ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Đinh Đức Tài

23TNT1 - 23122013

BÁO CÁO BONUS 2

Chủ đề: Câu hỏi mở rộng về xử lý số

Môn: Hệ thống máy tính

GV: Vũ Thị Mỹ Hằng

TP. Hồ Chí Minh - 2025

MỤC LỤC

Mục lục .....

Chương 1. Nội dung .....

1.1. Giới thiệu .....

1.2. Overflow, underflow .....

1.2.1. Overflow (tràn số) .....

1.2.2. Thiếu số (underflow) .....

1.3. Cộng, trừ, nhân, chia trên số thực .....

1.3.1. Giới thiệu .....

1.3.2. Phép cộng, phép trừ .....

1.3.3. Phép nhân .....

1.3.4. Phép chia .....

1.4. Quy tắc làm tròn .....

1.4.1. Các kiểu làm tròn .....

1.4.2. Các chiến lược để giảm thiểu lỗi làm tròn .....

1.5. NaN và nguyên tắc phát sinh .....

1.6. Quiet NaN và Signaling NaN .....

1.6.1. Quiet NaN (qNaN) .....

1.6.2. Signaling NaN (sNaN) .....

Chương 2. Kết luận .....

2.1. Kết quả đạt được .....

2.2. Khuyến nghị .....

Tài liệu tham khảo .....

Chương 1

Nội dung

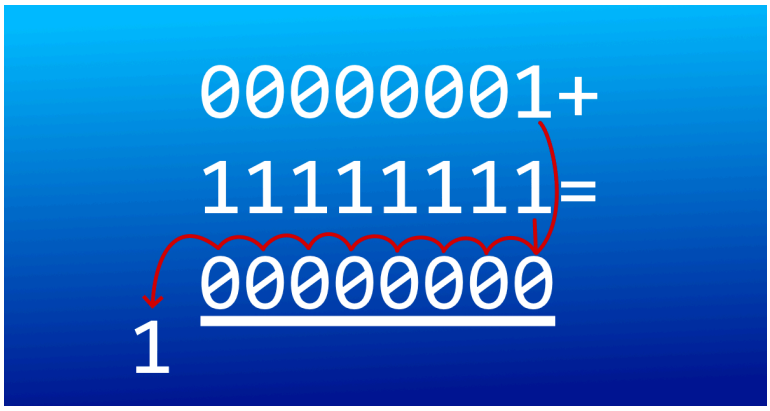
1.1. Giới thiệu

Trong lĩnh vực số học máy tính, việc biểu diễn và thao tác với các con số đặt ra nhiều thách thức do giới hạn về bộ nhớ và độ chính xác. Báo cáo này đi sâu vào các khái niệm cơ bản như tràn số (overflow), thiếu số (underflow), cách các phép toán trên số thực được thực hiện, các quy tắc làm tròn số phổ biến, cũng như giá trị đặc biệt NaN (Not a Number) và sự khác biệt giữa Quiet NaN và Signaling NaN. Hiểu rõ những khái niệm này là rất quan trọng để phát triển các ứng dụng phần mềm đáng tin cậy và chính xác, đặc biệt trong các lĩnh vực như tính toán khoa học, mô hình hóa tài chính và xử lý dữ liệu lớn.

