

实验 2 RSA 密码算法

姓名：____杨行____ 学号：____200111325____

一、运行截图

分组方式 1：密文储存于文本时用数字存储

```
F:\RSA\bin\Debug\RSA.exe
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
1
第一个大素数p: 9677, 第二个大素数q: 17959, 两个素数的乘积n: 173789243, n的欧拉函数: 173761608
生成的私钥: 69403157, 公钥: 18149
公钥私钥对已存入文件中
```

生成密钥对的运行结果截图

```
miyao.txt - 记事本

文件  编辑  查看

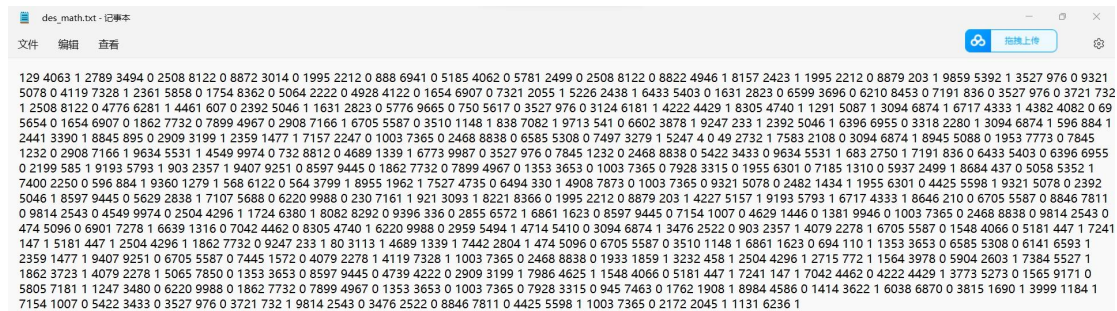
(e,n): 18149 , 173789243
(d,n): 69403157 , 173789243
```

生成的密钥对储存于文本中

```
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
2
1. 使用自动生成的密钥
2. 自己输入密钥
1
请选择密文存储于文本的方式, 这将决定密文的分组方式: 1. 数字 2. 乱码字符
1
请输入要加密的文件名, 该文件必须和本程序在同一个目录
lab2-Plaintext.txt
开始加密.....
加密后的密文长度: 723

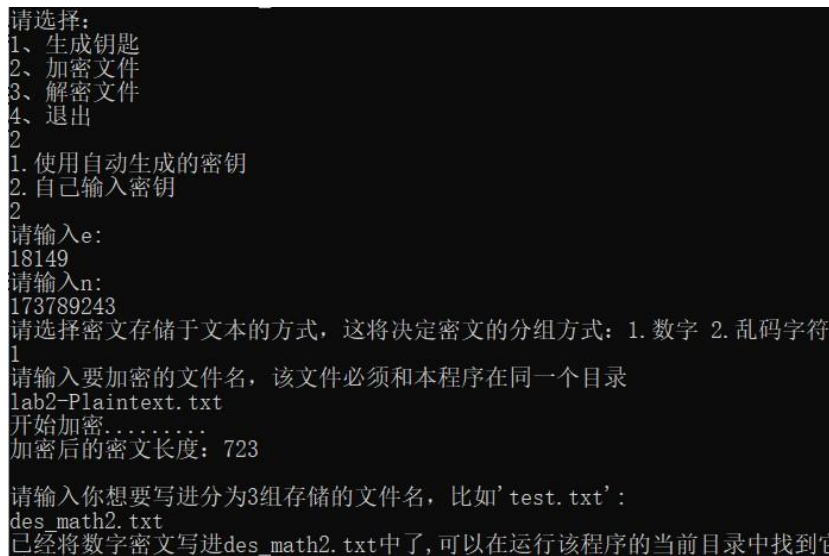
请输入你想要写进分为3组存储的文件名, 比如' test.txt ':
des_math.txt
已经将数字密文写进des_math.txt中了, 可以在运行该程序的当前目录中找到它。
```

