

## 《信息安全基础综合实验》课程实验报告

**实验题目：**Fermat 素性检验算法

**班级：**1718039      **学号 1：**17180210027      **姓名 1：**李欣

**班级：**1718039      **学号 2：**17189110002      **姓名 2：**祝佳磊

### 一、实验目的

#### 实验目的

1. 使用 C 语言的 miracl 库进行 Fermat 大数素性验证
2. 熟悉 miracl 库的安装、配置及使用。
3. 熟悉 miracl 库中一些基本函数的使用

#### 实验环境

Windows10

Visual Studio 2017

Miracl.lib

### 二、方案设计

在密码学的实验中，对于大数是否为素数的判断十分重要。检验一个数是否为素数的方法有很多，其中包括 Fermat 素性检验算法

#### Fermat 素性检验算法：

Fermat 素性检验算法是一个概率性算法。

根据费马小定理，给定素数  $p, a \in Z$ ，则有  $a^{p-1} \equiv 1(\text{mod } p)$ 。

那么如果有一个整数， $(a, m) = 1$ ，使得  $a^{m-1} \equiv 1(\text{mod } m)$ ，

那么这个整数 m 是不是一个素数。答案是他不一定是一个素数，还有可能是一个合数。因此，费马小定理反推不一定成立。但是却可以得出以下推断：

任取一个奇整数 m，若取一证书  $2 \leq a \leq m-2$ ，使得  $a^{m-1} \equiv 1(\text{mod } m)$

那么 m 至少有二分之一的概率为素数。也就是说，要判断一个奇整数是不是一个素数，我们可以随机选取同 m 互质的一个整数 a，判断 m 和 a 是否满足费马小定理。如果满足，m 就有不低于二分之一的概率为一个素数。不断的去选取这个 a，重复上述过程 n 次以后，这个 m 则有一减二的 N 次分之一的概率为一个素数

### 三、方案实现

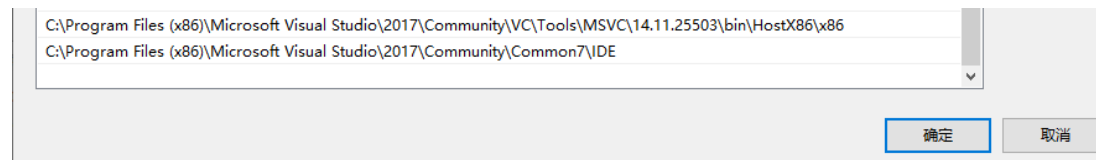
#### 环境配置

##### 配置环境变量：

因为编译 miracl 包需要调用 cl 命令编译文件，为了能够正常编译 miracl 库，我们需要先配置一下计算机的环境变量，使得可以在 cmd 中使用 cl 命令