1.2. Overflow, underflow

1.2.1. Overflow (tràn số)

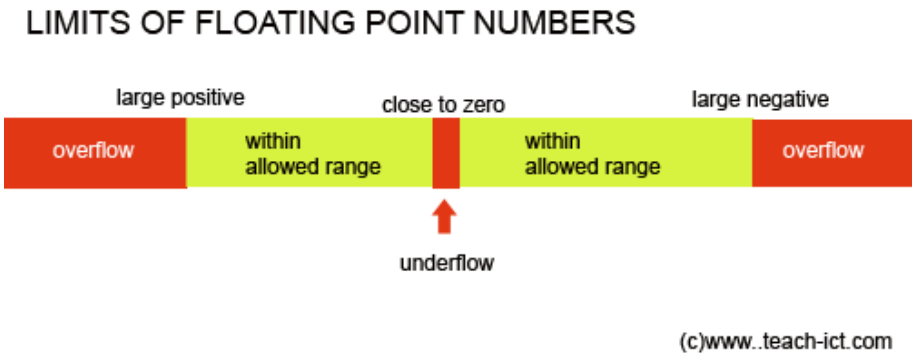
- **Tràn số (overflow)** [1] xảy ra khi kết quả của một phép toán số học vượt quá phạm vi giá trị mà một biến có thể chứa. Điều này có thể xảy ra với cả số nguyên và số thực. Đối với số nguyên, khi một phép toán tạo ra một giá trị lớn hơn giá trị lớn nhất hoặc nhỏ hơn giá trị nhỏ nhất mà kiểu dữ liệu có thể biểu diễn, thì xảy ra tràn số.
- Một hậu quả phổ biến của tràn số nguyên là giá trị bị "quấn vòng" (wrap around) trở lại phạm vi biểu diễn, thường là modulo lũy thừa của 2. Ví dụ, trong phép cộng số nguyên 8-bit có dấu, $127 + 1$ có thể dẫn đến kết quả là -128. Các ngôn ngữ lập trình khác nhau có thể xử lý tràn số khác nhau, từ việc bỏ qua (như trong C) đến việc gây ra ngoại lệ (như trong MIPS hoặc tùy chọn trong Java).[2] Tràn số có thể dẫn đến các lỗi chương trình khó phát hiện và gỡ lỗi, đặc biệt khi chúng chỉ xảy ra với các bộ dữ liệu đầu vào rất lớn.



Hình 1.1 — Một ví dụ về tràn số (overflow)

1.2.2. Thiếu số (underflow)

- **Thiếu số (underflow)** [3] thường xảy ra trong các phép toán trên số thực (floating-point) khi kết quả của một phép tính có giá trị tuyệt đối quá nhỏ để có thể biểu diễn một cách chính xác trong bộ nhớ máy tính. Thiếu số có thể được xem như là tràn số âm của phần mũ trong biểu diễn số thực. Khi một kết quả nhỏ hơn giá trị nhỏ nhất có thể biểu diễn dưới dạng số thực thông thường, nó có thể trở thành một số dưới chuẩn (subnormal number) hoặc bị làm tròn về không (flush to zero).
- Chuẩn IEEE 754 ưu tiên phương pháp thiếu số dần (gradual underflow) bằng cách sử dụng các số dưới chuẩn để lấp đầy khoảng trống giữa số dương nhỏ nhất thông thường và số không, giúp duy trì độ chính xác tốt hơn cho các phép tính liên quan đến số rất nhỏ. [4] Tuy nhiên, các số dưới chuẩn có thể có ít bit độ chính xác hơn so với các số thông thường.