加密文件的运行截图(分组方式采用 1 数字, 自动采用生成的密钥)



```
des_math.txt - 记事本
文件 编辑 查看
129 4063 1 2789 3494 0 2508 8122 0 8872 3014 0 1995 2212 0 888 6941 0 5185 4062 0 5781 2499 0 2508 8122 0 8822 4946 1 8157 2423 1 1995 2212 0 8879 203 1 9859 5392 1 3527 976 0 9321
5078 0 4119 7328 1 2361 5858 0 1754 8362 0 5064 2222 0 4928 4122 0 1654 6907 0 7321 2055 1 5226 2438 1 6433 5403 0 1631 2823 0 6599 3696 0 6210 8453 0 7191 836 0 3527 976 0 3721 732
1 2508 8122 0 4776 6281 1 4461 607 0 2392 5046 1 1631 2823 0 5776 9665 0 750 5617 0 3527 976 0 3124 6181 1 4222 4429 1 8305 4740 1 1291 5087 1 3094 6874 1 6717 4333 1 4382 4082 0 69
5654 0 1654 6907 0 1862 7732 0 7899 4967 0 2908 7166 1 6705 5587 0 3510 1148 1 838 7082 1 9713 541 0 6602 3878 1 9247 233 1 2392 5046 1 6396 6955 0 3318 2280 1 3094 6874 1 596 884 1
2441 3390 1 8845 895 0 2909 3199 1 2359 1477 1 7157 2247 0 1003 7365 0 2468 8838 0 6585 5308 0 7497 3279 1 5247 4 0 49 2732 1 7583 2108 0 3094 6874 1 8945 5088 0 1953 7773 0 7845
1232 0 2908 7166 1 9634 5531 1 4549 9974 0 732 8812 0 4689 1339 1 6773 9987 0 3527 976 0 7845 1232 0 2468 8838 0 5422 3433 0 9634 5531 1 683 2750 1 7191 836 0 6433 5403 0 6396 6955
0 2199 585 1 9193 5793 1 903 2357 1 9407 9251 0 8597 9445 0 1862 7732 0 7899 4967 0 1353 3653 0 1003 7365 0 7928 3315 0 1955 6301 0 7185 1310 0 5937 2499 1 8684 437 0 5058 5352 1
7400 2250 0 596 884 1 9360 1279 1 568 6122 0 564 3799 1 8955 1962 1 7527 4735 0 6494 330 1 4908 7873 0 1003 7365 0 9321 5078 0 2482 1434 1 1955 6301 0 4425 5598 1 9321 5078 0 2392
5046 1 8597 9445 0 5629 2838 1 7107 5688 0 6220 9988 0 230 7161 1 921 3093 1 8221 8366 0 1995 2212 0 8879 203 1 4227 5157 1 9193 5793 1 6717 4333 1 8646 210 0 6705 5587 0 8846 7811
0 9814 2543 0 4549 9974 0 2504 4296 1 1724 6380 1 8082 8292 0 9396 336 0 2855 6572 1 6861 1623 0 8597 9445 0 7154 1007 0 4629 1446 0 1381 9946 0 1003 7365 0 2468 8838 0 9814 2543 0
474 5096 0 6901 7278 1 6639 1316 0 7042 4462 0 8305 4740 1 6220 9988 0 2959 5494 1 4714 5410 0 3094 6874 1 3476 2522 0 903 2357 1 4079 2278 1 6705 5587 0 1548 4066 0 5181 447 1 7241
147 1 5181 447 1 2504 4296 1 1862 7732 0 9247 233 1 80 3113 1 4689 1339 1 7442 2804 1 474 5096 0 6705 5587 0 3510 1148 1 6861 1623 0 694 110 1 1353 3653 0 6585 5308 0 6141 6593 1
2359 1477 1 9407 9251 0 6705 5587 0 7445 1572 0 4079 2278 1 4119 7328 1 1003 7365 0 2468 8838 0 1933 1859 1 3232 458 1 2504 4296 1 2715 772 1 1564 3978 0 5904 2603 1 7384 5527 1
1862 3723 1 4079 2278 1 5065 7850 0 1353 3653 0 8597 9445 0 4739 4222 0 2909 3199 1 7986 4625 1 1548 4066 0 5181 447 1 7241 147 1 7042 4462 0 4222 4429 1 3773 5273 0 1565 9171 0
5805 7181 1 1247 3480 0 6220 9988 0 1862 7732 0 7899 4967 0 1353 3653 0 1003 7365 0 7928 3315 0 945 7463 0 1762 1908 1 8984 4586 0 1414 3622 1 6038 6870 0 3815 1690 1 3999 1184 1
7154 1007 0 5422 3433 0 3527 976 0 3721 732 1 9814 2543 0 3476 2522 0 8846 7811 0 4425 5598 1 1003 7365 0 2172 2045 1 1131 6236 1
```

