

```

1  #include "../include/get_sinx.h"
2
3  double est_sinx(double x, int taylor_deg);
4
5  double get_sinx(double x) {
6      // Handel x < 0.
7      if (x<0) {
8
9          if (x < 2*PI) {
10             x = 2 * PI * ((x / (2 * PI)) - ceil(x / (2 * PI)));
11         }
12
13         if (x <=-3*PI/2) {
14             return est_sinx(x+2*PI, T_DEG);
15         }
16
17         if (x <= -PI) {
18             return est_sinx(-(x+PI), T_DEG);
19         }
20
21         if (x <= -PI/2) {
22             return -est_sinx(PI + x, T_DEG);
23         }
24         else {
25             return -est_sinx(-x, T_DEG);
26         }
27     }
28
29     // Handel x > 0.
30     if (x>=0) {
31         if (x >= 2*PI)
32         {
33             x = 2*PI*((x / (2 * PI)) - floor(x / (2 * PI)));
34         }
35
36         if (x >= PI)
37         {
38             return -est_sinx(x - PI, T_DEG);
39         }
40
41         if (x >= PI/2)
42         {
43             return est_sinx(PI - x, T_DEG);
44         }
45         else
46         {
47             return est_sinx(x, T_DEG);
48         }
49     }
50 }
51
52 // Estimate sin(x) using a Taylor series of taylor_deg degree.
53 double est_sinx(double x, int taylor_deg) {
54     double old_num = x;
55     double new_num = 0;
56     int old_denom = 1;
57     int new_denom = 0;
58     double sum = old_num/old_denom;
59
60     for (int n = 1; n <= taylor_deg; n++) {
61         new_num = -(x)*(x)*old_num;

```

```
62 |         new_denom = old_denom * 2 * n * (2 * n + 1);
63 |         sum = sum + new_num/new_denom;
64 |         old_denom = new_denom;
65 |         old_num = new_num;
66 |     }
67 |     return sum;
68 | }
69 |
70 |
71 |
```