Implementação de uma Calculadora Científica utilizando Séries

Paulo Henrique de O. Bonfim, Pedro Bezerra Heinzelmann

2024.1

UERJ - Instituto de Matemática e Estatística

Disciplina: Cálculo IV

Professora: Cristiane Oliveira de Faria

Implementação de *n*!

```
função fatorial(x):
        resultado ← 1
        para i ← 2 até x faça
                resultado *= i

        retorna resultado
```

Apesar de simples, valores de **x** muito grandes, como os usados nas séries infinitas, podem exceder facilmente o espaço de memória em que a variável de retorno está alocada.

Variações da função utilizando recursão excederam o número de chamadas recursivas suportadas em alguns testes.

Implementação de π

Série de Leibniz de π	$\frac{\pi}{4} = \sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1} \Rightarrow \sum_{n=0}^{\infty} \frac{2}{(4n+1)(4n+3)}$
Pseudo-código Python	**função** pi(): resultado ← 0 **para** i ← 0 **até** 10^6 **faça** resultado += 2 / (4*i + 1) * (4*i + 3) **retorna** resultado * 4

Foi escolhido o intervalo [0,10^6] devido a precisão até a 6º casa decimal: 3.141592

Implementação de e^x

Série	$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$
Pseudo-código Python	**função** exp(x): resultado ← 0 **para** i ← 0 **até** 700 **faça** resultado += (x**i) / **fatorial**(i) **retorna** resultado

Implementação de sen(x) e cos(x)

Séries de Taylor	$\text{sen}(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$; $\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$
Pseudo-código Python	**função** cos(x): x ← x*(**pi**() / 180) #conversão para radianos resultado ← 0 **para** i ← 0 **até** 50 **faça** resultado += ((-1)**i)*(x**(2*i)) / **fatorial**(2*i) **retorna** resultado #O mesmo se aplica para a função sen(x).

Implementação de tan(x)

Série/Identidade	$(i) \quad \mathrm{tg}(x) = \sum_{n=0}^{\infty} \frac{B_{2n}(-4)^n(1-4^n)x^{2n+1}}{(2n)!}$ $(ii) \quad \mathrm{tg}(x) = \frac{sen(x)}{\cos(x)}$
Pseudo-código Python	**função** tan(x): resultado ← **sen**(x) /**cos**(x) **retorna** resultado

(i) além de computacionalmente custosa, também dependia da implementação da Números de Bernoulli. Logo, a alternativa (ii) é mais simples (mas há menor precisão) pois se aproveita das chamadas de duas outras séries já implementadas.

Implementação de ln(x)

Série	$(i) \quad \ln(x+1) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1} x^n}{n}$ $(ii) \ \ln(x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1} (\frac{a}{10})^n}{n} - (b+1) \sum_{n=1}^{\infty} \frac{(-0.9)^{n+1} (-1)^{n+1}}{n}$
Pseudo-código Python	**função** ln(x): a,b ← **notacãocientifica***(x) resultado ← 0 **para** i ← 0 **até** 370 **faça** arg1 = ((-1)**(i+1))*((a/10)**i) / i arg2 = (b+1) * (((-0.9)**(i+1))*((-1)**(i+1)) / i) resultado += arg1 – arg2 **retorna** resultado

(i) possui o intervalo de convergência **|x|** $\leq$ **1**, o que é um problema devido a entradas "x" maiores. Dessa forma, a sequência pode ser reescrita como (ii) para aceitar os valores **a** e **b**, **sendo x = a * 10^b**, para **1 < 0.1a < 1**.

*A função não existe, apenas representa um conjunto de procedimentos de formatação de funções padrão do Python 3.

Implementação de $log_{10}(x)$

Identidade	$$log_{10}(x) = \frac{ln(x)}{\ln(10)}$$
Pseudo-código Python	**função** log(x): resultado ← **ln**(x) /**ln**(10) **retorna** resultado

Implementação de $\sqrt{x}$

Série	$(i)\ \sqrt{N^2 = d} = \sum_{n=1}^{\infty} \frac{(-1)^n 2n!}{(1-2n)\, n!^2\, 4^n} \frac{d^2}{N^{2n}}$ $(ii)\quad \sqrt{x} = e^{(\ln(x)/2)}$
Pseudo-código Python	**função** raiz(x): resultado ← **exp**((**ln**(x) / 2)) **retorna** resultado

A implementação de (i) é custosa devido a presença de diversas potências e principalmente de fatoriais cada vez maiores.

