

# Information Encoding

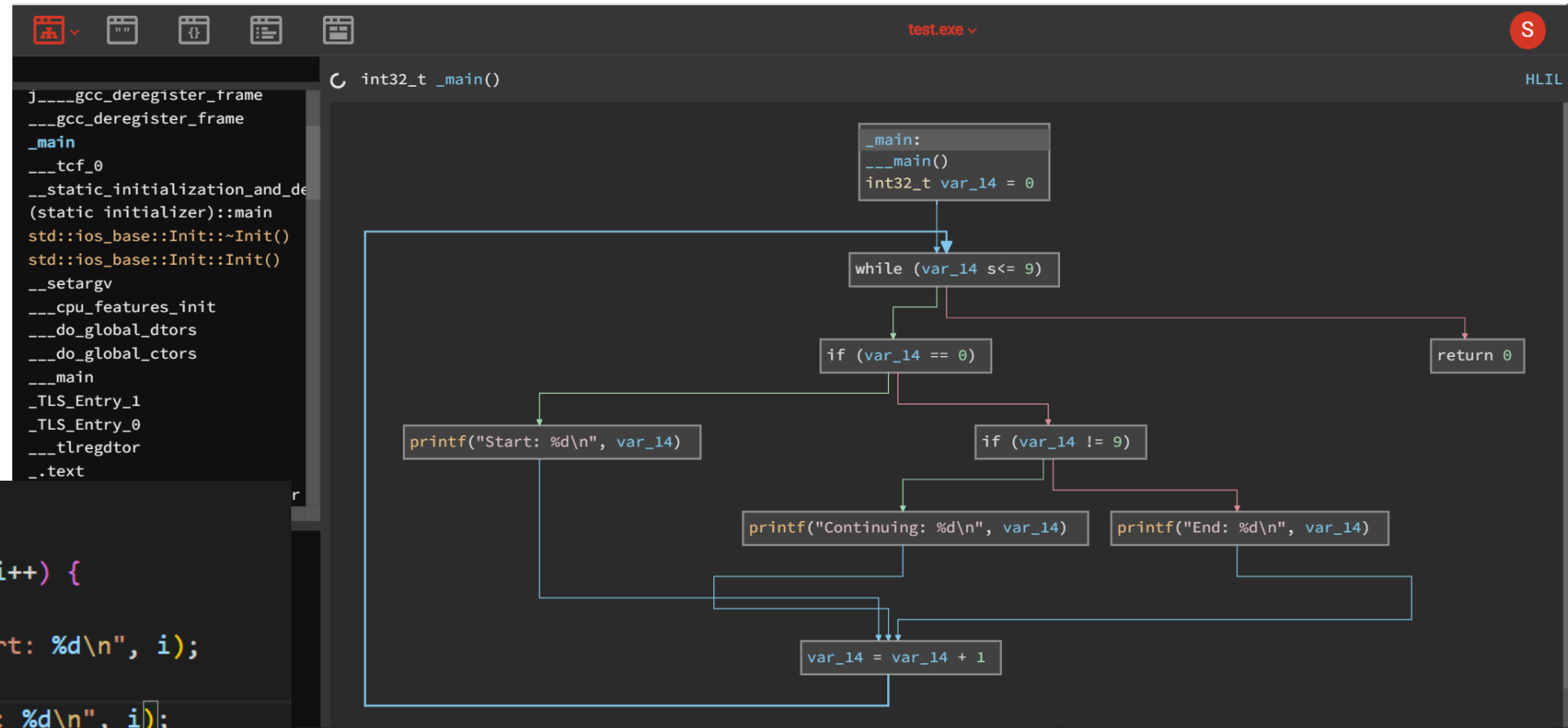
许珈铭

2023-09-20

# New Tool & New Tutorial

- Binary Ninja
  - Disassemble
  - Discompile
  - Analyze

```
1  #include<iostream>
2  int main() {
3      for (int i = 0; i < 10; i++) {
4          if (i == 0) {
5              std::printf("Start: %d\n", i);
6          } else if (i == 9) {
7              std::printf("End: %d\n", i);
8          } else {
9              std::printf("Continuing: %d\n", i);
10         }
11     }
12 }
```



# New Tool & New Tutorial

- The Missing Semester of Your CS Education (MIT)

