + 2$ 位的量 $q +$ *偏置*，其中偏置指数前两位的值合在一起是 0、1 或 2。

c) 一个 t 位尾部符号字段 T，包含 $J \times 10$ 位，包含大部分符号。当该字段与组合字段的前导符号位相结合时，该格式共编码 $p = 3 \times J + 1$ 个十进制数。

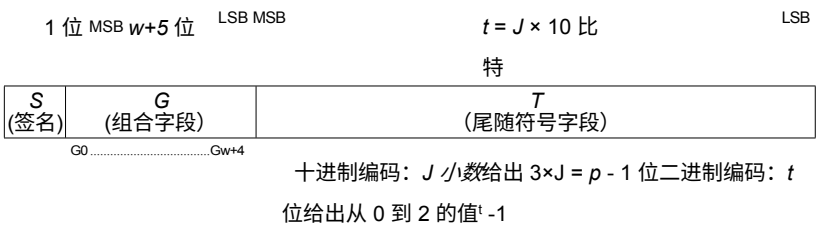

图 3.2--十进制交换浮点格式

表 3.6 列出了十进制交换格式的 *k*、*p*、*t*、*w* 和 *偏置*值（见 3.6）。

浮点数据的表示形式 *r* 和所表示的浮点数据的值 *v* 是根据组成字段推断出来的，具体如下：

a) 此外，如果 $_{G5}$ 为 1，则 *r* *为* sNaN；否则 *r* 为 qNaN。*G* 的其余位被忽略，*T* 是有效载荷，可用来区分各种 NaN。

   有效载荷的编码类似于下文所述的有限位数，*G* *的*处理方式是将所有比特均视为零。有效载荷对应于有限数字的符号位，被解释为一个

整数，最大值为 $10^{(3\times J)} - 1$，指数字段将被忽略（视为零）。如果 $G6$ 至 G_{w+4} 位为零，且有效载荷的编码为规范编码，则 NaN 为首选（规范）编码。

b) 如果 $G0$ 到 $G4$ 都是 11110，那么 r 和 $v = (-1)^S \times (+\infty)$。$G$ 中其余位的值，以及 T 则被忽略。无穷大的两种典型编码是 $G5$ 至 $G_{w+4} = 0$ 位和 $T = 0$ 位。

c) 对于有限数字，r 是（S，E - *偏置*，C），$v = (-1)^S \times 10^{(E-bias)} \times C$，其中 C 是组合字段 G 和尾部意义字段 T 的前导意义位或比特的连接，偏置指数 E 在组合字段中编码。这些字段内的编码取决于实现方法是使用十进制还是二进制编码来表示符号。

1) 如果使用*十进制*编码，那么指数的最小有效位为 $G5$ 至 G_{w+4}。偏置指数最有意义的两位和十进制数字串 $d0\,d1\ldots dp-1$ 由位 $G0$ 至 $G4$ 和 T 组成，如下所示：

i) 当 G 的最有效五位为 110xx 或 1110x 时，前导有效数字 $d0$ 为 $8 + G4$，数值为 8 或 9，前导偏指数位为 $2G2 + G3$，数值为 0、1 或 2。

ii) 当 G 的最有效五位为 0xxxxx 或 10xxx 时，前导有效数字 $d0$ 为 $4G2 + 2G3 + G4$，数值范围为 0 至 7，前导偏指数位为 $2G0 + G1$，数值范围为 0、1 或 2。因此，如果 T 为 0，G 的最有效五位是 00000、01000 或 10000，则 $v = (-1)^S \times (+0)$。

$p - 1 = 3 \times J$ 个十进制数字 $d1\ldots dp-1$ 由 T 编码，T 包含以密集十进制编码的 J 个小数。

如表 3.3 和表 3.4 所示，一个规范符号只有规范小数点。计算运算接受操作数中所有 1024 个可能的小数。除了静音计算运算（见 5.5）外，计算运算只产生 1000 个规范小数。

2) 另外，如果执行中使用*二进制*编码来表示有效数字，则

i) 如果 $G0$ 和 $G1$ 共为 00、01 或 10 之一，则偏置指数 E 由 $G0$ 至 G_{w+1} 组成，而示数则由 G_{w+2} 至编码结束位（包括 T）组成。

ii) 如果 $G0$ 和 $G1$ 共为 11，$G2$ 和 $G3$ 共为 00、01 或 10 之一，则偏置指数 E 由 $G2$ 至 G_{w+3} 组成，而示数则由 T 的 4 位前缀（$8 + G_{w+4}$）组成。

二进制编码符号的最大值与相应的十进制编码符号的最大值相同；即 $10^{(3\times J+1)} - 1$ （当 T 用作 NaN 的有效载荷时，则为 $10^{(3\times J)} - 1$ ）。如果数值超过最大值，则符号 c 是非规范值，c 的值为 0。

计算运算一般只产生规范符号，并始终接受操作数中的非规范符号。

注--当 k 是 32 的正倍数时，对于这些编码，以下所有情况均为真：

$$k = 1 + 5 + w + t = 32 \times \text{ceiling}((p + 2)/9)$$
$$w = k - t - 6 = k/16 + 4$$
$$t = k - w - 6 = 15 \times k/16 - 10$$
$$p = 3 \times t/10 + 1 = 9 \times k/32 - 2$$
$$\textit{最大值} = 3 \times 2^{(w-1)}$$
$$emin = 1 - emax$$
$$\textit{偏差} = emax + p - 2。$$

密集十进制解码表 3.3 将一个有 10 个位 $\mathbf{b_{(0)}}$ 至 $\mathbf{b_{(9)\text{的十进制}}}$数解码为 3 个十进制数 $\mathbf{d_{(1)}}$、$\mathbf{d_{(2)}}$、$\mathbf{d_{(3)}}$。第一列为二进制，"x "表示 "无所谓 "位。这样，所有 1024 种可能的 10 位模式都将被接受，并被映射成 1000 种可能的 3 位数组合，其中包含一些冗余。

表 3.3- 将 10 位密集十进制解码为 3 位十进制数

$\mathbf{B_{(6)}}$、$\mathbf{B_{(7)}}$、$\mathbf{B_{(8)}}$、$\mathbf{B_{(3)}}$、$\mathbf{B_{(4)}}$	$\mathbf{d_{(1)}}$	$\mathbf{d_{(2)}}$	$\mathbf{d_{(3)}}$
0 x x x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$4b_{(7)} + 2b_{(8)} + b_{(9)}$
1 0 0 x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$8 + b_{(9)}$
1 0 1 x x	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$8 + b_{(5)}$	$4b_{(3)} + 2b_{(4)} + b_{(9)}$
1 1 0 x x	$8 + b_{(2)}$	$4b_{(3)} + 2b_{(4)} + b_{(5)}$	$4b_{(0)} + 2b_{(1)} + b_{(9)}$
1 1 1 0 0	$8 + b_{(2)}$	$8 + b_{(5)}$	$4b_{(0)} + 2b_{(1)} + b_{(9)}$
1 1 1 0 1	$8 + b_{(2)}$	$4b_{(0)} + 2b_{(1)} + b_{(5)}$	$8 + b_{(9)}$
1 1 1 1 0	$4b_{(0)} + 2b_{(1)} + b_{(2)}$	$8 + b_{(5)}$	$8 + b_{(9)}$
1 1 1 1 1	$8 + b_{(2)}$	$8 + b_{(5)}$	$8 + b_{(9)}$

密集十进制编码表 3.4 将 3 个十进制数 $\mathbf{d_{(1)}}$、$\mathbf{d_{(2)}}$和 $\mathbf{d_{(3)}}$（每个十进制数有 4 个位，可用第二个下标 $\mathbf{d_{(1,0:3)}}$、$\mathbf{d_{(2,0:3)}}$和 $\mathbf{d_{(3,0:3)}}$表示，其中第 0 位最重要，第 3 位最不重要）编码成一个十进制数，有 10 个位 $\mathbf{b_{(0)}}$至 $\mathbf{b_{(9)}}$。大多数计算操作只产生表 3.4 中定义的 1000 个 10 位典型模式。

表 3.4- 将 3 位十进制数编码为 10 位密集十进制数

$\mathbf{d_{(1,0)}}$, $\mathbf{d_{(2,0)}}$, $\mathbf{d_{(3,0)}}$	$\mathbf{b_{(0)}}$、$\mathbf{b_{(1)}}$、$\mathbf{b_{(2)}}$	$\mathbf{B_{(3)}}$、$\mathbf{B_{(4)}}$、$\mathbf{B_{(5)}}$	$\mathbf{b_{(6)}}$	$\mathbf{B_{(7)}}$、$\mathbf{B_{(8)}}$、$\mathbf{B_{(9)}}$
0 0 0	$d_{(1,1:3)}$	$d_{(2,1:3)}$	0	$d_{(3,1:3)}$
0 0 1	$d_{(1,1:3)}$	$d_{(2,1:3)}$	1	0, 0, $d_{(3,3)}$
0 1 0	$d_{(1,1:3)}$	$d_{(3,1:2)}$, $d_{(2,3)}$	1	0, 1, $d_{(3,3)}$
0 1 1	$d_{(1,1:3)}$	1, 0, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
1 0 0	$d_{(3,1:2)}$, $d_{(1,3)}$	$d_{(2,1:3)}$	1	1, 0, $d_{(3,3)}$
1 0 1	$d_{(2,1:2)}$, $d_{(1,3)}$	0, 1, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
1 1 0	$d_{(3,1:2)}$, $d_{(1,3)}$	0, 0, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$
1 1 1	0, 0, $d_{(1,3)}$	1, 1, $d_{(2,3)}$	1	1, 1, $d_{(3,3)}$

计算操作的结果中不会产生 01x11x111x、10x11x111x 或 11x11x111x（其中 "x "表示 "无所谓 "位）形式的 24 个非规范模式。但是，如表 3.3 所列，这 24 个位模式确实映射到 0 至 999 范围内的值。NaN 尾随符号字段中的位模式会影响 NaN 的传播方式（见 6.2）。

3.6 交换格式参数

交换格式支持在实现之间交换 浮点数据。在每个弧度中，交换格式的精度和范围由其大小决定；因此，给定大小的浮点数据交换总是精确的，不可能出现溢出或下溢。

本标准定义了宽度为 16、32、64 和 128 位的二进制交换格式，以及一般情况下至少 128 位的任何 32 位倍数的二进制交换格式。十进制交换格式是为至少 32 位的任何 32 位倍数定义的。

对于二进制交换格式，表 3.5 列出了每种交换格式宽度的参数 p 和 $emax$；对于十进制交换格式，表 3.6 列出了每种交换格式宽度的参数 p 和 $emax$。交换格式的编码如 3.4 和 3.5.2 所述；表 3.5 和 3 .6 还列出了每种交换格式宽度的编码参数。

表 3.5- 二进制交换格式参数

参数	二进制16	二进制 32	二进制 64	二进制 128	二进制{k} ($k \geq 128$)
k，存储宽度（位	16	32	64	128	32 的倍数
p，精度（比特	11	24	53	113	k - round($4 \times \log_2(k)$) + 13
$emax$，最大指数 e	15	127	1023	16383	$2^{(k-p-1)} - 1$
编码参数					
偏差，$E - e$	15	127	1023	16383	*最大值*
符号位	1	1	1	1	1
w，指数字段宽度（比特	5	8	11	15	round($4 \times \log_2(k)$) - 13
t，尾部符号字段宽度（比特	10	23	52	112	$k - w - 1$
k，存储宽度（位	16	32	64	128	$1 + w + t$

在表 3.5 中，round( )舍入到最接近的整数。

例如，二进制 256 的 $p = 237$，$emax = 262143$。

表 3.6- 十进制交换格式参数

参数	小数32	小数64	小数 128	小数{k} ($k \geq 32$)
k，存储宽度（位	32	64	128	32 的倍数
p，精度（位数	7	16	34	$9 \times k / 32 - 2$
最大值	96	384	6144	$3 \times 2^{(k/16+3)}$
编码参数				
偏差，$E - q$	101	398	6176	*最大值* $+ p - 2$
符号位	1	1	1	1

w +5，组合字段宽度（比特	11	13	17	*k* / 16 + 9
t，尾部符号字段宽度（比特	20	50	110	15 × *k* / 16 - 10
k，存储宽度（位	32	64	128	1 + 5 + *w* + *t*

例如，decimal256 的 $p = 70$，$emax = 1572864$。

注--本标准将单个数据定义为概念性的第 4 级实体。在不同实施方案之间交换数据的 应用程序必须交流描述数据格式和布局的某些参数。除了影响所有数据交换的字节顺序等问题外，还必须考虑本标准允许的某些实施选项：

— 对于二进制格式，如何区分信号 NaN 和静态 NaN

— 对于十进制格式，无论使用的是二进制还是十进制编码。

本标准没有规定如何传递这些参数。

3.7 扩展和可扩展精度

建议使用扩展和可扩展精度格式，以扩展基本格式之外的运算精度。具体来说

— **扩展精度格式**是对支持的基本格式进行扩展的算术格式，具有更高的精度和更宽的范围。

— **可扩展精度格式**是一种算术格式，其精度和范围由用户控制定义。

这些格式的特征是参数 *b*、*p* 和 *emax*，它们可能与交换格式的参数相匹配，也应与交换格式的参数相匹配：

— 提供根据 3.2 和 3.3 中参数定义的浮点数据的所有表示形式

— 为该格式提供第 5 条定义的本标准的所有操作。

这些格式的编码应为固定宽度，并可与交换格式的编码相匹配。浮点数的每个表示形式都有唯一的规范编码，也可能有非规范编码。这些格式中的每个 NaN 都有一个有效载荷，可能编码诊断信息。每个 NaN 有效载荷都有一个规范编码，也可能有非规范编码。这些格式的编码的所有其他方面均由执行定义。

语言标准应为每个支持的弧度定义支持可扩展精度的机制。支持可扩展精度的语言标准应允许用户指定 *p* 和 *emax*。在这种情况下，当二进制格式中 $p \geq 237$ 位或十进制格式中 $p \geq 51$ 位时，语言标准应将 *emax* 定义为至少 $1000 \times p$。

语言标准或实现应支持扩展精度格式，以扩展该弧度所支持的最宽基本格式。表 3.7 规定了每种基本格式的扩展精度格式的最小精度和指数范围。

表 3.7- 浮点数的扩展格式参数

	相关的扩展格式：				
参数	**二进制 32**	**二进制 64**	**二进制 128**	**小数64**	**小数 128**
p 数字≥	32	64	128	22	40
emax ≥	1023	16383	65535	6144	24576

注 1 - 对于扩展格式，最小指数范围是下一个更宽基本格式（如果有的话）的最小指数范围，而最小精度介于给定基本格式和下一个更宽基本格式之间。

注 2 - 对于二进制浮点数据的交换，允许对至少 *p* *比特*的符号进行编码的最小格式的宽度 *k* （比特）由以下公式给出：

$k = 32 \times \text{ceiling}((p + \text{round}(4 \times \log_2(p + \text{round}(4 \times \log_2(p))) - 13)) - 13)/32)$，其中 round( )舍入到最接近的整数，且 $p \geq 113$；对于更小的 *p* 值，请参见表 3.5。

对于十进制浮点数据的交换，允许对至少 *p* 位数的符号进行编码的最小格式的位宽 *k* *由以下公式*给

出：

$k = 32 \times \text{ceiling}((p + 2)/9)$，其中 $p \geq 1$。

在这两种情况下，所选格式的精度可能更高（见 3.4 和 3.5.2）。

注 3 - 对于二进制格式，精度 p 至少应为 3，因为某些数字特性在精度较低时不成立。

同样，*emax* 至少应为 2，以支持 9.2 中列出的操作。

4. 属性和四舍五入

4.1 属性说明

属性在逻辑上与 与 h 程序块相关联，以修改其数值和异常语义。用户可以为属性参数指定一个常量值。

某些属性对本标准的大多数个别操作具有隐含参数的作用；语言标准应规定

— 舍入方向属性（见 4.3），并应指

定

— 备用异常处理属性（见第 8 条）。

其他属性改变了语言表达到本标准操作的映射；允许不止一种映射的语言标准应提供支持：

— 首选宽度属性（见 10.3）

— 改变价值的优化属性（见 10.4）

— 再现性属性（见第 11 条）。

对于属性规范，执行应提供语言定义的方法（如编译器指令），为块中的操作指定属性参数的常量值；属性值的范围是与其相关的块。语言标准应规定属性的默认值和每个特定值的常量。

4.2 属性的动态模式

本标准中的属性应支持 4.1 的常量规范。特别是为了支持调试，语言标准还应支持属性的动态模式规范。

通过动态模式规范，用户可指定属性参数取动态模式变量的值，而该变量的值在程序执行前可能是未知的。本标准没有规定常量属性或动态模式的基本实现机制。

对于动态模式规范，实现应提供语言定义的方法来指定属性参数在块中动态模式规范范围内的 操作中承担动态模式变量的值。实现将动态模式变量初始化为动态模式的默认值。在语言定义的（动态）范围内，动态模式变量值的更改由用户通过 9.3.1 和 9.3.2 中的操作控制。

动态模式变量的以下方面是由语言定义的；语言标准可以明确地将这些定义交由实现来处理：

— 静态属性规范和动态模式分配的优先级。

— 在异步事件中（如在另一个线程或信号处理程序中）更改动态模式变量值的效果。

— 是否可以通过调试器等非编程方式确定动态模式变量的值。

注--属性的常量值可通过动态模式规范来指定，并通过常量值的适当范围来满足 4.1 的要求。

4.3 舍入方向属性

四舍五入取一个被认为是无限精确的数字（ nfinitely precise），并在必要时对其进行修改，使其符合目的地的格式，同时在适当的时候发出不精确异常、下溢或溢出信号（参见第 7 条）。除另有说明外，每次操作的执行都应如同首先产生一个无限精确且范围无界的中间结果，然后根据本条款中的一个属性对该结果进行四舍五入。

除另有说明外，舍入方向属性影响所有可能不精确的计算操作。非精确数值浮点运算结果的符号总是与未四舍五入结果的符号相同。

舍入方向属性会影响精确零和的符号（见 6.3），也会影响溢出（见 7.4）和欠流（见 7.5）信号的阈值。

同时支持十进制和二进制格式的实现应分别提供二进制和十进制的舍入方向属性、二进制舍入方向和十进制舍入方向。以浮点格式返回结果的操作应使用与结果弧度相关的舍入方向属性。从浮点格式操作数转换为整数格式结果或外部字符序列（见 5.8 和 5.12）的操作，应使用与操作数弧度相关的舍入方向属性。

NaN 不四舍五入（但请参阅 6.2.3）。

4.3.1 将方向属性舍入到最接近的值

在下面两个四舍五入属性中，幅度至少为 $b^{emax} \times (b - \frac{1}{2} b^{1-p})$ 的无限精确结果应四舍五入为 ∞，符号不变；这里的 *emax* 和 *p* 由目标格式决定（见 3.3）。与

