Answer on Question #52304-Programming-Other

Write a function to raise a number to a power when both are floating point number

Answer

We can use $pow(double\ base, double\ exponent)$ library function.

Or we can write it using identity

$$x^y = \exp(\ln x^y) = \exp(y \cdot \ln x).$$

So in this way we alsu need functions for calculating $\exp(\mathrm{x})$ and $\ln x$.

```
#define MAX_DELTA_DOUBLE 1.0E-15

#define EULERS_NUMBER 2.718281828459045


double MathAbs_Double (double x) {

   return ((x >= 0) ? x : -x);

}


int MathAbs_Int (int x) {

   return ((x >= 0) ? x : -x);

}


double MathPow_Double_Int(double x, int n) {

   double ret;

   if ((x == 1.0) || (n == 1)) {

      ret = x;

   } else if (n < 0) {

      ret = 1.0 / MathPow_Double_Int(x, -n);

   } else {

      ret = 1.0;

      while (n--) {

         ret *= x;

      }
```

```
    }
    return (ret);
}

double MathLn_Double(double x) {
    double ret = 0.0, d;
    if (x > 0) {
        int n = 0;
        do {
            int a = 2 * n + 1;
            d = (1.0 / a) * MathPow_Double_Int((x - 1) / (x + 1), a);
            ret += d;
            n++;
        } while (MathAbs_Double(d) > MAX_DELTA_DOUBLE);
    } else {
        printf("\nerror: x < 0 in ln(x)\n");
        exit(-1);
    }
    return (ret * 2);
}

double MathExp_Double(double x) {
    double ret;
    if (x == 1.0) {
        ret = EULERS_NUMBER;
    } else if (x < 0) {
        ret = 1.0 / MathExp_Double(-x);
    } else {
        int n = 2;
        double d;
        ret = 1.0 + x;
        do {
            d = x;
            for (int i = 2; i <= n; i++) {
                d *= x / i;
            }
            ret += d;
            n++;
        } while (d > MAX_DELTA_DOUBLE);
    }
    return (ret);
}

double MathPow_Double_Double(double x, double a) {
    double ret;
    if ((x == 1.0) || (a == 1.0)) {
        ret = x;
    } else if (a < 0) {
        ret = 1.0 / MathPow_Double_Double(x, -a);
    } else {
        ret = MathExp_Double(a * MathLn_Double(x));
    }
    return (ret);
}
```