Có 1 vấn đề vẫn thường hay gặp phải và trở nên rất phổ biến trong tất cả các ngôn ngữ lập trình đó là:

“**Sai số ở các tính toán trên số thập phân**”, ví dụ:

Phép cộng: 0.1 + 0.2 = 0.299999999

Phép nhân: 0.7 \* 12 = 8.399999999

Không ít lập trình viên chịu bó tay không lý giải được hiện tượng trên.

Bài viết sau đây sẽ nêu lên được nguyên nhân cốt lõi của vấn đề:

https://quantrimang.com/nhung-van-de-voi-tinh-toan-dau-cham-dong-32936

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Hầu hết các ngôn ngữ lập trình đều có kiểu dữ liệu dấu chấm động và đều hỗ trợ tính toán thập phân trực tiếp hoăc thông qua các thư viện. Ngoại trừ một trường hợp đặc biệt và rất đáng tiếc là JavaScript/JScript:

    § Java (thông qua BigDecimal)

    § Visual Basic.NET và C# của Microsoft (thông qua class Decimal của môi trường .NET)

    § C++, Eiffel và Python (thông qua các thư viện)

    § Visual Basic (với kiểu Currency)

Như vậy khi gặp các phép tính trên số thực dấu chấm động hay còn gọi là số thập phân thì nhất định, bắt buộc các bạn phải sử dụng đúng class như trên:

    § Trên Java thì sử dụng BigDecimal không dùng kiểu Float,

    § Trên .NET thì sử dụng Decimal không dùng kiểu Double

Riêng với trường hợp của javascript: vì không có class hỗ trợ nên để xử lý các phép tính bị sai số:

    a/ Giải pháp trước đây: dùng mẹo “nhân lên rồi chia ra”

**Ví dụ 1:**

             var x = 0.1;

             var y = 0.2;

             var z = x + y                                     // z nhận được kết quả là 0.299999999 (sai)

             Áp dụng nhân lên và chia ra với 10

             var z = (x \* 10 + y \* 10) / 10;             // z nhận được kết quả 0.3 (đúng)

**Ví dụ 2:**

             Debug cho thấy kết quả của phép nhân 700 \* 9.70 là 6789.999999999: sai, trong khi kết quả đúng phải là 6790

             Áp dụng cách sau: nhân lên 10^x để chuyển số thập phân thành số nguyên, rồi sau đó chia lại cho 10^x

             Bước 1: xác định số mũ x bằng việc chuyển 9.70 thành số nguyên

                          9.70 là số thập phân có 2 chữ số, muốn chuyển thành số nguyên phải nhân 10^2 .Vậy x là 2

             Bước 2: Thực hiện phép tính nhân với 10^2

                          A = 700 \* (9.70 \* 10^2) = 679000

             Bước 3: Chia lại cho 10^2

                          KQ = A / 10^2 = 6790

    b/ Giải pháp đề nghị: xây dựng library dùng cho việc định nghĩa và xử lý trên số thực.

             Hiện tại đang có mã nguồn MIT hỗ trợ giải quyết vấn đề này. Thư viện và ví dụ đang để ở folder sau:

             S:\(BiMat)Common\【社外秘】教育チーム\Curriculum\CACH TAO MOT PG TOT\CAC PROBLEM PHO BIEN\Decimal tren JS

              Việc ứng dụng thư viện trên ở AIT, nhờ các phụ trách PM, PL xem xét và cân nhắc cho từng dự án của mình.

     Ngoài ra có thể sử dụng thêm các cặp số sau để kiểm chứng tính đúng đắn của phép toán:

             cặp số thứ 1:      0.7 \* 12 = 8.399999999

             cặp số thứ 2:      0.7 \* 90000= 62999.999999999

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vì vấn đề thuộc cơ bản và ít được nhắc đến trong các tài liệu hướng dẫn và qui cách nên rất thường bị lãng quên. Tất cả mọi người 1 lần nữa, hãy đọc lại và nhìn nhận thật đúng đắn.

Từ đây trở về sau, các bạn phải **nhất định** "vấn đề trên đây sẽ không có khả năng gây phát sinh lỗi ở AIT".

Hết.