github链接

github

题目摘要

Calculate a + b and output the sum in standard format -- that is, the digits must be separated into groups of three by commas (unless there are less than four digits).Each input file contains one test case. Each case contains a pair of integers a and b where -1000000 <= a, b <= 1000000. The numbers are separated by a space.

换句话说就是输入两个给定范围的整数，求其和，最后将和以从个位到高位依次每三个一组，中间由逗号隔开输出。

思路分析

刚开始看到题第一感觉是不是要用转换的方法或者使用一些技巧，但是能力有限，后来想一想还是按部就班思考，只能用最实际的方法，想的就很直接，就是把要输出的整数拆分成各个部分，然后再按要求拼接起来输出，所以操作过程中有两大关键过程：**1)**计算出每一部分的三个数 **2)**按要求输出，并且输出要符合实际。然后就开始操作了，过程中遇到了这么一些问题：

- 当和值为**2,000,000**时，输出结果是**2, 0, 0** 最开始也是郁闷了半天不知道为什么，然后就意识到应该是位 的问题，空位要用要用**0**补位，补位的方法老师上课有讲过。 错误代码

```
printf("%d,%3d,%3d",m,k,left);
```

正确代码

```
printf("%d,%03d,%03d",m,k,left);
```

- 负数问题。因为是直接对**sum**取整数或者是余数，如果**sum**是负数的话运算时也会带有符号，然后就会出现这种尴尬局面：**sum**为**-2,222,222**时，输出为**-2,-222.-222**。处理这种情况先开始我就是直接把负数情况给列出来单独考虑，于是就增加了两条**if**语句，后来又有了系统的想法，就把正负情况合并考虑。

```
if(sum<=-1000000) printf("%d,%03d,%03d",m,-k,-left);
```

后来又有了系统的想法，就把正负情况合并考虑了。

- 再就是一些写法上面的问题，还有表示上有的时候会把量写错。后来改进的时候也出现了一些小错误导致有几组数据错误，好在最后找出了错误。

代码截图

1月26日 19:57	答案正确	20	1001	C (gcc 4.7.2)	9	384
1月26日 19:42	部分正确	9	1001	C (gcc 4.7.2)	7	384
1月25日 22:31	部分正确	18	1001	C (gcc 4.7.2)	7	384
1月25日 22:27	部分正确	18	1001	C (gcc 4.7.2)	7	384
1月25日 22:25	部分正确	9	1001	C (gcc 4.7.2)	12	384
1月25日 22:18	答案正确	20	1001	C (gcc 4.7.2)	9	452
1月25日 22:18	部分正确	19	1001	C (gcc 4.7.2)	5	384
1月25日 21:20	答案正确	20	1001	C (gcc 4.7.2)	11	264
1月25日 20:56	答案正确	20	1001	C (gcc 4.7.2)	4	384
1月24日 21:07	部分正确	11	1001	C (gcc 4.7.2)	6	384
1月24日 20:56	部分正确	9	1001	C (gcc 4.7.2)	7	384

提交列表

时间	结果	得分	题目	语言	用时(ms)	内存(kB)	用户
1月26日 19:57	答案正确	20	1001	C (gcc 4.7.2)	9	384	hua

测试点

测试点	结果	用时(ms)	内存(kB)	得分/满分
0	答案正确	2	256	9/9
1	答案正确	3	256	1/1
10	答案正确	5	268	1/1
11	答案正确	3	256	1/1
2	答案正确	4	384	1/1
3	答案正确	4	256	1/1
4	答案正确	9	384	1/1
5	答案正确	4	256	1/1
6	答案正确	4	256	1/1
7	答案正确	3	384	1/1
8	答案正确	2	256	1/1
9	答案正确	3	256	1/1

```
int a,b,sum,temp,m,k,left;
scanf("%d %d",&a,&b);
sum=a+b;
temp=abs(sum);
if(temp>=1000000)
{
  m=temp/1000000;
  k=temp/1000%1000;
  left=temp%1000;
  if(sum>0) printf("%d,%03d,%03d",m,k,left);
  else printf("%d,%03d,%03d",-m,k,left);
}
else if(temp>=1000)
{
  k=temp/1000;
  left=temp%1000;
if(sum>0) printf("%d,%03d",k,left);
else printf("%d,%03d",-k,left);
}
else printf("%d",sum);
return 0;
```

总结

因为这个方法比较直接，只要在拆分计算的时候不出错，错误就会比较容易出现在输出的格式上，所以最后解决起来思路还是蛮清晰的，提交的次数也不是很多。

先开始的时候因为负数问题，单纯为了解决问题就直接也把负数的情况都列出来了，所以最后写出来的程序也很长。后来想一想发现正负情况其实可以合并，可以先不考虑正负，在输出的时候再通过判断改变输出的符号就行。这样写出来的程序比较清晰，后来检查的时候也比较直观，能节省出相当多的时间。