用分数形式精确表达有理数和循环无理数

学过计算机编程的就知道，在计算机中，浮点数是不可能用浮点数精确的表达的，如果你需要精确的表达这个小数，我们最好是用分数的形式来表示，而且有限小数或无限小数都是可以转化为分数的形式。比如下面的几个小数：

0.3333(3) = 1/3的(其中括号中的数字是表示循环节)
0.3 = 3 / 10
0.25 = 1 / 4
0. 285714(285714) = 2 / 7

为了简化编程，在这里，我们假定输入的数据都是以0.开始的，没有负数。
（1）、对于有限小数的情况很好分析，我们只要得到小数的位数n，然后用这个小数除以10^n就能得到比如小数形式为0.a1a2a3a4...an = a1a2a3a4....an / 10^n然后化简为最简分式就能得到。
（2）、对于无限小数，情况要复杂许多，假定无限小数为
0.a1a2....an(b1b2....bm)，我们做如下转换有

X = 0.a1a2....an(b1b2....bm)
X * 10^n=a1a2....an + 0. b1b2....bm
设Y = 0. b1b2....bm有
10^m * Y = b1b2....bm + 0.b1b2....bm
=b1b2....bm + Y
所以Y = b1b2....bm / (10^m - 1)带入上面得到
X = (a1a2....an + Y) / 10^n = ((a1a2....an) * (10^m - 1) + (b1b2....bm)) / ((10^m - 1) * 10^n)

由此我们可以得到无限小数的精确表达式，下面就是代码实现：

```
#include <iostream>
#include <string>

using namespace std;

unsigned long long GCD(unsigned long long a, unsigned long long b);
/**
*    author: w397090770
*    Date: 2012.08.31
*    Email:wyphao.2007@163.com
```

```
*   仅用于学习交流，转载请注明这些标识。
**/
void floatPrecisionExpress(string numberStr){
  //寻找 (
  string::size_type start = 0;
  //寻找 )
  string::size_type end = 0;
  //标记是否找到 ( 符号
  bool isFind = false;
  //记录字符串的长度
  int len = 0;
  int m = 0, n = 0;
  //分子，分母
  unsigned long long molecular = 0, denominator = 1;
  int i = 0;

  //
  unsigned long long gcd = 1;
  start = numberStr.find('(', 0);
  end =  numberStr.find(')', 0);

  //只有找到 ( 和 ) 才是对的，要么都不找到，找到一个地情况下是错误的，直接返回
  //当然我这里假设了用户输入的是0.XXXX格式的字符串，也就是一定是以0.开头的，
  //不考虑以别的开始的
  if(start == string::npos && end == string::npos){
    isFind = false;
  }else if(start != string::npos && end != string::npos){
    isFind = true;
  }else{
    cerr << "Input Error!" << endl;
    return;
  }

  //有限小数
  if(!isFind) {
    len =  numberStr.length();
    n = len - 2;  //2是除去 0.

    //计算分子
    for(i = 2; i < len; i++){
      molecular = molecular * 10 +  numberStr[i] - '0';
    }
    //cout << molecular << endl;

    //计算分母
    for(i = 0; i < n; i++){
```

```
    denominator *= 10;
   }
   //cout << molecular << "₩n" << denominator << endl;
   //将分子、分母化简为最简式,得到两数的最大公约数
   gcd= GCD(molecular, denominator);
   cout << "浮点数" <<  numberStr << "的分数精确表示为: " << molecular / gcd << "/" << denom
inator / gcd << endl;
  }else{
   n = start - 2;  //2是除去 0.
   m = end - start - 1;
   //cout << n << "₩t" << m << endl;
   unsigned long long temp1 = 0, temp2 = 0, temp3 = 1, temp4 = 1;
   for(i = 2; i < start; i++){
    temp1 = temp1 * 10 + numberStr[i] - '0';
   }

   for(i = start + 1; i < end; i++){
    temp2 = temp2 * 10 + numberStr[i] - '0';
   }

   //cout << temp1 << "₩t" << temp2 << endl;
   for(i = 0; i < n; i++){
    temp3 *= 10;
   }

   for(i = 0; i < m; i++){
    temp4 *= 10;
   }

   //cout << temp1 << "₩t" << temp2 << "₩t" << temp3 << "₩t" << temp4 << endl;
   molecular = temp1 * (temp4 - 1) + temp2;
   denominator = (temp4 - 1) * temp3;
   gcd= GCD(molecular, denominator);
   //cout << gcd << endl;
   cout << "浮点数" <<  numberStr << "的分数精确表示为: " << molecular / gcd << "/" << denom
inator / gcd << endl;

  }
}

unsigned long long GCD(unsigned long long a, unsigned long long b){
  if(a < b){
   return GCD(b, a);
  }

  if(b == 0){
```

```
    return a;
  }else{
    if(a & 0x1){  //奇数
      if(b & 0x1){
        return GCD(b, a - b);
      }else{
        return GCD(a, b >> 1);
      }
    } else{
      if(b & 0x1){
        return GCD(a >> 1, b);
      }else{
        return GCD(a >> 1, b >> 1) << 1;
      }
    }
  }
}

int main(){
  floatPrecisionExpress("0.285714(285714)");
  floatPrecisionExpress("0.33(3)");
  floatPrecisionExpress("0.25");
  floatPrecisionExpress("0.30");
  floatPrecisionExpress("0.3(000)");
  floatPrecisionExpress("0.3333(3333)");
  return 0;
}
```

程序运行结果：