- **1/13:** [课程概览与 shell](#)
- **1/14:** [Shell 工具和脚本](#)
- **1/15:** [编辑器 \(Vim\)](#)
- **1/16:** [数据整理](#)
- **1/21:** [命令行环境](#)
- **1/22:** [版本控制\(Git\)](#)
- **1/23:** [调试及性能分析](#)
- **1/27:** [元编程](#)
- **1/28:** [安全和密码学](#)
- **1/29:** [大杂烩](#)
- **1/30:** [提问&回答](#)

Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✓	Update	✓	Chinese	✓
Solution	✗	Update	✓	Chinese	✓
Solution	✗	Update	✓	Chinese	✓

# 目录

- 整型
  - 编码方式
  - 运算
- 浮点型
  - 编码方式
  - 运算律
  - INF & NAN

# 整型-编码

- 信息 = 位 + 上下文
- 位相关
  - 同宽类型有符号与无符号互转，位表示不变，实际表示值可能变化
  - 宽  $w$  bit 的有符号数与无符号数在位表示相同的情况下模  $2^w$  同余
- 上下文相关
  - 补码（默认编码方式）、反码、原码
  - 原码和反码的位级运算和上下文结果不符
  - 反码中 0 有 +0 和 -0 两种表示

# 整型-运算

- 溢出考虑corner case
- Signed (补码)
  - $1 < < 31 - 1 \ \& \ -1 < < 31$
  - $0 \ \& \ -1$
- Signed (反码)
  - $+0 \ \& \ -0$
- Signed (原码)
  - $1 < < 31 - 1 \ \& \ - (1 < < 31 - 1)$
  - $+0 \ \& \ -0$
- Unsigned
  - $1 < < 31$
  - $0$

# 整型-运算

- unsigned short -> int
- short -> unsigned int
- **都是先变大小再变符号**
- unsigned short -> int: 零扩展
- short -> unsigned int: 符号扩展

# Integer Promotion

- 对于 char, unsigned char, short 等范围小于 int 的变量, 在做任何运算前都会被隐式扩展为 int (先变大小再变符号)

```
int main() {  
    unsigned char uc = 128;  
    char c = 128;  
    printf ("%d %d\n", uc == c, uc + c);  
}
```



# Integer Promotion

- `int` 及以上，存在一系列按等级排列的类型
- 对表达式中的常数，依次检验各等级类型是否能放下
- 符号看作对常数做的一元运算
- 类型等级为

`int -> unsigned -> long -> unsigned long -> long long (C90)`

`int -> long -> long long (C99+)`

3. 在 32 位平台上，按 C90 标准以下语句中，结果为假的是 **B**

A. `return INT_MIN < INT_MAX;`

B. `return -2147483648 < 2147483647;`

C. `int a = -2147483648; return a < 2147483647;`

D. `return -2147483647-1 < 2147483647;`

参考信息：

C90 的转换顺序：`int -> long -> unsigned -> unsigned long`

$2^{31} = 2147483648$

# 例题 1

(多选) 假设下列 `int` 和 `unsigned` 数均为 32 位

```
int x = 0x80000000;
```

```
unsigned y = 0x00000001;
```

```
int z = 0x80000001;
```

以下表达式正确的是

A.  $(-x) < 0$

B.  $(-1) > y$

C.  $(z \ll 3) == (z * 8)$

D.  $y * 24 == z \ll 5 - y \ll 3$

# 例题 1

答案: ABC

- A. `(int)0x80000000` 的相反数还是自己
- B. `signed` 和 `unsigned` 比较, 都转成 `unsigned`
- C. 恒等
- D. 移位运算优先级低于加减

## 例题 2

下面表达式中为真的是：

- A. `(unsigned) -1 < -2`
- B. `2147483647 > (int) 2147483648U`
- C. `(0x80005942 >> 4) == 0x09005942`
- D. `2147483647 + 1 != 2147483648`

## 例题 2

答案: B(D)

A. `(unsigned) -1 == 2^32 - 1`

B. 两边都是 `int`, 右边  $== T_{min}$

C. 略

D. 见后 integer promotion 部分

32 位 C90 机器会将右侧识别为 `unsigned`, 所以左右相等, 不能选 D

64 位 C99 以上机器将右侧识别为 `long`, 所以左右不相等 (左边会先溢出为 `-0x80000000` 再转成 `long`), 可以选 D

## 例题 3

在 x86-64 机器上有以下程序片段，

```
int x = foo();
```

```
int y = bar();
```

```
unsigned ux = x;
```

```
unsigned uy = y;
```

