目标：

编写C++程序来达到：输入正整数x和y，计算其最大公因数和最小公倍数并且计算时间消耗。

原理：

通过循环输入直到输入满足条件的数为止。

通过辗转相除法计算给定的两个数的最大公因数，同时用（第一个数 / 最大公因数 \* 第二个数）来计算给定两个数的最小公倍数。（在这里计算最小公倍数的公式是为了减少因为数目过大而越界的情况。如果先把两个数相乘，溢出的可能比上述算法的溢出可能更大）。

同时用时间类来进行对时间的计算操作。在这里用到了循环N次计算总时间再除以N计算单次时间的方法。

测试过程中可以检测越界数据（用逆推法，最小公倍数 / 第一个数 \* 最大公因数 = 第二个数，如果越界则不会相等）。

在自动测试中，用到了：

1. 已知数据测试法
2. 质数倍数测试法（两个不同质数a和b分别和c相乘，结果ac和bc的最大公因数是c，最小公倍数是ab/c）

**如何编程和运行：**

* 软件构件库部分：

构建所需的类和最基本的计算函数。包括数对类和时间类。数对类中包括计算最大公因数和最小公倍数的函数，时间类中包括开始、停止计时和计算单次运行真实时间的函数。

* 构件库测试部分：

编写所需要的测试函数，自动测试部分还包括测试所需要的一些数据。

* 构件库应用部分：

编写main函数，在函数中提供选择自动和手动测试的功能。

结果：

以下测试数据均在64位操作系统下用VS 2017 测试得到。

自动测试程序运行结果：

------------------------------------------------------------

第1类测试开始！此测试使用已知数据测试！

计算得到2与96的最大公约数为2，用时0.11微秒；计算得到2与96的最小公倍数为96，用时0.15微秒！

计算得到3与9的最大公约数为3，用时0.12微秒；计算得到3与9的最小公倍数为9，用时0.15微秒！

计算得到89与5的最大公约数为1，用时0.13微秒；计算得到89与5的最小公倍数为445，用时0.17微秒！

计算得到9560与5的最大公约数为5，用时0.09微秒；计算得到9560与5的最小公倍数为9560，用时0.12微秒！

计算得到7与84的最大公约数为7，用时0.11微秒；计算得到7与84的最小公倍数为84，用时0.13微秒！

计算得到38与19的最大公约数为19，用时0.1微秒；计算得到38与19的最小公倍数为38，用时0.1微秒！

第1类测试结束，共测试6组数据。每组数据计算两个数（最大公约数和最小公倍数），其中0个测试结果错误！

------------------------------------------------------------

第2类测试开始！此测试使用两质数293和97的倍数测试！

第2类测试结束，共测试20000组数据。每组数据计算两个数（最大公约数和最小公倍数），其中0个测试结果错误！

------------------------------------------------------------

第3类测试开始！此测试使用两质数29333和29947的倍数测试！

第3类测试结束，共测试20000组数据。每组数据计算两个数（最大公约数和最小公倍数），其中0个测试结果错误！

------------------------------------------------------------

第4类测试开始！此测试使用两质数49993和49999的倍数测试！

第4类测试结束，共测试20000组数据。每组数据计算两个数（最大公约数和最小公倍数），其中0个测试结果错误！

------------------------------------------------------------

第5类测试开始！此测试使用两质数9982499和9522827的倍数测试！

第5类测试结束，共测试20000组数据。每组数据计算两个数（最大公约数和最小公倍数），其中0个测试结果错误！

------------------------------------------------------------

请按任意键继续. . .

自动测试程序验证结果报告：

* 对于64位的计算机

在测试的过程中发现对于测试的数据没有发现错误的计算结果。

手动测试部分某次测试结果：

1 1

1 0.08

1 0.1

9223372036854775807 9223372036854775807

9223372036854775807 0.07

9223372036854775807 0.1

9223372036854775807 9223372036854775806

1 0.11

-9223372036854775806 0.12

已越界！最小公倍数计算错误！

156468415 1564165163514564685

5 0.21

-1953152563206325993 0.22

已越界！最小公倍数计算错误！

1532 1532

1532 0.09

1532 0.12

15348 1564523415

3 0.23

8004101791140 0.24

1541865 15486352

1 0.23

23877864126480 0.27

5468512 16485132

4 0.24

22537285540896 0.28

1 31548

1 0.07

31548 0.11

684153 16854185

1 0.31

11530841230305 0.33

4564186 51564168541

1 0.3

235348456156472626 0.33

15648674 86515616531

1 0.41

1353854679002629894 0.45

156854 165416574863415

1 0.29

7499507359916544794 0.31

已越界！最小公倍数计算错误！

35 95

5 0.13

665 0.15

315 1548

9 0.14

54180 0.17

9223372036854775808 9223372036854775808

输入错误，请重新输入！

a 153

输入错误，请重新输入！

0 315

输入错误，请重新输入！

-1 -25

输入错误，请重新输入！

35 -3

输入错误，请重新输入！

程序验证结果报告：

* 对于64位的计算机（使用的是long long）

在测试的过程中发现读取数据部分合法输入为1 ~ 9223372036854775807。

不正确的输入并不会导致程序出错，而是会提示输入错误，重新输入。

只要输入数据在上述合法输入范围，可以正确计算最大公因数，但最小公倍数有可能错误（溢出的原因）。此时会报错（不只限于数据得出负数的情形）。

结论：

* 时间代价与数的大小的关系

在测试的过程中发现时间代价与数的大小并无直接关系。如计算1与1和计算9223372036854775807与9223372036854775807用时差不多（这是由算法决定的），而决定时间代价的是两个数之间的关系（它们是否互质，大小差别）。