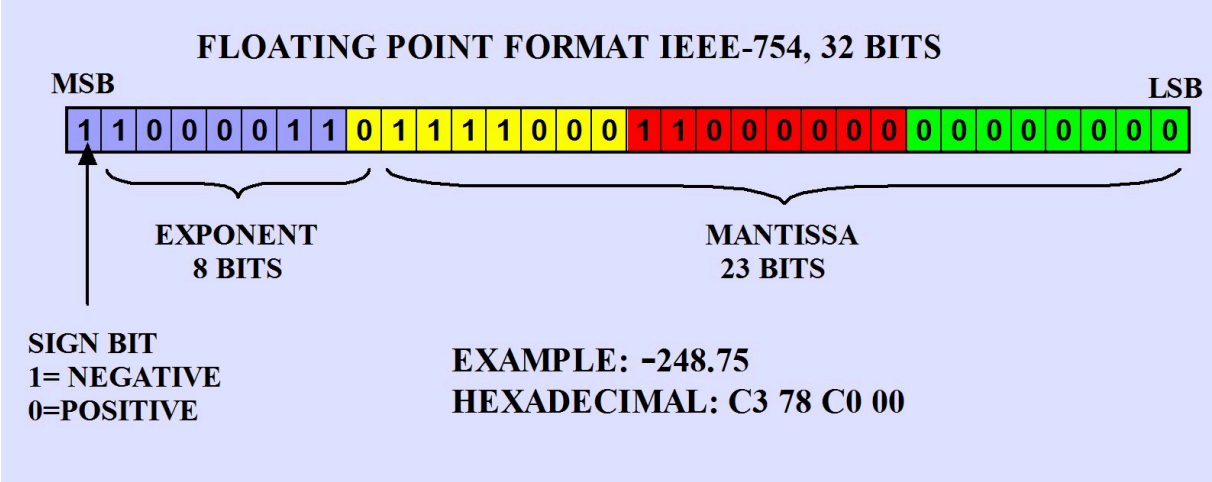
Hình 1.2 — Minh họa về thiếu số (underflow)

Việc hiểu rõ giới hạn của phạm vi có thể biểu diễn là rất quan trọng để tránh những vấn đề này.

1.3. Cộng, trừ, nhân, chia trên số thực

1.3.1. Giới thiệu

Số thực trong máy tính được biểu diễn bằng hệ thống dấu phẩy động (floating-point), thường tuân theo chuẩn IEEE 754 [5]. Biểu diễn này bao gồm ba thành phần chính: bit dấu (sign), phần mũ (exponent) và phần định trị (mantissa hoặc significand). Phần mũ xác định độ lớn của số, còn phần định trị xác định độ chính xác của nó. Các định dạng phổ biến bao gồm độ chính xác đơn (32 bit) và độ chính xác kép (64 bit), tương ứng với các kiểu float và double trong nhiều ngôn ngữ lập trình.



Hình 1.3 — Tiêu chuẩn IEEE-754, 32 bit

Các phép toán cộng, trừ, nhân, chia trên số thực được thực hiện thông qua các thuật toán phức tạp để mô phỏng các phép toán số học trên tập số thực, nhưng với độ chính xác hữu hạn. Do giới hạn về số lượng bit để biểu diễn một số thực, hầu hết các phép toán sẽ dẫn đến kết quả gần đúng và có thể phát sinh sai số làm tròn. Sai số này có thể tích lũy qua nhiều phép toán, dẫn đến kết quả cuối cùng không chính xác như mong đợi. Ví dụ, các số như 0.1 không thể được biểu diễn chính xác trong hệ nhị phân, dẫn đến sự khác biệt nhỏ khi thực hiện các phép toán lặp đi lặp lại. Để giảm thiểu vấn đề này, có thể sử dụng các thư viện hỗ trợ số thập phân với độ chính xác cao hơn, chẳng hạn như module decimal trong Python.

1.3.2. Phép cộng, phép trừ

- So sánh số mũ: Tìm số mũ lớn hơn trong hai số hạng.

- Dịch phần định trị: Dịch phần định trị của số có số mũ nhỏ hơn sang phải sao cho số mũ của nó bằng với số mũ lớn hơn. Việc dịch này có thể làm mất đi một phần độ chính xác.
- Cộng hoặc trừ phần định trị: Thực hiện phép cộng hoặc trừ trên phần định trị đã được điều chỉnh.
- Chuẩn hóa kết quả: Chuẩn hóa kết quả nếu cần (ví dụ, nếu kết quả có phần định trị quá lớn hoặc quá nhỏ).
- Làm tròn: Làm tròn kết quả nếu cần.