- roundTiesToEven，应提供最接近无限精确结果的浮点数；如果两个最接近无限精确结果的浮点数同样接近，则应提供最小有效数字为偶数的浮点数；如果不可能，则应提供较大的浮点数（这可能发生在一位数精度的情况下，例如，在将 9.5 舍入到一位数时，9 和 1×10^1 的有效数字均为奇数）。
- roundTiesToAway，则应交付最接近无限精确结果的浮点数；如果无法表示无限精确结果的两个最近浮点数同样接近，则应交付幅度较大的浮点数。

4.3.2 定向四舍五入属性

还定义了另外三个用户可选的舍入方向属性，即定向舍入属性 roundTowardPositive、roundTowardNegative 和 roundTowardZero。与

- roundTowardPositive，结果应是最接近且不小于无限精确结果的格式浮点数（可能是 +∞）。
- roundTowardNegative，结果应是格式浮点数（可能是-∞），最接近且不大于无限精确的结果
- roundTowardZero，其结果应为最接近无限精确结果的格式浮点数，且大小不大于无限精确结果。

4.3.3 四舍五入属性要求

本标准的实施应提供 roundTiesToEven 和三个定向舍入属性。本标准的十进制格式实施应提供 roundTiesToAway 作为用户可选的舍入方向属性。二进制格式的执行不需要舍入属性 roundTiesToAway。

对于二进制格式的结果，roundTiesToEven 四舍五入方向属性应为默认的四舍五入方向属性。十进制格式结果的默认舍入方向属性由语言定义，但应是 roundTiesToEven。

5. 业务

5.1 概述

除下文所述外，所有符合本标准 的实施均应为所有支持的算术格式提供本条款所列的运算。除非另有规定，本标准规定的返回数值结果的每种计算运算，在执行时应首先产生一个无限精度和无限制范围的中间结果，然后对中间结果进行四舍五入（如有必要），以适应目标格式（见第 4 条和第 7 条）。第 6 条扩展了以下规范，以涵盖 ±0、±∞ 和 NaN。第 7 条描述了默认异常处理。

在本标准中，操作被写成命名函数；在特定的编程环境中，它们可能用操作符表示，或用特定格式的函数族表示，或用操作或函数表示，其名称可能与本标准中的不同。

根据其产生的结果和异常情况，操作大致可分为四类：

- 一般计算操作产生浮点或整数结果，根据第 4 条对所有结果进行四舍五入，并可能发出第 7 条的浮点例外信号。
- 静态计算操作产生浮点结果，不会发出浮点异常信号。
- 信令计算运算不产生浮点结果，但可能发出浮点异常信号；比较运算属于信令计算运算。
- 非计算操作不会产生浮点结果，也不会发出浮点异常信号。

前三类操作统称为 "计算操作"。

根据结果格式和操作数格式之间的关系，操作也可分为两种：

- 同质运算，其中浮点操作数和浮点结果的格式都相同
- *formatOf*操作，表示结果的格式，与操作数的格式无关。

语言标准可能允许在表达式中使用其他类型的操作和操作组合。语言标准通过其表达式评估规则规定了何时以及如何将这些操作和表达式映射到本标准的操作中。操作（重编码操作除外）不必接受不同编码的操作数或产生不同编码的结果。

除 5.5 另有规定外，操作结果应是规范的。

在后面的操作说明中，操作数和结果格式用"..... "表示：

— 表示同质浮点操作数格式的*源代码*

— *源 1*、*源 2*、*源 3* 表示非同质浮点操作数格式

— *int* 表示整数操作数格式

— 布尔值，表示*假值*或*真值*（例如 0 或 1）

— *枚举*来表示一小部分枚举值中的一个值

— *sourceFormat* 表示*与源*格式相同的目标格式

— *integralFormat* 表示包含积分值的格式

— *logBFormat* 表示 *logB* 运算的目标*积分格式*和 *scaleB* 运算的标度指数操作数

— *decimalCharacterSequence* 表示十进制字符序列

— *hexCharacterSequence* 表示十六进制符号字符序列

— *conversionSpecification* 表示与语言相关的转换规范

— *十进制*表示支持的十进制浮点格式

— *decimalEncoding* 表示以十进制编码的十进制浮点格式

— *binaryEncoding* 表示以二进制编码的十进制浮点格式

— *exceptionGroup* 用于将一组异常表示为一组*布尔值*

— *标记*来表示一组状态标记

— *binaryRoundingDirection* 表示二进制的四舍五入方向。

— *decimalRoundingDirection* 表示小数的四舍五入方向

— *modeGroup* 表示可动态指定的模式

— *void* 表示操作没有显式操作数或没有显式结果；操作数或结果可能是隐式的。

formatOf 表示操作的名称指定了浮点目标*格式*，该格式可能与浮点操作数的格式不同。这些操作的 *formatOf* 版本适用于所有支持的算术格式。

intFormatOf 表示操作名称指定了整数目标格式。

在后面的操作说明中，语言定义了操作数和结果所对应的类型。具有有符号和无符号整数类型的语言应同时支持有符号和无符号 *int* 以及 *intFormatOf* 操作数和结果。

5.2 小数指数计算

如 3.5 所述，一个浮点数 可能有多种十进制格式。因此，十进制运算不仅涉及计算适当的数值结果，还涉及选择浮点数同族中的适当成员。

除**量化**操作外，浮点运算结果的值（及其同位数）由操作和操作数的值决定；它从不依赖于操作数的表示或编码。

如下所述，浮点运算结果的特定表示形式的选择取决于操作数的表示形式，但不受操作数编码的影响。

除另有说明外，对于所有计算运算，如果结果不精确，则使用可能指数最小的同族成员，以获得最大有效位数。如果结果是精确的，则根据该运算结果的首选指数（输入指数的函数）选择同族成员。因此，对于有限的 x，根据零的表示方法，$0 + x$ 可能会导致 x 的同族中出现不同的成员。如果运算结果的队列中没有首选指数的成员，则使用最接近首选指数的成员。

对于**量化**和 **roundToIntegralExact**，无论结果是否精确，有限结果都具有首选指数。

在接下来的运算描述中，$\mathrm{Q}(x)$ 是有限浮点数 x 表示的指数 q。

5.3 同构通用计算操作

5.3.1 一般业务

实现应为所有支持的算术格式提供以下同构通用计算运算：

— *sourceFormat* **roundToIntegralTowards ToEven**(*source*) *sourceFormat* **roundToIntegralTowards ToAway**(*source*)
sourceFormat **roundToIntegralTowardZero**(*source*)
sourceFormat **roundToIntegralTowardPositive**(*source)*
sourceFormat **roundToIntegralTowardNegative**(*source*)

详见 5.9。

首选指数为 max（Q（*源*），0）。

— *sourceFormat* **roundToIntegralExact**(*source*)

详见 5.9。

首选的指数是 max(Q(*源*),0)，即使在发出不精确异常信号时也是如此。

— *sourceFormat* **nextUp**(*source*)
sourceFormat **nextDown**(*source*)

nextUp(*x*) 是 *x* 格式中比 *x* *大*的最小浮点数。如果 *x* 是 *x* 格式中大小最小的负数，**nextUp**(*x) 是* -0。**nextUp**(±0) 是 *x* 格式中大小最小的正数，**nextUp**(+∞) 是 +∞，**nextUp**(-∞) 是大小最大的有限负数。当 *x* 为 NaN 时，结果按照 6.2 计算。

首选指数尽可能小。

nextDown(*x*) 是 **-nextUp**(-*x*).

— *sourceFormat* **remainder**(*源, 源*)

当 $y \neq 0$ 时，余数 r = **remainder**(x, y) 的数学关系式 $r = x - y \times n$ 定义了有限 x 和 y 的余数，而不管舍入方向属性如何，其中 n 是最接近精确数 x/y 的整数；只要 n - x/y | = ½，那么 n 就是偶数。因此，余数总是精确的。如果 $r = 0$，它的符号就是 x 的符号。

首选指数为 min（Q(x), Q(y)）。

注--2008 版标准中的 minNum 和 maxNum 操作已被 9.6 建议的操作所取代。

5.3.2 小数运算

支持 十进制格式的实现应为所有支持的十进制算术格式提供以下同质通用计算操作：

- *sourceFormat* **quantize**（*源格式，源代码*）

  对于格式相同的有限十进制操作数 *x* 和 *y*，**quantize**(*x*, *y*) 是一个格式相同的浮点数，如果可能，它的数值与 *x* 相同，量子与 *y* 相同。如果指数被增大，则可能会根据适用的舍入方向属性进行舍入：结果是一个不同的浮点表示形式，如果结果的数值与 *x* 不相同，则会发出不精确异常信号；如果指数被减小，且结果的示值将超过 *p* 位，则会发出无效操作异常信号，结果为 NaN。如果一个或两个操作数都是 NaN，则遵循 6.2 中的规则。否则，如果只有一个操作数是无限的，则发出无效操作异常信号，结果为 NaN。如果两个操作数都是无限的，那么结果就是带有 *x 符号的*规范 ∞。

  首选指数为 Q(*y*)。

支持十进制格式的实现应为所有支持的十进制算术格式提供以下同质通用计算操作：

- *sourceFormat* **quantum**（*源格式*）

  如果 *x* 是有限数，则操作 **quantum**(*x*) 就是由（0，*q*，1）表示的数，其中 *q 是 x* 的指数。

  首选指数为 Q(*x*)。

5.3.3 *logBFormat* 操作

实现应为所有可用于算术运算的支持浮点格式提供 以下通用计算运算。

对于每种支持的算术格式，语言都定义了相关的 *logBFormat*，以包含 **logB**(*x*) 的积分值。*logBFormat* 的范围应足以包含 ±2 × (*emax* + *p*) （含）之间的所有整数，其中包括用于在最大和最小有限数值之间缩放的比例因子。

— *sourceFormat* **scaleB**(*源*、*logBFormat*)

  对于整数值 *N*，**scaleB**(*x*, *N*) 是 $x \times b^N$。计算结果如同形成精确乘积，然后根据适用的舍入方向属性舍入到目标格式。当 *logBFormat* 为浮点格式时，第二个操作数为非整数时，**scaleB** 的行为由语言定义。对于 *N* 的非零值，**scaleB**(±0, *N*) 返回 ±0，**scaleB**(±∞, N) 返回 ±∞。对于 *N* 的零值，**scaleB**(*x*, *N*) 返回 *x*。

  首选指数为 Q(*x*) + *N*。

— *logBFormat* **logB**(*源*)

  logB(*x*) 是 *x* 的指数 *e*，是一个带符号的积分值，它的确定就好像 *x 是*用无限范围和最小指数来表示的。因此，当 *x* 为正且有限时，$1 \le$ **scaleB**(*x*, **-logB**(*x*)) $< b$，**logB**(1) 为 +0。

  当 *logBFormat* 是浮点格式时，**logB**(NaN) 是 NaN，**logB**(∞) 是 +∞，而 **logB**(0) 是 -∞，并发出 divideByZero 异常信号。如果 *logBFormat* 是整数格式，则 **logB**(NaN)、**logB**(∞) 和 **logB**(0) 返回语言定义的范围之外的值

$\pm 2 \times (emax + p - 1)$ 并发出无效操作异常信号。首选指数为 0

。

注 -对于正有限 x，**logB**(x) 的二进制值为 $\text{floor}(\log_2(x))$，十进制值为 $\text{floor}(\log_{10}(x))$。

5.4 一般计算操作格式

5.4.1 算术运算

实现应为所有支持的算术格式的目标运算符，以及每个目标运算符格式的操作数，提供 以下生成算术运算的格式 。

— **formatOf-addition**(*source1, source2*)

操作 **addition**(*x*, *y*) 计算 *x* + *y*。

首选指数为 min（Q(*x*), Q(*y*)）。

— **formatOf-subtraction**(*source1, source2*)

操作 **subtraction**(*x*, *y*) 计算 *x* - *y*。

首选指数为 min（Q(*x*), Q(*y*)）。

— **formatOf-multiplication**(*source1, source2*)

乘法运算（*x*, *y*）计算 *x* × *y*。

首选指数为 Q(*x*) + Q(*y*)。

— **formatOf-division**(*source1, source2*)

除法运算（*x*, *y*）计算 *x* / *y*。

首选指数为 Q(*x*) - Q(*y*)。

— **formatOf-squareRoot**(*source1*)
操作**平方根**（*x*）计算 √ *x*。对于所有操作数 ≥ 0，它的符号都是正的，除了 **squareRoot**(-0) 应为 -0。

首选指数是 floor(Q(*x*)/2)。

— **formatOf-fusedMultiplyAdd**(*source1, source2, source3*))

操作 **fusedMultiplyAdd**(*x*, *y*, *z*) 计算 (*x* × *y*) + *z* 就像计算无限制范围和精度一样，只对目标格式进行一次四舍五入。只有对精确值 (*x* × *y*) + *z* 进行**四舍五入**时，才会出现**下溢**、溢出或不精确异常（见第 7 条）。

首选指数为 min(Q(*x*)+Q(*y*),Q(*z*))。

— *formatOf-convertFromInt*(*int*)

所有支持的有符号和无符号整数格式均可转换为所有支持的算术格式。只要整数格式和浮点格式都能表示整数值，整数值就能准确地从整数格式转换为浮点格式。如果转换后的值不能在目标格式中精确表示，则会根据适用的舍入方向属性确定结果，并出现第 7 条规定的不精确或浮点溢出异常，就像算术运算一样。保留整数零的符号。没有符号的整数零将转换为 +0。

首选指数为 0。

实现应为所有支持的整数格式的目的地和所有支持的算术格式的操作数提供以下 *intFormatOf* 通用计算操作。

— **intFormatOf-convertToIntegerTiesToEven**(*source*)
intFormatOf-convertToIntegerTowardZero(*source*)
intFormatOf-convertToIntegerTowardPositive(source)
intFormatOf-convertToIntegerTowardNegative(*source*)
intFormatOf-convertToIntegerTiesToAway(*source*)

详见 5.8。

— **intFormatOf-convertToIntegerExactTiesToEven**(*source*)
intFormatOf-convertToIntegerExactTowardZero(*source*)
intFormatOf-convertToIntegerExactTowardPositive(source)
intFormatOf-convertToIntegerExactTowardNegative(*source*)
intFormatOf-convertToIntegerExactTiesToAway(source)。

详见 5.8。

注--当本子条款中的某些操作和 5.4.2 **中的 convertFormat** 操作序列能产生正确的数值、量子和异常结果时，实施可以将其作为子条款 5.4 中一个或多个操作子集的序列来提供。

5.4.2 浮点格式和十进制字符序列的转换操作

实现应提供以下 *formatOf* 转换操作：从所有支持的浮点 ng- point 格式到所有支持的浮点格式，以及十进制字符序列之间的转换。某些格式转换操作会产生与操作数不同弧度的结果。

- formatOf-convertFormat(*source*)

如果转换为不同弧度的格式，或转换为同一弧度中较小精度的格式，结果应按第 4 条的规定四舍五入。转换到弧度相同但精度和范围更宽的格式时，结果总是精确的。

对于从二进制格式到十进制格式的非精确转换，首选指数是可能的最小值。对于二进制到十进制的精确转换，首选指数为 0。

对于十进制格式之间的转换，首选指数是 Q（*源*）。

注意--源格式和目标格式完全相同的 **formatOf-convertFormat** 操作是一种规范化操作，与**复制**操作（5.5.1）不同，该操作会因操作数为 NaN 而发出无效操作异常信号。

— *formatOf-convertFromDecimalCharacter*(*decimalCharacterSequence*)

详见 5.12。首选指数是 Q(*十进制字符序列*)，即十进制字符序列的示值中最后一位数字的指数值 *q*。

— *decimalCharacterSequence* **convertToDecimalCharacter**（*源代码*、*转换规范*）

详见 5.12。*转换规范*（*conversionSpecification*）指定了*转换*结果的精度和格式。
decimalCharacterSequence result.

5.4.3 二进制格式的转换操作

实现应提供与所有支持的二进制浮点格式之间的以下 *formatOf* 转换操作。

— *formatOf-convertFromHexCharacter*(*hexCharacterSequence*) 详

情请参见 5.12。

— *十六进制字符序列* **convertToHexCharacter**（*源代码*、*转换规范*）
详见 5.12。*转换规范*（*conversionSpecification*）指定了 *转换*结果的精度和格式。
*十六进制字符序列*结果。

5.5 安静的计算操作

5.5.1 符号位操作

实现应为所有支持的算术交换格式提供 omogeneous quiet-computational sign bit 运算；这些运算只影响符号位。这些运算对浮点数和 NaN 都一视同仁，无一例外。这些操作可能会传播非规范编码。

- *sourceFormat* **copy**(*源代码*)
  sourceFormat **negate**(*源代码*
) *sourceFormat* **abs**(*源代码*)

  copy(*x*) 将浮点操作数 *x* 复制到相同格式的目标操作数，符号位不变。

  negate(*x*) 与 **subtraction**(0, *x*) 不同（参见 6.3）。

  abs(*x*) 将浮点操作数 *x* 复制到相同格式的目标，并将符号位设置为 0（正）。

- *源格式* **copySign**(*源、源*)

  copySign(*x*,*y*)将浮点操作数 *x* 复制到与 *x* 格式相同的目标操作数，但要加上 *y* 的符号位。

本标准不指定非交换格式的编码。

对于所有受支持的非交换格式，实现应提供**复制**、**否定**、**abs** 和
复制符号操作，与上述表示层的符号位操作相匹配（见 3.2 和 3.3）：对于数字 *x*、

copy(*x*) 表示与 *x* *相同*，但符号不变 **negate**(*x*) 表示与 x 相同，但符

号相反 **abs**(*x*) 表示与 *x* 相同，但符号为 0（正）。

copySign(*x*,*y*)的表示方法与 *x* 相同，但如果 *y* 是数值，则使用 *y* 的符号，如果 y 是 NaN，则使用未指定的符号。

如果 *x* 是 NaN，则运算结果的表示形式与 *x* 相同（qNaN 或 sNaN）。

如果编码允许，非交换格式的操作应遵循交换格式的符号位操作规范。

5.5.2 十进制重新编码操作

对于 mat 支持的每种十进制交换，实现应提供以下操作，以便在十进制格式和该格式的两种编码之间进行转换（见 3.5.2）。这些操作可使独立于十进制格式实现编码的可移植程序访问以任一编码表示的数据。如果结果格式的符号编码与操作数格式的符号编码相同，那么这些操作应为**复制**操作。这些操作可能会传播非规范编码；如果结果格式的符号编码与操作数格式的符号编码不同，则结果应为规范编码。

— *decimalEncoding* **encodeDecimal**(*decimal*)
使用十进制编码对操作数的值进行编码。

— **decodeDecimal**(*decimalEncoding*) 对十进制编码的操作数进行解码。

— *二进制编码* **encodeBinary**（*十进制*）
使用二进制编码对操作数的值进行编码。

— *decimal* **decodeBinary(***binaryEncoding*)
对二进制编码的操作数进行解码。

5.6 信号-计算操作

5.6.1 比较

对于算术格式中所有支持的相同弧度的浮 点 运 算 数， 实现应 ，提供以下 C 比较运算：

- *布尔值* **compareQuietEqual**(*source1, source2*) *布尔值*

  compareQuietNotEqual(source1, source2) 布尔值

  compareSignalingEqual(source1, *source2*) *布尔值*
  compareSignalingGreater(*source1, source2*)
  布尔值 **compareSignalingGreaterEqual**（*信号源 1*、*信号源 2 布尔值* **compareSignalingLess**（信号 *源 1*、*信号源 2*）

