1. 运算表达要规范，使用整数就写5，浮点数一定要表示成5.0，这样最不容易出错
2. 不借助多余变量的交换如下，但普通的三变量更常用，切忌多余的卖弄技巧

```
a = a + b;
b = a - b;
a = a - b;
```

3. keep it simple and stupid，解决问题即可，黑盒测试的特点或许可以加以利用

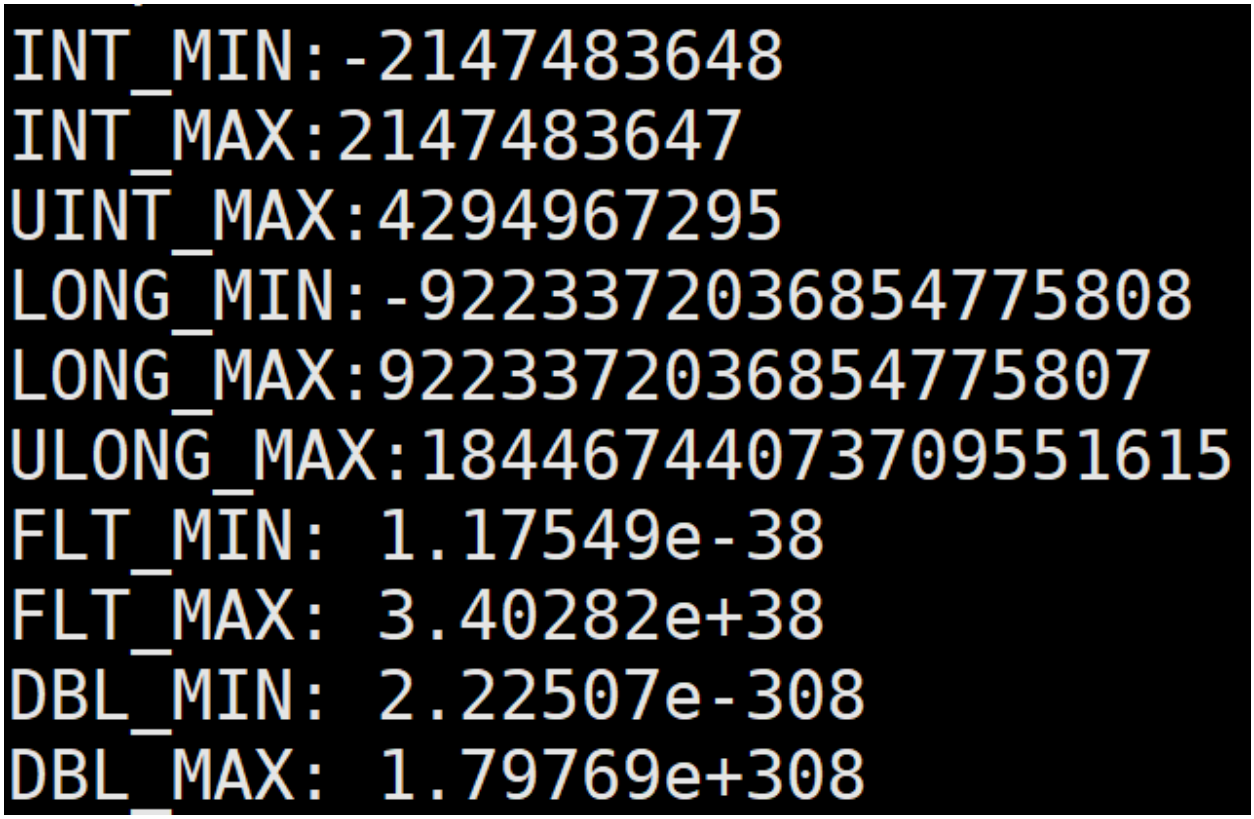
```
scanf("%d%d", &a, &b);
printf("%d %d\n", b, a);
```

4. C语言中inf与nan
    1. inf：1.0/0.0, log(0)等
        - 表示“**无穷大**”，超出浮点数的表示范围（溢出，即阶码部分超过其能表示的最大值）
        - +inf大于任何数（除了它自己和nan），-inf小于任何数（除了它自己和nan）
        - 得到inf时就查看是否有溢出或者除以0
    2. nan： 0.0/0.0, long(-1), sqrt(-1)等
        - not a number，表示“**无效数字**”
        - nan是**无序的**（unordered），无法对其进行逻辑运算。它不大于、小于或等于任何数（包括它自己），将<，>，<=，和>=作用于nan产生一个exception
        - 得到nan时就查看是否有非法操作，如果表达式中含有nan，那么表达式的结果为nan
    3. nan和inf的判断
        - #include <math.h>
        - int isfinite(x)，判断x是否有限，是返回1，其它返回0；
        - int isnormal(x)，判断x是否为一个数（非inf或nan），是返回1，其它返回0；
        - int isnan(x)，当x时nan返回1，其它返回0；
        - int isinf(x),当x是正无穷是返回1，当x是负无穷时返回-1，其它返回0。有些编译器不区分
5. 注意到4中对nan和inf的讨论在浮点数域进行

```
printf("1.0/0.0: %f\n", 1.0/0.0); //inf
printf("1.0/0.0: %d\n", 1.0/0.0); //1
printf("1.0/0.0: %d\n", 1/0); //Floating point exception (core dumped)
printf("1.0/0.0: %f\n", 1/0); //Floating point exception (core dumped)

printf("0.0/0.0: %f\n", 0.0/0.0); //nan
printf("0.0/0.0: %d\n", 0.0/0.0); //1
printf("0.0/0.0: %d\n", 0/0); //Floating point exception (core dumped)
printf("0.0/0.0: %f\n", 0/0); //Floating point exception (core dumped)
```

6. 各种数据类型的界限



- 关于整型的详细设置可以查看 /usr/include/limits.h中的设置
- 关于浮点数的要#include <float.h>，具体位置未找到