1.3.3. Phép nhân

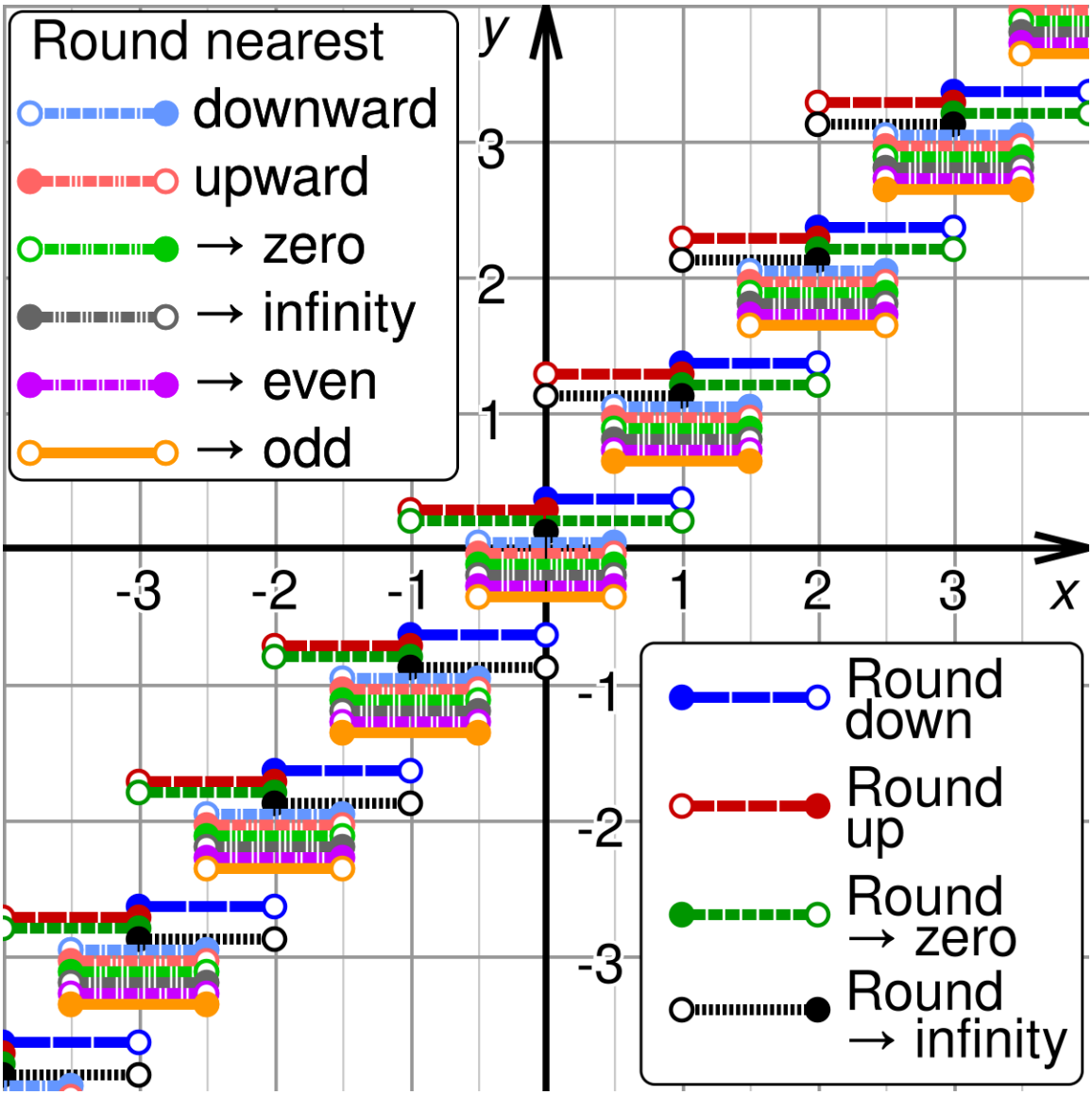
- Nhân phần định trị: Nhân phần định trị của hai số hạng.
- Cộng số mũ: Cộng số mũ của hai số hạng.
- Xác định dấu: Dấu của kết quả là tích của dấu của hai số hạng (dương * dương = dương, dương * âm = âm, âm * âm = dương).
- Chuẩn hóa kết quả: Chuẩn hóa kết quả nếu cần.
- Làm tròn: Làm tròn kết quả nếu cần.