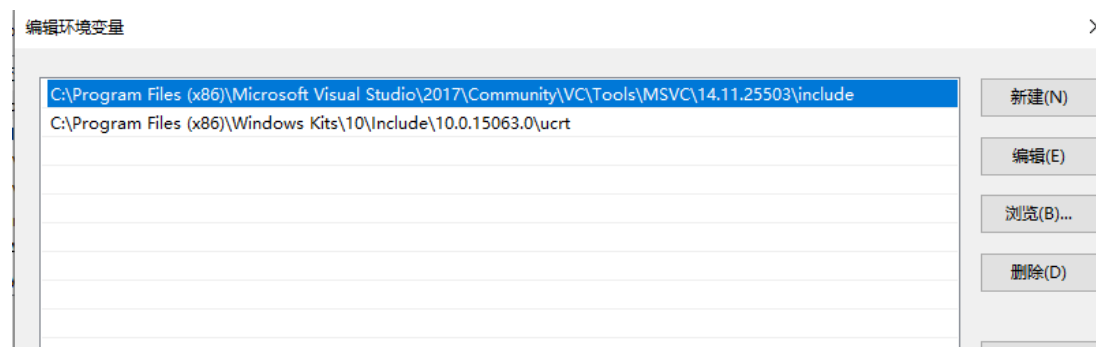
##### 1. 修改系统变量中的 PATH

将 Visual Studio 中 VC 的 bin 路径中 x86 和 Comon7 中的 IDE 添加到 PATH 变量中



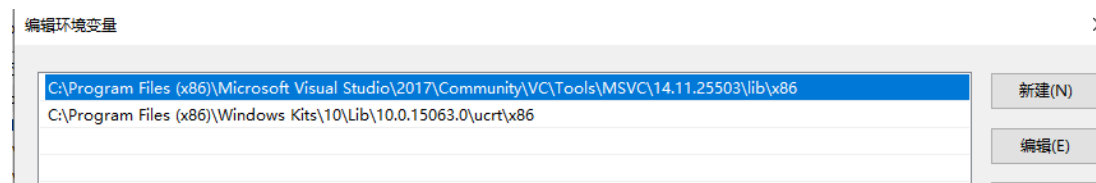
##### 2. 创建 INCLUDE 变量

创建 INCLUDE 变量，并将 Visual Studio 中 VC 的 include 路径和 Windows Kits 中的 ucrt 路径添加到新创建的变量



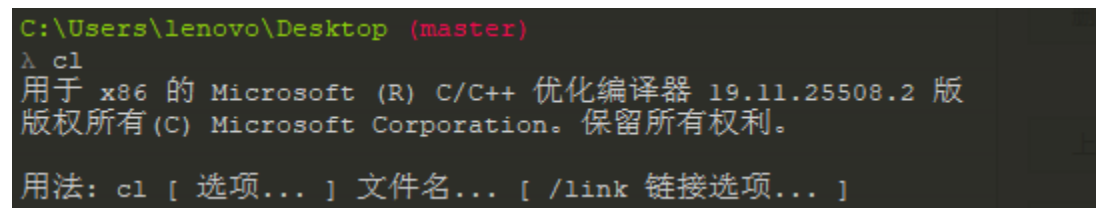
##### 3. 创建 LIB 变量

创建 LIB 变量，并将 Visual Studio 中 VC 的 lib 路径、Windows Kits 中的 ucrt 路径等路径添加到新创建的变量



##### 4. 测试 CL 命令是否配置成功

打开 cmd，并在其中输入 cl



## 编译 miracl 包:

### 1. 下载 miracl 包:

在 GitHub 上下载 miracl 包, 地址:

<https://github.com/miracl/MIRACL/archive/master.zip>

### 2. 解压并调整文件位置

根据 miracl 包的官方说明, 为了编译成功, 需要将 include 路径、lib 路径和 source 路径下的所有文件放在同一文件夹中

» 储存盘 (D:) » Professional software » miracl »

名称	修改日期	类型	大小
avr4.mcs	2019/9/22 16:12	MCS 文件	15 KB
bc32doit	2019/9/22 16:12	Windows 批处理...	4 KB
bcldoit	2019/9/22 16:12	Windows 批处理...	3 KB
bcxdoit	2019/9/22 16:12	Windows 批处理...	3 KB
big.cpp	2019/9/22 16:12	C++ 源	14 KB
big	2019/9/22 16:12	C Header File	16 KB
big	2019/9/22 16:53	3D Object	188 KB
blackfin.mcs	2019/9/22 16:12	MCS 文件	6 KB
bmark	2019/9/22 16:12	C Source File	36 KB
bmark	2019/9/22 16:53	3D Object	43 KB
bp160.ecs	2019/9/22 16:12	ECS 文件	1 KB
bpt160.ecs	2019/9/22 16:12	ECS 文件	1 KB
brent	2019/9/22 16:12	C Source File	3 KB
brent.cpp	2019/9/22 16:12	C++ 源	3 KB
brent	2019/9/22 16:53	3D Object	167 KB
brent_mt	2019/9/22 16:12	C Source File	3 KB
brick	2019/9/22 16:12	C Source File	2 KB
brick.cpp	2019/9/22 16:12	C++ 源	1 KB
brick	2019/9/22 16:12	C Header File	1 KB
brute	2019/9/22 16:12	C Source File	3 KB
brute.cpp	2019/9/22 16:12	C++ 源	2 KB
brute	2019/9/22 16:53	3D Object	156 KB
c.mcs	2019/9/22 16:12	MCS 文件	5 KB

### 3. 创建系统变量 vcvarall

找到 vcvarsall 的路径(可以使用 32 或 64 版本), 创建系统变量

系统 (C:) » Program Files (x86) » Microsoft Visual Studio » 2017 » Community » VC » Auxiliary » Build