请分别判断右侧表达式是否恒真。

```
x / 8 == x >> 3
```

```
ux / 8 == ux >> 3
```

```
x * x >= 0
```

```
ux * ux >= 0
```

```
x <= 0 || -x < 0
```

```
x >= 0 || -x > 0
```

```
!x || (x | -x) >> 31 == -1
```

```
ux > -1
```

## 例题 3

$x / 8 == x \gg 3$

F 负数不对

$ux / 8 == ux \gg 3$

T

$x * x \geq 0$

F 可能溢出为负数

$ux * ux \geq 0$

T 左边是 unsigned 值

$x \leq 0 \mid \mid -x < 0$

T

$x \geq 0 \mid \mid -x > 0$

F 反例:  $-0x80000000$

$!x \mid \mid (x \mid -x) \gg 31 == -1$

T  $x \neq 0$  则  $(x \mid -x)$  最高位为 1

$ux > -1$

F 左边是 unsigned 值

## 例题 4

判断下列表达式是否恒成立 (x 为整型)

$$((x \gg 1) \ll 1) \leq x$$

$$((x / 2) * 2) \leq x$$

$$-(x \wedge y \wedge (\sim x)) - y == -(y \wedge x \wedge (\sim y)) - x$$



## 例题 4

判断下列表达式是否恒成立 (x 为整型)

$$((x \gg 1) \ll 1) \leq x \quad \text{T}$$

$$((x / 2) * 2) \leq x \quad \text{F 负奇数不对}$$

$$-(x \wedge y \wedge (\sim x)) - y == -(y \wedge x \wedge (\sim y)) - x \quad \text{T}$$

异或运算具有交换律和结合律

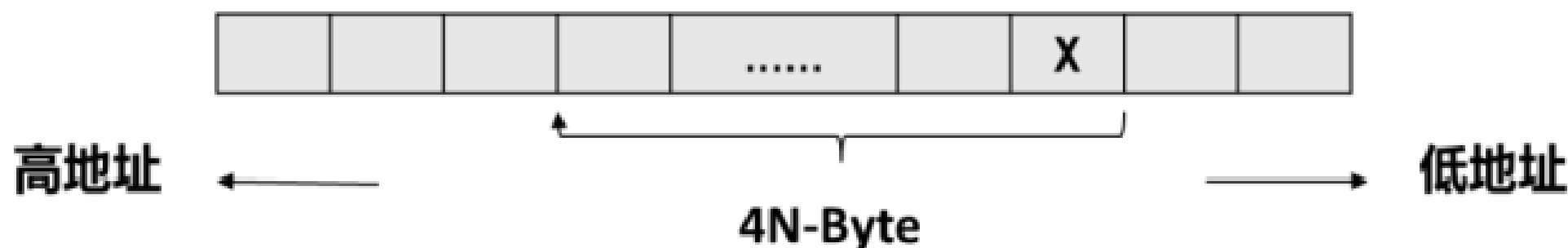
$$\text{左边} = -(x \wedge (\sim x) \wedge y) - y = -(0xffffffff \wedge y) - y$$

$$= -(\sim y) - y = 1$$

右边同理

## 例题 5

1. 假定一个特殊设计的计算机，将 `int` 型数据的长度从 4-Byte 扩展为 4N-Byte，采用大端法 (Big Endian)。现将该 `int` 型所能表示的最小负数写入内存中，如下图所示。其中每个小矩形代表一个 Byte，请问 X 位置这个 Byte 中的值是多少？



- A.  $00000000_2$     B.  $01111111_2$     C.  $10000000_2$     D.  $11111111_2$

## 例题 5

答案：C

4N 个 byte 的 int 型能表达的最小负数为  $0x8000\dots0$

图示要求低地址开始的第一个 byte，由于机器是大端法，所以第一个 byte 就是  $0x80$

## 例题 6

x 是整型, `ux = (unsigned)x;`

判断下列两式是否等值

`(x == 1) && (ux - 2 < 2)`

`(x == 1) && (!!ux - 2 < 2)`

## 例题 6

答案：不等

!!ux 是有符号数，只考虑  $x == 1$  时已经不等

## 例题 7

判断下列程序的输出

```
short sx = -12345;    /* 0xcfc7 */
unsigned short usx = sx;
int x = sx;
unsigned ux = usx;
printf("%d %x\n", sx, sx);
printf("%u %x\n", usx, usx);
printf("%d %x\n", x, x);
printf("%u %x\n", ux, ux);
```