Uma alternativa eficiente para o cálculo da raiz quadrada é utilizar a identidade (ii) que utiliza, por sua vez as funções das séries que convergem para e^xe ln(x) mostradas anteriormente.

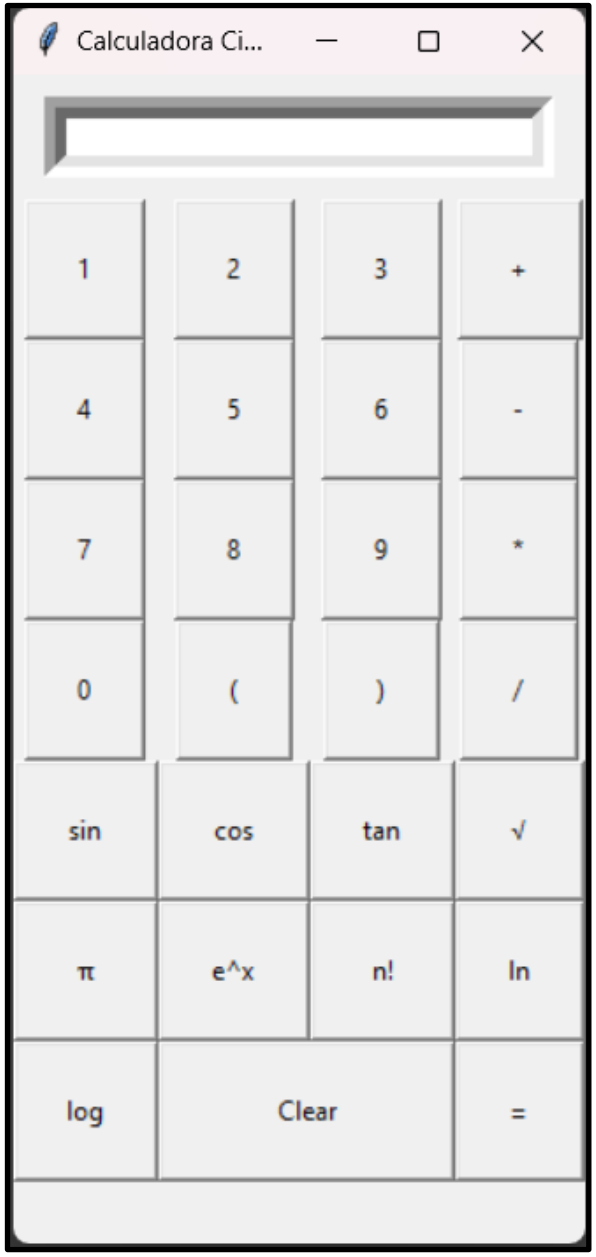
Interface da Calculadora

Aproveitamos o código de uma interface gráfica de usuário para uma calculadora científica e alteramos as chamadas das funções da biblioteca math para as funções desenvolvidas a partir das séries e dos algoritmos apresentados.

Utilizamos assim, somente o pacote tkinter para o uso da interface gráfica e as funções da biblioteca padrão do Python 3, além das desenvolvidas.