名称	修改日期	类型	大小
Microsoft.VCToolsVersion.default.pr...	2017/8/17 20:44	Project Property...	1 KB
Microsoft.VCToolsVersion.default	2017/8/17 20:44	文本文档	1 KB
vcvars32	2017/9/18 16:42	Windows 批处理...	1 KB
vcvars64	2017/9/18 16:42	Windows 批处理...	1 KB
vcvarsall	2017/8/17 20:44	Windows 批处理...	9 KB

编辑系统变量

变量名(N): VCVarsAll






变量值(V): m Files (x86)\Microsoft Visual Studio\2017\Community\VC\Auxiliary\Build\vcvarsall.bat

浏览目录(D)... 浏览文件(F)... 确定 取消

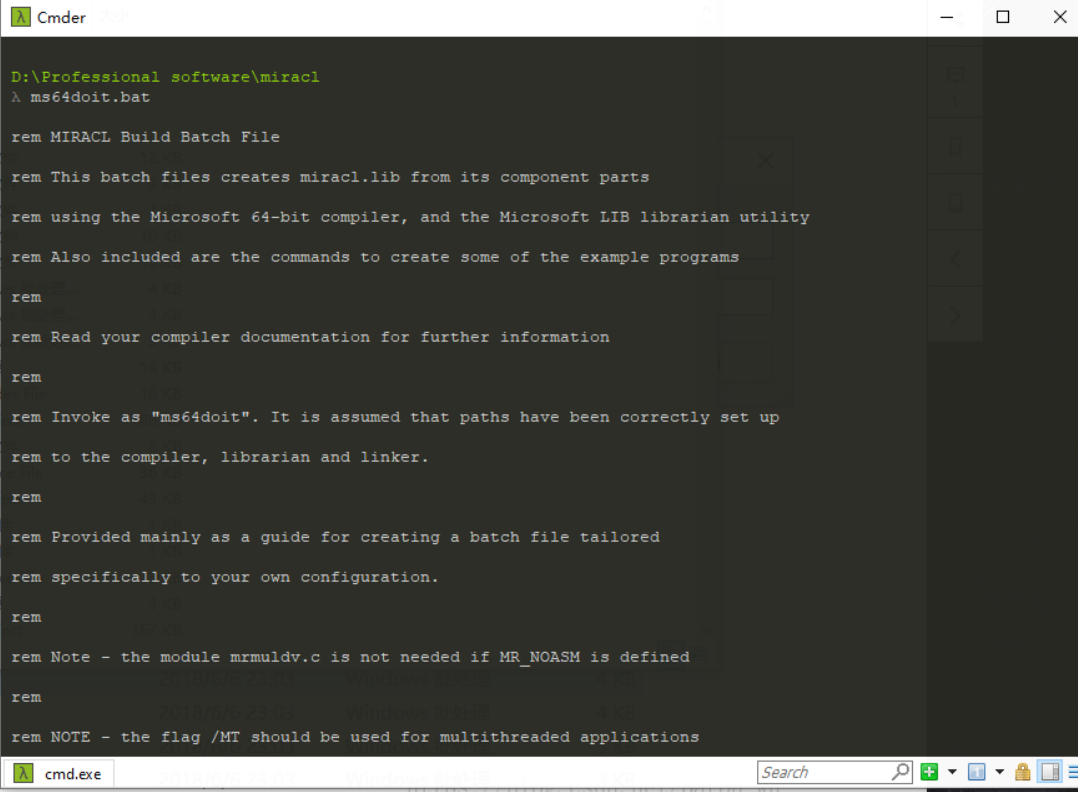
### 4. 环境搭建完毕, 编译生成 miracl 库

环境配置完成后, 找到 ms64doit.bat 或 ms32doit.bat 文件, 生成对应版本,

具体生成 64 位版本还是 32 位版本看具体需要







	ms32doit	2019/9/22 16:12	Windows 批处理...	4 KB
	ms64doit	2019/9/22 16:12	Windows 批处理...	4 KB
	ms64doit_cpp	2019/9/22 16:12	Windows 批处理...	4 KB
	ms86.mcs	2019/9/22 16:12	MCS 文件	7 KB
	msiodoit	2019/9/22 16:12	Windows 批处理...	3 KB