  布尔值 **compareSignalingLessEqual**(*source1, source2*) *布尔值* **compareSignalingNotEqual**(source1, *source2*) *布尔值*

  compareSignalingNotGreater(source1, *source2*) *布尔值*

  compareSignalingLessUnordered(source1, *source2*) *布尔值*
  compareSignalingNotLess(*source1, source2*)
  布尔值 **compareSignalingGreaterUnordered**(*source1, source2*) *布尔值* **compareQuietGreater**(*source1, source2*)

  布尔值 **compareQuietGreaterEqual**（*源 1，源 2 布尔值*

  compareQuietLess（源 *1，源 2*）

  布尔值 **compareQuietLessEqual**(*source1, source2*) *布尔值*

  compareQuietUnordered(source1, *source2*) *布尔值*

  compareQuietNotGreater(source1, *source2*) *布尔值*

  compareQuietLessUnordered(*source1, source2*) *布尔值*
  compareQuietNotLess(*source1, source2*)
  布尔值 **compareQuietGreaterUnordered**(*source1, source2*)

  布尔值 **compareQuietOrdered(***source1, source2*)。

详见 5.11。

5.7 非计算操作

5.7.1 一致性谓词

实现应提供以下非计算操作，只有当且仅当所述条件为真时才为真：

— *boolean* **is754version1985**(*void*)

is754version1985( ) 为真，当且仅当该编程环境符合 1985 版标准。

— *boolean* **is754version2008**(*void*)

is754version2008( ) 为真，前提是且仅当该编程环境符合 2008 版标准。

— *boolean* **is754version2019**(*void*)

is754version2019( ) 为真，当且仅当该编程环境符合该标准。

在翻译时（如适用），如果可以确定这些谓词的值，则实现时应提供这些谓词。

5.7.2 一般业务

实现应为所有支持的算术格式提供以下非运算操作，并应为所有支持的交换格式提供这些操作。即使是信号 NaN，它们也绝不例外。

— *枚举***类**（*源*）

class(*x*)表示 *x* 属于以下十个类别中的哪一类： 信号
NaN
负无穷负正常负次
正常负零正零正次
正常正正常正无穷
。

— *布尔* **isSignMinus**(*source*)

当且仅当 *x* *为*负号时，**isSignMinus**(*x*) 为真。

— *布尔***值 isNormal**(*source*)

isNormal(*x*) 为真，当且仅当 *x* 为正态（非零、次正态、无限或 NaN）。

— *布尔值* **isFinite**(*source*)

isFinite(*x*) 为真，当且仅当 *x* 为零、亚正态分布或正态分布（非无限或 NaN）。

— *布尔* **isZero**(*source*)

当且仅当 *x* 为 ±0 时，**isZero**(*x*) 为真。

— *布尔***值 isSubnormal**(*source*)

isSubnormal(*x*) 为真，当且仅当 *x* 是次正态分布时。

— *布尔值* **isInfinite**(*source*)

isInfinite(*x*) 为真，当且仅当 *x* 是无限的。

— *布尔***值 isNaN**(*source*)

isNaN(*x*) 为真，当且仅当 *x* 是 NaN。

— *布尔* **isSignaling**(*source*)

isSignaling(*x*) 为真，当且仅当 *x* 是一个信号 NaN。

— *布尔* **isCanonical**(*source*)

isCanonical(*x*) 为真，当且仅当 *x* 是正则表达式的有限个数、无穷小或 NaN。对于非交换格式的格式，实现应以适合这些格式的方式扩展 **isCanonical**(*x*)，这些格式可能有，也可能没有非规范的有限个数、无穷个数或 NaN。

— *枚举***radix**(*源*)
radix(*x*) 是 *x* 格式的弧度 *b*，即 2 或 10。

— *布尔值* **totalOrder**（*来源，来源*）

totalOrder(*x, y*) 的定义见 5.10。

— *布尔值* **totalOrderMag**（*来源，来源*）

totalOrderMag(*x, y*) 是 **totalOrder**(**abs**(*x*), **abs**(*y*))。

在翻译时（如适用），如果可以确定这些谓词的值，则实现时应提供这些谓词。

5.7.3 十进制操作

支持 十进制格式的实现应为所有支持的十进制算术格式提供以下非运算操作：

- *布尔***相同量子**（*源、源*）

  对于格式相同的十进制数值操作数 *x* 和 *y*，如果 *x* 和 *y* 的指数相同，即 Q(x) = Q(*y*)，则 **sameQuantum**(*x*, *y*) 为真，否则为假。**sameQuantum**(NaN, NaN) 和 **sameQuantum**(∞, ∞) 为真；如果正好有一个操作数是无穷大或正好有一个操作数是 NaN，则 **sameQuantum** 为假。

5.7.4 对标志子集的操作

实现应提供以下 ng 非运 算 操作，这些操作可共同作用于多个状态标志；这些操作不发出异常信号：

— *void* **lowerFlags**(*exceptionGroup*)
降低（清除）与*异常组中*指定的异常相对应的标记
操作数，它可以代表任何异常子集。

— *void* **raiseFlags**(*exceptionGroup*)

唤起（设置）与 *exceptionGroup* 操作符中指定的异常相对应的标志，*exceptionGroup* 可以代表异常的任何子集。

— *boolean* **testFlags**(*exceptionGroup*)
查询是否有与*异常组中*指定的异常相对应的标志
操作数，可以代表任何异常子集。

— *boolean* **testSavedFlags**(*flags, exceptionGroup*)

查询 *flags 操作数*中与 *exceptionGroup* 操作数中指定的异常（可代表异常的任意子集）相对应的任何标志是否被触发。

— *void* **restoreFlags**(*flags*, *exceptionGroup*)

将与 *exceptionGroup* 操作数中指定的异常（可以是异常的任意子集）相对应的标志恢复到 *flags* 操作数中的状态。

— *flags* **saveAllFlags**(*void*)

返回所有状态标志的状态表示。

saveAllFlags 操作的返回值将作为 **restoreFlags** 或
本标准不要求支持任何其他用途。

5.8 从浮点格式转换为整数格式的详细信息

实现应提供从所有支持的算术 ormats 到所有支持的有符号和无符号整数格式的转换操作。只要数值可以用浮点格式和整数格式表示，整数值就会从浮点格式精确转换为整数格式。

转换为整数时，应按第 4 条的规定进行四舍五入；操作名称表示四舍五入的方向。

当 NaN 或无限操作数无法用目标格式表示，且无法以其他方式表示时，应发出无效操作异常信号。当数值操作数将转换为超出目标格式范围的整数时，如果无法以其他方式说明这种情况，则应发出无效操作异常信号。

当转换操作的结果值与其操作数值不同，但可以用目的格式表示时，下面规定了一些转换操作发出不精确异常信号，另一些转换操作则不发出不精确异常信号。

如果语言标准允许隐式转换或表达式涉及混合类型，则应要求使用下面的非精确信号转换操作来实现这些转换或表达式。

将浮点数转换为特定有符号或无符号整数格式而不发出不精确异常信号的操作是

— **intFormatOf-convertToIntegerTiesToEven**(*source*)
convertToIntegerTiesToEven(*x*) 将 *x* 四舍五入为最接近的整数值，半途舍入为偶数。

— **intFormatOf-convertToIntegerTowardZero**(*source*) *(源代码)*
convertToIntegerTowardZero(*x*) 将 *x* 四舍五入为整数值，取整为零。

— **intFormatOf-convertToIntegerTowardPositive**(*source*)
convertToIntegerTowardPositive(*x*) 将 *x* 舍入为指向正无穷的积分值。

— **intFormatOf-convertToIntegerTowardNegative**(*source*) *(源代码)*
convertToIntegerTowardNegative(*x*) 将 *x* 舍入为指向负无穷的整数值。

— **intFormatOf-convertToIntegerTiesToAway**(*source*)
convertToIntegerTiesToAway(*x*) 将 *x* 四舍五入为最接近的整数值，半途舍入为零。

将浮点数转换为特定有符号或无符号整数格式的操作（如果不精确，则发出信号）有

— **intFormatOf-convertToIntegerExactTiesToEven**(*source*)
convertToIntegerExactTiesToEven(*x*) 将 *x 四舍五入*为最接近的整数值，半途舍入为偶数。

— **intFormatOf-convertToIntegerExactTowardZero**(*source*) *(源代码)*
convertToIntegerExactTowardZero(*x*) 将 *x* 四舍五入为趋近于 0 的整数值。

— **intFormatOf-convertToIntegerExactTowardPositive**(*source*)
convertToIntegerExactTowardPositive(*x*) 将 *x* 舍入为指向正无穷的积分值。

— **intFormatOf-convertToIntegerExactTowardNegative**(*source*)
convertToIntegerExactTowardNegative(*x*) 将 *x* 舍入为指向负无穷的整数值。

— **intFormatOf-convertToIntegerExactTiesToAway**(*source*)
convertToIntegerExactTiesToAway(*x*) 将 *x 四舍五入*为最接近的整数值，半途舍入为零。

5.9 将浮点数据舍入为积分值的操作详情

将浮点数舍入积分浮点数的几种操作 i n 格式相同。

四舍五入与第 4 条规定的四舍五入类似，但四舍五入只从格式中的整数浮点数中选择。这些操作将零操作数转换为相同符号的零结果，将无限操作数转换为相同符号的无限结果。

对于以下操作，舍入方向由操作名称指定，而不取决于舍入方向属性。除了 NaN 输入信号外，这些操作不会发出任何异常信号。

— *源格式* **roundToIntegralTiesToEven**(*source*)

roundToIntegralTiesToEven(*x*) 将 *x* 四舍五入为最接近的积分值，半途舍入为偶数。

— *sourceFormat* **roundToIntegralTowardZero**(*source*)

roundToIntegralTowardZero(*x*) 将 *x* 取整为零。

— *sourceFormat* **roundToIntegralTowardPositive**(*source*)

roundToIntegralTowardPositive(*x*) 将 *x* 的积分值向正无穷方向舍入。

— *sourceFormat* **roundToIntegralTowardNegative**(*source*)

roundToIntegralTowardNegative(*x*) 将 *x* 的积分值向负无穷方向舍入。

— *sourceFormat* roundToIntegralTiesToAway（*源格式*）

roundToIntegralTiesToAway(*x*) 将 *x* 四舍五入为最接近的积分值，半途舍入为零。

对于以下操作，舍入方向是适用的舍入方向属性。如果操作数为 NaN，则此操作会发出无效操作异常信号；如果操作数为 al，则此操作会发出不精确异常信号。

— *sourceFormat* roundToIntegralExact（*源格式*）

roundToIntegralExact(*x*) 根据适用的舍入方向属性将 *x* *舍入*为整数值。

5.10 totalOrder 谓词的详细信息

对于每种受支持的算术格式， ementation 应提供以下谓词，以定义特定格式中所有操作数的排序：

- *布尔值* **totalOrder**（*来源，来源*）

totalOrder(*x,y*)对 *x* 和 *y* 格式的规范成员进行总排序：

a) 如果 *x* < *y*，则 **totalOrder**(*x, y*) 为真。

b) 如果 *x* > *y*，则 **totalOrder**(*x, y*) 为假。

c) 如果 *x* = *y*：

1) **totalOrder**(-0, +0) 为 true。

2) **totalOrder**(+0, -0) 为假。

3) 如果 *x* 和 *y* 表示同一个浮点基准：

i) 如果 *x* 和 *y* *都*是负号、
当且仅当 *x* 的指数≥ *y* 的指数时，**totalOrder**(*x, y*) 为真

ii) 否则
当且仅当 *x* 的指数小于 *y* 的指数时，**totalOrder**(*x, y*) 为真。

d) 如果 *x* 和 *y* 在数值上无序，因为 *x* 或 *y* 是 NaN：

1) **totalOrder**(-NaN, *y*) 为 true，其中 -NaN 表示带负号的 NaN，*y* *表示*浮点数。

2) **totalOrder**(*x*, -*NaN*) 为 false，其中 -*NaN* 表示带负号的 NaN，*x* 表示浮点数。

3) **totalOrder**(*x*, +NaN) 为 true，其中 +NaN 表示带正号的 NaN，*x* 表示浮点数。

4) **totalOrder**(+NaN, *y*) 为 false，其中 +NaN 表示带正号的 NaN，*y* *表示*浮点数。

5) 如果 *x* 和 *y* 都是 NaN，那么 **totalOrder** 反映的是基于以下条件的总排序：

i) 正号下面的负号指令

ii) +NaN的信号顺序低于静音，-NaN的信号顺序相反

iii) 否则，NaNs 的顺序由执行定义。

信号 NaN 和静态 NaN 都不表示异常。对于规范 *x* 和 *y*，**totalOrder**(*x, y*) 和
totalOrder(*y, x*) 都为真。

无符号 NaN（在非交换格式中可能出现）应像带正符号位的 NaN 一样排序。

注意 - **TotalOrder** 并不对格式中的所有编码进行总排序。特别是，它不 会 区分同一浮点表示法的不同编码，例如一种或两种编码都是非规范编码。

5.11 比较谓词的详细信息

对于每种支持的算术格式，，可以将一个浮点数据与该格式的另一个浮点数据进行比较（见 5.6.1）。此外，只要操作数的格式具有相同的弧度，就可以将一个浮点数据与另一个不同格式的浮点数据进行比较。

有四种相互排斥的关系：*小于*、*等于*、*大于*和*无序*；当至少有一个操作数是 NaN 时，就会出现*无序*。每个 NaN 都应*无序地*与包括自身在内的一切进行比较。比较应忽略零的符号（因此 +0 = -0）。符号相同的无限操作数应进行*等价比较*。

语言标准应定义表 5.1 和表 5.2 中的比较谓词。这些谓词提供真-假响应，并成对定义。每个谓词都是真，当且仅当它所表示的任何关系为真时。一对谓词中的每个成员都是另一个谓词的逻辑否定。使用前缀（如 NOT）来否定一个谓词会颠倒其相关条目的真假意义，但不会改变*无序*关系是否发出无效操作异常的信号。

无效运算是比较谓词唯一可以发出的异常信号。所有谓词都会在操作数为 NaN 时发出操作无效的异常信号。名为 "安静 "的谓词不会发出任何异常信号，除非操作数是示意 NaN。名为 "信号 "的谓词应就静态 NaN 操作数发出无效操作异常信号。

对于常用数学符号 = ≠ > ≥ < ≤，许多语言标准都定义了不明确表示信号或静音的符号。在表 5.1 中，这些符号用常用数学符号表示。语言标准应将符号 = 和 ≠ 的标记与表 5.1 中的 "安静 "谓词相对应，将符号 > ≥ < ≤ 的标记与表 5.1 中的 "信号 "谓词相对应。

表 5.1-推荐用于常用数学符号的必要谓词和否定词

数学 符号	**语词** *真关系*	数学 符号	**否定** *真关系*
=	**compareQuietEqual** *与...相当*	≠, NOT =	**compareQuietNotEqual** *小于、大于、无序*
>	**比较信号强度** *大于*	不 >	**compareSignalingNotGreater** *小于、等于、无序*
≥	**比较信号大小相等** *等于, 大于*	不≥	**compareSignalingLessUnordered** *小于，无序*
<	**compareSignalingLess** *不到*	不 <	**compareSignalingNotLess** *等于、大于、无序*
≤	**compareSignalingLessEqual** *小于、等于*	不≤	**compareSignalingGreaterUnordered** *大于，无序*

注意 - 当出现 NaN 时，操作数具有*无序*关系，因此三分法不适用。例如，NOT (X < Y) 在逻辑上不等同于 (X ≥ Y)。表 5.1 中的信号谓词会对安静的 NaN 操作数发出无效运算异常信号，以警告假设三分法编写的程序可能出现的不正确行为。

表 5.2- 其他必要谓词和否定词

语词 *真关系*	**否定** *真关系*
compareSignalingEqual *与...相当*	**compareSignalingNotEqual** *小于、大于、无序*
compareQuietGreater *大于*	**compareQuietNotGreater** *小于、等于、无序*
compareQuietGreaterEqual *等于, 大于*	**compareQuietLessUnordered** *小于，无序*
compareQuietLess *不到*	**compareQuietNotLess** *等于、大于、无序*
compareQuietLessEqual *小于、等于*	**compareQuietGreaterUnordered** *大于，无序*
compareQuietUnordered *无序*	**compareQuietOrdered** *小于、等于、大于*

5.12 浮点数据与外部字符序列之间的转换细节

本条款 规定了受支持格式与外部字符序列之间的转换。请注意， 不同半径的受支持格式之间的转换要按 5.4.2 中所述正确舍入并正确设置异常，但须遵守下文 5.12.2 中所述的限制。

实现应提供每种支持的二进制格式和外部十进制字符序列之间的转换，以便在 roundTiesToEven 条件下，从支持的格式转换到外部十进制字符序列，然后再转换回原来的浮点表示形式，但有信号的 NaN 可能会转换为无信号的 NaN。详见 5.12.1 和 5.12.2。

实现应提供从每种支持的十进制格式到外部十进制字符序列的精确转换，并应提供能恢复原始浮点表示的返回转换，但有信号的 NaN 可转换为无信号的 NaN。详见 5.12.1 和 5.12.2。

实现应提供从每种支持的二进制格式到外部字符序列的精确转换，外部字符序列表示符号为十六进制数的数字，并应提供恢复原始浮点表示的转换，但信号 NaN 可转换为静态 NaN。详见 5.12.1 和 5.12.3。

本条款主要讨论程序执行过程中的转换；有一个特殊的考虑因素适用于与程序执行分离的程序翻译：如果程序文本中没有其他规定，在翻译时将程序文本中的常量从外部字符序列转换为支持的格式，应使用本标准的默认舍入方向和语言定义的异常处理。实施方案还可以提供一些方法，允许在执行时用执行时有效的属性和执行时产生的异常来翻译常量。

本标准未定义字符编码（ASCII、Unicode 等）问题。

5.12.1 代表零、无穷大和 NaN 的外部字符序列

从支持的格式到外部字符序列 es 的转换（5.4.2 中描述）以及恢复原始浮点表示的转换，应恢复零、无穷大、静态 NaN 以及非零有限数。特别是，零和无穷小的符号应予以保留。

将支持格式中的无穷小转换为外部字符序列，应产生语言定义的 "inf "或 "infinity"，或除大小写外等价的序列（如 "Infinity "或 "INF"），如果输入为负数，则在前面加上减号。如果输入是正数，转换是否会在前面产生一个正号是由语言定义的。

将外部字符序列 "inf "和 "无穷大"（无论大小写）转换为支持的浮点格式，应产生一个无穷大（符号与输入相同）。

将支持格式中的静态 NaN 转换为外部字符序列时，应产生语言定义的 "nan "或除大小写外等价的序列（*如*"NaN"），前面可选择符号。(本标准不解释 NaN 的符号）。

