电子科技大学信息与软件工程学院

实 验 报 告

学　　号 2018091618008

姓　　名 袁昊男

（实验）　课程名称 信息安全数学基础

理论教师 熊虎

实验教师 熊虎

电 子 科 技 大 学

实　验　报　告

学生姓名：袁昊男　　学号：2018091618008　　指导教师：熊虎

实验地点：信软楼 303　　实验时间：2019.12.15

一、 实验名称：模多项式求逆的实现

二、 实验学时：2 学时

三、 实验目的：

1、 熟悉一种模多项式求逆算法；

2、 运用高级程序设计语言完成一种模多项式求逆算法的程序，加深对多项式运算的理解；

3、 提供该算法的源代码及测试用例，给出运行结果分析。

四、 实验原理：

1、 **定理 6.1.2（多项式的带余除法）**设 $f(x),g(x)\in F[X]$， $g(x)\neq 0$，则一定存在多项式 $q(x),r(x)\in F[X]$ 使得

$$f(x)=q(x)g(x)+r(x) \tag{6.1}$$

其中 $\deg r(x)<\deg g(x)$ 或者 $r(x)=0$，而且 $q(x),r(x)$ 是唯一的。 $q(x)$ 称为 $g(x)$ 除 $f(x)$ 的商式，记为 $f(x)\ \mathrm{div}\ g(x)$； $r(x)$ 称为 $g(x)$ 除 $f(x)$ 的余式，记为 $f(x)\bmod g(x)$。

2、 **定义 6.1.5** 设 $f(x),g(x)\in F[X]$，如果存在 $q(x)\in F[x]$，使得 $f(x)=q(x)\cdot g(x)$，则称 $g(x)$ 整除 $f(x)$，记为 $g(x)\mid f(x)$。记 $g(x)\nmid f(x)$ 为 $g(x)$ 不整除 $f(x)$。当 $g(x)\mid f(x)$ 时，称 $g(x)$ 为 $f(x)$ 的因式，而称 $f(x)$ 为 $g(x)$ 的倍式。

3、 **定理 6.1.3** 设 $f(x),g(x)\in F[X]$， $g(x)\neq 0$，则 $g(x)$ 整除 $f(x)$ 的充要条件是 $g(x)$ 除 $f(x)$ 的余式为零。

4、 **定义 6.2.1** 如果 $h(x)$ 即是 $f(x)$ 的因式，又是 $g(x)$ 的因式，则称 $h(x)$ 是 $f(x)$ 和 $g(x)$ 的公因式。若 $f(x)$ 和 $g(x)$ 的首项系数为 1 的公因式 $d(x)$ 满足 $f(x)$ 和 $g(x)$ 的公因式都是 $d(x)$ 的因式，则称 $d(x)$ 是 $f(x)$ 和 $g(x)$ 的最大公因式，记为 $d(x)=(f(x),g(x))$。根据带余除法和多项式整除的性质，如果有等式

$$f(x)=q(x)g(x)+r(x)$$

成立，那么 $f(x),g(x)$ 和 $g(x),r(x)$ 有相同的公因式，因此有

$$\gcd(f(x),g(x))=\gcd(g(x),r(x))$$

5、**定理 6.2.1** 对于 $F[X]$ 中的多项式 $f(x)$ 和 $g(x)$，一定存在最大公因式 $d(x)\in F[X]$，且 $d(x)$ 可以表示成 $f(x)$ 和 $g(x)$ 的一个组合，即存在 $u(x),v(x)\in F[X]$，使得

$$d(x)=u(x)f(x)+v(x)g(x)$$

6、**定义 6.2.2** 如果 $F[X]$ 中的多项式 $f(x)$ 和 $g(x)$ 满足 $\gcd(f(x),g(x))=1$，则称 $f(x)$ 与 $g(x)$ 互素。

7、**定理 6.2.2** $F[X]$ 中的两个多项式 $f(x)$ 和 $g(x)$ 互素的充要条件是存在 $u(x),v(x)\in F[X]$，使得

$$u(x)f(x)+v(x)g(x)=1$$

8、**定理 6.3.3** 设 $f(x),g(x)\in F[X]$ 为非零多项式，$g(x)$ 模 $f(x)$ 有乘法逆元当且仅当 $\gcd(f(x),g(x))=1$。