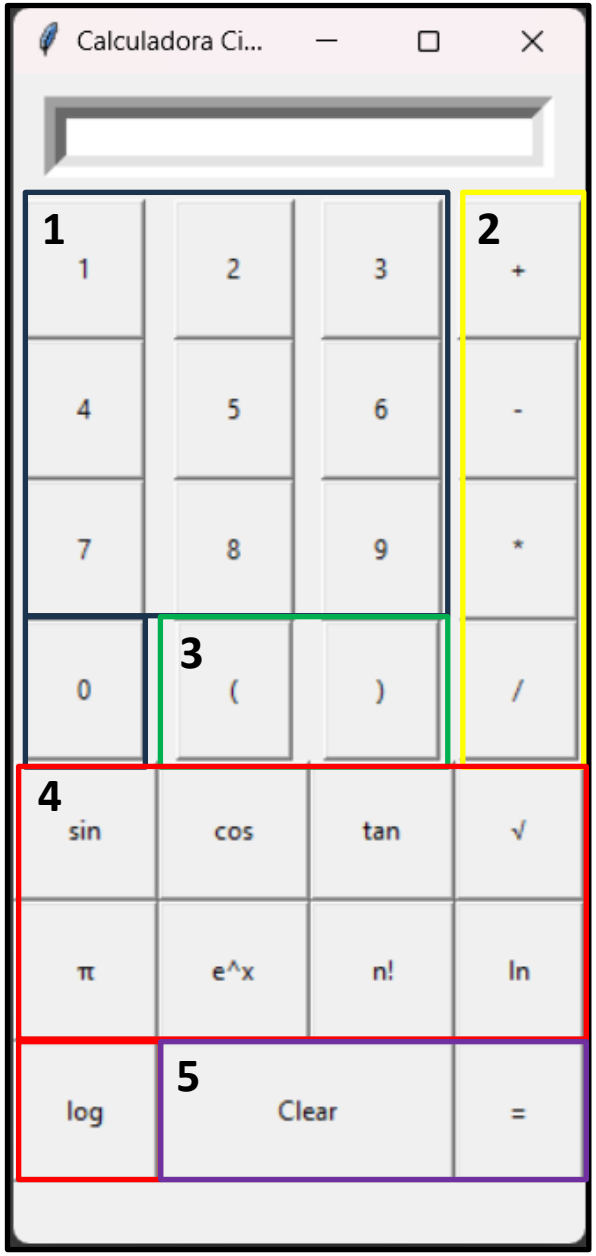
Em destaque:

fatorial(); pi(); sin(); cos(); tan(); exp(); ln(); log() e sqrt()



Demonstração de uso

Interface



Funções

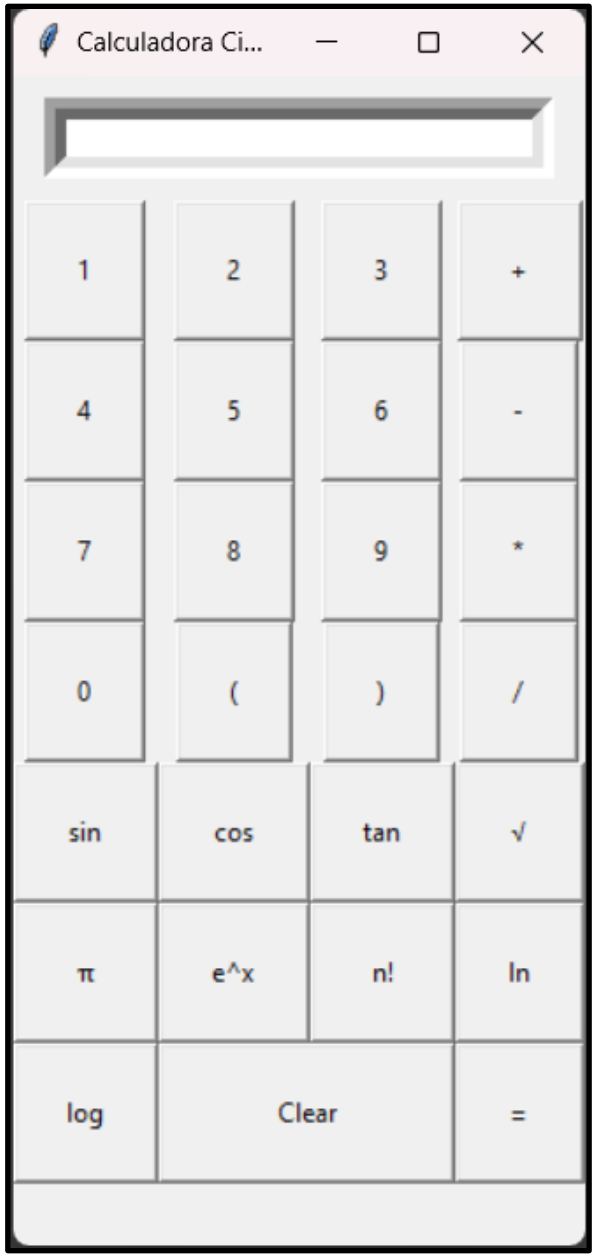
1 - Números

2 - Operações

3 – Regra de Precedência “( )“

4 – Funções com aplicações de séries

5 – Botão para avaliar a expressão e apagar



Restrições

- Não é possível armazenar histórico
- Todas funções que implementamos precisam de ter parenteses entre o valor computado