**实现说明**

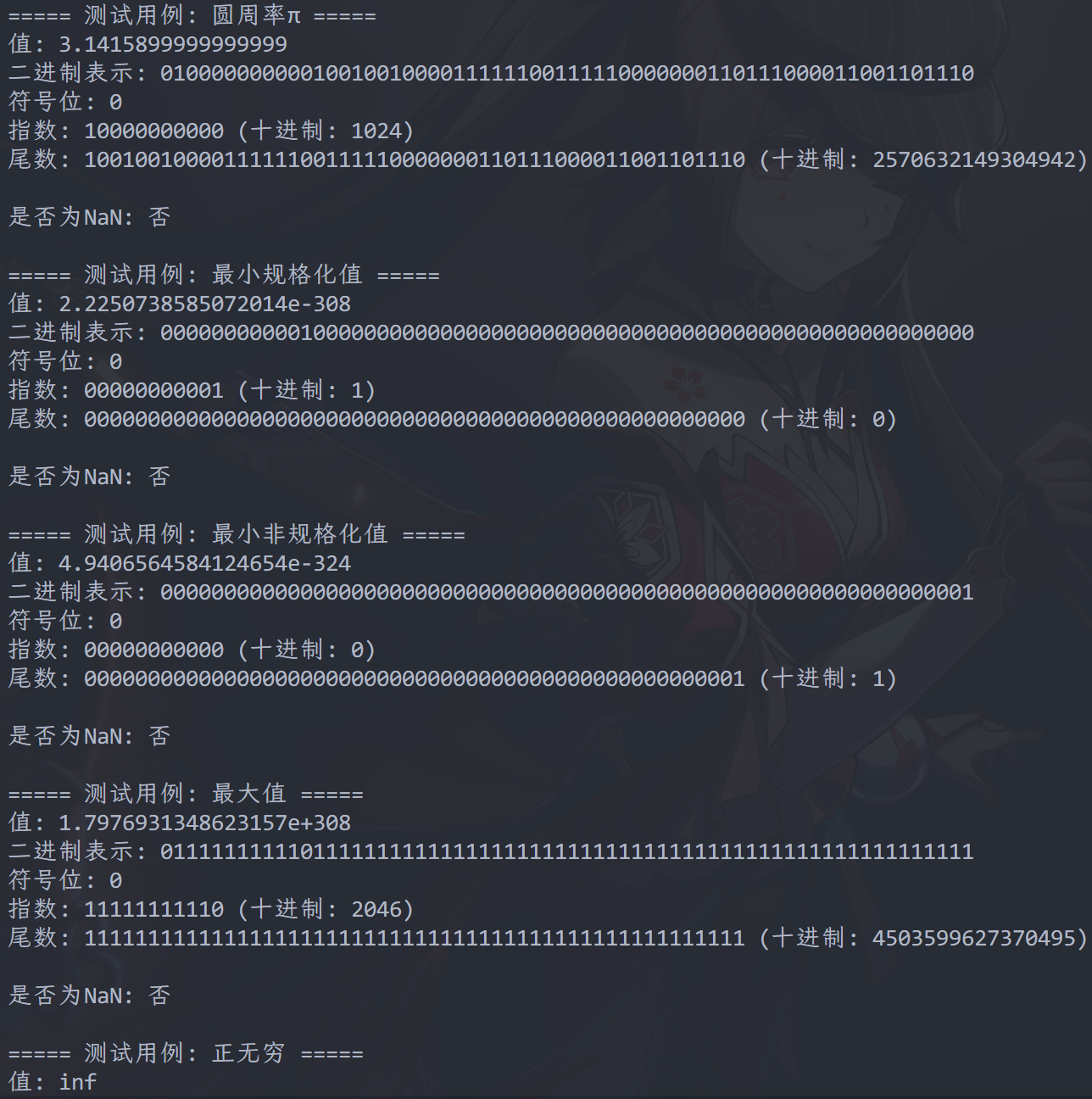
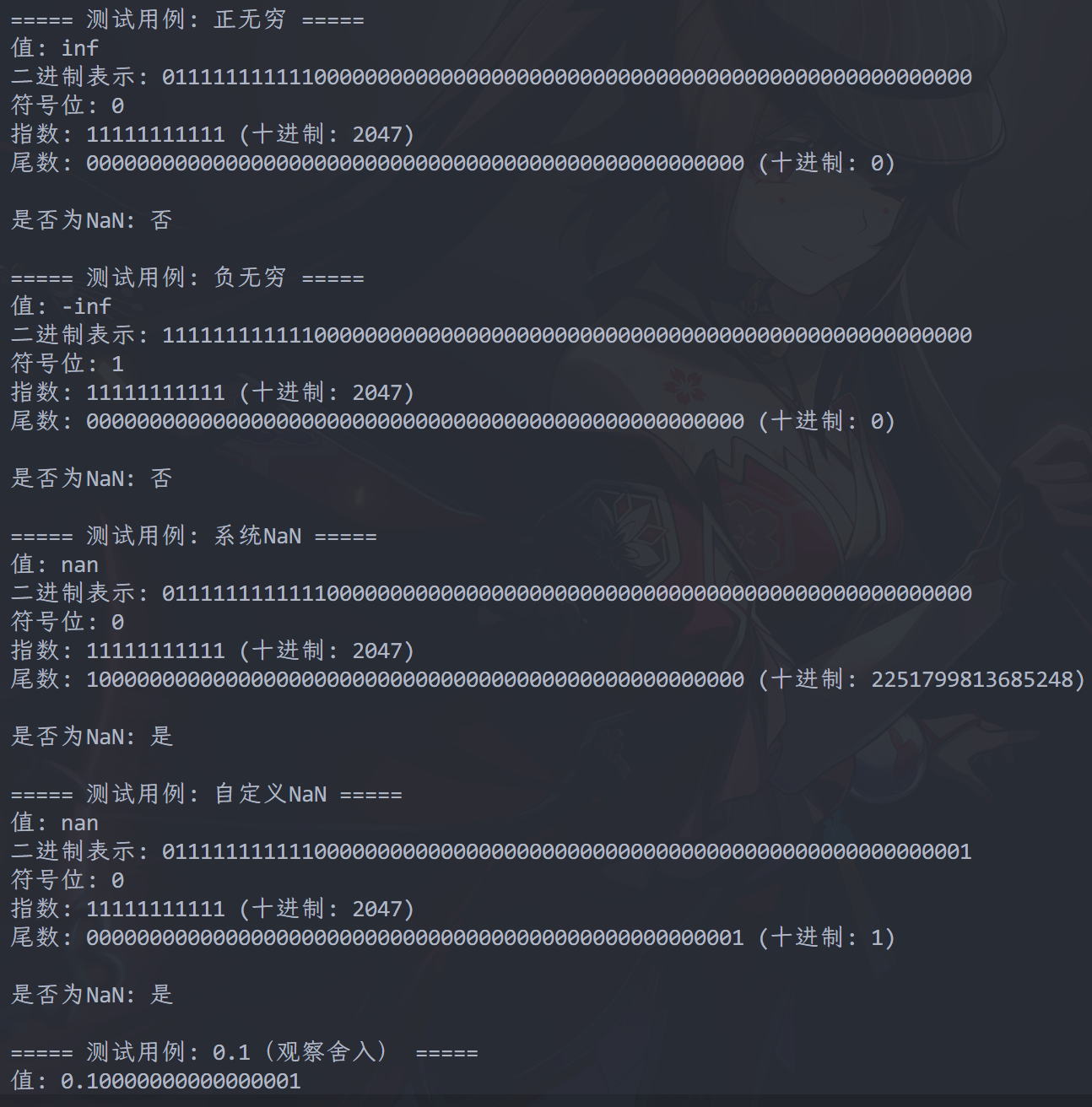
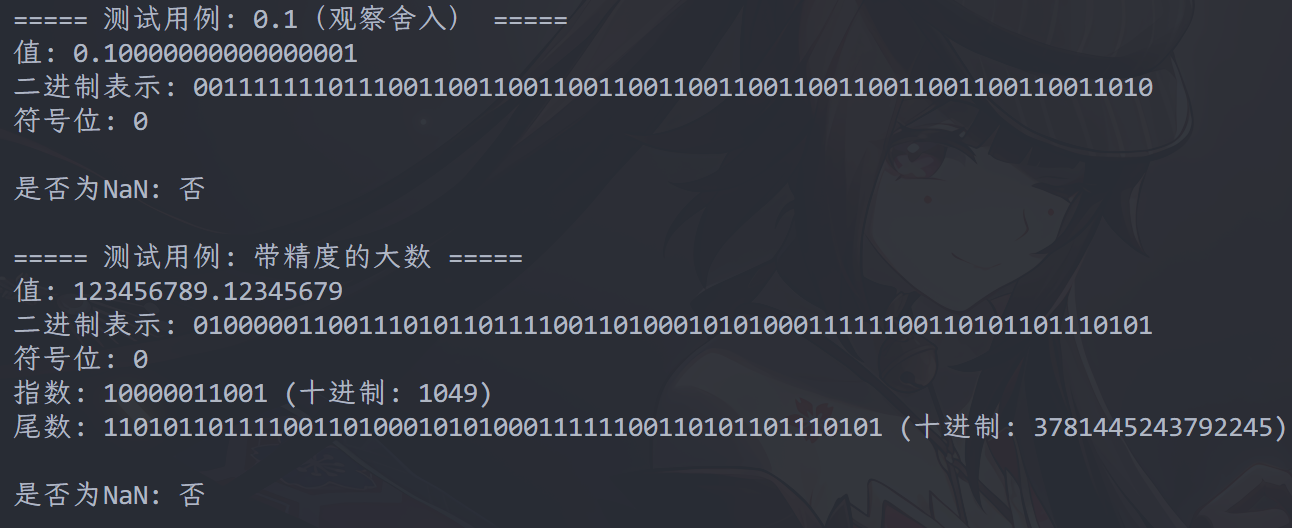
1. **面向对象设计**：
   * 创建了DoubleAnalyzer类来封装浮点数分析功能
   * 提供了完整的接口来查询浮点数的各个组成部分
2. **共用体技术**：
   * 使用DoubleUnion共用体实现对同一内存区域的不同解释
   * 允许以不同方式访问浮点数：整体值、整数位模式和IEEE 754的组成部分