将受支持格式中的信令 NaN 转换为外部字符序列时，应产生语言定义的 "snan "或 "nan"，或除大小写外等价的序列，并带有可选的前置符号。如果信号 NaN 转换产生的是 "nan "或除大小写外等价的序列，并带有可选的前置符号，则应发出无效操作异常信号。

将外部字符序列 "nan"（无论大小写）转换为支持的浮点格式时，应产生静态 NaN。

将外部字符序列 "snan"（无论大小写）转换为支持的格式时，如果前面带有可选符号，则要么产生一个有信号的 NaN，要么产生一个无信号的 NaN，并发出无效操作异常信号。

语言标准应提供将支持格式中的 NaN 转换为外部字符序列的可选方法，即在基本 NaN 字符序列上附加一个可表示有效载荷的后缀（见 6.2）。有效负载后缀的形式和解释由语言定义。语言标准应要求在将外部字符序列转换为受支持格式时，接受任何此类可选输出序列作为输入。

5.12.2 代表有限数字的外部十进制字符序列

实现应提供将所有支持的浮点 ng-point 格式转换为外部十进制字符序列的操作（见 5.4.2）。对于有限数值，这些操作可视为由源格式、结果中的有效位数（如果指定）以及是否保留量子（对于十进制格式）参数化的。需要注意的是，指定有意义位数和指定量子保留是互不相容的。指定有意义位数和指定量子保留的方法由语言定义，通常体现在 5.4.2 的*转换规范（conversionSpecification）*中。

实现还应提供将外部十进制字符序列转换为所有支持格式的操作。这些操作可视为由结果格式参数化的。

在本条款规定的范围内，除非需要四舍五入，否则两个方向的转换均应保留数值，并应保留其符号。如果需要四舍五入，则应使用正确的四舍五入，并正确显示不精确和其他例外情况。

从外部字符序列到支持的十进制格式的所有转换都应保留量子（见 5.4.2），除非需要四舍五入。每种支持的十进制格式至少有一种转换应保留量子以及数值和符号。

如果转换到外部字符序列需要指数，但指数宽度不足以避免溢出或下溢（见 7.4 和 7.5），则应通过适当的语言定义的字符序列向用户指示溢出或下溢。

为了讨论正确四舍五入换算的限制，请定义以下数量：

— 对于二进制 16，*Pmin* (binary16) = 5

— 对于二进制 32，*Pmin* (binary32) = 9

— 对于二进制 64，*Pmin* (binary64) = 17

— 对于二进制 128，*Pmin*（二进制 128）= 36

— 对于所有其他二进制格式 *bf*，*Pmin* (*bf*) = 1 + ceiling($p \times \log10(2)$)，其中 *p* 是 *bf* 中的有效位数

— M = max(*Pmin* (*bf*)) 适用于所有支持的二进制格式 *bf*

— 对于十进制 32，*Pmin*（十进制 32）= 7

— 对于十进制 64，*Pmin* (decimal64) = 16

— 对于十进制 128，*Pmin*（十进制 128）= 34

— 对于所有其他十进制格式 *df*，*Pmin* (*df*) 是 *df* 的有效位数。

无论要求或给出多少位数，与受支持的十进制格式之间的转换都应正确四舍五入。

在支持的二进制格式之间，通过正确的四舍五入可以转换的有效位数可能有一个实施定义的限制。该限制（H）应使 H≥ M + 3，且 H 应是无限制的。

对于所有支持的二进制格式，转换操作应支持从 1 到 H 的所有有效位数（即，如果 H 无限制，则支持所有可表达的位数）到外部字符序列或从外部字符序列到外部字符序列的正确舍入转换。

从支持的二进制格式转换为外部字符序列时，如果指定的有效位数超过 H 位，则应填充尾部的零。

从超过 H 个有效位数或指数范围大于目标二进制格式的 字符序列转换而来的字符序列，应首先按照适用的四舍五入方向正确地四舍五入到 H 个位数，并发出异常信号，如同从更宽的格式缩小，然后按照适用的四舍五入方向正确地四舍五入转换所产生的 H 个位数的字符序列。

注 1 - 综上所述，以下情况属实：

— 小数格式的四舍五入正确无误。

— 对于二进制格式，所有有效位数为 H 或更少的转换都会根据适用的四舍五入方向正确四舍五入；有效位数大于 H 的转换可能会在最后一位产生 $10^{(M-H)} < 10^{-3}$ 单位数 量 级 的 额外四舍五入。

— 即使给出的有效数字超过 H 位，也会遵守定向四舍五入限制：定向四舍五入误差的符号在任何情况下都是正确的，并且在幅度的最后一位都不会超过 1 + 1/1000 单位。

— 转换是单调的；将支持的浮点数转换为外部字符序列后，增加浮点数的值不会减少其值，而将支持的浮点数转换为外部字符序列后，增加外部字符序列的值也不会减少其值。

— 从支持的二进制格式 *bf* 转换到外部字符序列，然后再转换回来，只要指定的有效位数至少有 *Pmin* (*bf*)，并且在两次转换过程中有效的舍入方向属性是四舍五入到最接近的舍入方向属性，就能得到原始数字的副本。

— 从受支持的十进制格式 *df* 转换到外部字符序列，然后再转换回来，只要转换到外部字符序列时能保留量子，就能得到原始数字的规范副本。

— 从受支持的十进制格式 *df* *转换*为外部字符序列，然后再转换回来，只要指定的有效位数至少有 *Pmin* (*df*)，就能恢复原始数字的值（但不一定是量）。

— 所有实现都交换等效的十进制序列：如果两个十进制字符序列表示相同的值（和量子，对于十进制格式），它们就是等效的；如果两个实现都支持给定的格式，当指定相同的位数且（对于二进制格式）指定的位数不大于 H（对于两个实现），或（对于十进制格式）指定量子保留转换时，它们会将该格式中的任何浮点表示转换为等效的十进制字符序列。

— 同样，当两个实现都支持有效位数和结果格式时，任何两个实现都能将等效的十进制序列转换为相同的浮点数（对于十进制格式，量子相同）。

注 2 -H 应尽可能大，注意在许多系统中 "实用 "可能包括 "无限制"，因为任何 H 至少与最长精确十进制表示法所需的位数一样大，实际上与无限制一样好。对于二进制 128 来说， 最长精确十进制表示法的长度不到一万二千位。

5.12.3 代表有限数字的外部十六进制符号字符序列

语言标准应提供所有支持的二进制格式与外部 nal hexadecimal-significand 字符序列之间的转换。有限数字的外部十六进制符号字符序列应由以下语法描述，该语法定义了*十六进制序列（hexSequence*）：

```
符号                [+-]
数字                [0123456789]
十六进制数字        [0123456789abcdefABCDEF]

hexExpIndicator     [Pp]
十六进制指示符      "0" [Xx]
hexSignificand      ( {hexDigit} * "." {hexDigit}+ | {hexDigit}+ "." | {hexDigit}+ )
decExponent{hexExpIndicator}   {sign}?{digit}+
十六进制序列{符号} ？ {hexIndicator} {hexSignificand} {decExponent} {十六进制符号}?
```

其中，"[...]"表示从括号中列出的终端字符中选择一个，"{...}"表示前面命名的规则，"(...|...|...) "表示从三个备选规则中选择一个，直双引号表示终端字符，"?表示前一项没有实例或只有一个实例，'*' 表示前一项有零个或多个实例，'+'表示前一项有一个或多个实例。

*十六进制符号*被解释为十六进制常数，其中每个*十六进制数字*代表 0 至 15 范围内的一个值，字母 "a "至 "f "代表 10 至 15（无论大小写）。在*十六进制符号*中，第一个（最左边的）字符是最重要的。如果存在句号，则句号定义了十六进制小数部分的开始；如果句号在所有十六进制数字的右边，则*十六进制符号*是一个整数。*decExponent* 被解释为十六进制表示的十进制整数，同样是最重要的数字在前。

hexSequence 的值是 *hexSignificand 的值*乘以 *decExponent 值的*幂，如果有前导"-"号，则予以否定。*hexIndicator* 和 *hexExpIndicator* 对数值没有影响。

在没有明确精度说明的情况下转换为十六进制标识符字符序列时，应使用足够的十六进制字符来准确表示二进制浮点数。在有明确精度说明的情况下转换为十六进制符号字符序列，以及从十六进制符号字符序列转换为支持的二进制格式时，都会根据适用的二进制舍入方向属性正确舍入，并适当提示所有异常情况。

注 -此处描述的外部十六进制符号字符序列遵循 ISO/IEC 9899:2018(E) Programming languages - C (C17) 中为有限数字指定的字符序列：

6.4.4. 2 浮动常数
7.21.6.1 fprintf（转换指定符 "a "和 "A）
7.21.6.2 fscanf（转换指定符 "a）
7.22.1.3 strtod.

6. 无穷大、NaNs 和符号位

6.1 无穷大运算

无穷大在浮点运算中的表现是根据操作数任意大的实数运算的极限情况推导出来的。无穷大应从仿射意义上解释，即-∞<{每个有限数}<+∞。

对无限操作数的运算通常是精确的，因此没有例外信号，其中包括

— **加法**（∞，x）、**加法**（x，∞）、**减法**（∞，x）或**减法**（x，∞），适用于有限 x

— **乘法**（∞，x）或**乘法**（x，∞），针对有限或无限 $x \neq 0$

— **除法**（∞，x）或**除法**（x，∞），适用于有限 x

— **平方根**(+∞)

— **余数**（x，∞）为有限正态 x

— 将一个无穷小转换成另一种格式的相同无穷小。

只有在下列情况下，才会发出与无穷大有关的异常信号

— ∞是无效操作数（见 7.2）

— 通过溢出（见 7.4）或除以零（见 7.3）从有限操作数中生成 ∞

— **remainder**(subnormal, ∞) 信号为下溢。

6.2 对 NaN 的运算

在所有浮点运算中，应支持两种不同的 NaN：信令 NaN（si gnaling）和静音 NaN（quiet）。信号 NaN 可用于表示未初始化变量和类似算术的 增强（如复杂无穷大或极宽范围），但不在本标准范围内。静态 NaN 应通过实施者自行决定的方式，提供从无效或不可用数据和结果中继承的追溯诊断信息。为便于传播 NaN 中包含的诊断信息，应在 NaN 运算结果中尽可能多地保留该信息。

在默认异常处理下，除非另有说明，否则任何发出无效操作异常信号的操作，如果要输出浮点运算结果，都将输出静态 NaN。有关非默认处理方法，请参见 8.

除 5.12 所述的转换外，信令 NaN 应为保留操作数，可为每个一般运算和信令运算操作发出无效操作异常信号（见 7.2）。

除了 **fusedMultiplyAdd** 可能发出无效运算异常信号（参见 7.2）外，所有涉及一个或多个输入 NaN 的一般运算和静态运算操作（均 不发出信号）都不应发出异常信号。对于有静态 NaN 输入的运算，除非另有说明，否则如果要输出浮点运算结果，其结果应为规范的静态 NaN。格式转换（包括受支持格式与外部字符序列表示之间的转换）可能无法输出相同的 NaN。静态 NaN 表示某些不提供浮点结果的操作出现异常；这些操作，即比较和转换为无 NaN 的格式，将在 5.6、5.8 和 7.2 中讨论

。

6.2.1 二进制交换格式中的 NaN 编码

所有二进制 NaN 位字符串的符号位 *S* *都*设为 0 或 1， ，偏指数字段 *E* *的*所有位都设为 1（见 3.4）。静态 NaN 比特串编码时，尾部符号字段 *T* *的*第一位（d_1）应为 1。如果尾随符号字段的第一位为 0，尾随符号字段的其他位必须为非 0，以区分 NaN 和无穷大。在上述首选编码中，信号 NaN 应通过将 $_{d1}$ 设置为 1 来消除，*T* 的其余位保持不变。位 $_{d2\ d3}$ ...

尾部符号字段的 d_{p-1} 包含有效载荷的编码，可能是诊断信息（见 6.2）。

6.2.2 十进制交换格式中的 NaN 编码

十进制信号 NaN 应通过清除 $_{G5}$ 并保留数字 $_{d1}$ 至 *d2* 的值来消除。
尾部符号字段的 $_{dp-1}$ 不变（见 3.5）。

对于十进制格式，有效载荷是 3.5 中定义的尾部符号字段。

6.2.3 NaN 传播

将 NaN 操作数传播到结果的操作 ，如果输入是单个 NaN，则应产生一个 NaN，其有效载荷为输入的 NaN（如果可以用目的格式表示）。

如果两个或两个以上的输入是 NaN，那么生成的 NaN 的有效载荷应与其中一个输入 NaN 的有效载荷相同（如果可以用目的格式表示的话）。本标准没有规定由 哪个输入 NaN 提供有效载荷。

将静态 NaN 从较窄的格式转换为相同弧度的较宽格式，然后再转换回相同的较窄格式，除了使静态 NaN 有效载荷规范化外，不应以任何方式改变静态 NaN 有效载荷。

将静态 NaN 转换为相同或不同弧度的浮点格式时，如果无法保留有效载荷，则应返回静态 NaN，并提供一些语言定义的诊断信息。

应提供从外部字符序列读写有效载荷的方法（见 5.12.1）。

除 5.5 另有规定的操作外，NaN 结果应是规范的，即使该 NaN 结果是由非规范的 NaN 操作数产生的。

6.3 符号位

当 输入或结果均为 NaN 时，本标准不解释 NaN 的符号。但是，对位字符串的操作（**copy**、**negate**、**abs**、**copySign**）**会**指定 NaN 结果的符号位，有时是基于 NaN 操作数的符号位。逻辑谓词 **totalOrder** 和 **isSignMinus** 也受 NaN 操作数符号位的影响。对于所有其他操作，本标准不指定 NaN 结果的符号位，即使只有一个输入 NaN 或 NaN 由无效操作产生。

当输入和结果都不是 NaN 时，积或商的符号是操作数符号的排他 OR；和的符号或被视为和 $x + (-y)$ 的差 $x - y$ *的符号*最多与加数符号中的一个不同；转换、**量化操作**、整数取整操作和**整数取整操作**（见 5.3.1）的结果符号是第一个或唯一操作数的符号。即使操作数或结果为零或无限，这些规则也应适用。

当两个符号相反的操作数之和（或两个符号相同的操作数之差）正好为零时，在所有四舍五入属性下，该和（或差）的符号应为 +0，但 roundTowardNegative 属性除外；在该属性下，正好为零的和（或差）的符号应为 -0。 然而，在所有四舍五入属性下，当 x *为*零时，$x + x$和 $x - (-x)$ 的符号为 x。

当 $(a \times b) + c$ 恰好为零时，**fusedMultiplyAdd**(a, b, c) 的符号应按上述操作数之和的规则确定。当 $(a \times b) + c$ 的精确结果不为零，但由于四舍五入的原因，**fusedMultiplyAdd 的结果为零时**，零结果取精确结果的符号。

除了 **squareRoot**(-0) 应为-0 外，所有数字 **squareRoot** 结果都应为正号。

7. 异常和默认异常处理

7.1 概述：异常和标记

本条款规定了五种异常，当这些异常出现时，应及时发出 gnaled 信号；信号会调用默认或备用的异常处理方法。对于每种异常，实现都应 提供相应的状态标志。

本条款还规定了异常信号的默认不间断异常处理，即交付默认结果、继续执行并引发相应的状态标志（精确下溢的情况除外，参见 7.5）。第 8 条规定了这些信号的备用异常处理属性；语言标准可能会规定实现其中一些属性，然后定义用户启用这些属性的方法。对一种异常的默认或备用异常处理也可能发出其他异常信号（见溢出和下溢，7.4 和 7.5）。因此，可以通过默认、备用异常处理或明确的用户操作（见 5.7.4）来引发状态标志。

在默认的异常处理过程中，状态标志的升起通常表示相应的异常已发出信号并按默认方式进行了处理。只有在出现精确下溢的情况下，才会在不升起状态标志的情况下处理异常；只有在用户提出要求时，才会在未发出异常信号的情况下升起状态标志。只有在用户提出要求时，才会降低状态标志。用户应能单独或集体测试和更改状态标志，并能一次性保存和恢复所有状态标志（见 5.7.4）。

未从其他源程序继承状态标志的程序在开始执行时会降低所有状态标志。在没有用户明确说明的情况下，语言标准应指定默认值：

- 在程序明确设置或测试该标志的作用域之外，是否存在任何特定标志（即是否可通过调试器等非程序手段进行测试）。
- 当标志的作用域大于被调用函数时，异步事件（如在另一个线程或信号处理程序中升高或降低标志）是否以及何时会影响在该被调用函数中测试的标志；这包括程序中显式异步引起的事件，以及语言或实现引入的异步引起的事件。
- 当一个标志的作用域大于被调用函数的作用域时，该标志的状态是否可以在被调用函数中通过非编程方式（如调试器）确定。
- 被调用函数中的标志是否会在调用函数中引发标志。
- 在调用函数中引发的标志是否会在被调用函数中引发标志。
- 在没有明确程序声明的情况下，是否允许（如果允许） 指定标志为持久标志：
  - 特定函数开始执行时的标志是从外部环境（通常是调用函数）继承的。
  - 从调用函数返回或终止时，调用函数中的标志是返回或终止时被调用函数中的标志。

调用本标准所要求的任何操作时，最多只能发出一个异常信号；其他异常可能通过默认异常处理或第一个异常的备用异常处理发出信号。溢出的默认异常处理（见 7.4）发出不精确异常信号。如果默认结果不精确，则下溢的默认异常处理（见 7.5）会发出不精确异常信号。默认的无效异常处理（见 7.2）由于操作数为 NaN 而发出信号，因此可以通过无序信号谓词发出另一个无效操作异常信

号。

调用 **restoreFlags** 或 **raiseFlags** 操作（见 5.7.4）可能引发任意状态标志组合。在默认情况下处理所有异常时，调用本标准所要求的任何其他操作最多可能引发两个状态标志，即不精确溢出（见 7.4）或不精确欠溢（见 7.5）。

对于本标准中定义的计算操作，下文定义了例外情况，只有在出现特定条件时才会发出信号。因此，这些运算的计算方式应能避免用户在使用本标准时出现异常。

即使操作是在软件中使用其他异常信号操作实现的，也应为其他条件提供可观察到的异常信号。本标准中未规定的操作，如复数运算或某些超越函数，应根据本标准中定义的操作的下述定义发出异常信号，但这可能并不总是经济的。对本标准中未规定的运算发出异常信号是由语言定义的。

注--在默认异常处理下，用户无法检测到操作实现发 出 的 冗余异常信号。如果支持子异常（见 8.1），用户可以在（推荐的）备用异常处理（见 8.2）的 recordException 属性下检测到 此类冗余信号，如果是带有信号 NaN 操作数的无序信号谓词，用户也可以在其他备用异常处理属性下检测到此类冗余信号。

