asgn2 intial design

Ryan Hui

rhui1

Design

Implementing our own functions for e to the power of x and sqrt(x), approximate the value of pi and e. Compare our function values with the values generated by the math.h library.

Files

-e.c
-madhava.c
-euler.c
-bbp.c
-viete.c
-newton.c
-mathlib-test.c
-mathlib-test.h

Pseudocode

e.c

include necessary headers and files
set static global term variable
set static global counter variable

e function
k equals 0
e equals 0
for loop through k by 1 until term is less than EPSILON

term equals term * 1/k
set e equal to e plus term value
increment counter by 1

return e value

e terms function
return counter

madhava.c

include necessary headers and files
set static global term variable
set static global counter variable

pi madhava function
k equals 0
pi equals 0
for loop through k by 1 until term is less than EPSILON

numerator equals 1
denominator equals (1/(3 to the power of k) * (2k + 1)
term equals sqrt(12) * numerator/denominator
pi equals pi + term

return pi

pi madhava terms function
return counter

euler.c

include necessary headers and files
set static global term variable
set static global counter variable

pi euler function
k equals 0
pi equals 0
sum equals 0
for loop through k by 1 until term is less than EPSILON

term equals 1/k squared
sum equals sum plus term value
increment counter by 1
pi equals sqrt(6 * sum)

return pi

pi euler terms funciton return counter

bbp.c

include necessary headers and files
set static global term variable
set static global counter variable

pi bbp function
k equals 0
pi equals 0
for loop through k by 1 until term is less than EPSILON

numerator equals (k(120k+151)+47)
denominator equals (k(k(k(512k+1024)+712)+194)+15)
term equals 16 to the power of -k * numerator/denominator
pi equals pi plus term
increment counter by 1

return pi

pi bbp terms function
return counter

viete.c

include necessary headers and files
set static global term variable
set static global counter variable

pi viete function
k equals 0
pi equals 1

for loop through k by 1 until term is less than EPSILON

numerator equals (2+sqrt(2))
denominator equals 2
term equals numerator/denominator
pi equals pi * term

pi equals pi * 1/2

return pi

newton.c

include necessary headers and files
set static global counter variable

sqrt newton function
y equals 1
z equals 0
while(absolute(value(y - z) ¿ EPSILON))

z equals y
y equals 1/2 * (z * x / z)
increment counter by 1
return y

sqrt newton iters function
return counter

mathlib-test.c

include necessary headers and files
main function
get opt
case a
runs all tests
case e approximation
runs e program
case b
runs bailey approximation
case m
runs madhava approximation
case r
runs euler approximation
case v
runs viete approximation
case n
runs newton approximation
case s
enables statistics
case h
displays help message