**Bisection Method**

**Bisection Method**

二分法是一种查找方程零值点的方法，它反复地对一个区间均分，直到找到满足要求的零值点。二分法简单，但是收敛速度较慢，为线性收敛。

设有函数在区间上连续，并且与异号，则在区间上至少有一个解。在迭代的每一步中，都会计算中间点，将原始区间分为两个子区间和。在第一次迭代中，如果刚好，则点就是在区间上的零值点（不过这一般是不太可能的）；如果，就只有下面的两种情况了：

1. 如果与同号，则新的迭代区间变为，并且将赋值于

2. 如果与反号，则新的迭代区间变为，并且将赋值于

直到达到计算的要求：提前设置精度要求eps，当区间的绝对误差小于eps时，结束计算。

这里给出提前确定所需精度后（精确到小数点后p位），估计计算迭代次数的公式



**伪代码**

INPUT: 函数、区间，精度eps

if f(a)\*f(b) > 0

print\*, " error: f(a) \* f(b) < 0 is not satisfied! "

Do while ( b-a ) > eps

c ← (a + b) / 2

get f(c)

if f(c)==0 stop end

if f(a)\*f(c) > 0 then

a ← c; f(a) ← f(c)

else

b ← c; f(b) ← f(c)

end if

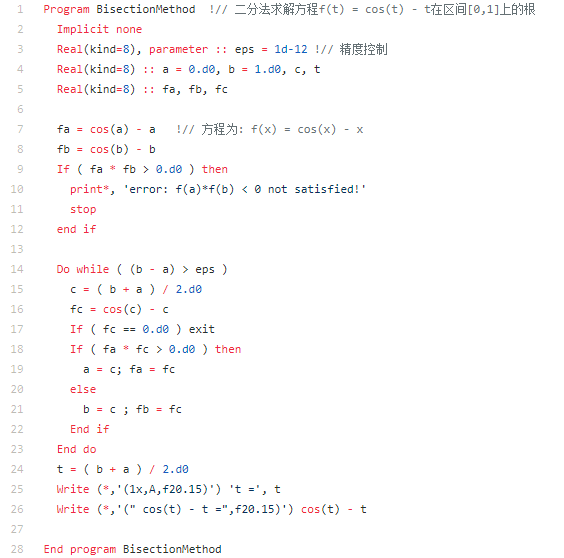
end do

**实例**

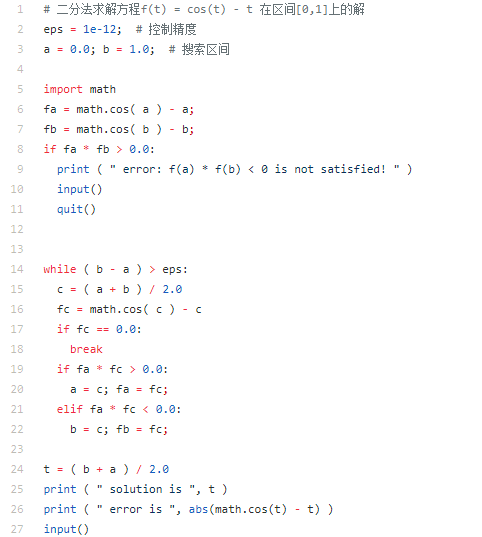
设函数在区间上有一零值点，求其零值点。

**源代码**

**Fortran**



**Python**



运行结果如下

