ARREDONDAMENTO DECIMAL NA LINGUAGEM C

Em C, o arredondamento decimal para números de ponto flutuante (float e double) é influenciado pela forma como esses números são representados internamente em binário. A representação binária não pode representar exatamente todos os números decimais, o que pode resultar em imprecisões durante cálculos aritméticos. Isso é conhecido como erro de arredondamento ou erro de precisão.

Os números de ponto flutuante são armazenados na memória usando um formato que inclui três componentes principais: sinal, expoente e mantissa. A representação é geralmente do tipo "notação científica em base 2", onde um número é representado como mantissa * 2^expoente. No entanto, a mantissa é normalmente normalizada para um valor entre 1 e 2 (ou -1 e -2 no caso de números negativos), de modo que comece com "1." em binário.

O erro de arredondamento ocorre quando um número decimal não pode ser representado exatamente nesse formato binário. Por exemplo, o número decimal 0.1 não pode ser representado precisamente em binário, o que pode resultar em pequenos erros quando usado em cálculos.

```
#include <stdio.h>

int main(void)

{

    float num = 0.1;

    printf("%.9f\n", num);//mostra 9 casas decimais

    return 0;

}
```

Neste exemplo, você verá que o número impresso não é exatamente 0.1, mas algo como 0.100000001. Isso ocorre devido à representação binária e à forma como o número é armazenado em ponto flutuante.

Considere um outro exemplo, uma simples soma de 0.1 + 0.2:

```
#include <stdio.h>

int main(void)

{

    float num1=0.1, num2=0.2;

    printf("%.9f\n", num1 + num2);

    return 0;

}
```

O resultado da soma é algo como 0.300000012.

Portanto, ao realizar arredondamentos com números de ponto flutuante em C, é importante estar ciente dessas limitações e das potenciais imprecisões resultantes da representação binária. Em muitos casos, isso não causará problemas significativos, mas em situações críticas onde a precisão é essencial, é importante considerar o uso de técnicas específicas para mitigar essas questões, como o uso de números decimais de precisão fixa ou a utilização de bibliotecas especializadas para cálculos de ponto flutuante de alta precisão.

Para minimizar esse tipo de erro, é importante estar ciente das limitações de representação de ponto flutuante e considerar estratégias como arredondamento apenas no momento da exibição do resultado, se necessário, em vez de confiar nas comparações exatas de valores de ponto flutuante. Bibliotecas matemáticas mais avançadas também podem ser usadas para cálculos mais precisos, se necessário.

DIFERENÇA ENTRE FLOAT E DOUBLE PARA O ARREDONDAMENTO DECIMAL

Tanto float quanto double são tipos de dados utilizados em programação para representar números de ponto flutuante, ou seja, números que possuem uma parte decimal e uma parte fracionária. A principal diferença entre eles está na precisão e no tamanho.

- ✓ Precisão:
  - float: O tipo float é de precisão simples, o que significa que ele ocupa 32 bits (4 bytes) na memória e oferece aproximadamente 7 dígitos de precisão decimal.
  - double: O tipo double é de precisão dupla e ocupa 64 bits (8 bytes) na memória, fornecendo cerca de 15 a 17 dígitos de precisão decimal.
- ✓ Faixa de Valores:
  - Ambos os tipos têm faixas de valores representáveis, mas o double pode representar uma gama maior de valores devido à sua maior quantidade de bits.
  - A faixa de valores típicos para um float em C é de aproximadamente $1.17549435 \times 10^{-38}$ a $3.40282347 \times 10^{38}$.
  - A faixa de valores típicos para um double em C é de aproximadamente $2.2250738585072014 \times 10^{-308}$ a $1.7976931348623157 \times 10^{308}$.

Quando se trata de arredondamento decimal, tanto float quanto double estão sujeitos a imprecisões devido à representação binária interna dos números de ponto flutuante. Muitos números decimais simples não podem ser representados exatamente em binário, levando a arredondamentos e erros de precisão.

Devido a essas imprecisões, operações aritméticas simples podem gerar resultados ligeiramente diferentes do que você esperaria ao lidar com números decimais.

Em geral, você deve escolher o tipo de dados (float ou double) com base na sua necessidade de precisão e no alcance de valores que você pretende lidar. Se você precisa de alta precisão ou está trabalhando com cálculos críticos, double é uma escolha melhor. Se você estiver lidando com cálculos onde a precisão um pouco menor é aceitável e deseja economizar espaço em memória, float pode ser suficiente.

QUAL O MELHOR TIPO DE DADOS PARA TRABALHAR COM VALORES MONETÁRIOS EM LINGUAGEM C?

Ao trabalhar com valores monetários em uma linguagem como C, é importante escolher o tipo de dados apropriado para garantir precisão e evitar problemas de arredondamento. No contexto de dinheiro e finanças, recomenda-se o uso do tipo double para representar valores monetários. Aqui estão algumas razões pelas quais double é frequentemente preferido:

Precisão Adequada: O tipo double é um tipo de ponto flutuante de precisão dupla, que oferece uma precisão maior do que o tipo float. Isso é importante ao lidar com valores monetários, onde a precisão decimal é crucial para evitar erros de arredondamento.

Faixa de Valores: O tipo double permite representar uma ampla faixa de valores, o que é importante ao lidar com transações financeiras que podem envolver grandes quantidades de dinheiro.

Compatibilidade com Bibliotecas: Muitas bibliotecas e APIs de cálculos financeiros e contábeis em C estão projetadas para trabalhar com valores de tipo double, facilitando a integração e o desenvolvimento de aplicações financeiras robustas.

No entanto, ao trabalhar com valores monetários, você também deve estar ciente de alguns cuidados adicionais:

Evitar Cálculos com Pontos Flutuantes: Como os números de ponto flutuante têm representação limitada, cálculos repetidos podem levar a pequenos erros de arredondamento. É recomendável evitar cálculos diretos de valores monetários sempre que possível, especialmente para operações financeiras críticas.

Utilizar Funções de Arredondamento Adequadas: Ao apresentar valores monetários para usuários ou realizar operações de arredondamento, use funções como **round**, **ceil** ou **floor** para garantir que os resultados sejam precisos e coerentes.

Por exemplo, ao lidar com valores monetários em C, você pode declarar uma variável usando o tipo double da seguinte forma:

```
#include <stdio.h>

int main(void)

{
```

```
    double valorMonetario = 1234.56;

    printf("Valor monetario: %.2f\n", valorMonetario); // Exibe o
valor com duas casas decimais

    return 0;

}
```

Nesse exemplo, valorMonetario é uma variável do tipo double que pode armazenar valores monetários com precisão decimal. A formatação %.2f na função printf é usada para exibir o valor com duas casas decimais.

Referências:

http://www.inf.ufes.br/~zegonc/material/Introducao_a_Computacao/Aula_Zegonc_Ponto_Flutuante_NEW.pdf

https://pt.stackoverflow.com/questions/219211/qual-a-forma-correta-de-usar-os-tipos-float-double-e-decimal