```c
#include "../include/get_sinx.h"

double est_sinx(double x, int taylor_deg);

double get_sinx(double x) {
    // Handel x < 0.
    if (x<0) {

        if (x < 2*PI) {
            x = 2 * PI * ((x / (2 * PI)) - ceil(x / (2 * PI)));
        }

        if (x <=-3*PI/2) {
            return est_sinx(x+2*PI, T_DEG);
        }

        if (x <= -PI) {
            return est_sinx(-(x+PI), T_DEG);
        }

        if (x <= -PI/2) {
            return -est_sinx(PI + x, T_DEG);
        }
        else {
            return -est_sinx(-x, T_DEG);
        }
    }

    // Handel x > 0.
    if (x>=0) {
        if (x >= 2*PI)
        {
            x = 2*PI*((x / (2 * PI)) - floor(x / (2 * PI)));
        }

        if (x >= PI)
        {
            return -est_sinx(x - PI, T_DEG);
        }

        if (x >= PI/2)
        {
            return est_sinx(PI - x, T_DEG);
        }
        else
        {
            return est_sinx(x, T_DEG);
        }
    }
}

// Estimate sin(x) using a Taylor series of taylor_deg degree.
double est_sinx(double x, int taylor_deg) {
    double old_num = x;
    double new_num = 0;
    int old_denom = 1;
    int new_denom = 0;
    double sum = old_num/old_denom;

    for (int n = 1; n <= taylor_deg; n++) {
        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
```