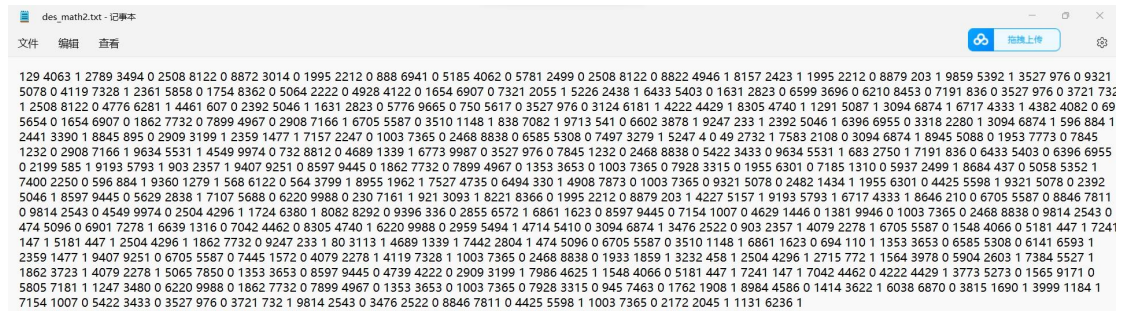
密文 des_math.txt 中的密文



```
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
2
1. 使用自动生成的密钥
2. 自己输入密钥
2
请输入e:
18149
请输入n:
173789243
请选择密文存储于文本的方式, 这将决定密文的分组方式: 1. 数字 2. 乱码字符
1
请输入要加密的文件名, 该文件必须和本程序在同一个目录
lab2-Plaintext.txt
开始加密.....
加密后的密文长度: 723

请输入你想要写进分为3组存储的文件名, 比如' test.txt ':
des_math2.txt
已经将数字密文写进des_math2.txt中了, 可以在运行该程序的当前目录中找到它。
```

加密文件的运行截图(分组方式采用 1 数字, 手动输入密钥)



```
des_math2.txt - 记事本
文件 编辑 查看
129 4063 1 2789 3494 0 2508 8122 0 8872 3014 0 1995 2212 0 888 6941 0 5185 4062 0 5781 2499 0 2508 8122 0 8822 4946 1 8157 2423 1 1995 2212 0 8879 203 1 9859 5392 1 3527 976 0 9321
5078 0 4119 7328 1 2361 5858 0 1754 8362 0 5064 2222 0 4928 4122 0 1654 6907 0 7321 2055 1 5226 2438 1 6433 5403 0 1631 2823 0 6599 3696 0 6210 8453 0 7191 836 0 3527 976 0 3721 732
1 2508 8122 0 4776 6281 1 4461 607 0 2392 5046 1 1631 2823 0 5776 9665 0 750 5617 0 3527 976 0 3124 6181 1 4222 4429 1 8305 4740 1 1291 5087 1 3094 6874 1 6717 4333 1 4382 4082 0 69
5654 0 1654 6907 0 1862 7732 0 7899 4967 0 2908 7166 1 6705 5587 0 3510 1148 1 838 7082 1 9713 541 0 6602 3878 1 9247 233 1 2392 5046 1 6396 6955 0 3318 2280 1 3094 6874 1 596 884 1
2441 3390 1 8845 895 0 2909 3199 1 2359 1477 1 7157 2247 0 1003 7365 0 2468 8838 0 6585 5308 0 7497 3279 1 5247 4 0 49 2732 1 7583 2108 0 3094 6874 1 8945 5088 0 1953 7773 0 7845
1232 0 2908 7166 1 9634 5531 1 4549 9974 0 732 8812 0 4689 1339 1 6773 9987 0 3527 976 0 7845 1232 0 2468 8838 0 5422 3433 0 9634 5531 1 683 2750 1 7191 836 0 6433 5403 0 6396 6955
0 2199 585 1 9193 5793 1 903 2357 1 9407 9251 0 8597 9445 0 1862 7732 0 7899 4967 0 1353 3653 0 1003 7365 0 7928 3315 0 1955 6301 0 7185 1310 0 5937 2499 1 8684 437 0 5058 5352 1
7400 2250 0 596 884 1 9360 1279 1 568 6122 0 564 3799 1 8955 1962 1 7527 4735 0 6494 330 1 4908 7873 0 1003 7365 0 9321 5078 0 2482 1434 1 1955 6301 0 4425 5598 1 9321 5078 0 2392
5046 1 8597 9445 0 5629 2838 1 7107 5688 0 6220 9988 0 230 7161 1 921 3093 1 8221 8366 0 1995 2212 0 8879 203 1 4227 5157 1 9193 5793 1 6717 4333 1 8646 210 0 6705 5587 0 8846 7811
0 9814 2543 0 4549 9974 0 2504 4296 1 1724 6380 1 8082 8292 0 9396 336 0 2855 6572 1 6861 1623 0 8597 9445 0 7154 1007 0 4629 1446 0 1381 9946 0 1003 7365 0 2468 8838 0 9814 2543 0
474 5096 0 6901 7278 1 6639 1316 0 7042 4462 0 8305 4740 1 6220 9988 0 2959 5494 1 4714 5410 0 3094 6874 1 3476 2522 0 903 2357 1 4079 2278 1 6705 5587 0 1548 4066 0 5181 447 1 7241
147 1 5181 447 1 2504 4296 1 1862 7732 0 9247 233 1 80 3113 1 4689 1339 1 7442 2804 1 474 5096 0 6705 5587 0 3510 1148 1 6861 1623 0 694 110 1 1353 3653 0 6585 5308 0 6141 6593 1
2359 1477 1 9407 9251 0 6705 5587 0 7445 1572 0 4079 2278 1 4119 7328 1 1003 7365 0 2468 8838 0 1933 1859 1 3232 458 1 2504 4296 1 2715 772 1 1564 3978 0 5904 2603 1 7384 5527 1
1862 3723 1 4079 2278 1 5065 7850 0 1353 3653 0 8597 9445 0 4739 4222 0 2909 3199 1 7986 4625 1 1548 4066 0 5181 447 1 7241 147 1 7042 4462 0 4222 4429 1 3773 5273 0 1565 9171 0
5805 7181 1 1247 3480 0 6220 9988 0 1862 7732 0 7899 4967 0 1353 3653 0 1003 7365 0 7928 3315 0 945 7463 0 1762 1908 1 8984 4586 0 1414 3622 1 6038 6870 0 3815 1690 1 3999 1184 1
7154 1007 0 5422 3433 0 3527 976 0 3721 732 1 9814 2543 0 3476 2522 0 8846 7811 0 4425 5598 1 1003 7365 0 2172 2045 1 1131 6236 1
```

