# AULA 2 – NOÇÕES BÁSICAS SOBRE ERROS

Prof. Gustavo Resque
gustavoresqueufpa@gmail.com

# REPRESENTAÇÃO DE NÚMEROS

- Exemplo 1

  - Ao calcular a área de uma circunferência de raio 100m obtivemos os seguintes resultados:

    - $A = 31400\ m^2$
    - $A = 31416\ m^2$
    - $A = 31415{,}92654\ m^2$

  - Como justificar tais diferenças?

  - É possível obter o número exato dessa área?

# REPRESENTAÇÃO DE NÚMEROS

- Exemplo 1 - Comentários
  - resultados:
    - $A = 31400\ m^2$
    - $A = 31416\ m^2$
    - $A = 31415{,}92654\ m^2$
  - O número pi é irracional, portanto não pode ser escrito com um número finito de dígitos. Então, dependendo do número de casas decimais utilizada gera os resultados acima. O valores utilizados foram:
    - 3,14
    - 3,1416
    - 3,141592654

# REPRESENTAÇÃO DE NÚMEROS

- Exemplo 2
  - Ao efetuar o somatório a seguir em uma calculadora (a) e em um computador (b):
    - $S = \sum_{i=1}^{30000} x_i$ para $x_i = 0.5$ e para $x_i = 0.11$
  - Obtivemos os seguintes resultados
    - Para $x_i = 0.5$
      - Na calculadora: $S = 15000$
      - No computador: $S = 15000$
    - Para $x_i = 0.11$
      - Na calculadora: $S = 3300$
      - No computador: $S = 3299.99691$
  - Como justificar a diferença entre os resultados obtidos?

# REPRESENTAÇÃO DE NÚMEROS

- Exemplo 2 – Comentários

  - Um número pode ter representação finita em uma base e infinita em outra

  - Neste caso o número 0.5 tem representação binária finita, mas o número 0.11 tem representação binária infinita.
    - $(0.5)_{10} = (0.1)_2$
    - $(0.11)_{10} = (0.00\overline{011100001010001111101}\ldots)$

  - Por tanto, uma vez que o computador (obviamente) precisa arredondar ou truncar esse número, a somatória gerará sempre um número aproximado.

# CONVERSÃO DE BASE

- Revisão
  - Converter o número 22 para binário
  - Converter 10101 para decimal

# CONVERSÃO DE BASE

- Revisão

  - Converter o número 22 para binário
    - 10110

  - Converter 10101 para decimal
    - 21

# CONVERSÃO DE BASE

- Todo número pode ser escrito do seguinte forma:

  - $(347)_{10} = 3 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$
  - $(10111)_2 = 1 \times 10^4 + 0 \times 10^3 + 1 \times 10^2 + 1 \times 10^1 + 1 \times 10^0$

- A conversão para uma base maior é mais direta:
  - $(10111)_2 = (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10} = (23)_{10}$

- Porém a conversão para uma base menor depende do processo inverso
  - $(23)_{10} = (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$

# CONVERSÃO DE BASE

- Porém a conversão para uma base menor depende do processo inverso
  - Vamos colocar a base em evidência
  - $(23)_{10} = (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$
  - $(23)_{10} = (2 \times (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (1 \times 2^1 + 0 \times 2^0) + 1) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (2 \times (1 \times 2^0) + 0) + 1) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (2 \times (+1) + 0) + 1) + 1) + 1)_{10}$

- Nesse caso, forma-se uma recorrência de multiplicações por 2 somado por um número binário.

# CONVERSÃO DE BASE

- Porém a conversão para uma base menor depende do processo inverso
  - Vamos colocar a base em evidência
  - $(23)_{10} = (1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0)_{10}$
  - $(23)_{10} = (2 \times (1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (1 \times 2^1 + 0 \times 2^0) + 1) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (2 \times (1 \times 2^0) + 0) + 1) + 1) + 1)_{10}$
  - $(23)_{10} = (2 \times (2 \times (2 \times (2 \times (+1) + 0) + 1) + 1) + 1)_{10}$

- Nesse caso, forma-se uma recorrência de multiplicações por 2 somado por um número binário.