9、求 $g(x)\bmod(f(x))$ 下的逆元：

根据定理 6.2.2 及 6.3.3，可得 $v(x)=g^{-1}(x)\bmod(f(x))$。

10、域 $\mathbb{Z}_p$ 上多项式乘法：

$$\begin{aligned}f(x)\cdot g(x)&=a_nb_mx^{n+m}+(a_nb_{m-1}+a_{n-1}b_m)x^{n+m-1}+\cdots+(a_1b_0+a_0b_1)x+a_0b_0\\&=\sum_{s=0}^{m+n}\left(\sum_{i+j=s}a_ib_j\right)x^s\end{aligned}$$

11、域 $\mathbb{Z}_p$ 上多项式带余除法：

$$f_1(x)=f(x)-a_nb_m^{-1}x^{n-m}g(x)=f(x)-q(x)g(x)$$

直到 $f_1(x)$ 的次数小于 $g(x)$ 的次数。

12、**算法 6.2.2** 适用于 $F[X]$ 的扩展的欧几里得算法。

输入：两个多项式 $f(x),g(x)\in F[X]$。

输出：$d(x)=(f(x),g(x))$ 以及多项式 $u(x),v(x)$ 满足 $u(x)f(x)+v(x)g(x)=d(x)$。

(1) 如果 $g(x)=0$，则令 $d(x)\leftarrow f(x)$，$u(x)\leftarrow 1$，$v(x)\leftarrow 0$，返回 $(d(x),u(x),v(x))$。

(2) 令 $u_2(x)\leftarrow 1$，$u_1(x)\leftarrow 0$，$v_2(x)\leftarrow 0$，$v_1(x)\leftarrow 1$。

(3) 当 $g(x)\neq 0$ 时作以下工作：

(3.1) $q(x)\leftarrow f(x)\ \mathrm{div}\ g(x)$，$r(x)\leftarrow f(x)-g(x)q(x)$；

(3.2) $u(x)\leftarrow u_2(x)-q(x)u_1(x)$，$v(x)\leftarrow v_2(x)-q(x)v_1(x)$；

(3.3) $f(x)\leftarrow g(x)$，$g(x)\leftarrow r(x)$；

(3.4) $u_2(x)\leftarrow u_1(x)$，$u_1\leftarrow u(x)$，$v_2(x)\leftarrow v_1(x)$，$v_1(x)\leftarrow v(x)$。