7.2 无效操作

当且仅当没有可用的定义结果时，才会发出操作无效的信号。在这种情况下，操作数对要执行的操作无效。

对于产生浮点格式结果的操作，发出无效操作异常信号的操作的默认结果应为静态 NaN，并提供一些诊断信息（参见 6.2）。这些操作包括

a) 信号 NaN 的任何一般运算操作（见 6.2），某些转换操作除外（见 5.12）

b) **乘法**：**乘法**（0，∞）或**乘法**（∞，0）

c) **fusedMultiplyAdd**： **fusedMultiplyAdd**（0，∞，*c*）或 **fusedMultiplyAdd**（∞，0，*c*），除非 *c* 是静态 NaN；如果 *c* 是静态 NaN，那么是否发出无效操作异常信号由实现定义

d) **MultiplyAdd**：无穷大的大小减法，如：**加法**、**减法**或**融合**：
加法（+∞，-∞）

e) **除法**：**除法**（0，0）或**除法**（∞，∞）

f) **余数**： **remainder**(*x*,*y*)，当 *y* 为零或 *x* 为无限且都不是 NaN 时

g) 如果操作数小于零，则为**平方根**

h) 当结果不符合目标格式或一个操作数是有限的而另一个操作数是无限的时，进行**量化处理**

对于不产生浮点格式结果的运算，发出无效运算异常信号的运算有

i) 信号 NaN 上的任何信号计算操作（见 6.2）；然后，在默认异常处理下，用安静 NaN 代替信号 NaN 操作数对操作进行求值，以确定结果，对于无序信号比较，可能会发出另一个无效操作异常信号

j) 将浮点数转换为整数格式，如果源值是 NaN、无穷大或在适用的四舍五入属性下将转换为结果格式范围之外的整数的值

k) 通过表 5.2 中列出的无序信号谓词进行比较，当操作数为
无序

l) **logB**(NaN)、**logB**(∞) 或 **logB**(0)，当 *logBFormat* 为整数格式时（见 5.3.3）。

7.3 除以零

当且仅当对有限操作数的操作定义了精确的无限结果时，dividByZero 除 离子才会发出信号。除以零的默认结果应是根 据 操作正确签名的∞：

— 对于**除法**，当被除数为零而红被除数为有限的非零数时，无穷大的符号是操作数符号的排他 OR（见 6.3）。

— 对于 **logB**(0)，当 *logBFormat* 为浮点格式时，无穷大的符号为负号 (-∞)。

7.4 溢出

如果且仅当目标格式的最大有限数在指数范围无限制的情况下被四舍五入后的浮点结果（见第 4 条）超过时，才会发出溢出（overflow e ）异常信号。默认结果应由舍入方向属性和中间结果的符号决定，如下所示：

a) roundTiesToEven 和 roundTiesToAway 会 将所有溢出结果转为 ∞，并带有中间结果的符号。

b) roundTowardZero 会将所有溢出都转为格式中最大的有限数值，并保留中间结果的符号。

c) roundTowardNegative 将正数溢出到格式的最大有限数，将负数溢出到-∞。

d) roundTowardPositive 将负数溢出到格式的最负有限数，将正数溢出到 +∞。

此外，在溢出的默认异常处理中，溢出标志将被唤起，不精确异常将被发出信号。

7.5 下溢

当检测到微小的非零结果时，应发出下溢信号 eption。对于二进制格式，这应是

a) *四舍五入后*，当计算的非零结果好像指数范围没有限制时，将严格介于 $\pm b^{emin}$ 之间，或

b) 在*四舍五入之前*，如果指数范围和精度都没有限制，计算出的非零结果将严格介于 $\pm b^{emin}$ 之间。

实现者可选择检测锡度的方式，但应以相同方式检测弧度为 2 的所有运算的锡度，包括二进制舍入属性下的转换运算。

对于十进制格式，在四舍五入之前会检测到微小性--此时，如果指数范围和精度都没有限制，计算出的非零结果将严格位于 $\pm b^{emin}$ 之间。

对下溢的默认异常处理应始终提供四舍五入的结果。检测微小性的方法不会影响舍入后的结果，舍入后的结果可能是零、次正常或 $\pm b^{emin}$ 。

此外，在默认的下溢异常处理中，如果舍入的结果是不精确的，即与指数范围和精度均未受限时的

计算结果不同，则将引发下溢标志，并发出不精确（见 7.6）异常信号。如果四舍五入的结果是精确的，则不会引发标志，也不会发出不精确异常信号。这是本标准中唯一一种接受默认处理而不引发相应标志的异常信号。在默认处理方式下，这种下溢信号没有可观察到的影响。

7.6 不精确

除非规范 另有说明（例如 5.8 和 5.9），否则如果四舍五入后的结果与指数范围和精度均无约束的情况下计算出的结果不同，则在没有其他异常信号的情况下，输出数值结果的操作应发出不精确信号。四舍五入后的结果应传送到目的地。

注意- 默认的溢出异常处理会引发溢出标志和不精确信号。当舍入结果不是精确的次正常值时，默认的下溢异常处理会引发下溢标志和不精确信号。当默认处理所有这些异常时，当溢出或下溢标志被触发时，不精确标志总是被触发。

8. 备用异常处理属性

8.1 概述

语言标准 ds 应定义并要求实施 ions 为用户提供将替代异常处理属性与块关联起来的方法（见 4.1）。备用异常处理程序指定了异常列表，以及在发出信号的情况下对每个列出的异常所要采取的操作。语言标准应定义包含第 7 条所列任何异常子集的异常列表：无效运算、除以零、溢出、下溢或不精确。语言标准还应定义包含以下内容的异常列表

— **所有例外情况**：第 7 条列出的所有五种例外情况，或

— 子异常的任何子集--第 7 条中异常的子案例（*如* 7.2 中无效操作异常的子案例）；当且仅当操作属于子异常的指定案例时，才会对列出的子异常采取行动；子异常名称由语言定义。

语言标准应定义本条款的所有备用异常处理属性。特别是，语言标准应为第 7 条所列的五种例外情况中的每一种例外情况定义至少一个延迟的备用异常处理属性（见 8.3）。此类属性值规范的语法和范围由语言定义。

更改异常处理属性不会发出任何异常信号。

8.2 恢复备用异常处理属性

将恢复备用异常处理属性与 一个程序块相关联意味着：根据指定的恢复属性处理隐含异常，并恢复相关程序块的执行。实现应支持这些恢复属性：

— **默认**

在相关程序块中提供默认的异常处理（见第 7 条），尽管在更大范围内可能会有替代的异常处理。

— **raiseNoFlag**

提供默认的异常处理（见第 7 条），而不引发相应的状态标志。

— **可升旗**

提供默认的异常处理方式（见第 7 条），但语言可以定义是否引发标记。各种语言可根据性能要求遵照执行。

— **记录异常**

提供默认的异常处理（见第 7 条），并在第 7 条规定触发标记时记录相应的异常。记录异常意味着存储异常描述，包括语言定义的细节，其中可能包括当前操作和操作数，以及异常的位置。语言标准定义了将异常描述转换为字符序列或从字符序列转换为异常描述的操作，以及检查、保存和恢复异常描述的操作。

— **代入(*x*)**

可为任何异常指定：用变量或表达式 *x* 替换此类异常操作的默认结果。

— **substituteXor(*x*)**

可指定乘法或除法运算中出现的任何异常情况：类似于 substitute(*x*)，但用 |x| 替换此类异常运算的默认结果，如果 |x| 不是 NaN，则从操作数符号的 XOR 中获取符号位。

— **突然下溢**

当检测到微小的非零结果而发出下溢信号时，用相同符号的零或相同符号的最小正常舍入结果替换默认结果，唤起下溢标志，并发出不精确异常信号。当 roundTiesToEven、roundTiesToAway 或

适用 roundTowardZero 属性时，舍入后的结果幅度应为零。当 roundTowardPositive 属性适用时，舍入后的结果幅度应为正极小值结果的最小正常幅度，负极小值结果的最小正常幅度应为零。当 roundTowardNegative 属性适用时，舍入的结果幅度应为负微小结果的最小正常幅度，正微小结果的最小正常幅度应为零。该属性对子正常操作数的解释没有影响。

对于返回两个结果（*x* 运算 *y* 及其误差，其中运算符为 +、- 或 ×）的增强算术运算 (9.5)，如果因为使用 roundTiesTowardZero 对 *x* 运算 *y* *进行*四舍五入会发出下溢信号，而出现下溢信号，则两个结果都为零，*x* 运算的符号为 *y*.如果由于误差（*x* 运算 *y* - roundTiesTowardZero(*x* 运算 *y*)）非零且严格位于 $\pm b^{emin}$ 之间而发出下溢信号，则默认误差结果将被一个符号为（*x* 运算 *y* - roundTiesTowardZero(*x* 运算 *y)*）的零所替代。这些情况会引发下溢标志，并发出不精确异常信号。

8.3 立即和延迟交替异常处理属性

将交替异常处理与程序块关联起来意味着：根据指定的属性处理指定的 异常。如果指示的异常已发出信号，那么根据异常和异常处理属性，相关程序块的执行可能会被立即放弃，也可能会继续默认处理。在后一种情况下，异常处理将被延迟，并在相关程序块正常终止时进行。延迟异常处理是完全确定的，而立即异常处理则需要许可，但并不要求执行者以确定性换取性能，因为在关联代码块中计算的中间结果可能不是确定的。

语言标准应定义并要求实施提供这些属性：

— 与程序块相关联的立即交替异常处理程序块：如果提示异常，则根据语言语义，尽快放弃执行相关程序块并执行处理程序块，然后继续执行相关程序块正常终止后本应继续执行的程序块。

— 与程序块相关联的延迟交替异常处理程序块：如果指示的异常发出信号，则默认处理该异常，直到相关程序块正常终止，然后执行处理程序块，然后根据语言的语义，在相关程序块正常终止后继续执行。

— 与程序块相关的立即转移：如果出现异常信号，则尽快转移控制权；不能返回。

— 与程序块相关的延迟转移：如果出现指示的异常信号，则默认处理该异常，直到相关程序块正常终止，然后转移控制权；不能返回。

当出现下溢信号时，无论默认结果是精确的还是不精确的，都应调用立即的下溢替代异常处理。只有当下溢信号与不精确的默认结果相对应，且下溢标志会被触发时，才会调用延迟的下溢替代异常处理。

注 1 - 第 7 条所列异常（但不包括子异常）的延迟备用异常处理可通过测试状态标志来实现。无论如何执行，都应在相关程序块开始之前保存与所指示的异常相对应的状态标志，然后将其降低。在关联程序块结束时，应保存当前的状态标志，并恢复先前保存的状态标志。然后测试最近保存的状态标志，以确定是执行处理程序块还是转移控制。

注 2 - 对异常的即时交替异常处理可以通过陷阱来实现，对于第 7 条所列的除下溢外的异常，也可以通过在每次操作后或相关块结束时测试状态标志来实现。因此，对于第 7 条所列的除下溢以外的异常，如果没有更好的实现机制，可采用与延迟异常处理相同的机制来实现即时异常处理。无论以何种方式执行，如果所指示的异常没有在关联程序块中发出信号，则相应的状态标志不应更改。如果所指示的异常在关联程序块中发出信号，导致执行处理程序块或转移控制，那么相应状态标志的状态可能不是确定的。

注 3 -转移是一种特定于语言的惯用语，用于不可递归的控制转移。语言标准可能提供多种转移习语，例如

— **break（中断）：** break：放弃相关程序块并继续执行，根据语言语义，在相关程序块正常终止后继续执行。

— **throw exceptionName：抛出**异常名（throw exceptionName）：不在本地处理异常名，而是根据语言的语义，向下一个处理范围发出信号，也许是调用当前子程序的函数。调用者可能会默认处理异常名，也可能采用其他处理方式，例如向下一级调用子程序发出异常名信号。

— **标签：**跳转；根据语言的语义，标签可以是局部的，也可以是全局的。

9. 建议的操作

第 5 条完整规定了所有支持的算术格式所需的操作。

本条款规定了推荐使用的其他操作。在特定的编程环境中，这些操作可以用运算符符号或函数符号表示。特定编程环境中使用的函数名称可能与相应数学函数的名称或本标准相应操作的名称不同。

9.1 符合语言和实施定义的操作

对于一种或多种格式，语言标准和实现可以定义一种或 多种浮点运算，本文档中没有其他定义，但这些浮点运算符合本标准，满足本条款的所有要求。特别是，语言标准应定义尽可能多的 9.2 至 9.7 中适合该语言的操作，并根据本条款加以实现。

9.2 其他数学运算

语言标准应定义表 9.1 中尽可能多的操作，并根据本子条款实施 mented。与本标准的其他操作一样，表 9.1 中的操作名称不一定与任何特定编程语言使用的名称一致。

在本子条款中，运算域是指相应数学函数定义明确的仿射扩展实数子集。

符合要求的操作应针对其域中所有操作数的适用舍入方向正确返回舍入结果。

操作结果应符合规范。

除此处规定的情况外，操作将根据第 7 条规定发出所有适当的异常信号。返回浮点结果的操作，如果操作数中存在示意 NaN，则应返回静态 NaN 作为结果。返回浮点运算结果的操作，如果操作数中有静音 NaN，则应返回静音 NaN，本子条款另有规定的情况除外。

- 无效操作：对于所有操作，NaN 操作数信号应为无效操作异常信号。

  在其域之外的操作应返回安静的 NaN 值，并发出无效操作异常信号。

— divideByZero：对于某些有限浮点操作数的简单极点的操作，应发出 divideByZero 异常信号，并默认返回无穷大。

— 不精确：如果结果不精确，操作应发出不精确异常信号。如果结果精确，则操作不应发出不精确异常信号。

其他例外情况见表 9.1。

表 9.1- 附加数学运算

运行	功能	域名	其他例外情况；另见 9.2.1
exp **expm1** **exp2** **exp2m1** **exp10** **exp10m1**	e^x $e^x - 1$ 2^x $2^x - 1$ 10^x $10^x - 1$	$[-\infty, +\infty]$	溢出；下溢
log **log2** **log10**	$\log_e(x)$ $\log_2(x)$ $\log_{10}(x)$	$[0, +\infty]$	$x = 0$： 除以零； $x < 0$：无效操作
logp1 **log2p1** **log10p1**	$\log_e(1 + x)$ $\log_2(1 + x)$ $\log_{10}(1 + x)$	$[-1, +\infty]$	$x = -1$: divideByZero； $x < -1$：无效操作；下溢
假设(x, y)	$\sqrt{(x^2 + y^2)}$	$[-\infty, +\infty] \times [-\infty, +\infty]$	溢出；下溢
rSqrt	$1/\sqrt{x}$	$[0, +\infty]$	$x < 0$：操作无效； x 为 ±0： 除以零
复合(x, n)	$(1+x)^n$	$[-1, +\infty] \times \mathbf{Z}$	$x < -1$：无效操作；溢出；下溢
rootn(x, n)	$x^{1/n}$	$[-\infty, +\infty] \times \mathbf{Z}$	$n = 0$：无效操作； $x < 0$ 且 n 为偶数：操作无效； $n = -1$：溢出、下溢 $x = 0$ 且 $n < 0$： 除以零
pown(x, n)	x^n	$[-\infty, +\infty] \times \mathbf{Z}$	溢出；下溢
pow(x, y)	x^y	$[-\infty, +\infty] \times [-\infty, +\infty]$	溢出；下溢
powr(x, y)	x^y	$[0, +\infty] \times [-\infty, +\infty]$	溢出；下溢

有关圆三角函数和双曲三角函数的运算，请参见后面的续页。

表 9.1- 附加数学运算（续）

运行	功能	域名	其他例外情况；另见 9.2.1
罪过	sin(*x*)	(-∞, +∞)	| x| = ∞：无效运算；下溢
系数	cos(*x*)	(-∞, +∞)	| x| = ∞：无效操作
黄棕色	tan(*x*)	(-∞, +∞)	| x| = ∞：无效运算；下溢
sinPi	sin(π × *x*)	(-∞, +∞)	| x| = ∞：无效运算；下溢
cosPi	cos(π × *x*)	(-∞, +∞)	| x| = ∞：无效操作
tanPi	tan(π × *x*)	(-∞, +∞)	| x| = ∞：无效操作；下溢； | x| = (*n* + 0.5) 为整数 *n*： 除以零
asin	当于(*x*)	[-1, +1]	| x| > 1：无效操作；下溢
acos	acos(*x*)	[-1, +1]	| x| > 1：无效操作
庵	atan(*x*)	[-∞, +∞]	下溢
atan2(*y*, *x*)	见 9.2.1	[-∞, +∞] × [-∞, +∞]	下溢
asinPi	asin(*x*)/π	[-1, +1]	| x| > 1：无效操作；下溢
acosPi	acos(*x*)/π	[-1, +1]	| x| > 1：无效操作
atanPi	atan(*x*)/π	[-∞, +∞]	下溢
atan2Pi(*y*, *x*)	见 9.2.1	[-∞, +∞] × [-∞, +∞]	下溢
sinh	sinh(*x*)	[-∞, +∞]	溢出；下溢
杯垫	cosh(*x*)	[-∞, +∞]	溢出
黄褐色	tanh(x)	[-∞, +∞]	下溢
asinh	asinh(*x*)	[-∞, +∞]	下溢
弊	acosh(*x*)	[+1, +∞]	*x* < 1：无效操作
atanh	atanh(*x*)	[-1, +1]	下溢； | x| = 1: divideByZero； | x| > 1：无效操作

域使用区间符号：与括号相邻的值包含在域中，与括号相邻的值不包含在域中。**Z** 是整数集合。

域中的符号 A × B 表示元素（*a*，*b*）有序对的集合，其中 *a* 是 A 的元素，*b* 是 B 的元素。

sin、**cos**、**tan**、**asin**、**acos**、**atan** 和 **atan2** 运算以弧度为单位测量角度。**sinPi**、**cosPi**、**tanPi**、**asinPi**、**acosPi**、**atanPi** 和 **atan2Pi** 运算以半周为单位测量角度。

对于由偶数数学函数定义的运算 f，对于所有四舍五入属性的整个域和范围，*f*(-*x*) 都是 *f*(*x*)。*对于*

奇数数学函数定义的运算 f，对于 roundTiesToEven、**roundTiesToAway** 和 **roundTowardZero** 的整个域和范围，$f(-x)$ 都是$-f(x)$。

注：相对于指数范围而言精度非常高的非交换格式可能会提示表 9.1 中未列出的其他异常。例如，**log 表示**下溢，**tan** 表示溢出。

9.2.1 特殊价值

对于 **sin**、 **tan**、**sinPi**、**tanPi**、**asin**、**atan**、**asinPi**、**atanPi**、**sinh**、**tanh**、**asinh**、**atanh**、**expm1**、**exp2m1**、**exp10m1**、**logp1**、**log2p1** 和 **log10p1 等**运算，$f(+0)$ 均为 +0，$f(-0)$ 均为 -0，无一例外。

对于 **cos**、**cosPi**、**cosh**、**exp**、**exp2** 和 **exp10** 运算，$f(\pm 0)$ 无一例外都是 +1。