1.3.4. Phép chia

- Chia phần định trị: Chia phần định trị của số bị chia cho phần định trị của số chia.
- Trừ số mũ: Trừ số mũ của số chia khỏi số mũ của số bị chia.
- Xác định dấu: Dấu của kết quả là thương của dấu của hai số hạng (dương / dương = dương, dương / âm = âm, âm / âm = dương).
- Chuẩn hóa kết quả: Chuẩn hóa kết quả nếu cần.
- Làm tròn: Làm tròn kết quả nếu cần.

1.4. Quy tắc làm tròn

Khi kết quả của một phép toán số học không thể được biểu diễn chính xác, cần phải áp dụng các quy tắc làm tròn [6] để đưa kết quả về một giá trị có thể biểu diễn được.



Hình 1.4 — Minh họa các quy tắc làm tròn

1.4.1. Các kiểu làm tròn

- **Làm tròn đến số chẵn gần nhất** (Round to nearest, ties to even - Banker's Rounding): Đây là phương pháp làm tròn mặc định trong IEEE 754 và nhiều ngôn ngữ lập trình. Khi một số nằm chính giữa hai số có thể biểu diễn, nó sẽ được làm tròn đến số có chữ số cuối cùng là chẵn. Mục đích của phương pháp này là giảm thiểu sự thiên vị khi làm tròn một lượng lớn các số. Ví dụ, 2.5 được làm tròn thành 2, và 3.5 được làm tròn thành 4.
- **Làm tròn lên** (Round up, towards +Infinity - Ceiling): Số luôn được làm tròn đến số nguyên gần nhất lớn hơn hoặc bằng nó. Ví dụ, 2.3 được làm tròn

thành 3, và -2.3 được làm tròn thành -2. Trong Python, hàm math.ceil() được sử dụng cho mục đích này.

- **Làm tròn xuống** (Round down, towards -Infinity - Floor): Số luôn được làm tròn đến số nguyên gần nhất nhỏ hơn hoặc bằng nó. Ví dụ, 2.7 được làm tròn thành 2, và -2.7 được làm tròn thành -3. Trong Python, hàm math.floor() được sử dụng cho mục đích này.

- **Làm tròn về không** (Round towards zero - Truncation): Phần thập phân của số bị loại bỏ, và số được giữ nguyên phần nguyên của nó. Ví dụ, 2.9 được làm tròn thành 2, và -2.9 được làm tròn thành -2.

Các quy tắc làm tròn khác cũng tồn tại, chẳng hạn như làm tròn ra xa số không (round away from zero) và các biến thể của làm tròn đến số gần nhất khi có sự hòa (ties), ví dụ như làm tròn nửa lên (round half up) hoặc làm tròn nửa xuống (round half down). Việc lựa chọn quy tắc làm tròn có thể ảnh hưởng đến độ chính xác và tính ổn định của các phép tính số học, đặc biệt trong các ứng dụng nhạy cảm với sai số.

1.4.2. Các chiến lược để giảm thiểu lỗi làm tròn

- Sử dụng các kiểu dữ liệu có độ chính xác cao hơn (ví dụ: double thay vì float).
- Sử dụng các thuật toán số ổn định hơn và ít bị tích lũy lỗi hơn.
- Thận trọng với việc so sánh bằng nhau của các số dấu chấm động; sử dụng một dung sai nhỏ (epsilon).
- Cân nhắc các biểu diễn thay thế như số thập phân cho các phép tính tài chính, nơi biểu diễn thập phân chính xác là rất quan trọng.
- Thực hiện phép nhân trước phép cộng khi xử lý các số có độ lớn khác nhau.
- Làm tròn kết quả đến một số lượng chữ số thập phân cụ thể khi thích hợp.

