Newton's Method

```
#include <stdio.h>
#include <math.h>

double f(double x)
{
    return x * x - 2;
}

double df(double x)
{
    return 2 * x;
}

double g(double x)
{
    return x - f(x) / df(x);
}

double error(double x)
{
    return x - sqrt(2);
}

double Mn(double en, double en_1)
{
    return en_1 / (en * en);
}

int main()
{
    double x = 1, err[9], M[5];
    int i;

    err[0] = error(x);
    printf("%d\tx = %10.20e\terror = %10.20e\n", 0, x, err[0]);

    for(i = 1; i < 9; i++)
    {
        x = g(x);

        err[i] = error(x);

        printf("%d\tx = %10.20e\terror = %10.20e\n", i, x, err[i]);
    }

    for(i = 1; i < 5; i++)
    {
        M[i - 1] = Mn(err[i], err[i + 1]);
```

```
49
50          printf("\tM%d = %10.20e\n", i, M[i - 1]);
51      }
52
53      return 0;
54  }
```

Output

```
 0   x = 1.00000000000000000000e+00  error = -4.14213562373095145475e-01
 1   x = 1.50000000000000000000e+00  error = 8.57864376269048545254e-02
 2   x = 1.41666666666666674068e+00  error = 2.45310429357159520691e-03
 3   x = 1.41421568627450988664e+00  error = 2.12390141474116944664e-06
 4   x = 1.41421356237468986983e+00  error = 1.59472435257157485466e-12
 5   x = 1.41421356237309514547e+00  error = 0.00000000000000000000e+00
 6   x = 1.41421356237309492343e+00  error = -2.22044604925031308085e-16
 7   x = 1.41421356237309514547e+00  error = 0.00000000000000000000e+00
 8   x = 1.41421356237309492343e+00  error = -2.22044604925031308085e-16
     M1 = 3.33333333333331038872e-01
     M2 = 3.52941176468276551770e-01
     M3 = 3.53522384487245822093e-01
     M4 = 0.00000000000000000000e+00
```

Secant Method

```
1   #include <stdio.h>
2   #include <math.h>
3
4   double f(double x)
5   {
6       return (x * x) - 2;
7   }
8
9   double error(double x)
10  {
11      return x - sqrt(2);
12  }
13
14  double Mn(double err0, double err1)
15  {
16      return fabs(err1) / pow(fabs(err0), 1.618);
17  }
18
19  int main()
20  {
21      int i = 2;
22
23      double q0, q1, x0 = 0, x1 = 1, x, err[9], M[8];
24
25      q0 = f(x0);
```

```
26      q1 = f(x1);
27
28      err[0] = error(x0);
29      err[1] = error(x1);
30
31      printf("%d\tx = %10.20e\terror = %10.20e\n", 0, x0, err[0]);
32      printf("%d\tx = %10.20e\terror = %10.20e\n", 1, x1, err[1]);
33
34      for(i = 2; i < 9; i++)
35      {
36          // compute xi
37          x = x1 - q1 * (x1 - x0) / (q1 - q0);
38
39          // update values
40          x0 = x1;
41          q0 = q1;
42          x1 = x;
43          q1 = f(x);
44
45          // calculate error
46          err[i] = error(x);
47
48          printf("%d\tx = %10.20e\terror = %10.20e\n", i, x, err[i]);
49      }
50
51      for(i = 1; i < 8; i++)
52      {
53          M[i - 1] = Mn(err[i], err[i+1]);
54          printf("\tM%d = %10.20e\n", i, M[i - 1]);
55      }
56
57      return 0;
58  }
```

Output

```
0   x = 0.00000000000000000000e+00  error = -1.41421356237309514547e+00
1   x = 1.00000000000000000000e+00  error = -4.14213562373095145475e-01
2   x = 2.00000000000000000000e+00  error = 5.85786437626904854525e-01
3   x = 1.33333333333333348136e+00  error = -8.08802290397616641116e-02
4   x = 1.40000000000000013323e+00  error = -1.42135623730950122479e-02
5   x = 1.41463414634146333881e+00  error = 4.20583968368193339415e-04
6   x = 1.41421143847487007505e+00  error = -2.12389822507041969857e-06
7   x = 1.41421356205732040578e+00  error = -3.15774739689800298947e-10
8   x = 1.41421356237309536752e+00  error = 2.22044604925031308085e-16
    M1 = 2.43820586365651825744e+00
    M2 = 1.92149900902253506496e-01
    M3 = 8.31407582617175511253e-01
    M4 = 4.09995753025747533549e-01
    M5 = 6.16225988349691222723e-01
```

```
M6 = 4.765133320627968904 76e-01
M7 = 5.229348857928102223 27e-01
```

Floating Point Method

```
#include <stdio.h>
#include <math.h>

double f(double x)
{
    return x * x - 2;
}

double df(double x)
{
    return 2 * x;
}

double d2f(double x)
{
    return 2.0;
}

double h(double x)
{
    return x - ((f(x) * df(x)) / ((df(x) * df(x)) - (f(x) * d2f(x))));
}

double error(double x)
{
    return x - sqrt(2);
}

int main()
{
    double x = 1, err[9];

    int i;

    err[0] = error(x);
    printf("%d\tx = %10.20e\terror = %10.20e\n", 0, x, err[0]);

    for(i = 1; i < 9; i++)
    {
        x = h(x);

        err[i] = error(x);

        printf("%d\tx = %10.20e\terror = %10.20e\n", i, x, err[i]);
    }
```

```
46
47      return 0;
48  }
```

Output

```
 0   x = 1.00000000000000000000e+00  error = -4.14213562373095145475e-01
 1   x = 1.33333333333333325932e+00  error = -8.08802290397618861562e-02
 2   x = 1.41176470588235281056e+00  error = -2.44885649074233491262e-03
 3   x = 1.41421143847487007505e+00  error = -2.12389822507041969857e-06
 4   x = 1.41421356237150019908e+00  error = -1.59494639717649988597e-12
 5   x = 1.41421356237309492343e+00  error = -2.22044604925031308085e-16
 6   x = 1.41421356237309514547e+00  error = 0.00000000000000000000e+00
 7   x = 1.41421356237309492343e+00  error = -2.22044604925031308085e-16
 8   x = 1.41421356237309514547e+00  error = 0.00000000000000000000e+00
```