密文 des_math2.txt 中的密文

```
F:\RSA\bin\Debug\RSA.exe
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
5
1. 使用自动生成的密钥
2. 自己输入密钥
2
请选择密文存储于文本的方式, 这取决于密文的分组方式: 1. 数字 2. 乱码字符(可以打开文档查看)
2
请输入d:
69403157
请输入n:
173789243
请输入要解密的文件名, 该文件必须和本程序在同一个目录
des_math.txt
开始解密.....
2002 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization of a public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely used encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both theoretical computer science and mathematics.
明文如上
是否需要导出明文: 1.Yes 2.No
1
请输入你想要写进明文的文件名, 比如'test.txt':
answer.txt
已经将字符明文写进answer.txt中了, 可以在运行该程序的当前目录中找到它。
```

```
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
5
1. 使用自动生成的密钥
2. 自己输入密钥
2
请选择密文存储于文本的方式, 这取决于密文的分组方式: 1. 数字 2. 乱码字符(可以打开文档查看)
2
请输入d:
69403157
请输入n:
173789243
请输入要解密的文件名, 该文件必须和本程序在同一个目录
des_math.txt
开始解密.....
2002 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization of a public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely used encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both theoretical computer science and mathematics.
明文如上
是否需要导出明文: 1.Yes 2.No
2
```