1.5. NaN và nguyên tắc phát sinh

NaN [7], [8], viết tắt của "Not a Number", là một giá trị đặc biệt trong số học dấu phẩy động được sử dụng để biểu thị kết quả của một phép toán không xác định hoặc không thể biểu diễn dưới dạng một số thực. NaN không giống như vô cực (infinity) hoặc tràn số/thiếu số. Một số nguyên nhân phổ biến dẫn đến việc phát sinh giá trị NaN bao gồm:

- **Phép toán không xác định:** Ví dụ, chia 0 cho 0 $\left(\frac{0}{0}\right)$, chia vô cực cho vô cực $\left(\frac{\infty}{\infty}\right)$, nhân 0 với vô cực $(0 * \infty)$, hoặc vô cực trừ vô cực $(\infty - \infty)$.
- **Phép toán trên số thực cho kết quả phức:** Ví dụ, căn bậc hai của một số âm $(\sqrt{-1})$, logarit của một số âm, hoặc các hàm lượng giác ngược (arcsin, arccos) với đầu vào nằm ngoài phạm vi $[-1, 1]$.
- **So sánh không có thứ tự:** Khi so sánh hai giá trị NaN, kết quả là "không có thứ tự" (unordered), và các phép so sánh như bằng, lớn hơn, nhỏ hơn đều trả về false. Đặc biệt, NaN không bằng chính nó (NaN == NaN là false).
- **Kết quả của một chuỗi phép toán có chứa NaN:** Hầu hết các phép toán số học với một toán hạng là NaN sẽ cho kết quả là NaN. Điều này cho phép NaN lan truyền qua các phép tính và giúp phát hiện lỗi ở giai đoạn cuối mà không cần kiểm tra từng bước trung gian.

	First Name	Gender	Salary	Bonus %	Senior Management	Team
0	Douglas	Male	97308.0	6.945	True	Marketing
1	Thomas	Male	61933.0	NaN	True	NaN
2	Jerry	Male	NaN	9.340	True	Finance
3	Dennis	n.a.	115163.0	10.125	False	Legal
4	NaN	Female	NaN	11.598	NaN	Finance
5	Angela	NaN	NaN	18.523	True	Engineering
6	Shawn	Male	111737.0	6.414	False	NaN
7	Rachel	Female	142032.0	12.599	False	Business Development
8	Linda	Female	57427.0	9.557	True	Client Services
9	Stephanie	Female	36844.0	5.574	True	Business Development
10	NaN	NaN	NaN	NaN	NaN	NaN

Hình 1.5 — Một bảng dữ liệu có giá trị NaN

1.6. Quiet NaN và Signaling NaN

Chuẩn IEEE 754 định nghĩa hai loại NaN riêng biệt: Quiet NaN (qNaN) và Signaling NaN (sNaN).

1.6.1. Quiet NaN (qNaN)

- **Quiet NaN (qNaN):** [9] Đây là loại NaN phổ biến hơn, được sử dụng để lan truyền lỗi từ các phép toán không hợp lệ hoặc các giá trị không xác định. Các phép toán số học thông thường với qNaN thường sẽ trả về một qNaN mà không gây ra ngoại lệ. qNaN cho phép các phép tính tiếp tục mà không bị gián đoạn, và lỗi có thể được kiểm tra ở kết quả cuối cùng.
- Trong biểu diễn nhị phân theo IEEE 754, qNaN thường có bit quan trọng nhất của phần định trị (sau bit ẩn) được đặt là 1.