(4) 令 $d(x)\leftarrow f(x)$，$u(x)\leftarrow u_2(x)$，$v(x)\leftarrow v_2(x)$。

(5) 返回 $(d(x),u(x),v(x))$。

五、 实验内容：

1、 模多项式求逆算法，包括：

(1) 域上多项式乘法；

(2) 域上多项式带余除法；

(3) 域上多项式的扩展的欧几里得算法。

2、 提供算法的源代码及测试用例，给出运行结果分析；

3、 按照标准实验报告整理实验内容。

六、 实验器材（设备、元器件）：

电脑一台。

七、 实验步骤：

1、 学习实验原理，尤其是适用于 $F[X]$ 的扩展的欧几里得算法，理解算法实现的过程；

2、 分析题目需求，设计数据结构；

3、 编码实现，并按测试用例进行输入输出测试；

4、 分析实验结果，撰写实验报告。

八、 实验结果与分析（含重要数据结果分析或核心代码流程分析）

1、 代码分析

```
1. int Euclid(int f[], int g[])
2. {
3.     u2[0] = 1, u2[1] = MAX;
4.     int u2t = 1;
5.     u1[0] = 0, u1[1] = MAX;
6.     int u1t = 0;
7.     v2[0] = 0, v2[1] = MAX;
8.     int v2t = 0;
9.     v1[0] = 1, v1[1] = MAX;
10.    int v1t = 1;
11.    int gt = checktimes(g);
12.    int gn = checkn(g);
13.    int ft = checktimes(f);
14.    int fn = checkn(f);
15.    while (gt != 0 && gn != 0)
16.    {
17.        //  memset(q, 0, sizeof(q));
18.        //  memset(r, 0, sizeof(r));
19.//      memset(q_new, 0, sizeof(q_new));
20.        int qt = Div(f, g);
21.        int rt = checktimes(r_new);
22.        Mul(q_new, u1); //结果是 c
```

```
        int ut = Sub(u2, c);        //结果是 c
        memcpy(u, c, sizeof(int)*(ut + 2));

        Mul(q_new, v1); //结果是 c
        int vt = Sub(v2, c);
        memcpy(v, c, sizeof(int)*(vt + 2));

        //          memset(v, 0, sizeof(v));
           //   memcpy(v, c, sizeof(int)*(vt + 2));
        memset(f, 0, sizeof(f));
        memcpy(f, g, sizeof(int)*(gt + 2));
        memset(g, 0, sizeof(g));
        memcpy(g, r_new, sizeof(int)*(rt + 2));

        u1t = checktimes(u1);
        u2t = checktimes(u2);
        v1t = checktimes(v1);
        v2t = checktimes(v2);
        memset(u2, 0, sizeof(u2));
        memcpy(u2, u1, sizeof(int)*(u1t + 2));
        memset(u1, 0, sizeof(u1));
        memcpy(u1, u, sizeof(int)*(ut + 2));
        memset(v2, 0, sizeof(v2));
        memcpy(v2, v1, sizeof(int)*(v1t + 2));
        memset(v1, 0, sizeof(v1));
        memcpy(v1, v, sizeof(int)*(vt + 2));
        gt = checktimes(g);
        gn = checkn(g);
    }
    memset(d, 0, sizeof(d));
    ft = checktimes(f);
    memcpy(d, f, sizeof(int)*(ft + 2));
    memset(u, 0, sizeof(u));
    u2t = checktimes(u2);
    memcpy(u, u2, sizeof(int)*(u2t + 2));
    memset(v, 0, sizeof(v));
    v2t = checktimes(v2);
    memcpy(v, v2, sizeof(int)*(v2t + 2));
    int vt = checktimes(v);
    return vt;
}
```

说明：模多项式求逆的主要依靠适用于$F[X]$的扩展的欧几里得算法来实现，这里以扩展的欧几里得算法作为主要代码来说明（完整代码见文末）。Euclid 函数带两个参数：待操作的两个多项式的系数数组 f 及 g。本算法是在实数域内的模多项式求逆，因此理论上 f 与 g 的系数可以取到实数域的所有数。然后按照算法 6.2.2 的步骤依次执行代码。由于本实验目的是要求$g(x) \bmod (f(x))$下的逆元，因此 Euclid 函数返回值为逆元多项式$v(x)$的最高次数，且$v(x)$多项式的系数将会保存在全局数组 v 中。另$d(x)$、$u(x)$的系数同样保存在全局数组 d、u 中。当$d(x)=1$时说明$f(x)$与$g(x)$互素，此时的$v(x)$即为所求。Euclid 函数中还调用了多项式乘法 Mul 函数、多项式减法 Sub 函数来完成多项式模的运算操作；除此之外，还有一些辅

助函数，如返回多项式最高次数的函数 checktimes、返回多项式项数的函数 checkn 等。

2、 测试用例

注：本实验所测试的多项式都是在实数域内的多项式。

求： $g(x) \bmod (f(x))$

序号	输入 $(f(x), g(x))$	预期输出 $(d(x), u(x), v(x))$	实际输出 $(d(x), u(x), v(x))$
1	$f(x)=x^4+2x^3-x^2-4x-2$ $g(x)=x^4+x^3-x^2-2x-2$	$d(x)=x^2-2$ $u(x)=-x-1$ $v(x)=x+2$	$d(x)=x^2-2$ $u(x)=-x-1$ $v(x)=x+2$
2	$f(x)=x^4+x^2+x+1$ $g(x)=x^5+x^4+x^3+x+1$	$d(x)=x+1$ $u(x)=x^3+x+1$ $v(x)=-x^2-x$	$d(x)=x+1$ $u(x)=x^3+x+1$ $v(x)=-x^2-x$
3	$f(x)=x^4-x^3-4x^2+4x+1$ $g(x)=x^2-x-1$	$d(x)=1$ $u(x)=-x-1$ $v(x)=x^3+x^2-3x-2$	$d(x)=1$ $u(x)=-x-1$ $v(x)=x^3+x^2-3x-2$

3、 算法过程分析

a) 测试用例 1

初始化阶段			
$f(x)$	$x^4+2x^3-x^2-4x-2$		
$g(x)$	$x^4+x^3-x^2-2x-2$		
$u_2(x)$	1	$u_1(x)$	0
$v_2(x)$	0	$v_1(x)$	1