对于正整数 n， **sinPi**(+n) 为 +0，**sinPi**(-*n*) 为 -0。这意味着，在*四舍五入*为偶数、四舍五入为异数和四舍五入为零时，对于所有 *x*，**sinPi**(-*x*) 和 **-sinPi**(*x*) *都*是相同的数（或都是 NaN）。这意味着对于所有 *x*，在所有四舍五入方向上，**cosPi**(-*x*) 和 **cosPi**(*x*) *都*是同一个数（或都是 NaN）。

对于整数 n≥0：

tanPi($2 \times n + 0.5$)为 +∞，并发出 "除以零 "异常信号

tanPi($2 \times n + 1$)为-0

tanPi($2 \times n + 1.5$)为-∞，提示除法归零异常

tanPi($2 \times n + 2$) 为 +0

对于整数 $n \le 0$：

tanPi($2 \times n - 0.5$)为-∞，提示除法归零异常

tanPi($2 \times n - 1$)为 +0

tanPi($2 \times n - 1.5$)为 +∞，表示除零值异常

tanPi($2 \times n - 2$)为-0

这意味着，在四舍五入、四舍五入和四舍五入为零的条件下，**tanPi**(-*x*) 和
对于所有 *x*，-**tanP**i(*x*) 都是同一个数（或都是 NaN）。

acos(1)、**acosPi(**1) 和 **acosh**(1) 均为 +0。

atan(±∞)是 ±π/2 舍入的，因此应发出不精确异常信号。

atan2(*y, x*) 是点（*x, y*）与 *x 轴*正值在原点所夹的角度；该角度也是复数 $x + \mathrm{i}\,y$ 的对数的参数或相位或虚部。

对于有正符号位的 *y*，在有限非零数字 *x 的*一般情况下，**atan2**(*y, x*) 可以从下面的表达式中正确舍入：

的 **atan2**(*y, x)* 是 **atan**(|*y*/*x*|)，这可能是下溢异常信号
对于有限 $x < 0$，**atan2**(*y, x)* 为 π - **atan**(|*y*/*x*|)。

atan2(*y, x*) 中涉及 0 和 ∞ 的特例是常量，当结果为非零时，它们应该发出不精确异常信号，但它们没有发出其他异常信号：

atan2(±0, -0) 为 ±π

atan2(±0, +0) 为 ±0

atan2(±0, *x)* 在 $x < 0$ 时为

±π **atan2(**±0, *x) 在* $x > 0$ 时

为 ±0 atan2(*y*, *±0) 在 y* < 0

时为 -π/2 **atan2**(*y*, ±0) 在 *y*

> 0 时为 +π/2

atan2(±*y*, -∞) 对于有限 *y* > 0是

±π **atan2**(±*y*, +∞) 对于有限 *y* > 0

是 ±0 **atan2(**±∞, *x)* 对于有限 *x*

是 ±π/2 **atan2(**±∞**,** -∞) 是 ±3π/4

atan2(±∞**,** +∞) 是 ±π/4。

对于某些格式，在某些四舍五入属性下，**atan** (**atan2**) 的四舍五入幅度范围可能会超过 π/2 (π) 的非四舍五入幅度。因此，程序员必须注意妥善处理反向运算时可能出现的异常流形跳转。

atanPi(±∞) 无一例外都是±½。

atan2Pi(*y, x*) 是点（*x, y*）与正 *x 轴*在原点所成的角度。其范围为
atan2Pi 为 [-1，+1]。

对于有正符号位的 *y*，在有限非零数字 *x 的*一般情况下，**atan2Pi**(*y, x*) 可以通过以下表达式正确舍入：

对于有限 *x* > 0，**atan2Pi**(*y, x*) 为 **atan**(| *y*/*x*|)/π，这可能是下溢异常的信号
对于有限 *x* < 0，**atan2Pi**(*y, x*) 为 1- **atan**(|*y*/*x*|)/π。

涉及 0 和 ∞ 的 **atan2Pi**（*y*，*x*）的特殊情况是精确常数，没有例外：

atan2Pi（±0，-0）为±1

atan2Pi(±0, +0) 为 ±0

atan2Pi(±0, *x)* 在 *x* < 0 时为

±1 **atan2Pi(±0**, *x)* 在 *x* > 0

时为 ±0 **atan2Pi**(*y*, ±0) 在 *y*

< 0 时为 -½ **atan2Pi**(*y*, ±0)

在 *y* > 0 时为 +½

atan2Pi（±y，-∞）对有限 *y* > 0为

±1 **atan2Pi**（±*y*，+∞）对有限 *y* >

0 为 ±0 **atan2Pi**（±∞，*x*）对有限

x 为±½ **atan2Pi**（±∞，-∞）为 ±¾

atan2Pi（±∞，+∞）为 ±¼.

sinh(±∞) 和 **asinh**(±∞) 无一例外都是±∞。 **cosh**(±∞) 和 **acosh**(+∞) 无一例外都是＋∞。
tanh(±∞)为±1，没有异常。**atanh**(±1)为±∞，并发出了除零异常信号。

对于运算 **exp**、**exp2** 和 **exp10**，无一例外，*f* (+∞) 是 +∞，*f* (-∞) 是 +0。对于运算 **expm1**、**exp2m1** 和 **exp10m1**，*f* *(*+∞）为 +∞，*f*（-∞）为 -1，无一例外。

对于 log、**log2**、**log10**、**logp1**、**log2p1** 和 **log10p1 的**运算，*f* (+∞) 是 +∞，没有例外。对于 **log**、**log2** 和 **log10** 运算，*f* (±0) 为 -∞，并发出除法归零异常信号，*f* (1) 为 +0。 对于 **logp1**、**log2p1** 和 **log10p1 运算**，f *(*-1) 为 -∞，并发出除法归零异常信号。

对于 **hypot** 运算，**hypot**（±0，±0）为 +0，**hypot**（±∞，qNaN）为 +∞，**hypot**（qNaN，±∞）为 +∞。

rSqrt(+∞)为 +0，没有异常。**rSqrt**(±0)为 ±∞，并发出 divideByZero 异常信号。

对于**复合**运算、**rootn** 和 **pown** 运算，*n* 是*积分格式*中的有限积分值。当 *integralFormat* 是浮点格式时，当第二个操作数是非积分或无限时，这些运算的行为由语言定义。

对于**复合**操作：

当 *x* ≥ -1 时，**化合物** (*x*, 0) 为 1，否则为安静的 NaN
复数（-1，*n*）为 +∞，并在 *n* < 0 时发出 divideByZero 异常信号

当 n > 0 时，**化合物** (-1, *n*) 为

+0 当 n < 0 时，**化合物** (±0, n)

为 1 当 *n* > 0 时，**化合物** (+∞,

n) 为 +∞ 当 *n* < 0 时，**化合物**

(+∞, *n)* 为 +0
复数（*x*，*n*）为 qNaN，并在 *x* < -1 时发出无效运算异常信号

对于 $n \neq 0$，**化合物**（qNaN，n）为 qNaN。

对于 **rootn** 操作：

rootn（±0，n）为±∞，并在奇数 $n < 0$ 时发出 divideByZero 异常信号

rootn（±0，n）为 +∞，并在偶数 $n < 0$ 时发出 divideByZero 异常信号

rootn（±0，n）在偶数 $n > 0$ 时为 +0

对于奇数 $n > 0$，**根号**（±0，n

）为 ±0 对于奇数 $n > 0$，**根号**

（+∞，n）为+∞ 对于奇数 $n >$

0，**根号（**-∞，n）为 -∞

根号（-∞，n）为 qNaN，在偶数 $n > 0$ 时发出无效运算异常信号
当 $n < 0$ 时，**rootn**（+∞，n）为 +0
对于奇数 $n < 0$，**rootn**（-∞，n）为-0
rootn (-∞, n) 是 qNaN，在偶数 $n < 0$ 时发出无效操作异常信号。

注意 **-rootn** (-0, 2) 与 **squareRoot(**-0) 不同，因为它们有不同的一致性考虑。

用于 **pown** 操作：

如果 *x* 不是信号 NaN，则 **pown** (*x*, 0) 为 1

pown (±0, *n*) 为 ±∞，在奇数 *n* < 0 时发出 divideByZero 异常信号 **pown**

(±0, *n*) 为 +∞，在偶数 *n* < 0 时发出 divideByZero 异常信号 **pown (±0**, *n)*

为 +0，在偶数 *n* > 0 时发出 divideByZero 异常信号

pown (±0, *n*) 在奇数 *n* > 0时为

±0 **pown** (+∞, *n) 在 n* > 0 时为

+∞ **pown (**-∞, *n) 在*奇数 *n* > 0

时为 -∞ **pown (**-∞, n) 在偶数 n >

0 时为 +∞ pown (+∞, *n*) 在奇数

n < 0 时为 -0 pown (-∞, n) 在偶

数 n < 0 时为 +0。 **pown (**+∞,

n) 在 *n* < 0 时为 +0 **pown (**-∞, n)

在奇数 *n* < 0时为 -0 **pown** (-∞,

n) 在偶数 *n* < 0时为 +0。

用于**粉末**操作：

如果 *x* 不是信号 NaN，则 **pow** (*x*, ±0) 为 1
pow（±0，*y*）为±∞，并在 *y* 为奇数整数 < 0 时发出 divideByZero 异常信号

pow (±0, -∞) 是 +∞，无一例外 **pow (**±0,

+∞) 是 +0，无一例外 **pow (**±0, *y*) 是 ±0，

对于有限 *y* > 0 的奇整数 **pow (**-1, ±∞) 是 1

，无一例外

pow (+1, *y*) 对任何 *y 都*是 1（即使是静态 NaN）
当 -1 < *x* < 1 时，**pow** (*x*, +∞) 为 +0
当 *x* < -1 或 1 < x时，**pow**（*x*，+∞）为 +∞ *(*包括 ±∞）。
当 -1 < *x* < 1 时，**pow** (*x*, -∞) 为 +∞
对于 *x* < -1 或对于 1 < *x (*包括 ±∞），**pow**（*x*，-∞）为 +0
当数字 *y* < 0 时，**pow** (+∞, *y*) 为 +0
对于数 *y* > 0，**pow**（+∞，*y*）为 +∞

pow（-∞，*y*）在有限 *y* < 0 时为-0，是奇整数 **pow**（

-∞，*y*）在有限 *y* > 0 时为-∞，是奇整数 **pow (**-∞，*y*

）在有限 y < 0 时为 +0，不是奇整数 **pow (**-∞，*y*）

在有限 *y* > 0 时为 +∞，不是奇整数

pow（±0，*y*）为 +∞，并在有限 *y* < 0 且非奇数整数时发出 divideByZero 异常信号
pow(±0, *y*) 在有限 *y* > 0 且不是奇整数时为 +0
pow(*x*, *y*) 表示有限 *x* < 0 和有限非整数 *y* 的无效运算异常。

注--为了支持十进制浮点数中可能出现而二进制浮点数中不可能出现的特殊情况，当 x 为负数而 y 不是奇整数时，语言标准可能会定义另一种幂运算，即 **powd**，其规范将上述最后五条规则扩展为

对于有限的 *y* < 0，**powd**（-∞，*y*）为 +0，为偶数整数
对于有限的 *y* > 0，**powd**（-∞，*y*）为 +∞，是一个偶数整数

powd (+0, *y*) 为 +∞，当有限的 *y* < 0 且不是奇整数时，发出除法归零异常信号 **powd** (-0, *y*) 为 +∞，当有限的 *y* < 0 且不是偶整数时，发出除法归零异常信号 **powd** (+0, *y*) 为 +0，当有限的 *y* > 0且不是奇整数时，发出除法归零异常信号
对于 *y* > 0 的有限偶数，**powd**(-0, *y*) 为 +0
对于最简单形式为 *m* / *n* 且 *m* 为偶数、n 为奇数的有限非整数 *y*，**powd** (-1, *y*) 为 +1 对于最简单形式为 *m* / *n* 且 *m* 和 *n* 均为奇数的有限非整数 *y*，**powd** (-1, y) 为 -1 对于最简单形式为 *m* / n且 n 为偶数的有限非整数 *y*，*powd* (-1, *y*) 为 qNaN 并发出无效运算异常信号
对于有限非整数 *y* 和负 *x*（包括-0、有限负 *x* 和-∞），**powd**(*x*, *y*) 是 **powd** (-1, *y*) × **powd** (abs(*x*), *y*)。

用于**供电**操作：
对于有限 *x* > 0，**powr**（*x*，±0）为 1
powr（±0，*y*）为 +∞，并在有限 *y* < 0 时发出 divideByZero 异常信号
powr (±0, -∞) is +∞ **powr**
(±0, *y*) is +0 for *y* > 0 **powr**
(+1, *y*) is 1 for finite *y*
powr (*x*, *y*) 在 *x* < 0 时发出无效运算异常信号
powr（±0，±0）发出无效操作异常信号

powr（+∞，±0）表示无效操作异常 **powr**（+1，

±∞）表示无效操作异常 **powr**（*x*，qNaN）为

qNaN，对于 $x \geq 0$

powr（qNaN，*y*）为 qNaN。

注- 本标准定义了几种将 *x* 提升到给定幂次的运算：
对于任意 *x*，**pown**(*x*, *n*) 只接受整数幂 *n*
当 y 包含一个等于整数 *n* 的值时，**pow**(*x*, *y*) 的行为与 **pown**(*x*, *n*) 相同

powr(*x*, *y*) 的定义是考虑 **exp**(*y* × **log**(*x*))，因此其域不包括负 *x*。

9.2.2 首选指数

子条款 9.2 中的运算首选指数为 0，以下情况除外：Q(**hypot**(*x*, *y*)) 为 min(Q(*x*), Q(*y*))

Q(**rSqrt**(*x*))为-floor(Q(*x*)/2)Q(**compound**(*x*,

n)) 为 floor(*n* × min(0, Q(*x*)))

Q(**rootn**(*x*, *n*)) 是 floor(Q(*x*)/*n*)

Q(**pow**(*x*, *y*)) 是 floor(*y* × Q(*x*))

Q(**pown**(*x*, *n*)) 是 floor(*n* × Q(*x*))

Q(**powr**(*x*, *y*)) 是 floor(*y* × Q(*x*))。

9.3 动态模式操作

9.3.1 对单个 dy namic 模式的操作

为二进制或十进制舍入方向定义动态模式规范（见 4.2）的语言标准应定义相应的非计算操作，以获取和设置每个指定的动态模式舍入方向的适用值。根据语言的范围规则，舍入方向的适用值可能是 通过常量属性规范或动态模式赋值设置的。如果在舍入方向动态规范的静态范围之外使用这些操作，其效果是由语言定义的（也可能是未指定的）。

定义二进制舍入方向动态模式规范的语言标准应定义

— *二进制包围***方向 getBinaryRoundingDirection**(*void*)
— *void* **setBinaryRoundingDirection**(*binaryRoundingDirection)*.

定义十进制四舍五入方向动态模式规范的语言标准应定义

— *decimalRoundingDirection* **getDecimalRoundingDirection**(*void*)
— *void* **setDecimalRoundingDirection**(*decimalRoundingDirection*).

为其他属性定义动态模式规范的语言标准应定义相应的操作来获取和设置这些动态模式。

9.3.2 所有动态模式的操作

定义动态 模式规范的语言标准，应为所有可动态指定的模式集体定义以下非计算操作：

— *模式组* **saveModes(***void*)

将所有动态可指定模式的值保存为一组。

— *void* **restoreModes**(*modeGroup*)

将所有可动态指定的模式作为一组恢复其值。

— *void* **defaultModes**(*void*)

将所有可动态指定的模式设置为默认值。

modeGroup 表示可动态指定的模式集。**saveModes** 操作的返回值可用作同一程序中 **restoreModes 操作的**操作数；本标准不要求支持任何其他用途。

9.4 削减作业

语言标准应为所有支持的算术格式定义以下还原运算。与本标准中的其他运算不同，这些运算以一种格式对操作数向量进行运算，并以相同格式返回结果。实现时可以任意顺序关联，或以更宽的格式进行运算。

矢量长度操作数应具有语言定义格式 *integralFormat 的*积分值。如果 *integralFormat* 是浮点格式，其精度应至少与 *sourceFormat* 相同，并具有相同的弧度。当向量长度操作数为非整数或负数时，这些操作的行为由语言定义。

数值结果和异常行为，包括无效操作异常及其子异常，可能会因中间操作的精度和求值顺序而不同。不过，每次调用还原运算时只发出一个异常信号，不会对每个异常中间操作数或结果发出异常信号。所有还原运算中，如果有任何操作数是示意 NaN，都会发出无效运算异常信号。一旦出现无效运算条件信号，由于信号 NaN、
$\infty - \infty$，或 $0 \times \infty$，矢量元素的处理可能会停止。

在默认异常处理下，当这些操作发出溢出或不完全下溢信号时，也会发出不完全异常信号。否则，不精确异常是否发出信号未作规定。

对于这些运算，没有规定首选指数和结果的同族成员。语言标准应定义以下和的还原：

— *sourceFormat* **sum**（*源向量，积分格式*）
sum(*p, n*) 是$\sum_{(i=1, n)} pi$ 的实现定义近似值，其中 *p* 是长度为 *n* 的向量。
— *sourceFormat* **dot**（*源向量、源向量、积分格式*）
dot(*p, q, n*) 是$\sum_{(i=1, n)} (pi \times qi)$ 的实现定义近似值，其中 *p* 和 *q* 是长度为 *n* 的向量。
— *源格式* **sumSquare**（*源向量，积分格式*）
sumSquare(*p, n*) 是$\sum_{(i=1, n)} pi2$ *的*实现定义近似值，其中 *p* 是长度为 *n* 的向量。
— *sourceFormat* **sumAbs**（*源*矢量，*积分格式*）
sumAbs(*p, n*) 是$\sum_{(i=1, n)} | pi|$ 的实现定义近似值，其中 *p* 是长度为 *n* 的向量。

在**求和**与**点运算**中，如果任何操作数元素为 NaN，则返回安静的 NaN。如果乘积为 $\infty \times 0$，则会出现无效运算异常。不同符号的无穷小之和会出现无效操作异常。否则，同符号的无穷小之和将返回该无穷小，不会出现异常。否则，计算和时不会出现可避免的中间异常情况，并根据中间结果确定最终结果。如果最终结果溢出，则发出溢出信号。如果最终结果出现下溢，则发出下溢信号。

对于 **sumSquare** 和 **sumAbs**，如果任何操作数元素为无穷大，则返回 $+\infty$。否则，如果任何操作数元素是 NaN，则返回安静的 NaN。否则，将在计算过程中避免出现可避免的中间异常情况，并据此确定最终结果。如果最终结果溢出，则发出溢出信号。如果最终结果出现下溢，则发出下溢信号。

当向量长度操作数为零时，返回值毫无例外地为 +0。语言标准应定义以下按

比例乘积缩减操作：

— (*sourceFormat, integralFormat*) **scaledProd**（*源矢量，积分格式*）
scaledProd(*p, n*) 返回 {pr, *sf*}，因此 **scaleB**(*pr, sf*) 是 $\prod_{(i = 1, n)} pi$, *的*实现定义近似值, 其中 *p* 是长度为 *n* *的*向量。