手动输入的解密过程

分组方式 2：密文储存于文本时用字符存储

```
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
1
第一个大素数p: 27631, 第二个大素数q: 26597, 两个素数的乘积n: 734901707, n的欧拉函数: 734847480
生成的私钥: 556484147, 公钥: 121283
公钥私钥对已存入文件中
请选择:
1、生成密钥
2、加密文件
3、解密文件
4、退出
2
1. 使用自动生成的密钥
2. 自己输入密钥
1
请选择密文存储于文本的方式, 这将决定密文的分组方式: 1. 数字 2. 乱码字符
2
请输入要加密的文件名, 该文件必须和本程序在同一个目录
lab2-Plaintext.txt
开始加密.....
加密后的密文长度: 1205
请输入你想要写进分为5组存储的文件名, 比如'test.txt':
des_char.txt
已经将字符密文写进des_char.txt中了, 可以在运行该程序的当前目录中找到它。
```

```
des_char.txt - 记事本
文件 编辑 查看

S
;S0G400 W0G0
3Y 0/
O00& 0_00
=0*Ea 0W0G0 0R- C%0/
O00000 0JR0000O3070a8*%0"S.G00RFH0,NV(0P0>c085000[
500:0Vb0QV0*0EUQ00C&J00
00^0=S0bK$0O3070;00c0W0G0X00" 090/c9
C&J0070U# 0/& O3070Q;50 $!C00HBQM007(10"0E*0}%b00AP00
0$00[
50000 0,EW00KGY0076X0 0)=50:03A00^0#0/00/0A: 0/c9
@S0\0R00 "0E*0L
HLO0Kc
0@K
-F1$0E0(00W58004 +0
%000)-30 0?*[W6 000Y0 `A
00*0E*0F<0701
-B0@C000KGY00
0FG004 :0HB'000*ZR000
J003070@C000
%000000&0
0FG00000bK.50EUQ00@S0\0000.007.0 J0*0<0 c0N&0K00T 0,EW00G#RW04 +01U20]G000C
&0000090~0c00L00X0 `^0 L
HLO<Jb600aRK0UG0W0Q3800#0 000=H0'S%04 +0a8*%003( 0)G[00\0 00a8*%0/c9
N&000
J00 T070000*00R0)0KR0000
00/
O00000_00E007.0 ]%b00J*0076X0 0[000=04C004 :00*^'000#002'X 0$; 7Tc
0
[M+0N&0K07N0c0;^+0* E_04 +0
%000=04C07NSA $DK00
00b 0a
00HBQM00000*0ISUL0:T6H "0E*0N\000J0*J0K*4R 76X0 30;-0 J0Q XK\@0 J0Q 0*^'00T 0 JA: 0'80(00*ZR00@5007NSA 76X0 0)=50
```

加密过程与密文展示

```
F:\RSA\bin\Debug\RSA.exe
请选择:
1. 生成密钥
2. 加密文件
3. 解密文件
4. 退出
1
第一个大素数p: 21577, 第二个大素数q: 30803, 两个素数的乘积n: 664636331, n的欧拉函数: 664583952
生成的私钥: 266966225, 公钥: 6689
公钥私钥对已存入文件中
请选择:
1. 生成密钥
2. 加密文件
3. 解密文件
4. 退出
2
1. 使用自动生成的密钥
2. 自己输入密钥
1
请选择密文存储于文本的方式, 这将决定密文的分组方式: 1. 数字 2. 乱码字符
2
请输入要加密的文件名, 该文件必须和本程序在同一个目录
lab2-Plaintext.txt
开始加密.....
加密后的密文长度: 1205
请输入你想要写进分为6组存储的文件名, 比如' test.txt':
des_char.txt
已经将字符密文写进des_char.txt中了, 可以在运行该程序的当前目录中找到它。
请选择:
1. 生成密钥
2. 加密文件
3. 解密文件
4. 退出
3
1. 使用自动生成的密钥
2. 自己输入密钥
1
请选择密文存储于文本的方式, 这取决于密文的分组方式: 1. 数字 2. 乱码字符(可以打开文档查看)
2
请输入要解密的文件名, 该文件必须和本程序在同一个目录
des_char.txt
开始解密.....
2002 A.M. TURING AWARD. RSA, an acronym for Rivest, Shamir and Adleman, uses algorithmic number theory to provide an efficient realization of a public-key cryptosystem, a concept first envisioned theoretically by Whitfield Diffie, Martin Hellman and Ralph Merkle. RSA is now the most widely used encryption method, with applications throughout the Internet to secure on-line transactions. It has also inspired breakthrough work in both theoretical computer science and mathematics.
明文如上:
是否需要导出明文: 1. Yes 2. No
```

解密过程

二、实验过程中遇到的问题有哪些？你是怎么解决的。

答：实验过程中遇到的问题如下：

1.扩展欧几里得算法求乘法逆的时候，结果可能为负数，影响实验。

解决方法：利用模的性质，将得到的乘法逆加上模后在对模取余数保证得到的为正

数。

2.应用 c 语言编写代码时，快速幂算法中的乘法存在数据过大溢出的可能。

解决方法：同时采用快速积求余的算法避免溢出。

3.密文的分组方式没有较好的想法。

解决方法：本次实验中采用 c 语言，通过 `rand()` 函数生成大素数的范围在 `1000~RAND_MAX(32767)` 之间，故而 $p*q$ 的最大值十进制不超过 10 位，基于此得到的一个加密的密文的最大值十进制不超过 10 位。以此来考虑分组储存密文。但如果 p,q 特别大，仍然没有较好的思路对密文分组。

