**古典密码学实验**

1. **算法原理**

仿射变换（及乘积密码）的加密解密分别是：

c = E（m） ≡ a + b(mod 26)

m = D(c) ≡ a-1 (c - b)(mod 26)

其中，a,b是密钥，为满足0≤a,b≤25和gcd(a,26)等于1的整数。

其中gcd(a,26)表示a和26的最大公因数，gcd(a,26)=1表示a和26是

互素的，a-1表示a的逆元，即a-1\*a ≡ 1mod26。

1. **算法参数**
2. 参数Str 存放明文或密文的字符串，
3. 数组coprime为存放与n互素的元素
4. 秘钥空间K=（a，b）
5. c 判断加密解密操作，当c=1时，为加密，c=0为解密

**三、算法流程**

输入秘钥**入秘钥**

计算逆元

输入字符串并选择加密解密

C=1，进行加密计算 c=0，进行解密计算

输出结果

四、算法实现

#include <stdio.h>

#include <assert.h>

#define N 26 //仿射变换默认模数为26

int getGcd(int value1, int value2)//获取最大公约数

{

int gcd = 0; //最大公约数

int divisor = 0;

do

{

divisor = value1 % value2;

gcd = value2;

value1 = value2;

value2 = divisor;

}while(divisor);

return gcd;}

void setCoprime(int coprime[], int n)

{

int i = 1;

for (; i < n; i++)

if (1 == getGcd(n, i))//判断n,i是否互素

\*(coprime++) = i; //把i存入coprime中

}

int get\_a(int coprime[], int a, int n)//在coprime中寻找a的模n可逆元\_a

{

int i = 0;

for (; coprime[i] != 0; i++)

if (1 == (a\*coprime[i])%n)

return coprime[i];

return 0;

}

char \*encode(char \*c\_str, int a, int b, int n)//加密算法

{

char \*p\_str = c\_str; //减小副作用

assert (c\_str); //判断明文字符串c\_str是否为NULL

while (\*c\_str)

{

if (' ' == \*c\_str)

{

++c\_str;

continue;

}

if ((\*c\_str < 'a') || (\*c\_str > 'z')) //不是'A'到'Z'之间的就中断

assert(0);

\*c\_str -= 'a'; //将字符转化为对应数字

\*c\_str = (a\*(\*c\_str) + b)%n;//加密核心算法

\*c\_str += 'a'; //把数字转化为字符

++c\_str;

}

return p\_str;

}

char \*decode(char \*m\_str, int a, int b, int n)//解密算法

{

char \*p\_str = m\_str; //减小副作用

int coprime[32] = {0}; //存放小于n并且与n互素的元素

int \_a = 0; //存放a的模n可逆元

int i = 0; //迭代因子

assert (m\_str); //判断密文字符串m\_str是否为NULL

for (; i < 32; i++) //将数组元素赋为0

coprime[i] = 0;

setCoprime(coprime, n);//设置数组coprime存放与n互素的元素

\_a = get\_a(coprime, a, n);//在coprime中寻找a的逆元\_a

while (\*m\_str)

{

if (' ' == \*m\_str)

{

++m\_str;

continue;

}

if ((\*m\_str < 'a') || (\*m\_str > 'z')) //不是'A'到'Z'之间的就中断

assert(0);

\*m\_str -= 'a'; //把字符转化为对应数字

\*m\_str = (\_a\*(\*m\_str - b + n))%n;//解密核心算法

\*m\_str += 'a'; //将数字转化为字符

++m\_str;

}

return p\_str;

}

int main()

{

int a = 0;

int b = 0;

int c;

char str[128] = "";//str存储明文

printf("please input a,b\n");

scanf("%d %d", &a, &b);

getchar();

printf("please input str\n");

gets(str);

printf("please choose your operation(1 or 0)\n");

scanf("%d",&c);

if(c==1){

//加密

encode(str, a, b, N);

printf("ciphertext：%s\n", str);}

if(c==0){

//解密

decode(str, a, b, N);

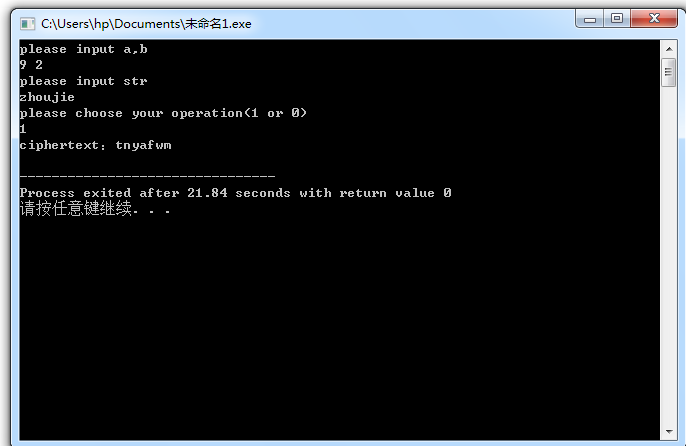
printf("plaintext：%s\n", str);}

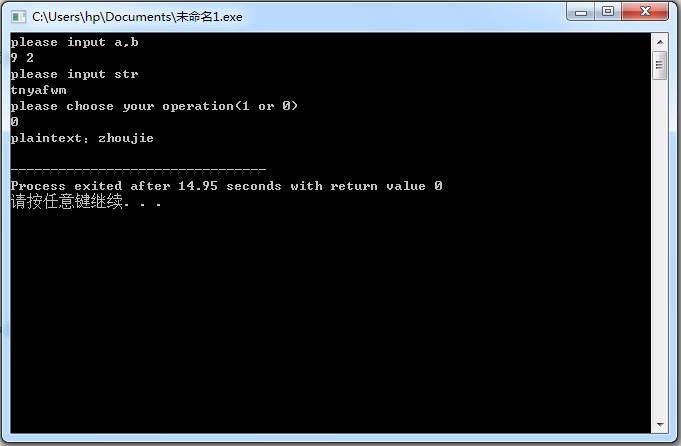
return 0;

}

**五、加密解密过程演示**

1.加密明文I am a student of SDUPSL, I love china. ZHANG,秘钥为（9,2）



2.解密密文tnyafwm,秘钥为（9,2）

**六、总结体会**

仿射密码为单表代替密码，也是一种线性变换。在进行本次古典密码实验前，我查询了大量资料及网上论坛的代码原文，通过学习他们的算法思想及主要代码的具体实现语句，我逐渐有了仿射密码代码的大体框架。之后又在不断修改代码的过程中对仿射密码以及C语言的语句思想有了更加深刻的认识。

在编程过程中遇到很多问题，看似简单的算法却不容易实现。小小的一个差错就可能使程序出错，无法得出结果。

仿射密码算法中有几个关键点：

其一，密钥的限制，k1、k2是模26的整数（即0-25），而且k1必须与26互素，输入密钥后即进行判断，只有在符合要求的情况下程序才能继续执行，但我的程序中没有这一部分。

其二，字符的转换，只有正确转换字符才能得出结果。

最后，要得到解密密钥，需要对k1和26进行处理才能获得，我在次使用一个函数gcd()来实现。