1.6.2. Signaling NaN (sNaN)

- **Signaling NaN (sNaN):** [10] Đây là một loại NaN đặc biệt, khi được sử dụng làm toán hạng trong hầu hết các phép toán, sẽ gây ra một ngoại lệ "invalid operation". sNaN được thiết kế để hỗ trợ các tính năng nâng cao như kết hợp tính toán số và ký hiệu, hoặc các mở rộng khác cho số học dấu phẩy động cơ bản. sNaN có thể được sử dụng để đánh dấu các biến chưa được khởi tạo hoặc các điều kiện lỗi đặc biệt.
- Trong biểu diễn nhị phân, sNaN thường có bit quan trọng nhất của phần định trị (sau bit ẩn) được đặt là 0 và ít nhất một bit khác trong phần định trị phải khác 0 để phân biệt với vô cực. Khi một sNaN được sử dụng trong một phép toán, nó thường được "làm im lặng" thành một qNaN để tiếp tục lan truyền lỗi. Tuy nhiên, sự hỗ trợ cho sNaN có thể khác nhau tùy thuộc vào phần cứng và ngôn ngữ lập trình. Python, ví dụ, không cung cấp sự hỗ trợ trực tiếp và dễ dàng cho việc tạo và xử lý sNaN. Giá trị float('nan') trong Python thường tương ứng với một qNaN.

Chương 2

Kết luận

2.1. Kết quả đạt được

Báo cáo này đã trình bày các khái niệm quan trọng trong số học máy tính, bao gồm tràn số, thiếu số (cho cả số nguyên và số thực), nguyên tắc của số học dấu phẩy động và chuẩn IEEE 754, tầm quan trọng của các quy tắc làm tròn, và vai trò của NaN trong việc xử lý các điều kiện ngoại lệ. Việc hiểu rõ những khái niệm này là nền tảng để phát triển phần mềm số học chính xác, đáng tin cậy và mạnh mẽ. Bỏ qua những chi tiết cấp thấp này có thể dẫn đến những lỗi tiềm ẩn và hành vi không mong muốn trong các ứng dụng. Do đó, việc lựa chọn kiểu dữ liệu phù hợp và nhận thức được các vấn đề về độ chính xác và sự lan truyền sai số trong tính toán số là vô cùng quan trọng.

2.2. Khuyến nghị

- Trong các ứng dụng quan trọng liên quan đến số học dấu chấm động, các nhà phát triển phải nhận thức được các lỗi làm tròn tiềm ẩn, mất độ chính xác và các điều kiện tràn số/thiếu số và sử dụng các chiến lược thích hợp để giảm thiểu tác động của chúng.
- Đối với các ứng dụng tài chính hoặc bất kỳ tình huống nào đòi hỏi biểu diễn thập phân chính xác, hãy cân nhắc sử dụng các kiểu dữ liệu thập phân hoặc các thư viện số học có độ chính xác tùy ý để tránh các hạn chế của dấu chấm động nhị phân.

Tài liệu tham khảo

[1] Arithmetic Overflow Definition 2025. https://limeup.io/glossary/arithmetic-overflow.

[2] Organization of Computer Systems, §3: Computer Arithmetic 2025. https://www.cise.ufl.edu/~mssz/CompOrg/CDA-arith.html.

[3] Arithmetic Underflow Definition 2025. https://limeup.io/glossary/arithmetic-underflow/.

[4] CIE A-Level Computer Science Notes: Underflow and Overflow 2025. https://www.tutorchase.com/notes/cie-a-level/computer-science/13-3-6-underflow-and-overflow.

[5] Computer Science: Real Numbers - IEEE Standard For Floating Point 2025. http://www.multiwingspan.co.uk/a23.php?page=real.

[6] Rounding (GNU C Language Manual) 2025. https://www.gnu.org/software/c-intro-and-ref/manual/html_node/Rounding.html.

[7] Infinity and NaN (GNU C Language Manual) 2025. https://www.gnu.org/software/libc/manual/html_node/Infinity-and-NaN.html.

[8] Floating-point arithmetic – all you need to know, explained interactively 2025. https://matloka.com/blog/floating-point-101.

[9] Quiet NaN 2025. https://mathworld.wolfram.com/QuietNaN.html.

[10] NaN is Not a Math 2025. https://learncodethehardway.com/blog/12-nan-is-not-a-math/.