— (*sourceFormat, integralFormat*) **scaledProdSum**(*源向量, 源向量, integralFormat*)
scaledProdSum(*p, q, n*) 返回 {pr, *sf*}，因此 **scaleB**(*pr, sf*) 是 $\prod_{(i = 1, n)} (pi + qi)$ 的实现定义近似值，其中 *p* 和 *q* 是长度为 *n* *的*向量。

— (*sourceFormat, integralFormat*) **scaledProdDiff**(source vector, *source vector, integralFormat*) **scaledProdDiff**(*p, q, n*) 返回 {pr, *sf*}，因此 **scaleB**(*pr, sf*) 是$\prod_{(i = 1, n)} (p_i - q_i)$的实现定义近似值，其中 *p* 和 *q* 是长度为 *n* 的向量。

矢量操作数和结果中的比例乘积成员应采用相同的格式。矢量长度操作数和结果中的比例因子成员应具有积分值，并采用相同的语言定义格式 *integralFormat*。

对于 **scaledProd**、**scaledProdSum** 和 **scaledProdDiff**，如果任何操作数元素为 NaN，则返回安静的 NaN。如果乘积为 ∞ × 0，则会出现操作无效异常。不同符号的无穷级数之和（或同类符号的无穷级数之差）会导致无效操作异常。否则，如果乘积中有无穷小，则返回无穷小，不提示无效操作异常。否则，如果乘积中存在零，则返回零，并且不提示无效操作异常。

如果没有上述情况，按比例计算的结果 *pr* 不会受到溢出或下溢的影响。即使使用 **logB** 实现这些操作，也不应发出 divideByZero 异常信号。如果缩放因子的量级太大，无法用 *sf 格式*精确表示，那么这些操作应发出无效操作异常信号，默认情况下，*pr* 返回静态 NaN，如果*积分格式*是浮点格式，*sf* 也返回静态 NaN。当向量长度操作数为零时，*pr* 为 1，*sf* 为 +0，无一例外。

9.5 增强算术运算

语言标准应根据 9.1 为二进制格式 ormats 定义适合该语言的本条款操作。这些同质操作会产生一对*源格式*结果，表示为 *（源格式，sourceFormat）*。本标准只推荐二进制格式的运算，因为十进制格式运算的增强要求尚 未 确定。

本标准规定了在本子条款中的操作中使用的单一舍入方向，定义为 roundTiesTowardZero：应提供最接近无限精确结果的浮点数；如果无法表示无限精确结果的两个最接近的浮点数同样接近，则应提供幅度较小的浮点数。但是，幅度大于 $b^{emax} \times (b - \frac{1}{2} b^{1-p})$ 的无限精确结果应舍入到 ∞，符号不变；这里的 *emax* 和 *p 由*目标格式决定（见 3.3）。因此，roundTiesTowardZero 会将所有溢出（见 7.4）带到 ∞，中间结果的符号不变。 大小等于 $b^{emax} \times (b - \frac{1}{2} b^{1-p})$ 的无限精确结果将舍入为 $b^{emax} \times (b - b^{1-p})$，符号不变。

在本子条款的规范中，roundTiesTowardZero(*x* 运算 *y*) 表示使用 roundTiesTowardZero 对 *x* 运算 *y 进行舍入的*无限精确结果，其中运算为 +、- 或 ×。

— *（来源格式，来源格式）* **augmentedAddition**（*来源，来源*）

操作 **augmentedAddition**(*x*, *y*) 既能计算使用 roundTiesTowardZero 舍入*到源格式*的无限精确和 x + *y*，也能计算舍入后的 *x* + *y 为*有限时的舍入误差。如果 roundTiesTowardZero(*x* + *y*) 是一个有限的数，**augmentedAddition 将**产生 roundTiesTowardZero(*x* + *y*) 和 *x* + *y* - roundTiesTowardZero(*x* + *y)*，如果 *x* + *y* - roundTies- TowardZero(*x* + *y)* 等于零，则返回 roundTiesTowardZero(*x* + *y*) 的符号。

此运算的异常行为与使用 roundTiesTowardZero 的**加法运算**（参见 5.4.1）相同，但有以下附加说明。如果任何输入值为 NaN，则操作的两个结果均为 NaN（参见 6.2.3）。如果 roundTiesTowardZero(*x* + *y*) 是无限的，则产生的两个结果都是 roundTiesTowardZero(*x* + *y)* 的结果，并且操作信号类似于使用 roundTiesTowardZero 的**加法**（x , *y*）。如果运算发出无效运算异常信号，则两个输出结果都是相同的静态 NaN。如果 *x* + *y* - roundTiesTowardZero(*x* + *y*) 非零，且严格位于 $\pm b^{emin}$ 之间，则会发出下溢异常信号。根据默认的异常处理方式，只有当 roundTiesTowardZero(*x* + *y*) 发生溢出时，运算才会发出不精确信号；运算的次正常结果和零结果都是精确的。

— *（源格式，源格式）* **augmentedSubtraction**（*源，源*）

操作 **augmentedSubtraction**(*x*, *y*) 既能计算使用 roundTiesTowardZero 舍入*到源格式的*无限精确差值 *x* - *y*，也能计算舍入后的 *x* - *y 为*有限时的差值舍入误差。如果 roundTiesTowardZero(*x* - *y*) 是一个有限的数字，**augmentedSubtraction 会**产生 roundTiesTowardZero(*x* - *y*) 和 *x* - *y* - roundTiesToward- Zero(x - *y)*，如果 *x* - *y* - roundTiesTowardZero(*x* - *y*) 等于零，则返回 roundTiesTowardZero(*x* - *y*) 的符号。

此运算的异常行为与使用 roundTiesTowardZero 的**减法运算**（参见 5.4.1）相同，但有以下附加说明。如果任何输入值为 NaN，则操作的两个结果都传播一个 NaN（见 6.2.3）。如果 roundTiesTowardZero(x - y) 是无限的，则产生的两个结果都是 roundTiesTowardZero(x - y) 的结果，并且操作信号类似于使用 roundTiesTowardZero 的**减法**（x , y）。如果运算发出无效运算异常信号，则两个输出结果都是相同的静态 NaN。如果 x - y - roundTiesTowardZero(x - y) 非零且严格位于 $\pm b^{emin}$ 之间，则会发出下溢异常信号。在默认异常处理下，只有当 roundTiesTowardZero(x - y) 发生溢出时，运算才会发出不精确信号；运算的次正常结果和零结果都是精确的。

— *(源格式，源格式)* **增强乘法**（*源，源*）

增强乘法运算（*x*, *y*）既能计算使用 roundTiesTowardZero 舍入*到源格式的*无限精确乘积 *x* × *y*，也能计算舍入后的 *x* × *y* 为有限时的乘积舍入误差，具体条件如下。如果 roundTiesTowardZero(*x* × *y*) 是有限个数，并且 *x* × *y* - roundTiesTowardZero(*x* × *y*) 可以在源格式中精确地表示为有限个数、如果 *x* × *y* - roundTiesTowardZero(*x* × y) 等于零，则返回 roundTiesTowardZero(*x* × *y*) 的符号。

此运算的异常行为与使用 roundTiesTowardZero 的**乘法运算**（参见 5.4.1）相同，但有以下附加说明。如果输入值为 NaN，则运算结果均为 NaN（参见 6.2.3）。如果 roundTiesTowardZero(*x* × *y*) 是无限的，则产生的两个结果都是 roundTiesTowardZero(*x* × *y*) *的结果*。如果运算发出无效运算异常信号，则两个输出结果都是相同的静态 NaN。 如果 *x* × *y* - roundTiesTowardZero(*x* × *y*) 非零，且严格位于 $\pm b^{emin}$ 之间，则应发出下溢异常信 号。如果 *x* × *y* - roundTiesTowardZero(*x* × *y*) 是有限的非零数字，且无法在 *sourceFormat 中*准确表示（因为某些非零数字严格位于±b 之间），则应发出下溢异常信号。
$\pm b^{(emin-p+1)}$），结果是 roundTiesTowardZero(*x* × *y*) 和使用 roundTiesTowardZero 将无限精确的结果 *x* × *y* - roundTiesTowardZero(*x* × *y*) 四舍五入到 *sourceFormat*。在这种情况下，默认异常处理会引发下溢标志并发出不精确异常信号。否则，在默认异常处理下，只有当 roundTiesTowardZero(*x* × *y*) 运算溢出时，才会发出不精确异常信号。

9.6 最小和最大操作

语言标准应为所有支持的算术格式定义以下 同质通用计算运算：

— *源格式*__最小值__（源，*源*） 源*格式*__最小数__（*源，源*） 源格式__最大值（源，源）__ *源格式*__最大数__（*源，源*）

根据 6.2，如果 *x* < *y*，则__最小值__（*x*, y）为 *x*；如果 *y* < *x*，则__最小值（__*y*）为 *y*；如果任一操作数为 NaN，则__最小值（__NaN）为静 NaN。否则（*即*当 *x* = *y* 且符号相同时），要么是 *x*，要么是 *y*。

如果 *x* < *y*，则返回 *x*；如果 *y* < *x*，则返回 y；如果一个操作数是数字，另一个操作数是 NaN，则返回数字。对于此操作，-0 小于 +0。如果 *x* = *y* 且符号相同，则返回 *x* 或 *y*。如果两个操作数都是 NaN，则根据 6.2 返回静态 NaN。如果其中一个操作数是信号 NaN，则会发出无效操作异常信号，但除非两个操作数都是 NaN，否则信号 NaN 将被忽略，并且不会转换为静态 NaN，如 6.2 所述。
6.2 用于其他操作。

根据 6.2，如果 $x > y$，则 **maximum**(x, y) 为 x；如果 $y > x$，则 maximum(y) 为 y；如果任何一个操作数为 NaN，则 maximum(x, y) 为静态 NaN。否则（*即*当 $x = y$ 且符号相同时），要么是 x，要么是 y。

如果 $x > y$，则返回 x；如果 $y > x$，则返回 y；*如果*一个操作数是数字，另一个操作数是 NaN，则返回数字。对于此操作，+0 比较大于-0。如果 $x = y$ 且符号相同，则返回 x 或 y。如果两个操作数都是 NaN，则根据 6.2 返回静态 NaN。如果其中一个操作数是信号 NaN，则会发出无效操作异常信号，但除非两个操作数都是 NaN，否则信号 NaN 将被忽略，也不会转换为静态 NaN，如 6.2 所述。

6.2 用于其他操作。

— *sourceFormat* **minimumMagnitude**(*源，源*) *sourceFormat*
minimumMagnitudeNumber(*源，源*) *sourceFormat*
maximumMagnitude(源，*源*) *sourceFormat*
maximumMagnitudeNumber(*源，源*)

minimumMagnitude(*x, y*) 如果 | x| < | *y*| *则*为 x，如果 | *y*| < | *x*| *则*为 y，否则为 **minimum**(*x,y*)。 **minimumMagnitudeNumber**(*x, y)* 如果 | x| < | *y*| *则为 x*，如果 | *y*| < | *x*| *则为 y*，否则为 **minimumNumber**(*x*,y)。**maximumMagnitude**(*x, y*) 如果 | x| > | *y*| *则为 x*，如果 | *y*| > | *x*| 则为 y，否则为 **maximum**(*x,y*)。 **maximumMagnitudeNumber**(*x, y*) 如果 | x| > | y| 则为 *x*，如果 | *y*| > | *x*| 则为 *y*，否则为 **maximumNumber**(*x,y*)。

如果结果为 *x*，则首选指数为 Q(*x*)；如果结果为 *y*，则首选指数为 Q(*y*)。

注意--当 *x* 和 *y* 是十进制浮点数中同一队列的不同表示时，不同实现的结果量子可能不同。

9.7 NaN 有效载荷操作

语言标准应定义 以下的同质静音计算操作，以提供对有效载荷的通用访问。这些操作没有例外信号。

— *sourceFormat* **getPayload**(*source*)

如果源操作数为 NaN，则结果为非负浮点整数的有效载荷。如果源操作数不是 NaN，则结果为-1。

首选指数为 0。

— *sourceFormat* **setPayload**(*source*)

如果源操作数是一个非负浮点整数，且其值是该操作可接受的有效载荷的实现定义集之一，则结果是一个带有该有效载荷的静态 NaN。否则，结果为 +0，首选指数为 0。

— *sourceFormat* **setPayloadSignaling**(*source*)

如果源操作数是一个非负浮点整数，且其值是该操作可接受的有效载荷的实现定义集之一，则结果是一个带有该有效载荷的信号 NaN。否则，结果为 +0，首选指数为 0。

注意 - 实现可能会限制可设置的有效载荷。因此，**getPayload** 返回的值可能不是 **setPayload** 或 **setPayloadSignaling** 可接受的操作数。(程序可通过对 **setPayload** 或 **setPayloadSignaling** 的结果应用 **isNaN** 操作进行检查）。

10. 表达评估

10.1 表达式评估规则

本标准第 5 条规定了单一算术运算的结果。每个运算结果都有一个显式或隐式的目的地。对于数字运算结果，需要进行一次四舍五入，以便将精确结果纳入目的格式。这种结果具有可重复性，即在相同属性下对相同操作数进行相同的运算，在所有语言的所有符合标准的实现中产生相同的结果。

程序语言标准可能会定义表达式的语法，这些表达式结合了本标准的一个或多个操作，产生的结果符合一个显式或隐式的最终目标。当一个已声明格式的变量是最终目的地时，如在对变量进行格式转换时，该变量的声明格式会影响其四舍五入。隐式终点的格式，或没有声明格式的显式终点的格式，由语言标准表达式的评估规则来定义。

编程语言标准规定了一种或多种表达式求值规则。表达式求值规则包括：

— 操作的评估顺序。

— 隐含中间结果的格式。

— 当分配到明确目的地时，四舍五入一次；当分配到明确目的地时，四舍五入两次（见下文）。

— 通用和非通用操作的参数格式。

— 一般业务成果的格式。

语言标准可能允许用户为表达式评估选择不同的语言定义规则，也可能允许实现定义额外的表达式评估规则并指定默认的表达式评估规则；在这种情况下，语言标准应定义如下所示的首选宽度属性。

有些语言标准会隐式地将浮点运算的操作数转换为通用格式。通常，操作数会被提升为操作数中最宽的格式或首选宽度（preferredWidth）格式。但是，如果通用格式不是操作数格式的超集，那么将操作数转换为通用格式可能无法保留操作数的值。例子包括

— 将定点或整数操作数转换为精度较低的浮点格式。

— 将浮点操作数从一个弧度转换为另一个弧度。

— 将浮点操作数转换为弧度相同但范围或精度较低的格式。

语言标准应禁止或警告会导致隐式转换从而改变操作数值的混合格式操作。

10.2 赋值、参数和函数值

许多表达式的最后一个操作是对显式最终目的变量赋值。作为表达式求值规则的 一部分，语言标准应规定什么情况下 倒数第二个操作最多四舍五入一次到显式最终目的格式，什么情况下最多四舍五入两次，先四舍五入到隐式中间格式，再四舍五入到显式最终目的格式。后一种情况与第 5 条中

的任何单一操作都不对应，而是意味着两个此类操作的序列。

无论在哪种情况下，在对后续表达式进行求值时，实现都不得使用分配给变量的更大前导值来代替分配给变量的存储值。

当函数的作用域中有明确声明的形式参数类型时，必要时应将实际参数取整为这些明确声明的类型。如果函数的作用域中没有明确声明的形式参数类型，或者是一个泛型操作，则应根据语言定义的规则对实际参数进行四舍五入。

当函数明确声明了返回值的类型时，返回值应按明确声明的类型四舍五入。当函数的返回值类型由语言标准规则隐式定义时，返回值应按隐式定义的类型四舍五入。

10.3 用于表达式评估的首选宽度属性

定义泛型运算的语言标准，在特定弧度中支持不止 一种算术格式，并定义或允许不止一种方法将该语言中的表达式映射到本标准的运算中，应为每个此类弧度定义 preferredWidth 属性。preferredWidth 属性由用户显式启用，并指定表达式求值的一个方面：语言定义的泛型运算的隐式目标格式。

在该标准中，返回数值结果的计算操作首先产生一个未四舍五入的结果，即无限精确的精确数。然后将未舍入结果舍入为目标格式。对于某些语言定义的通用操作，目标格式由操作数的宽度和当前有效的首选宽度属性暗示。

以下 preferredWidth 属性可禁用或启用表达式中的加宽操作，表达式可能简单如 $z = x + y$，也可能涉及对不同格式操作数的多个操作。

— **preferredWidthNone** 属性：每个此类语言标准都应定义并要求实现为用户提供为数据块指定 preferredWidthNone 属性的方法。目标宽度是操作数宽度的最大值：具有浮点操作数和相同弧度结果的通用运算将结果取整为操作数中最宽的格式。

— **首选宽度格式**属性：每个此类语言标准都应定义并要求实现为用户提供为数据块指定首选宽度格式属性的方法。目标宽度通常是首选 WidthFormat 宽度和操作数宽度的最大值：受影响的操作具有浮点操作数和结果（相同弧度），并将结果取整为操作数和首选 WidthFormat 中最宽的格式。受影响的操作不会缩小其操作数，因为操作数可能是加宽的表达式。preferredWidthFormat 只影响该格式弧度内的结果。

preferredWidth 属性不影响最终舍入到已声明格式的显式目标的宽度，该目标始终舍入为该格式。preferredWidth 属性不影响表达式中的显式格式转换，它们始终舍入为转换指定的格式。

10.4 字面意义和改变价值的优化

语言标准应定义程序源代码 ode 的字面意义，以便转换为本标准的操作。字面意义包含在操作顺序（由优先规则和括号控制）、操作的目标格式（隐式和显式）以及属性或动态模式规范的范围中。语言标准应要求语言实现在默认情况下，在未启用优化和未启用备用异常处理的情况下，保留源代码的字面意义。这意味着语言实现不会执行改变数值结果或标志的数值变换。