在 cmd 中执行对应文件



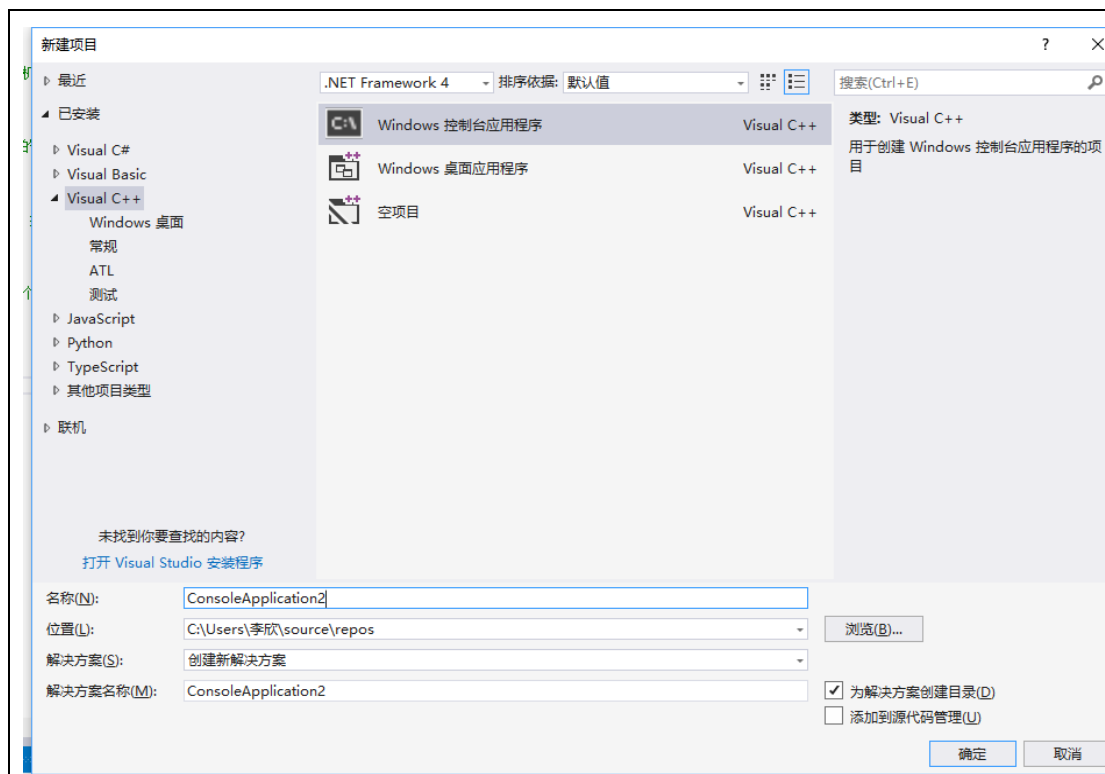
## 5. 检验编译结果

如果编译成功，则会在当前文件夹中生成 miracl.lib 文件

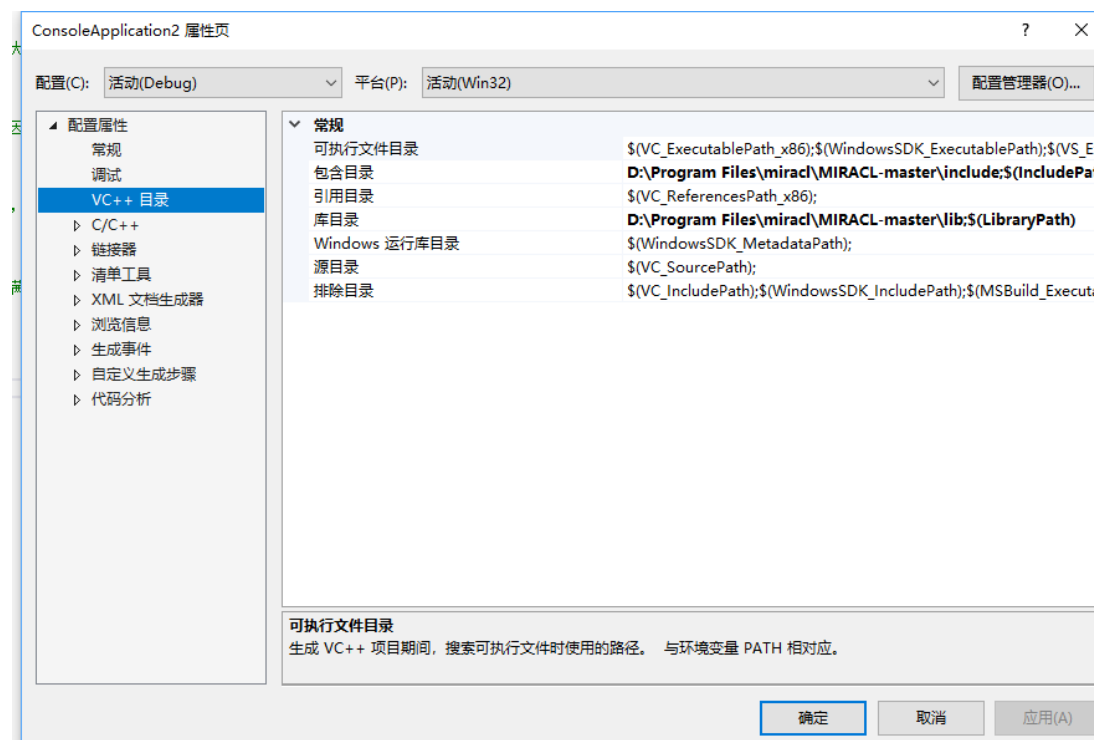
	miracl.lib	2019/9/22 19:15	对象文件库	506 KB
	miracl.mak	2019/9/22 16:12	Makefile	8 KB
	MIRACL-master	2019/9/22 15:54	压缩(zipped)文件...	1,926 KB
	mirdef	2019/9/22 16:12	文件	2 KB
	MIRDEF.AMD	2019/9/22 16:12	AMD 文件	1 KB
	mirdef.arm	2019/9/22 16:12	ARM 文件	1 KB

