Initial Design

Andrew Kuznetsov

January 2023

Initial Design

1 Description of Program

For assignment 2 the program contains six files that run different famous formulas to either compute an approximation of pi or an approximation of e. All these files can be run by Mathlib.h using commands to print out their unique approximations.

2 Files Included

1. e.c
   - first function approximates the value of e using a Taylor series, returns e approximation
   - second function returns number of terms it took to get to e approximation
2. madhava.c
   - first function approximates value of pi using madhava approximation, returns pi approximation
   - second function returns number of terms it took to get to pi approximation
3. euler.c
   - first function approximates the value of pi using euler's solution to the Basel problem, returns pi approximation
   - second function returns number of terms it took to get to pi approximation
4. bbp.c
   - first function approximates the value of pi using Bailey-Borwein-Plouffe formula, returns pi approximation

- second function returns number of terms it took to get to pi approximation

5. viete.c
   - first function approximates the value of pi using viete's formula, returns pi approximation
   - second function returns number of terms it took to get pi approximation
6. newton.c
   - first function approximates the value of the square root of the value passed to it, returns approximation of square root of value passed
   - second function returns number of terms it took to get square root approximation
7. mathlib-test.c
   - acts as they main hub for all files
   - able to run any one of the .c files using a command such as -e or -b etc.
8. mathlib.h
   - contains main functions for the other .c functions

3 Basic Sudo Code

In each of the files that approximates pi, e, or a square root we use the value EPSILON given to us, which is approximately equal to $1 * 10^{-14}$. This is the basic sudo code for the first function in every .c file that approximates a value of pi, e, or square root

value_being_approximated = 0
count = 0
while (current_iteration EPSILON) {
current_iteration = formula calculations for current count
value_being_approximated = value_being_approximated + current_iteration
count++
} return value_being_approximated = 0

Final Design

1 Description of the project

For Assignment 2 we were assigned the task of using famous formulas from famous scientists to approximate the values of pi, e, or a square root of a number in one case. The assignment specifically prohibited us from using the math library so these approximations came from implemented formulas that were given to us in the assignment.

2 Files Included

1. e.c
   - This .c file uses the Taylor series to calculate an approximation of e
2. madhava.c
   - This .c file uses the Madhava series formula to calculate the approximation of pi
3. euler.c
   - This .c file uses the formula derived from Euler's solution to the Basel problem to approximate the value of pi
4. bbp.c
   - This .c file uses the Bailey-Borwein-Plouffe formula to approximate the value of pi
5. viete.c
   - This .c file uses Viete's formula to approximate the value of pi
6. newton.c
   - This .c file approximates the square root of the value passed to it using the Newson-Raphson method
7. mathlib-test.c
   - This file acts as the main function for all the other files
   - Allows you to run any of the approximations and formats results in readable format
8. mathlib.h
   - This is the header file that connects all the .c file's functions
   - Allows .c files to use other .c file's functions

3 Basic Sudo Code

mainlib-test.c
Main function takes arguments argc and *argv[]
argc being number of arguments and *argv[] being a list of pointers to said arguments

for all arguments in *argv[]:
check if one of the arguments in -s
make boolean print_terms equal to true

while current argument != -1:
switch for the current argument:
case for arguments (-e, -r, -b, -m, -v) argument:
print out value approximation
print out math library approximation
print out the difference between both
case for argument -a:
make boolean print_all equal to true
case for argument -n:
for all values between 0 and 10 incrementing 0.1:
print sqrt approximation for val
print math library sqrt for val
print for difference between both
case for argument -h:
print help message with all possible arguments and what they do

Notes about sudo code
To run this you first would need to do ./mathlib-test followed by a - and one of the letters a, e, r, b, m, v, n, h

Makefile
The Makefile has four main capabilities, make, make clean, make format, make scan-build. The .c files that run approximations need to be converted to executables and object files we can do that with Makefile.

$(EXEC): $(OBJS)
$(CC) $(LDFLAGS) -o $(EXEC) $(OBJS)

The make clean capability in Makefile cleans and deletes all the executable files and .o files. This is done with the line...

clean:
rm -f $(CLEAN_EXEC) $(CLEAN_OBJS)

The make format in Makefile formats the .c files to be clanged, so that when

others run the files they are already clanged. This is the way the Makefile does this...

```
format:
	clang-format -i -style=file *.[ch]
```