## 例题 7

判断下列程序的输出

```
short sx = -12345;    /* 0xcfc7 */
unsigned short usx = sx;
int x = sx;
unsigned ux = usx;
printf("%d %x\n", sx, sx);      -12345 cfc7
printf("%u %x\n", usx, usx);    53191 cfc7
printf("%d %x\n", x, x);        -12345 fffffcfc7
printf("%u %x\n", ux, ux);      53191 cfc7
```

## 例题 8

判断下列程序的输出

```
short a = -3;  
unsigned b = a;  
printf(“%d, %x”, b, b);
```

- A. -3, 0xfffffffffd
- B. 65533, 0Xfffd
- C. 4294967293, 0xfffffffffd
- D. -3, 0xfffd



## 例题 8

答案：A

short 转 unsigned 先变大小再变符号

注意 %d 是按 signed int 输出

## 例题 9

C 语言中的 `int` 和 `unsigned` 类型的常数进行比较时，下列表达式及描述正确的是：

(位宽为 32 位,  $TMIN = -2147483648$ ,  $TMAX = 2147483647$ )

- A. `0 == 0U`, 按有符号数进行比较
- B. `2147483647U > -2147483647 - 1`, 按无符号数进行比较
- C. `(unsigned)-1 < -2`, 按无符号数进行比较
- D. `2147483647 > (int)2147483648U`, 按有符号数进行比较

## 例题 9

C 语言中的 `int` 和 `unsigned` 类型的常数进行比较时，下列表达式及描述正确的是：D

(位宽为 32 位,  $TMIN = -2147483648$ ,  $TMAX = 2147483647$ )

- A. `0 == 0U`, 按有符号数进行比较
- B. `2147483647U > -2147483647 - 1`, 按无符号数进行比较
- C. `(unsigned)-1 < -2`, 按无符号数进行比较
- D. `2147483647 > (int)2147483648U`, 按有符号数进行比较

## 例题 10

### 代码纠错

```
int main() {  
    char A[12] = "Hello World";  
    char B[12] = "Buggy Codes";  
    int pos;  
    for (pos = 0; pos - sizeof(B) < 0; pos++) {  
        B[pos] = A[pos];  
    }  
    printf("%s\n", B);  
}
```

## 例题 10

### 代码纠错

```
int main() {  
    char A[12] = "Hello World";  
    char B[12] = "Buggy Codes";  
    int pos;  
    for (pos = 0; pos - sizeof(B) < 0; pos++)  
        B[pos] = A[pos];  
}  
printf("%s\n", B);  
}
```

sizeof 返回值为 unsigned, 循环直接退出

## 例题 11

假设有下面  $x$  和  $y$  的程序定义

```
int x = a >> 2;
```

```
int y = (x + a) / 4;
```

那么有多少个位于闭区间  $[-8, 8]$  的整数  $a$  能使得  $x$  和  $y$  相等?

## 例题 11

假设有下面  $x$  和  $y$  的程序定义

```
int x = a >> 2;
```

```
int y = (x + a) / 4;
```

那么有多少个位于闭区间  $[-8, 8]$  的整数  $a$  能使得  $x$  和  $y$  相等  
枚举即可

## 例题 12

a, b 均为 int 型, 下列判断溢出的方式是否有效?

(1)

```
int x = a + b;
```

```
x - a == b;
```

(2)

```
int x = a * b;
```

```
x / b == a;
```



## 例题 12

$a, b$  均为 `int` 型, 下列判断溢出的方式是否有效?

