2022-程序设计2-第一次习题课

助教-张舒恒

2022/4/23

浮点数求高精度幂

编程题　　代码行：175 | 正确率：0.97 | 完成时间：31分钟　　源文件提交

【问题描述】有一个实数 R (0.0 < R < 99.999) , 要求写程序精确计算 R 的 n 次方。n 是整数并且 0 < n <= 25。

【输入形式】每行包括一个实数R和指数n，中间以空格隔开，其中R最多10位，可能有前导0和尾0如0.00010，可能是整数如10，0 < n <= 25。

【输出形式】对于每行输入，要求输出相应的一行，该行是完全精确的 R 的 n 次方。输出需要去掉前导0和尾0，如.0001是合法输出。如果输出是整数，不能输出小数点。

判断有无小数点

```
for(i=0;s[i]!='\0';i++)
{
    if(s[i]=='.')
    {
        flag=1;
        break;
    }
    point++;
}
```

逐位相乘，并且进位

```
for(i=1;i<=c;i++)
{
    memset(temp,0,sizeof(temp));//存完一次结果之后，下一次用之前的清零
    for(j=0;j<len;j++)
        for(k=0;k<len2;k++)
            temp[k+j]+=b[k]*a[j];
    for(j=0;j<len+len2;j++)
    {
        temp[j+1]+=temp[j]/10;
        temp[j]=temp[j]%10;
    }
    memcpy(b,temp,sizeof(temp));
    len2+=len;
}
```

去除前导零，逆序输出整数部分，判断小数部分是否为零，有小数部分逆序输出小数部分

```
while(b[len2]==0)//去除前导零
            len2--;
        for(i=len2;i>=point;i--)//逆序输出整数部分
            printf("%d",b[i]);
        for(i=0;i<point;i++)//判断小数部分是否为零
            if(b[i]!=0)
                break;
        if(i<point)//有小数部分
        {
            printf(".");
            for(j=point-1;j>=i;j--)//逆序输出小数部分
                printf("%d",b[j]);
        }
```

孙子问题

编程题 代码行：64 | 正确率：0.92 | 完成时间：36分钟 源文件提交

【问题描述】对于给定的正整数a1, a2, ... an，问是否存在正整数b1, b2, ... bn，使得对于任意的一个正 整数N，如果用N 除以a1 的余数是p1，用N 除以a2 的余数是p2……用N 除以an 的余数是 pn，那么M = p1 * b1 + p2 * b2 + ... + pn * bn 能满足M 除以a1 的余数也是p1，M 除以a2 的余数也是p2……M 除以an 的余数也是pn。如果存在，则输出b1, b2, ... bn。题中1 <= n <= 10，a1, a2, ... an 均不大于50。

【输入形式】输入包括多组测试数据，每组数据包括一行。在每组数据中，首先给出ai的个数n (1 <= n <= 10)，然后给出n个不大于50的正整数a1, a2, … an。最后一组测试数据中n = 0，表示输入的结束，这组数据不用处理。

【输出形式】对于每一组测试数据，输出一行，如果存在正整数b1, b2, … bn满足题意，则输出这n个正整数，相邻的正整数之间用一个空格隔开；否则，输出“NO”。

这里引用一位同学的做法

考虑两个同余方程

$$N \equiv k_1 (\mod a_1)$$
$$N \equiv k_2 (\mod a_2)$$

改写为不定方程

$$N = k_1 + a_1 x = k_2 + a_2 y$$
$$a_1 x - a_2 y = k_2 - k_1$$

根据不定方程的解乏，先通过扩展欧几里得算法得到解

$$a_1 x_0 + a_2 y_0 = d = (a_1, a_2)$$
$$x = x_0 (k_2 - k_1)/d$$
$$N = k_1 + a_1 x = k_1 + a_1 x_0 k_2 / d - a_1 x_0 k_1 / d = \frac{a_2 y_0}{d} \cdot k_1 + \frac{a_1 x_0}{d} \cdot k_2$$

所以可以将上述两个方程等价为

$$N \equiv \frac{a_2 y_0}{d} \cdot k_1 + \frac{a_1 x_0}{d} \cdot k_2 (\mod a_1 a_2 / d)$$

那么，如果假设我合并了前i个方程，得到了以下的等效方程

$$N \equiv b_1 k_1 + b_2 k_2 \cdots b_i k_i = S (\mod M)$$

希望将它和$N \equiv k_{i+1} (\mod a_{i+1})$合并。
根据之前合并两个方程的结论，先用扩展欧几里得算法得到两个解$M x_0 + a_{i+1} y_0 = (M, a_{i+1}) = d$，有

$$N \equiv \frac{y_0 a_{i+1}}{d} S + \frac{x_0 M}{d} k_{i+1} = \frac{y_0 a_{i+1}}{d} b_1 k_1 + \cdots \frac{y_0 a_{i+1}}{d} b_i k_i + \frac{x_0 M}{d} k_{i+1} (\mod M a_{i+1}/d)$$

于是，我们得到了新的一组解

$$b'_j = \frac{y_0 a_{i+1}}{d} \cdot b_j (1 \le j \le i)$$
$$b'_{i+1} = \frac{x_0 M}{d}$$
$$M' = M a_{i+1}/d$$

初始时
$N \equiv b_1 = 1 \cdot k_1 (\mod M = a_1)$
反复更新即可得到答案。

扩展欧几里得原理

首先，有 $\begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}\cdot\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}a\\b\end{bmatrix}$

令 $q=[\frac{a}{b}]$，即 q 为 $a\div b$ 的整数部分，构建矩阵 $\begin{bmatrix}0 & 1\\1 & -q\end{bmatrix}$