3. **NaN处理**：
   * 提供了专门的方法检测和创建NaN
   * 展示了NaN在内存中的二进制表示

**测试案例设计思路**

测试案例设计涵盖了浮点数的各种可能表示：

1. **基本值测试**：
   * 正零和负零（验证符号位表示）
   * 正常数值（1.0, -1.0, π）
2. **边界值测试**：
   * 最小规格化值和最小非规格化值（验证极小数处理）
   * 最大值（验证极大数处理）
   * 正无穷和负无穷（验证特殊值处理）
3. **特殊值测试**：
   * 系统生成的NaN
   * 自定义生成的NaN（验证NaN创建和识别）
4. **精度问题测试**：
   * 0.1（展示二进制无法精确表示某些十进制数）
   * 带精度的大数（展示精度损失）

**程序实际运行结果**

通过以上测试样例可见：

1. 正零和负零：它们仅在符号位上有区别，尾数和指数都为0。
2. 常规数值：可以观察到IEEE 754的三部分组成（符号位、指数、尾数）。
3. 边界值：
   * 最小规格化值有特殊的指数值和全0尾数
   * 最大值有接近全1的指数和尾数
4. 特殊值：
   * 无穷大有全1指数和全0尾数
   * NaN有全1指数和非零尾数
5. 精度问题：
   * 0.1在二进制中是无限循环小数，展示了舍入效果
   * 大数的小数部分展示了精度限制