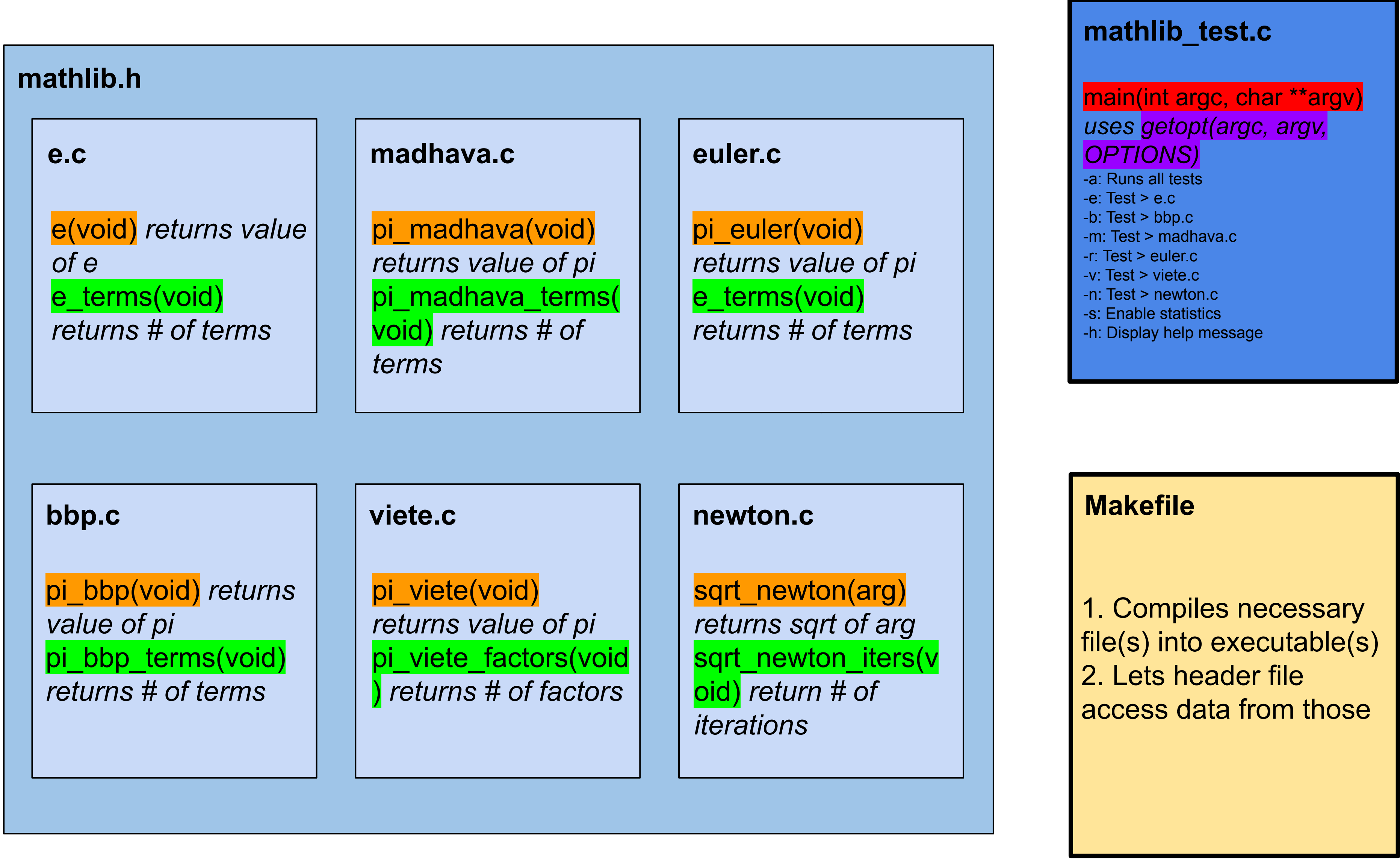

mathlib.h

e.c

e(void) *returns value of e*
e_terms(void) *returns # of terms*

madhava.c

pi_madhava(void) *returns value of pi*
pi_madhava_terms(void) *returns # of terms*

euler.c

pi_euler(void) *returns value of pi*
e_terms(void) *returns # of terms*

bbp.c

pi_bbp(void) *returns value of pi*
pi_bbp_terms(void) *returns # of terms*

viete.c

pi_viete(void) *returns value of pi*
pi_viete_factors(void) *returns # of factors*

newton.c

sqrt_newton(arg) *returns sqrt of arg*
sqrt_newton_iters(void) *return # of iterations*

mathlib_test.c

main(int argc, char **argv)
uses getopt(argc, argv, OPTIONS)
-a: Runs all tests
-e: Test > e.c
-b: Test > bbp.c
-m: Test > madhava.c
-r: Test > euler.c
-v: Test > viete.c
-n: Test > newton.c
-s: Enable statistics
-h: Display help message

