* factorials
* stop looping when reach EPSILON

Eulers:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!}$$

Madhava:

$$\pi = \sum_{k=0}^{\infty} \frac{(-3)^{-k}}{2k+1}$$

Eulers:

$$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$$

BBP:

$$\pi = 16^{-k}\left(\frac{(k(120k+151)+47)}{k(k(k(512k+1024)+712)+194)+15}\right)$$

Viet:

$$\pi = \frac{2}{\frac{\sqrt{2}}{2} \times \frac{\sqrt{2+\sqrt{2}}}{2} \times \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}}$$

Newton:

$$\underset{\swarrow\, z}{x_{k+1}} = x_k - \frac{f(x_k)}{f'(\underset{\downarrow\, y}{x_k})}$$

mathlib-test:

#define OPTIONS "a b r..."

diff(mathlib.h, math.h)

Psudo code so far:

- e.c():
  - for loop()
    - initial $k=0$; add $\frac{1}{\text{factorial of } k}$ until it reaches EPSILON
- Madhava():
  - for loop()
    - start at $k=0$ and take sum of $\frac{1}{(-3)^k \cdot (2k+1)}$ untill reaches epsilon
- euler():
  - first use for loop to sum $\frac{1}{k^2}$; k starts at 1 and goes on untill reaches EPSILON
  - Then use sqrt_Newton to find $\sqrt{6 \cdot \text{This whole sum}}$
- bbp():
  - use for loop to sum $\frac{1}{16^k}\left(\frac{k(120k+\cdots)}{k(k(k(k(\cdots))))}\right)$ while k starts at 0 and continues till reaches Epsilon.
- viete():
  - use for loop to generate a sequence of $\frac{\sqrt{2}}{2}$ and adding a $\sqrt{2}$ inside of each one like $\frac{\sqrt{2+\sqrt{2}}}{2} + \sqrt{2+\sqrt{2+\sqrt{2}}}$
  - use sqrt_newton for sqrt(2)
  - divide the entire sum FROM 2; like $\frac{2}{\frac{\sqrt{2}}{2}+\frac{\sqrt{2+\sqrt{2}}}{2}\cdots}$

Questions:

- when to use epsilons?
- for viete(), how do i put a $+\sqrt{2}$ inside of each one?
- what are my max for loops argument?
- How to test functions?

improved Psudo code:

bbP():

while (new term greater than epsilon){

for (multiply 16 by itself k times)

16^K

new term → $\frac{(K(120K+151)+47}{K(K(K(512K+1024)+712)+194)+15}$

sum of all → adding new terms

increment K }

return sum

return K for terms

Pi_euler():

counter

while (new term greater than epsilon){

new term → $1/K^2$ or $1/K*K$

K → increment

counter → increment

sum → add every new terms }

multiply 6 by sum

return sqrt_Newton → $\sqrt{6*sum}$

return counter for terms

e():

while (new term greater than epsilon){

new term init to 1

for (multiplying new denom by old denom)

new term multiplied by $\frac{1}{K}$ (factorial)

sum of all → adding new terms

increment K }

return sum

return K for terms

Pi_Madhava():

while(|new term| greater than Epsilon){

for(multiplying -3 by itself k times)

-3.^k

new term → $\frac{1}{3^k} \cdot \frac{1}{2k+1}$

increment k }

return sqrt_Newton → $\sqrt{12} * sum$

return k for counter

Improved Psudo Code

Pi_viete()
counter
init numerator → use sqrt_Newton → $\sqrt{2}$
init denomenator → 2
while (|1 - new term| greater than epsilon)
new term → nume/denome → init to $\frac{\sqrt{2}}{2}$
result → product of new terms
change new nume → add $2+\sqrt{2}$ → sqrt_Newton(2 + nume)
increment counter }
result = 2/result ≃ π
return result
return counter for factors

sqrt_Newton(): # used python translation in asgn2 pdf.
init. old term, new term, counter
while (|new - old| greater than EPSILON)
old term becomes the new term
new term → .5 * (new + argument / new)
increment counter }
return new term
return counter for terms

mathlib-test():
include header file, std math lib
the rest used from asgn pdf and TA section.
define user input options
use switch and case statements to run designated tests

Purpose:

- Coded different methods of solving for π and e w/out using the <math.h> library.
- e() for approximating e.
- Madhava(), euler(), bbp(), and viete() for approximating π.

★ All these formulas are shown earlier in this document.