配置 VS2017:

### 1. 新建一个项目



## 2. 设置项目属性:



修改包含目录和库目录为相应的 miracl 解压位置。



### 代码清单：

```
#define _CRT_SECURE_NO_WARNINGS
#include "miracl.h"
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#define round 6

int Fermatjdu_prime(big obj);

int main()
{
    FILE *fp;
    big obj;
    miracl *mip = mirsys(1500, 16); // 定义的这些变量最大长度都是5000位（这个位是后面进制的位数），输入、输出、运算用的进制都是16进制。
    mip->IOBASE = 16;
    obj = mirvar(0); // 初始化变量obj，obj是输入的需要判断是否为素数的大数
    if ((fp = fopen("data.txt", "r+")) == NULL) {
        printf("Open the file failure...\n");
        exit(0);
    } // 判断文件是否能够正确打开
    while (!feof(fp)) { // 检测文件结束符
        cinnum(obj, fp); // 从文件中读取一个数字进入，并将其强制转化为十六进制表的大数obj
        cotnum(obj, stdout); // 向屏幕上输出一个大数obj
        if (Fermatjdu_prime(obj))
            printf("This number has a %.4f%% probability of being a prime number.\n", 100 * (1 - pow(0.5, round)));
        else
            printf("This number is 100%% definitely a Composite number! \n");
    }

    fclose(fp);
    mirkill(obj); // 释放大数obj所占用的空间
    mirexit(); // 清楚miracl系统
    getchar();
    return 0;
}
```

```

int Fermatjdu_prime(big obj)

{
    big rando, tran, mgcd, tran1, r, num1, num2, cons;
    int i, j;
    int test, test1;
    miracl *mip = mirsys(1500, 16);
    mip->IOBASE = 16;
    rando = mirvar(0); //对函数中使用到的big型变量进行初始化
    tran = mirvar(0);
    mgcd = mirvar(0);
    tran1 = mirvar(0);
    r = mirvar(0);
    num1 = mirvar(1);
    num2 = mirvar(2);
    cons = mirvar(0);
    i = 0;
    j = 0;
    decr(obj, 2, tran); //trans=obj-2
    decr(obj, 1, tran1); //trans=obj-1
    srand((unsigned int)time(NULL));
    for (i = 0; i < round; i++)
    {
        bigrand(obj, rando); //生成所需要的随机数
        egcd(rando, obj, mgcd); //计算obj和生成的随机数的最大公因数
        test = mr_compare(mgcd, num1);
        if (!test) //判断obj和随机数是否互素，它们的最大公因数如果不是1的话，
compare函数将会返回1，不满足条件
        {
            powmod(rando, tran1, obj, r); //计算，如果r=1，则obj可能是素数，进入
下一个if语句
            test1 = mr_compare(r, num1);
            if (test1) j++; //j是判断因子，如果一个数能够满足在当前的轮数下，满足
上述的算法，则j能够计数；如果j不等于轮数，那么这个数就不是素数；

        }

    }
    j++;
    if (j == round)
        return 1;
    else
        return 0;
}