Makefile

1. Compiles necessary file(s) into executable(s)
2. Lets header file access data from those

bbp.c

```
K = 0
Bbp_sum = 0
EPSILON = .0000001
A = 4 / (8 * k + 1)
B = 2 / (8 * k + 4)
C = 1 / (8 * k + 5)
D = 1 / (8 * k + 6)
K_term = (1 / (16 ** k)) * (a - b - c - d)

While abs(k_term) > EPSILON:
    Bbp_sum += k_term
    K += 1
    A = 4 / (8 * k + 1)
    B = 2 / (8 * k + 4)
    C = 1 / (8 * k + 5)
    D = 1 / (8 * k + 6)
    K_term = (1 / (16 ** k)) * (a - b - c - d)
Return bbp_sum
```

e.c

```
Initialize k, e, fact_k, kth_term, EPSILON
Loop until the kth term is less than or equal to epsilon
    Calculate kth term
    Add kth term to e
    Add 1 to k
```

```
k = 0, e = 0, fact_k = 1, kth_term = 1, EPSILON = .98923
    While kth_term > epsilon:
        If k > 0:
            fact_k *= k
        kth_term = 1 / (fact_k)
        e += kth_term
        k += 1
```

euler.c

```
Initialize k, sum, k_term, and epsilon
While k_term is bigger than epsilon
    Add k_term to sum
    Update iteration (k)
    Calculate new k term
Return sqrt(sum * 6)
```

```
k = 1
e_sum = 0
k_term = 1
EPSILON = .000001

While abs(k_term) > EPSILON:
    E_sum += k_term
    K += 1
    K_term = 1 / (k * k)
Pi = sqrt(e_sum * 6)
```

madhava.c

```
Initialize k, _sum, k_term, EPSILON
As long the current term is larger than epsilon
    Calculate k term
    Add k term to sum
    Update iteration ( k )
```

```
k = 0
_sum = 0
k_term = 1
EPSILON = .00001
While abs(k_term) > EPSILON:
    k_term = (1 / ((-3) ** k)) / (2 * k + 1)
    _sum += k_term
    k += 1
Return _sum
```

Makefile

```
CC = clang
CFLAGS = -Wall Wpedantic -Werror -Wextra
OBJECTS = mathlib-test.o bbp.o euler.o madhava.o e.o viete.o

all: mathlib-test

Mathlib-test: $(OBJECTS)
    $(CC) -lm -o mathlib-test mathlib-test.o
Mathlib-test.o: mathlib-test.c
    $(CC) $(CFLAGS) -c mathlib-test.c
Clean:
    Rm -f *.o
```

mathlib_test.c

```
int main(int argc, char **argv) {
        int opt = 0;

        while ((opt = getopt(argc, argv, OPTIONS)) != -1) {
        bool stats = false;
        bool all = false;
        switch (opt) {
                case 's':
                stats = true;
                case 'a':
                all = true;
                case 'e':
                printf("e() = %16.15lf, M_E = %16.15lf, diff = %16.5lf\n", e(), M_E, fabs(e() - M_E));
                if (stats == true) {
                        printf("e() terms = %6d\n", pi_euler_terms());
                }
                if (all == false) {
                        break;
                }

                case 'b':
                printf("pi_bbp() = %16.15lf, M_PI = %16.15lf, diff = %16.5lf\n", pi_bbp(), M_PI, fabs(pi_bbp() - M_PI));
                if (stats == true) {
                        printf("pi_bbp() terms = %6d\n", pi_bbp_terms());
                }
                if (all == false) {
                        break;
                }

                case 'm':
                printf("pi_madhava() = %16.15lf, M_PI = %16.15lf, diff = %16.15lf\n", pi_madhava(), M_PI, fabs(pi_madhava() - M_PI));
                if (stats == true) {
                        printf("pi_madhava() terms = %6d\n", madhava_pi_terms());
                }
                if (all == false) {
                        break;
                }

                case 'r':
                printf("pi_euler() = %16.15lf, M_PI = %16.15lf, diff = %16.15lf\n", pi_euler(), M_PI, fabs(pi_euler() - M_PI));
                if (stats == true) {
                        printf("pi_euler() terms = %6d\n", pi_euler_terms());
                }
                if (all == false) {
                        break;
                }
```

newton.c

```
Initialize epsilon
Store guess in temp var
While the abs value between the temp var and guess^2
is larger than epsilon
    Update the guess
    Return guess
```

```
Epsilon = .230293
Guess = x
While abs(x - guess * guess) > epsilon:
    Guess = (Guess + (x / guess)) / 2
Return Guess
```