```c
#include <stdio.h>
int main (){
   printf("%d\n",-128 < 127);
   printf("%d\n", -32678 < 32677);
   printf("%d\n",-2147483648 < 2147483647);          //溢出
   printf("%d\n",-2147483648LL < 2147483647LL);     //解决办法1
   printf("%d\n",(int)-2147483648 < 2147483647);    //解决办法2
   getchar();
   return 0;
}
```

注明：使用不同的 C 编译器第三行输出结果会有不同

MSVC： 

GCC： 

解释原因：

如果你在Visual Studio (VS) 中运行代码，发现最后一行的输出结果是 `0` 而不是 `1`，这可能是由于整数溢出或类型转换的问题导致的。具体来说，32位有符号整数 `-2147483648` 在某些编译器和环境中可能会被视为常量 `2147483648`，然后取负号，但由于 `2147483648` 已经超出了32位有符号整数的最大值 `2147483647`，这会导致溢出。

在C语言中，当一个常量超出了其类型所能表示的范围时，行为是未定义的。在某些编译器中，`-2147483648` 可能会被处理成 `2147483648` 的补码形式，这实际上是一个非常大的正数，导致 `-2147483648` 实际上变成了 `2147483648`，从而使得 `-2147483648 < 2147483647` 变成了 `2147483648 < 2147483647`，结果为假（false），即 `0`。

为了确保正确的结果，可以显式地指定 `-2147483648` 为一个 `int` 类型的常量，或者使用 `(int)` 强制类型转换来避免这个问题。例如：

```c
printf("%d\n",(int)-2147483648 < 2147483647);
```