```



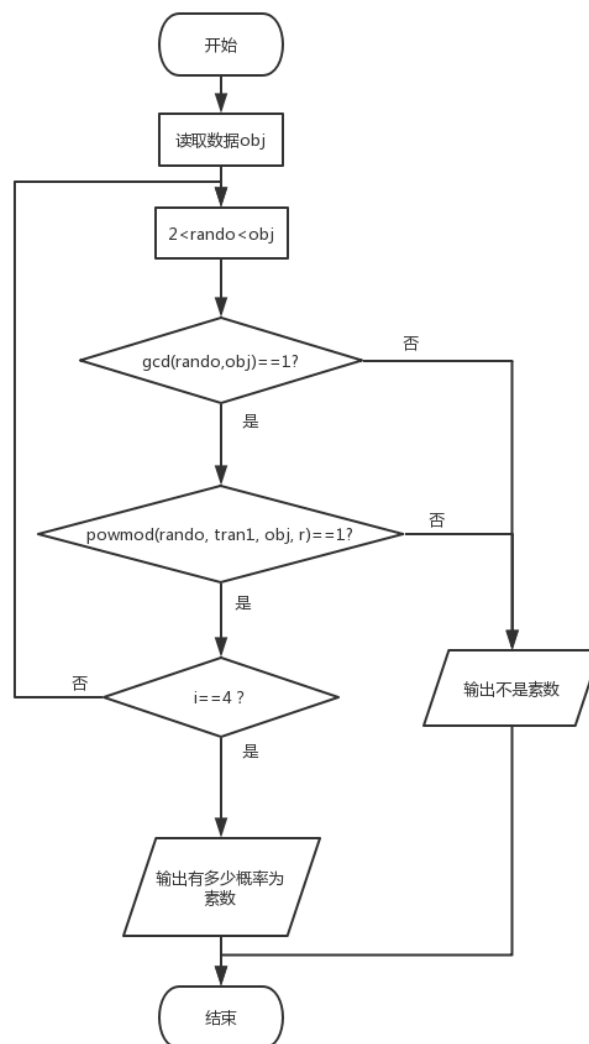
```

    mirkill(obj);
    mirkill(rando);
    mirkill(tran);
    mirkill(tran1);
    mirkill(r);
    mirkill(mgcd);
}

```

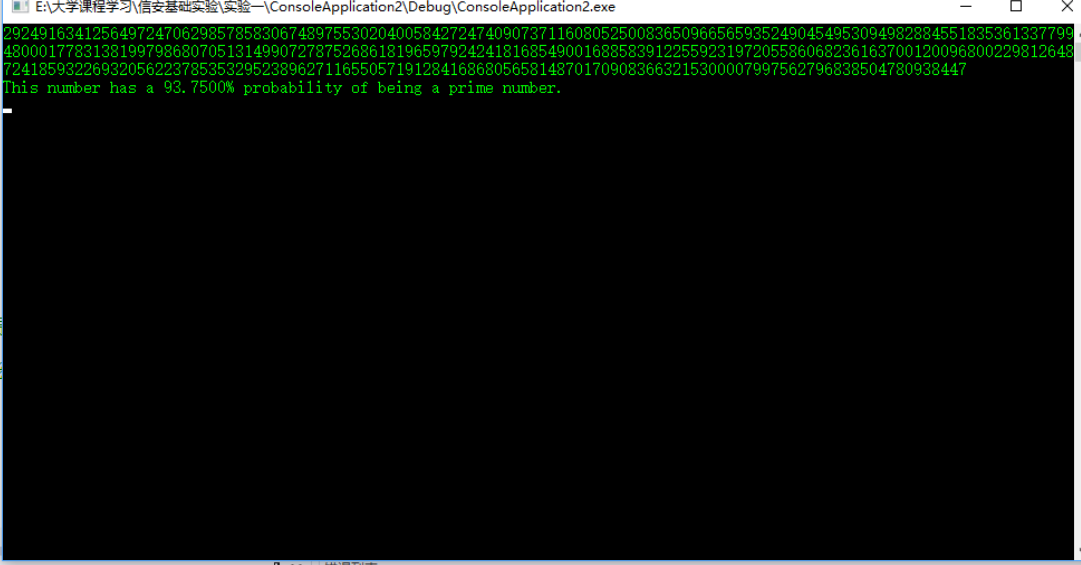
### 主函数分析：

主函数为 `Fermatjdu_prime` 函数，它接收一个大数对象作为参数，设置判断轮次为 4 轮，通过 `for` 循环进行四轮判断。每轮生成一个大于 3 小于要检验的大数的随机输。使用 `miracl` 库自带的 `egcd` 函数计算随机数和需要判断的数的最大公因数。判断最大公因数是否为一，弱不为一，说明需要判断的数并不是一个素数。如果为 1，则进行下一轮判断，知道轮数到达 4。再通过概率计算公式计算出需要判断的数为素数的概率