语言的实现保留了源代码的字面意义，例如

— 保留显式序列或括号中定义的操作顺序。

— 保留显性和隐性目的地的格式。

— 在浮点表达式中应用实数的属性时，只能保留数值结果和标志：

  — 换元法只适用于**加法**和**乘法**等运算，对于这些运算，结果的数值或结果的表示都不取决于操作数的顺序。

  — 只有当联立或分配律能保留数字结果时才应用，并竖起旗帜。

  — 只有在保持数字结果和标志的情况下，才应用同一律（$0 + x$ 和 $1 \times x$）。

— 相对于修改属性或动态模式的操作，保留受属性或动态模式影响的操作顺序；大多数计算操作都会受 到 属性或动态模式的影响。

— 相对于测试或保存状态标志的操作，保留恢复、降低或提升状态标志的操作顺序；大多数计算操作都能提升状态标志。

除其他外，以下改变值的转换可保留源代码的字面意义：

— 当 x 不为零且不是信号 NaN 时，应用同一性质 0 + x，且结果具有与 x 相同的指数。

— 当 x 不是信号 NaN 且结果与 x 具有相同指数时，应用同一性质 $1 \times x$。

— 更改静态 NaN 的有效载荷或符号位。

— 改变不同旗帜的升旗顺序。

— 当旗帜至少升起一次时，更改升旗次数。

语言标准还应定义并要求实现者提供允许或不允许对代码块单独或集体进行改变值的优化的属性。这些优化可能包括但不限于

— 应用联立或分配律。

— 由**乘法**和**加法**合成**融合的乘加**运算。

— 由一个操作合成一个 *formatOf 操作，*并对操 作 结果进行转换。

— 在表达式评估中使用更广泛的中间结果。

当数值或状态标志的相应变化可以接受时，程序员可以允许这些优化。

11. 可重复的浮点运算结果

可重现的浮点数值和状态标志 r 可重现的运算结果，具有可重现的属性，在可重现的格式上运行，每种语言的定义如下：

— 可重复操作是第 5 条中描述的操作之一，或者是 9.2、9.3、9.5 或 9.6 中支持的操作。

- 可重现属性是语言标准要求所有实施都必须具备的属性（见第 4 条）。

- 可重现状态标志是由无效运算、除以零或溢出异常（见 7.2、7.3 和 7.4）引发的标志。

— 可复制格式是一种算术格式，也是一种交换格式（见第 3 条）。

程序可以可靠地转化为对可重现格式进行可重现操作的显式或隐式序列，并产生可重现的结果。也就是说，会产生相同的数字和可重现的状态标志结果。

可重现的结果需要语言标准、语言处理器和用户的合作。语言标准应支持可重复编程。任何支持可重现编程的符合标准的语言标准都应

— 支持可重现结果属性。

— 支持可复制格式，为该格式提供所有可复制操作。

— 提供对该语言支持的可复制格式明确或隐含地指定任何可复制操作序列的方法。

并应明确定义：

— 哪种语言元素对应哪种支持的可复制格式。

— 如何在语言中指定每种支持的可复制格式上的每种可复制操作。

— 一个或多个无歧义的表达式评估规则，应可在该语言标准的所有符合性实施中供用户选择，而不将任何方面推迟到实施中。如果语言标准允许对本标准中的操作序列进行多种解释，则应提供一种方法来指定对该序列的明确评估（如通过规定性括号）。

— 4.1 中描述的可再现结果属性，其值表示何时需要或不需要可再现结果。语言标准规定了默认值。当用户选择需要可重复结果时：

  — 执行行为应保留源代码的字面含义（见 10.4）。

  - 与外部十进制字符序列之间的转换不应限制最大支持精度 H（见 5.12.2）。

  — 语言处 理 程 序 应说明在哪些情况下不能保证影响浮点运算结果的操作的可重复性。

  — 只能使用默认异常处理（见第 7 条）。

  如果一种语言支持单独编译的例程（*例如*常用函数的库例程），就必须有某种机制来确保行为的可重现性。

在所有支持这种语言标准的平台上，用户只需编写以下程序，即可获得相同的浮点数值和可重复的状态标志结果：

— 使用所需的 "可重复结果 "属性。

— 只使用可重复格式的浮点格式。

— 只明确使用或通过表达式隐含使用可重复的浮点运算。

— 仅使用四舍五入和首选宽度的所有实现中所需的属性。

— 只使用语言标准的所有实现所支持的整数和非浮点格式，并且只能以避免提示整数运算异常和其他实现定义的异常的方式使用。

以及

— 不要使用改变值的优化方法（见 10.4）。

— 不要超过系统限制。

— 请勿使用 **fusedMultiplyAdd**(0, ∞, *c)* 或 **fusedMultiplyAdd**(∞, 0, *c*)，其中 *c* 为静态 NaN。

— 不要使用信号 NaN。

— 的最小值和最大值运算时，不要依赖于十进制结果的量。
当 *x* 和 *y* 相等时为 9.6。

— 不要依赖静态 NaN 传播、有效载荷或符号位。

— 不要依赖下溢和不精确异常及标志。

— 不要依赖表 9.1 中十进制格式运算结果的量。

— 不要依赖编码。

附件 A（资料性） 参

考书目

参考书目是提供额外或有用材料的资源，但不需要理解或用于实施本标准。参考这些资源仅供参考之用。

[B1] Ahrens, P., Nguyen, H. D., and Demmel, J. "Efficient Reproducible Floating Point Summation and BLAS", EECS Department, UC Berkeley, Technical Report No. UCB/EECS-2016-121, June 18, 2016.[1]

[B2] ANSI INCITS 4-1986 Information Systems -Coded Character Sets-7-bit American National Standard Code for Information Interchange (7-Bit ASCII).[2]

[B3] Boldo, S., and Muller, J.-M., "Some functions computable with a fused-mac", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic,* ISBN 0-7695-2366-8, pp.[3]

[B4] Bruguera, J. D., and Lang, T., "Floating-point Fused Multiply-Add：Reduced Latency for Floating-Point Addition"，*Proceedings of the 17th IEEE Symposium on Computer Arithmetic*，ISBN 0-7695-2366-8、
pp.42-51, IEEE Computer Society, 2005. doi:10.1109/ARITH.2005.22

[B5] Coonen, J. T., "Contributions to a Proposed Standard for Binary Floating-point Arithmetic", PhD thesis, University of California, Berkeley, 1984.

[B6] Cowlishaw, M. F., "Densely Packed Decimal Encoding", IEE *Proceedings - Computers and Digital Techniques*, Vol. 149 #3, ISSN 1350-2387, pp.

[B7] Cowlishaw, M. F., "Decimal Floating-Point：计算机算法"，第 *16 届 IEEE 计算机算术研讨会论文集*，ISBN 0-7695-1894-X，第 104-111 页，IEEE 计算机协会，2003 年。doi:10.1109/ARITH.2003.1207666

[B8] Demmel, J. W., and Li., X., "Faster numerical algorithms via exception handling", *IEEE Transactions on Computers,* 43(8): pp.

[B9] de Dinechin, F., Ershov, A., and Gast, N., "Towards the post-ultimate libm", *Proceedings of the 17th IEEE Symposium on Computer Arithmetic,* ISBN 0-7695-2366-8, pp.

[B10] de Dinechin, F., Lauter, C. Q., and Muller, J.-M., "Fast and correctly rounded logarithms in double-precision", *RAIRO - Theoretical Informatics and Applications*, 41, pp.

[B11] Higham, N. J., *Accuracy and Stability of Numerical Algorithms,* 2nd edition, ISBN 0-89871-521-0, Society for Industrial and Applied Mathematics (SIAM), 2002.

[B12] IEC 60559:1989，微处理器系统的二进制浮点运算（原名 IEC 559:1989）。[4]

[B13] ISO/IEC 9899:2018(E)编程语言-C（C17）。[5]

[B14] ISO/IEC TS 18661-1:2014，信息技术-编程语言、其环境和系统软件接口-C 的浮点扩展-第 1 部分：二进制浮点运算。

[1]见 http://www.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-121.html。

[2]ANSI 出版物可从美国国家标准学会 (http://www.ansi.org/) 获取。

[3]IEEE 出版物可从电气与电子工程师学会（The Institute of Electrical and Electronics Engineers）获取，地址为 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/)。

[4]IEC 出版物可从国际电工委员会 (http://www.iec.ch/) 获取。在美国，也可从美国国家标准学会 (http://www.ansi.org) 获取 IEC 出版物。

[5]ISO/IEC 出版物可从 ISO 中央秘书处 (http://www.iso.org/) 获取。在美国，ISO/IEC 出版物可从美国国家标准学会 (http://www.ansi.org/) 获取。

[B15] ISO/IEC TS 18661-2:2015，信息技术-编程语言、其环境和系统软件接口-C 的浮点扩展-第 2 部分：十进制浮点运算。

[B16] ISO/IEC TS 18661-3:2015，信息技术-编程语言、其环境和系统软件接口-C 的浮点扩展-第 3 部分：交换和扩展类型。

[B17] ISO/IEC TS 18661-4:2015，信息技术-编程语言、其环境和系统软件接口-C 的浮点扩展-第 4 部分：补充函数。

[B18] ISO/IEC TS 18661-5:2016，信息技术-编程语言、其环境和系统软件接口-C 的浮点扩展-第 5 部分：补充属性。

[B19] Kahan, W., "Branch Cuts for Complex Elementary Functions, or Much Ado About Nothing's Sign Bit", *The State of the Art in Numerical Analysis,* (Eds. Iserles and Powell), Clarendon Press, Oxford, 1987.

[B20] Lefèvre, V., "New results on the distance between a segment and Z^2 . Application to the exact rounding", *Proceedings of 17th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-2366-8、pp.68-75, IEEE Computer Society, 2005. doi:10.1109/ARITH.2005.32

[B21] Lefèvre, V., and Muller, J.-M., "Worst cases for correct rounding of the elementary functions in double precision", *Proceedings of the 15th IEEE Symposium on Computer Arithmetic,* ISBN 0-7695-1150-3, pp.

[B22] Markstein, P., *IA-64 and Elementary Functions：速度与精度》*，ISBN 0-13-018348-2，Prentice Hall，Upper Saddle River，NJ，2000。

[B23] Montoye, R. K., Hokenek, E., and Runyon, S. L., "Design of the IBM RISC System/6000 floating point execution unit", *IBM Journal of Research and Development,* 34(1), pp.

[B24] Muller, J.-M., *Elementary Functions：算法与实现》*，2nd 版，第 10 章，ISBN 0-8176-4372-9，Birkhäuser，2006 年。

[B25] Overton, M. L., *Numerical Computing with IEEE Floating Point Arithmetic,* ISBN 0-89871-571-7, Society for Industrial and Applied Mathematics (SIAM), 2001.

[B26] Priest, D. M., "Algorithms for arbitrary precision floating point arithmetic".*第 10 届 IEEE 计算机算术研讨会论文集》*，ISBN 0-8186-9151-4，第 132-143 页，IEEE 计算机协会，1991 年。

[B27] Sayed, W. S., and Fahmy, H. A. H., "What are the Correct Results for the Special Values of the Operands of the Power Operation?", *ACM Transactions on Mathematical Software*, ISSN:0098-3500, 42(2), pp.

[B28] Schwarz, E. M., Schmookler, M. S., and Trong, S. D., "Hardware Implementations of Denormalized Numbers", *Proceedings of the 16th IEEE Symposium on Computer Arithmetic*, ISBN 0-7695-1894-X、pp.70-78, IEEE Computer Society, 2003. doi:10.1109/ARITH.2003.1207662

[B29] Stehlé, D., Lefèvre, V., and Zimmermann, P., "Searching Worst Cases of a One-Variable Function Using Lattice Reduction", *IEEE Transactions on Computers,* 54(3), pp.

[B30] 统一码标准，5.0 版，统一码联盟，Addison-Wesley Professional，2006 年 10 月 27 日，ISBN 0-321-48091-0。

附件 B

(资料性)

程序调试支持

B.1 概述

本标准的实施对性能和可调试性（调试能力）等特性的优先级各不相同。本附件介绍了一些编程环境特性，这些特性应由旨在支持最大调试能力的实现来提供。在某些实现中，与完全优化的代码相比，启用其中某些功能可能会在性能上付出很大代价。

调试包括查找数值敏感性或异常的根源和原因，查找编程错误（如访问未初始化的存储）（这些错误只表现为不正确的数值结果），以及测试发现问题的候选修复方法。

B.2 数值敏感性

调试器应能改变子程序内部四舍五入或异常处理的属性，即使这些子程序的源代码不可用 ；动态模式可用于此目的。例如，在执行过程中改变四舍五入的方向或精度可能有助于识别对四舍五入异常敏感的子程序，无论是由于所解决问题的条件不佳、所选算法的不稳定性，还是算法的设计只适用于一种四舍五入方向属性。最终目标是确定数值误差的责任，特别是在单独编译的子程序中。为实现这一终极目标而选择的方法是促进制作可重现的小型测试用例，以诱发出乎意料的行为。

B.3 数字例外

调试器应能在特定子程序或其调用的特定子程序中出现预先指定的异常信号时，检测并暂停正在调试的程序。为避免混淆，暂停应在导致暂停的事件发生后不久进行。暂停后，调试器应能继续执行，就好像异常已由指定的备用处理程序或默认处理程序处理过一样。暂停与异常相关联，但可能与定义明确的源代码语句边界无关；坚持要求暂停与源代码相关联的精确性可能会妨碍优化。

调试器应能提升和降低状态标志。

调试器应能检查子程序或整个程序执行结束时留下的所有状态标志。应通过将每个状态标志作为其来源和历史详细记录的引用来增强这些功能。在默认情况下，即使是预先调试好的子程序，也至少应在状态标志中插入对其名称的引用，并在无效操作的浮点运算结果产生的任何新的静态 NaN 的有效载荷中插入对其名称的引用。这些引用说明了异常或 NaN 的来源。

调试器应能维护静默 NaN 的历史记录表，并使用 NaN 有效载荷为表编制索引。

调试器应能在每次浮点运算时暂停，而不中断程序处理异常的逻辑。调试器应尽可能显示与机器指令相对应的源代码行。

出于各种目的，信号 NaN 可被用作对包含由异常历史、更宽指数或更宽符号扩展的数值记录的引用。因此，调试程序应能使通常无声的比特运算（如**否定**、**abs** 和 **copySign**）检测到信号 NaN。此外，信令 NaN 的信令属性应能启用或

在全局或特定范围内禁用，而不会干扰或影响程序默认或交替处理其他无效操作异常的逻辑。

B.4 编程错误

调试器应能将某些或所有 NaN 定义为信号 NaN，每次使用时都会发出异常信号。在具有非算术生成的多余比特模式的格式中，例如十进制格式中的非规范符号字段，调试器应能对包含此类比特模式的数据启用信号 NaN 行为。

调试器应该能够将堆和公共空间等未初始化的存储空间和变量设置为特定的位模式，如 all-zeros 或 all-one，这有助于发现这些变量的无意使用；如果这些使用涉及到同一物理存储空间的多个别名，则可能会对静态分析造成困难。

在调试环境中，所有浮点变量，包括每次在堆栈中分配的自动变量，都被初始化为NaNs信号，并引用符号表中描述其源于源程序的条目。在调试器能够在预设异常信号或标志出现时暂停执行的环境中，这种初始化尤其有用。

附件 C

(资料性)

业务清单

convertToIntegerExactTowardNegative 34, 40
convertToIntegerExactTowardPositive 34, 40
腹肌 35、38、50、79
acos 60、61
acosh 60-62
acosPi 60、61
33 号、48 号、52 号、68 号、74 号**增补件**
asin 60, 61
第 60-62 号决议
asinPi 60、61
atan 60-62
atan2 60, 61
atan2Pi 60-62
atanh 60-62
阿坦皮 60、61
增强添加 68
增强乘法 69
增强减法 68
第 38 **类**
compareQuietEqual 37, 43
compareQuietGreater 37
compareQuietGreater 44
compareQuietGreaterEqual 37, 44
compareQuietGreaterUnordered 37, 44
compareQuietLess 37, 44
compareQuietLessEqual 37, 44
compareQuietLessUnordered 37, 44
compareQuietNotEqual 37, 43
compareQuietNotGreater 37, 44
compareQuietNotLess 37, 44
compareQuietOrdered 37, 44
compareQuietUnordered 37, 44
compareSignalingEqual 37, 44
compareSignalingGreater 37
比较信号强度 43
compareSignalingGreaterEqual 37, 43
compareSignalingGreaterUnordered 37, 43
compareSignalingLess 37, 43
compareSignalingLessEqual 37, 43
compareSignalingLessUnordered 37, 43
compareSignalingNotEqual 37, 44
compareSignalingNotGreater 37, 43
compareSignalingNotLess 37, 43
化合物 59、62、64
convertFormat 34
convertFromDecimalCharacter 34
convertFromHexCharacter 34
convertFromInt 33
convertToDecimalCharacter 34
convertToHexCharacter 34
convertToIntegerExactTiesToAway 34, 40
convertToIntegerExactTiesToEven 34, 40

convertToIntegerExactTowardZero 34, 40
convertToIntegerTiesToAway 34, 40
convertToIntegerTiesToEven 34, 40
convertToIntegerTowardNegative 34, 40
convertToIntegerTowardPositive 34, 40
convertToIntegerTowardZero 34, 40
复制 34-36、50
copySign 35、50、79
余数 60、61
复数 60-62
cosPi 60、61
解码二进制 36
DecodeDecimal 36
默认模式 65
第 33、48、52、53 **分部**
点 66
encodeBinary 36
encodeDecimal 36
EXP 59、61、62
exp10 59、61、62
EXP10M1 59、61、62
exp2 59、61、62
exp2m1 59、61、62
expm1 59、61、62
fusedMultiplyAdd 14、33、48、50、52、74、76
getBinaryRoundingDirection 65
getDecimalRoundingDirection 65
getPayload 71
假设 59、60、62、64
is754version1985 37
is754version2008 37
is754version2019 37
isCanonical 38
isFinite 38
isInfinite 38
isNaN 38、71
isNormal 38
isSignaling 38
isSignMinus 38, 50
isSubnormal 38
isZero 38
日志 59、60、62
log10 59, 62
log10p1 59、61、62
log2 23, 59, 62
log2 23
log2p1 59、61、62
对数 B 29、32、52、53、67
logp1 59、61、62
lowerFlags 39
最大值 69、70
最大振幅 70

最大振幅数 70
最大数 69, 70
最低 69、70
最小幅度 70
最小振幅数 70
最小编号 69, 70
乘法 33、48、52、69、74
否定 35、50、79
下一个向下 31
下一页 31
第 59、63、64 段
powd 63
第 59、62-64 **页**
POWR 59、63、64
量化 30、32、50、52
量子 32
小数点后38**位**
raiseFlags 39, 51
余下部分 31、48、52
restoreFlags 39, 51
还原模式 65
根 59、62、64
根号 62
roundToIntegralExact 30、31、41、50
roundToIntegralTiesToAway 31, 41
roundToIntegralTiesToEven 31, 41
roundToIntegralTowardNegative 31, 41
roundToIntegralTowardPositive 31, 41
roundToIntegralTowardZero 31, 41
rSqrt 59、62、64
相同量子 39
saveAllFlags 39
保存模式 65
比额表 B 29、32、66、67
scaledProd 66, 67
缩放产品差值 67
缩放产品总和 66, 67
setBinaryRoundingDirection 65
设置十进制滚圆方向 65
setPayload 71
setPayloadSignaling 71
罪 60、61
sinh 60-62
sinPi 60、61
方根 33、48、50、52、62
减法 33、35、48、52、68
总和 66
sumAbs 66
和平方 66
黄褐色 60、61
第 60-62 页
tanPi 60、61
testFlags 39
testSavedFlags 39
订单总数 38、42、50
订单总数

共识

我们建造它。

与我们联系

Facebook: https://www.facebook.com/ieeesa

Twitter：@ieeesa

Linkedln: http://www.linkedin.com/groups/IEEESA-Official-IEEE-Standards-Association-1791118

IEEE-SA Standards Insight 博客：http://standardsinsight.com

YouTube：IEEE-SA 频道