- Problema: neste caso sabemos os valores das casas binárias e quantas foram, mas para um número qualquer não sabemos

# CONVERSÃO DE BASE

- Para um número qualquer, temos uma recorrência:
  - $x_0 = 2 \times (x_1) + (c_0)_2$
  - $x_1 = 2 \times (x_2) + (c_1)_2$
  - …
  - $x_n = 2 \times 0 + (c_n)_2$
- Resolvendo a recorrência e considerando que $(2)_{10} = (10)_2$, teremos o número em binário.
- Fazer exemplo para o número 25.
- Os algoritmos vão simplificar essa visão dividindo o valor recursivamente por 2 e considerando resto da divisão como os valores de $[c_0, c_1, \dots, c_n]$.

# CONVERSÃO DE BASE - FRAÇÃO

- A fração tem o mesmo princípio uma vez que:
  - $12.2 = 1 \times 10^{1} + 2 \times 10^{0} + 2 \times 10^{-1}$
  - $0.234 = 0 \times 10^{0} + 2 \times 10^{-1} + 3 \times 10^{-2} + 4 \times 10^{-3}$

- Assim também para um número binário:
  - $(0.101)_2 = 0 \times 2^{0} + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$

  - $(0.101)_2 = \frac{1}{2} + \frac{1}{8} = 0.625$

- Da mesma forma para trocar a base de decimal para binário
  - $0.101 = (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})$
  - $0.101 = \frac{1 + (0 \times 2^{-1} + 1 \times 2^{-2})}{2}$

# CONVERSÃO DE BASE - FRAÇÃO

- Da mesma forma para trocar a base de decimal para binário
  - $0.101 = (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3})$
  - $0.101 = \frac{1 + (0 \times 2^{-1} + 1 \times 2^{-2})}{2}$
  - $0.101 = \frac{1 + \frac{0 + (1 \times 2^{-1})}{2}}{2}$
  - $0.101 = \frac{1 + \frac{0 + \frac{1 + \frac{\mathbf{0}}{\mathbf{2}}}{2}}{2}}{2}$
- Perceba que a iteração agora que a recorrência vai dividindo, então para fazer o processo inverso é necessário multiplicar o valor decimal por 2.
  - Como trata-se de uma fração, caso $x_n \geq 1$, então $c_n = 1$ e $c_n = 0$ caso contrário. O algoritmo para quando a parte fracionária chega a zero.

# CONVERSÃO DE BASE - FRAÇÃO

- Perceba que para certos casos o algoritmo pode não parar.
  - Ex:
  - $(0.1)_{10} = \frac{0+0.2}{2}$
  - $(0.1)_{10} = \frac{0+\frac{(0+0.4)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{(0+0.8)}{2}\right)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{\left(0+\frac{(1+0.6)}{2}\right)}{2}\right)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{\left(0+\frac{\left(1+\frac{(1+0.2)}{2}\right)}{2}\right)}{2}\right)}{2}}{2} = (0.0\overline{0011})_2$

# CONVERSÃO DE BASE - FRAÇÃO

- Perceba que para certos casos o algoritmo pode não parar.
  - Ex:
  - $(0.1)_{10} = \frac{0+0.2}{2}$
  - $(0.1)_{10} = \frac{0+\frac{(0+0.4)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{(0+0.8)}{2}\right)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{\left(0+\frac{(1+0.6)}{2}\right)}{2}\right)}{2}}{2}$
  - $(0.1)_{10} = \frac{0+\frac{\left(0+\frac{\left(0+\frac{\left(1+\frac{(1+0.2)}{2}\right)}{2}\right)}{2}\right)}{2}}{2} = (0.0\overline{0011})_2$

- Agora entendemos melhor porque o somatório abaixo não dá resultado exato no computador:

$$S = \sum_{i=1}^{30000} 0.11$$

# REPRESENTAÇÃO - PONTO FLUTUANTE

- Por convenção os computadores representam os números reais em notação científica no seguinte formato

$$s0.d_1 d_2 \ldots d_m \times 10^n$$

- Onde:
  - $s$ é o sinal
  - $d_1 d_2 \ldots d_m$ são os dígitos da mantissa com m posições
  - $m$ é o tamanho da mantissa
  - $n$ é o valor do expoente

- Exemplo
  - $(101.111)_2$ é representado como $0.101111 \times 10^{11}$

# REPRESENTAÇÃO - PONTO FLUTUANTE

- O que ocorre se $m = 3$ e tentarmos representar o número $101.1$?