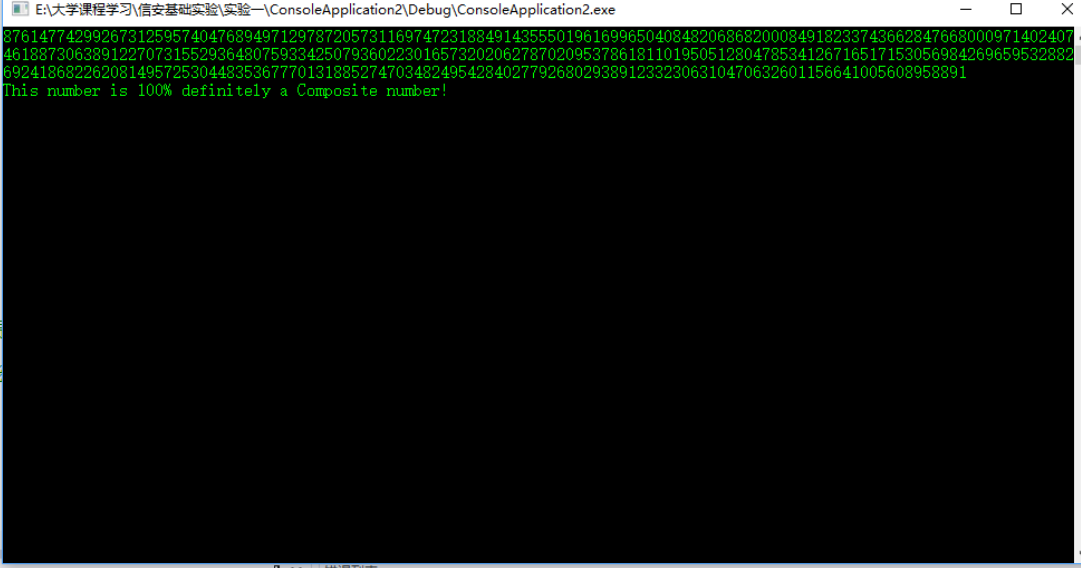
## 四、数据分析

测试数据 1:



```
EA\大学课程学习\信安基础实验\实验一\ConsoleApplication2\Debug\ConsoleApplication2.exe
292491634125649724706298578583067489755302040058427247409073711608052500836509665659352490454953094982884551835361337799
480001778313819979868070513149907278752686181965979242418168549001688583912255923197205586068236163700120096800229812648
724185932269320562237853532952389627116550571912841686805658148701709083663215300007997562796838504780938447
This number has a 93.7500% probability of being a prime number.
```

测试数据 2:



```
EA\大学课程学习\信安基础实验\实验一\ConsoleApplication2\Debug\ConsoleApplication2.exe
876147742992673125957404768949712978720573116974723188491435550196169965040848206868200084918233743662847668000971402407
461887306389122707315529364807593342507936022301657320206278702095378618110195051280478534126716517153056984269659532882
692418682262081495725304483536777013188527470348249542840277926802938912332306310470632601156641005608958891
This number is 100% definitely a Composite number!
```

## 五、总结

本次实验我们遇到的最主要的问题一共有两个：第一，通过批处理文件编译生成 .lib 文件，中途一直报错，不断添加环境变量的值，并复制相应的文件到相关文件夹，做了很久才成功编译出 lib 文件。第二，寻找相关前人的实现，发现不可以成功运行，编译器一直报链接错误，后来我们寻找错误原因，查看了相关头文件，发现 compare 函数名字应该是 mr\_compare，修改过后，就可以成功运行我们的程序了，整个实验还是很考验耐心的，环境变量配置实在挺繁琐，GitHub 上找的那个库的批处理文件写的太简单，报错都不提醒，还需要手动操作寻找错误。