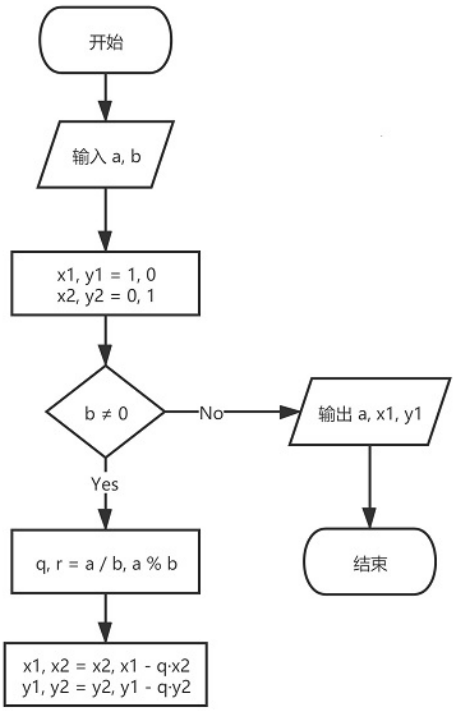
将等式两端同时左乘上述矩阵，则有 $\begin{bmatrix}0 & 1\\1 & -q\end{bmatrix}\cdot\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}b\\a-q\cdot b\end{bmatrix}=\begin{bmatrix}b\\a\ mod\ b\end{bmatrix}$

再令 $q'=[\frac{b}{a\ mod\ b}]$，重复上述步骤

最终有 $\begin{bmatrix}x_1 & y_1\\x_2 & y_2\end{bmatrix}\cdot\begin{bmatrix}a\\b\end{bmatrix}=\begin{bmatrix}gcd(a,b)\\0\end{bmatrix}$

即求出 $x_1\cdot a+y_1\cdot b=gcd(a,b)$

算法流程图




```
long long exgcd(long long a, long long b, long long &x0, long long &y0) {
    if(!b) {
        x0 = 1; y0 = 0;
        return a;
    }
    long long d = exgcd(b, a % b, x0, y0);
    std::swap(x0, y0);
    y0 -= (a/b) * x0;
    return d;
}
```

浮点数格式

编程题 代码行：46 | 正确率：0.97 | 完成时间：26分钟 源文件提交

【问题描述】输入n个浮点数，要求把这n个浮点数重新排列后再输出

【输入形式】第1行是一个正整数n(n<=10000)，后面n行每行一个浮点数，每个浮点数均有小数点且长度不超过50位。

【输出形式】n行，每行一个浮点数。要求浮点数的顺序和输入保持一致，每个浮点数的小数点在同一列上，首列不会全部是空格。

每输入一行后判断小数点位置，如果超过当前最大的小数点位置则更新，输出的时候不足的位置补空格。

```
string *numofnum = new string[num];
int shuzu[10006] = { 0 };
for (int i = 0; i < num; i++)
{
    cin >> numofnum[i];
    for (int j = 0; j<numofnum[i].length(); j++)
    {
        shuzu[i]++;//记录整数位数
        if (numofnum[i][j] == '.')
        {
            if (j >= max)
                max =j+1;
            break;
        }
    }
}
for (int i = 0; i < num; i++) {
    for (int j = max - shuzu[i]; j > 0; j--)
    {
        cout << " ";
    }
    cout << numofnum[i] << endl;
}
```

词典

编程题 代码行：43 | 正确率：0.98 | 完成时间：37分钟 源文件提交

【问题描述】你旅游到了国外的一个城市，却不能理解那里的语言。不过幸运的是，你有一本词典可以帮助你。词典中包含不超过100000 个词条，而且在词典中不会有某个外语单词出现两次。现在给你一个由外语单词组成的文档，文档不超过100000 行，而且每行只包括一个外语单词。所有单词都只包括小写字母，而且长度不会超过10。请你把这个输入：首先输入一个词典，每个词条占据一行。每一个词条包括一个英文单词和一个外语单词，两个单词之间用一个空格隔开。词典之后是一个空行，然后是外语文档。你需要将外语文档翻译成英文，每行输出一个英文单词。如果某个外语单词不在词典中，就把这个单词翻译成"eh"。

利用sscanf将字符串分割成外语单词和英语单词，利用std::map<string,string>将单词以映射关系存起来，之后每次读入一个外语单词，则在map中进行查找。

```
char a[30],b[15],c[15];
while(gets(a) && a[0] != '\0')
{
    sscanf(a,"%s%s",b,c);
    mp[c]=b;
}
map<string,string>::iterator p;
while(gets(a))
{
    p = mp.find(a);
    if(p == mp.end())
        printf("%s\n","eh");
    else
        cout << mp[a] << endl;
}
```

两倍

编程题　1　代码行：- | 正确率：1.00 | 完成时间：38分钟　源文件提交

【问题描述】给定2到15个不同的正整数，你的任务是计算这些数里面有多少个数对满足：数对中一个数是另一个数的两倍。

【输入形式】一串正整数

【输出形式】一个数字

【样例输入】12 3 5 7 6 24 2 18 20 9

【样例输出】4

【样例说明】(3,6)；(6,12)；(9,18)；(12,24)；

读入数组后进行排序，vis[] 记录数字的两倍的位置，每访问一个数字，若vis数组相应位置为True则说明存在两倍关系，同时将该数字的两倍的vis置为True。

```
memset(vis, 0, sizeof(vis));
p = ans = 0;
while (scanf("%d",&x) != EOF) {
    p++;
    a[p] = x;
}
sort(a + 1, a + p + 1);
for (int i = 1; i <= p; i++) {
    if (vis[a[i]] == 1)
        ans++;
    vis[2 * a[i]] = 1;
}
cout << ans << endl;
```