(1)

```
int x = a + b;
```

```
x - a == b;
```

无效。由于两边模  $2^w$  相等, 所以无论  $x, a, b$  有无符号, 只要位宽相等都成立  $x - a == b$

(2)

```
int x = a * b;
```

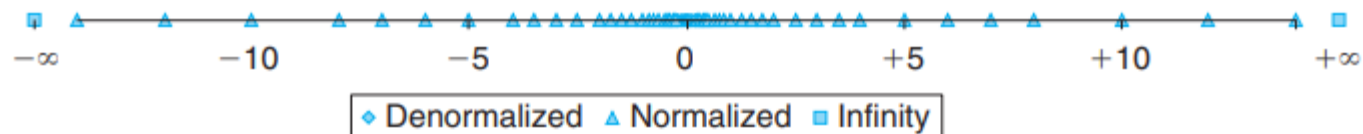
```
x / b == a;
```

有效。若乘法发生溢出则  $x / b$  的绝对值一定  $< a$

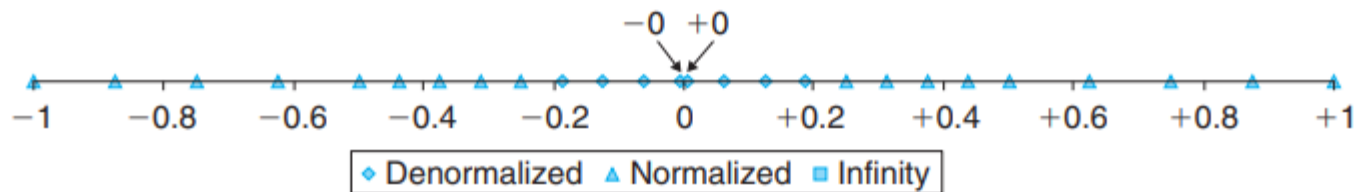
浮点型

# 浮点型-编码

- 信息 = 位 + 上下文
- 上下文相关
  - $S + \text{Exp} + \text{Frac}$
  - $(-1)^s * (M * 2^E)$



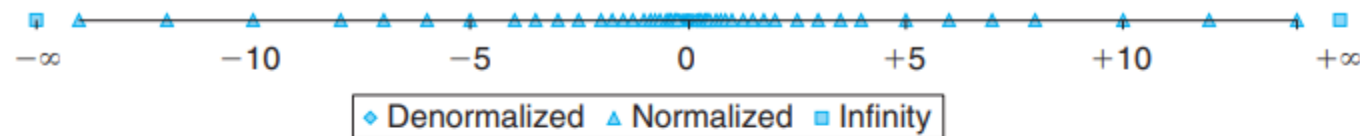
(a) Complete range



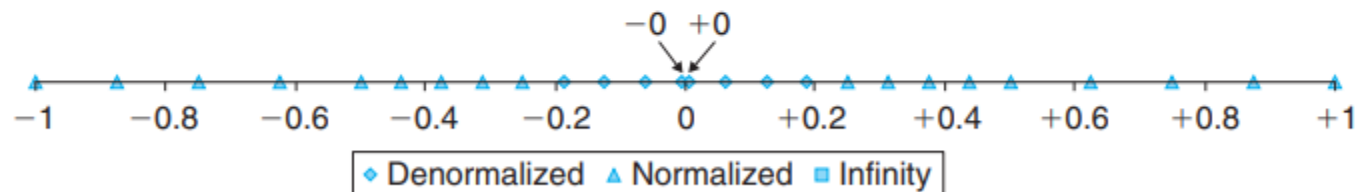
(b) Values between  $-1.0$  and  $+1.0$

# 浮点型-我的思考

- 截断小数 v.s. 浮点数
  - 舍入/密度 (精确度)
  - 范围
- 浮点型为什么要这么设计
  - 浮点型是整型的扩展 (包含性)
  - 正态分布/稀疏性
  - e-f trade-off



(a) Complete range



(b) Values between -1.0 and +1.0

# 浮点型-运算律

- 舍入
- 交换律
  - $a, b$  均非特殊值的情况下成立
- 结合律  $\times$

# 浮点型-INF & NaN

- $INF > 1.0 \Rightarrow 1$
- $-INF < -1.0 \Rightarrow 1$
- $-INF < INF \Rightarrow 1$
- $INF == INF \Rightarrow 1$
- $-INF == -INF \Rightarrow 1$
- $INF > INF \Rightarrow 0$
- $-INF > -INF \Rightarrow 0$
- $(-INF) + (INF) == NaN \Rightarrow 1$
- $NaN != NaN \Rightarrow 1$
- $NaN || NaN \Rightarrow 1$
- $NaN \&\& NaN \Rightarrow 1$
- 其他 NaN 参与判断式均输出 0, 算术表达式均输出 NaN

## 例题 13

4. 关于浮点数，以下说法正确的是
- A. 给定任意浮点数  $a$ ,  $b$  和  $x$ , 如果  $a > b$  成立 (求值为 1), 则一定  $a + x > b + x$  成立
  - B. 不考虑结果为 NaN、Inf 或运算过程发生溢出的情况, 高精度浮点数一定得到比低精度浮点数更精确或相同的结果
  - C. 不考虑输入为 NaN、Inf 的情况, 高精度浮点数一定得到比低精度浮点数更精确或相同的结果
  - D. 给定任意浮点数  $a$ ,  $b$  和  $x$ , 如果  $a > b$  不成立 (求值为 0), 则一定  $a + x > b + x$  不成立。