三、 请说明你的字符分组方式，以及关键的算法例如扩展欧几里德，素数检测，快速幂等。

答：字符分组方式：

明文分组方式：直接将字符的 ASCII 码组成一个四位十进制数处理。ASCII 码的范围为 0~127。存在两个 3 位数组合成六位的可能，但考虑到常用的字符（可显示字符）是 32~127。于是对大于 99 的 ASCII 码作出处理让其除以 100 取余数加上 1。（加 1 的目的是 d 的 ASCII 码为 100，取余数后为 0，当明文两两一组，分组不足 4 位时，最后一组补上零处理，会与 d 混淆，所以加 1）。可显示字符的码值范围即是 1~28，32~99 两两一组组成一个四位十进制数处理，例如 ao 的四位十进制数为 9712。

密文分组方式：本次实验中采用 c 语言，通过 `rand()` 函数生成大素数的范围在 `1000~RAND_MAX(32767)` 之间，故而 $p*q$ 的最大值十进制不超过 10 位，基于此得到的一个加密的密文的最大值十进制不超过 10 位。以此来考虑分组储存密文。于是想到两种方式，方式 1 即将 10 位密文补充成 12 位分为 3 组，每组为一个四位十进制数据。将数据按每组分空格存入文件，解密时跳过空格依次读取 3 个解密一个明文字符。方式 2 考虑不用空格将 10 位密文分为 5 组，每组两位十进制，直接将该数看作 ASCII 码转化成对应的字符存入文本，解密时依次读取五个解密一个明文字符。

扩展欧几里德算法：


```
//扩展欧几里得算法
int EX_Euclid(int a, int b, int *d) { //求a模b的乘法逆
    int x0=1, x1=0;
    int y0=0, y1=1;
    int x, y;
    int r = b%a;
    int q= (b-r)/a;
    int temp =b;

    while(r) {
        x=x0 - q*x1; y = y0 - q*y1;
        x0 = x1; y0=y1;
        x1 = x ; y1 =y;
        b = a ; a = r ; r = b%a;
        q=(b-r)/a;
    }

    *d = (y+temp)%temp ; // 乘法逆, 保证为正数
    return a;//a为余数
}
```

利用扩展欧几里得算法的递推式计算，

通过 $(y+temp)\%temp$ 的方式保证乘法逆为正数。

素数检测：

```
int prime[10]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29}; //素数测试数组

int Miller_Rabin(int x) {
    int i, j, k;
    int num=0;
    int t = x-1;
    if(x==2) return 1; //判断2
    if(x<2 || !(x&1)) return 0; //判断偶数或1
    while(!(t&1)) { //通过移位获得, 也可通过取余获得
        num++;
        t >>=1;
    }
    for(i=0; i<10&& (prime[i]<x); i++) {
        int temp = prime[i];
        int b = Quick_Power(temp, t, x);
        for(j = 1 ; j<=num; j++) {
            k=Quick_Multiply(b, b, x);
            // printf("M:%d, %d, %d\n", b, k, prime[i]);
            if(k==1 && b!=1 && b!=x-1) {
                return 0;
            }
            b = k; //继续二次
        }
        if(b!=1) return 0;
    }

    return 1;
}
```

采用 Miller_Rabin 算法,通过费马小定理和二次检测定理检测输入的数对测试数组是否都满足素数的性质,全部满足则判断为素数。

快速幂:

```
long long Quick_Power(int a,int b,int c) //快速幂求余, a为底数, c为取模的数, b为指数
{
    long long ans=1,res=a;
    while(b)
    {
        if(b&1)
            ans=Quick_Multiply(ans,res,c); //上一步的余乘当前的余在取余, 模的乘法
        res=Quick_Multiply(res,res,c); //下一次的余
        b>>=1;
    }
    return ans;
}
```

参考课程理论的要求采用快速幂,乘法部分采用快速积。

快速积:

```
long long Quick_Multiply(int a,int b,int c) //快速积求余, a为乘数, c为取模的数, b为乘数; a*b % c
{
    long long ans=0,res=a;
    while(b)
    {
        if(b&1) {
            ans=(ans+res)%c; //当前位为1, 上一次存在的余数加上此项的余数再取余, 模的加法
        }
        res=(res+res)%c; //下一位的余数
        b>>=1;
    }
    return ans;
}
```

应用 c 语言编写代码时,快速幂算法中的乘法存在数据过大溢出的可能,同时采用快速积求余的算法避免溢出并提高速度。