  - A representação produz um erro, pois com esse tamanho mantissa não é possível representar os 4 dígitos significativos desse número.
    - $101.1 \rightarrow 0.101 \times 10^{11} \rightarrow 101$

  - Sendo assim, essa máquina só consegue representar 3 dígitos significativos de qualquer número. Sem considerar a capacidade do expoente.

  - Ao considerar que o expoente tem 1 bit para o sinal e 2 bits para o número, qual o conjunto de números que podem ser representados?

# REPRESENTAÇÃO - PONTO FLUTUANTE

- O que ocorre se $m = 3$ e tentarmos representar o número $101.1$?
  - Ao considerar que o expoente tem 1 bit para o sinal e 2 bits para o número, qual o conjunto de números que podem ser representados?
    - O expoente $e$: vai de $-11 \leq e \leq 11$, ou seja $(-3)_{10} \leq e \leq (3)_{10}$
  - Ou seja, algumas situações limites podem ocorrer:
    - $1011.0 \rightarrow 0.111 \times 10^{11} \rightarrow 111.0$ (overflow)
    - $0.0000111 \rightarrow 0.011 \times 10^{-11} \rightarrow 0.000011$ (truncamento)

# REPRESENTAÇÃO - PONTO FLUTUANTE

- Aritmética de ponto flutuante
  - Ao efetuarmos cálculos com o computador, todos os números envolvidos (entradas da operação e saídas) devem estar dentro dos limites de representação da máquina.

  - Por exemplo

$$110.0 + 10.1 = 1000.1$$

  - Porém considerando a máquina do exemplo anterior
    - As entradas podem ser representadas na máquina
      - $110.0 \rightarrow 0.110 \times 10^{11}$ e $10.1 \rightarrow 0.101 \times 10^{10}$

    - Entretanto a saída deve ser aproximada
      - $1000.1 \rightarrow 0.111 \times 10^{11} \rightarrow 111.0$

# REPRESENTAÇÃO - PONTO FLUTUANTE

- Aritmética de ponto flutuante
  - Outro exemplo:
    - $100.01 + 100.01 - 11.1 = 101.0$
  - Ao realiza-lo na máquina:

# REPRESENTAÇÃO - PONTO FLUTUANTE

- Aritmética de ponto flutuante
  - Outro exemplo:
    - $100.01 + 100.01 - 11.1 = 101.0$
  - Ao realiza-lo na máquina:
    - Primeira parcela: $0.100 \times 10^{11} + 0.100 \times 10^{11} = 0.111 \times 10^{11}$
    - Segunda parcela: $0.111 \times 10^{11} - 0.111 \times 10^{10} = 0.111 \times 10^{10}$
    - Resultado: $0.111 \times 10^{10} \rightarrow \mathbf{11.1}$
  - Se fizermos primeiro a subtração, temos:
    - Primeira parcela: $0.100 \times 10^{11} - 0.111 \times 10^{10} = 0.100 \times 10^{0}$
    - Segunda parcela: $0.100 \times 10^{11} + 0.100 \times 10^{0} = 0.100 \times 10^{11}$
    - Resultado: $0.100 \times 10^{11} \rightarrow \mathbf{100.0}$
  - Observe que a propriedade associativa se perde para operações aproximadas!

# ERROS

- Como vimos, representações aproximadas estão passíveis de erros

- Então,
  - O que são erros?
  - Como são calculados?
  - Para que servem?

# ERROS

- Os erros podem ser:
  - Inerentes
  - Truncamento ou Arredondamento
  - Overflow
  - Underflow

- E podem ser medidos por:
  - Erro Absoluto
  - Erro Relativo
  - Erro Percentual

# ERROS

- Seja $\tilde{p}$ uma aproximação de $p$

  - Erro Absoluto
    - $e_a = |p - \tilde{p}|$

  - Erro Relativo
    - $e_r = \frac{e_a}{|p|} = \frac{|p-\tilde{p}|}{|p|}$

  - Erro Percentual
    - $e_p = 100e_r = 100\,\frac{|p-\tilde{p}|}{|p|}\%$

# EXERCÍCIOS

- Teste o exemplo 1 e o exemplo 2 do início dessa apresentação em uma linguagem de programação.

- Faça com que o computador apresente erros de overflow e arredondamento. Identifique a diferença entre os dois na prática.

- Por que o computador consegue mostrar o valor $(0.11)_{10}$ se esse número só tem solução aproximada em binário?