## 例题 13

- 答案: D
- A. 反例:  $a = \text{INF}$ ,  $b$  未溢出,  $b + x$  溢出
- B. 反例:  $0.1f + 0.2f == 0.3f$   
 $(\text{double})0.1 + (\text{double})0.2 != (\text{double})0.3$
- C. 同 B
- D. 若  $a$ ,  $b$  中有 NAN 显然正确, 否则讨论是否有 INF 即可



## 例题 14

- 判断
- 对于任意的单精度浮点数  $a$  和  $b$ , 如果  $a > b$ , 那么  $a + 1 > b$
- 对于任意的双精度浮点数  $d$ , 如果  $d < 0$ , 那么  $d * d > 0$
- 单精度浮点数  $a$  和  $b$ , 如果  $a > b$ , 那么  $a + b > b + b$
- 对于任意的双精度浮点数  $d$ , 如果  $d < 0$ , 那么  $d * 2 < 0$
- 对于任意的双精度浮点数  $d$ ,  $d == d$

## 例题 14

- 判断
- 对于任意的单精度浮点数  $a$  和  $b$ , 如果  $a > b$ , 那么  $a + 1 > b$  T
- 对于任意的双精度浮点数  $d$ , 如果  $d < 0$ , 那么  $d * d > 0$  N  
反例:  $d$  是最大的负双精度浮点数,  $d * d == 0$
- 单精度浮点数  $a$  和  $b$ , 如果  $a > b$ , 那么  $a + b > b + b$  N
- 对于任意的双精度浮点数  $d$ , 如果  $d < 0$ , 那么  $d * 2 < 0$  T
- 对于任意的双精度浮点数  $d$ ,  $d == d$  N

## 例题 15

- 在遵守 IEEE-754 标准的一台机器上声明如下三个变量 `double f, g, h`; `r()` 返回一个随机的 `int`; 判断以下断言是否恒真
- $f > g$  则  $f + 1 > g + 1$
- $f > g \ \&\& \ g > 1$  则  $f - 1 > g - 1$
- $f = r(), g = r(), h = r()$  则  
 $(f + g) + h == f + (g + h)$
- $f \neq 0.0$  则  $f / f * f == f$
- $f \neq 0.0$  则  $f * f / f == f$

# 例题 15

- 在遵守 IEEE-754 标准的一台机器上声明如下三个变量 `double f, g, h`; `r()` 返回一个随机的 `int`; 判断以下断言是否恒真

- `f > g` 则 `f + 1 > g + 1`                      N    反例: `f` 是最小的 `double`, `g == 0`
- `f > g && g > 1` 则 `f - 1 > g - 1`                      N    反例: `f == 253 + 6`, `g == 253 + 4`

- `f = r()`, `g = r()`, `h = r()` 则 `(f + g) + h == f + (g + h)`

T    `int` 转 `double` 做浮点运算满足交换律和结合律

- `f != 0.0` 则 `f / f * f == f`                      N    反例: `NAN`
- `f != 0.0` 则 `f * f / f == f`                      N    同上

- 单调性: `a >= b`  $\Rightarrow$  `a+1 >= b+1`
- 考虑 `INF` / `NaN`

## 例题 16

- 以下程序的输出结果最接近哪个常数?
- 已知该机器上 `int` 为 4 字节, 且  $1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots = \frac{\pi}{4}$

```
int main() {  
    double pi = 0.0; int k;  
    for (k = 0; k >= 0; ++k)  
        pi += (k & 1 ? -1 : 1) / (double)(2 * k + 1);  
        pi *= 4;  
        printf("%f\n", pi);  
}
```

## 例题 16

- 答案:  $2\pi$
- 两处 int 溢出: for 循环内  $k$  和  $2 * k + 1$
- 分  $0 \leq k \leq 2^{30} - 1$  和  $2^{30} \leq k \leq 2^{31} - 1$  两部分
- 第一步运算结果为

$$1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots - \frac{1}{2^{31}-1} - \frac{1}{2^{31}-1} + \dots + 1 \approx \frac{\pi}{2}$$

## 例题 17

```
void func(int *a, int *b) {  
    if (*a && ++*b) printf("%d\n", *b);  
    else printf("%d\n", *b);  
}  
int main() {  
    int a = 0, b = 0;  
    func(&a, &b); func(&b, &a);  
}
```

- 运行结果?

## 例题 17

- 运行结果: 0 0
- `a && b` 逻辑与
  - 当 `a == 0` 时不再计算 `b`