循环次数	1	2
$q(x)$	1	$x+1$
$r(x)$	x^3-2x	x^2-2
$u(x)$	1	$-x-1$
$v(x)$	-1	$x+2$
$f(x)$	$x^4+x^3-x^2-2x-2$	x^3-2x
$g(x)$	x^3-2x	x^2-2
$u_2(x)$	0	1

$u_1(x)$	1	$-x-1$
$v_2(x)$	1	-1
$v_1(x)$	-1	$x+2$
循环次数	**3**	**$g(x)=0$ 终止循环**
$q(x)$	x	
$r(x)$	0	
$u(x)$	x^2+x+1	
$v(x)$	$-x^2-2x-1$	输出：
$f(x)$	x^2-2	$d(x) \leftarrow f(x)=x^2-2$；
$g(x)$	0	$u(x) \leftarrow u_2(x)=-x-1$；
$u_2(x)$	$-x-1$	$v(x) \leftarrow v_2(x)=x+2$
$u_1(x)$	x^2+x+1	
$v_2(x)$	$x+2$	
$v_1(x)$	$-x^2-2x-1$	

b) 测试用例 2

初始化阶段			
$f(x)$	x^4+x^2+x+1		
$g(x)$	$x^5+x^4+x^3+x+1$		
$u_2(x)$	1	$u_1(x)$	0
$v_2(x)$	0	$v_1(x)$	1

循环次数	**1**	**2**
$q(x)$	0	$x+1$
$r(x)$	x^4+x^2+x+1	x^2+x
$u(x)$	1	$-x-1$
$v(x)$	0	1
$f(x)$	$x^5+x^4+x^3+x+1$	x^4+x^2+x+1
$g(x)$	x^4+x^2+x+1	x^2+x
$u_2(x)$	0	1
$u_1(x)$	1	$-x-1$
$v_2(x)$	1	0
$v_1(x)$	0	1

循环次数	3	4
$q(x)$	x^2+x	x
$r(x)$	$x+1$	0
$u(x)$	x^3+x+1	$-x^4-x^2-1$
$v(x)$	$-x^2-x$	x^3+x^2+1
$f(x)$	x^2+x	$x+1$
$g(x)$	$x+1$	0
$u_2(x)$	$-x-1$	x^3+x+1
$u_1(x)$	x^3+x+1	$-x^4-x^2-1$
$v_2(x)$	1	$-x^2-x$
$v_1(x)$	$-x^2-x$	x^3+x^2+1

循环次数	$g(x)=0$ 终止循环
$q(x)$	
$r(x)$	
$u(x)$	
$v(x)$	输出：
$f(x)$	$d(x)\leftarrow f(x)=x+1$；
$g(x)$	$u(x)\leftarrow u_2(x)=x^3+x+1$；
$u_2(x)$	$v(x)\leftarrow v_2(x)=-x^2-x$
$u_1(x)$	
$v_2(x)$	
$v_1(x)$	

c) 测试用例 3

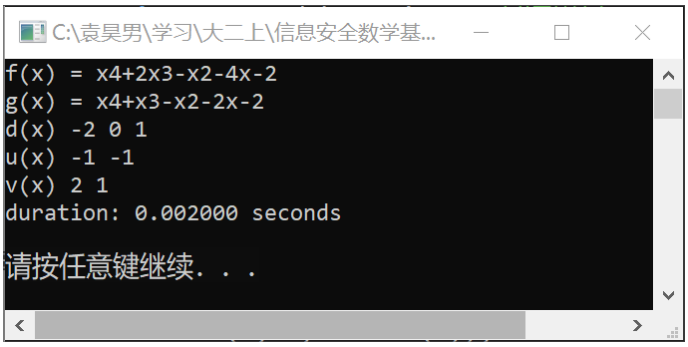
初始化阶段			
$f(x)$	$x^4-x^3-4x^2+4x+1$		
$g(x)$	x^2-x-1		
$u_2(x)$	1	$u_1(x)$	0
$v_2(x)$	0	$v_1(x)$	1

循环次数	1	2
$q(x)$	x^2-3	x
$r(x)$	$x-2$	$x-1$

$u(x)$	1	$-x$
$v(x)$	$-x^2+3$	x^3-3x+1
$f(x)$	x^2-x-1	$x-2$
$g(x)$	$x-2$	$x-1$
$u_2(x)$	0	1
$u_1(x)$	1	$-x$
$v_2(x)$	1	$-x^2+3$
$v_1(x)$	$-x^2+3$	x^3-3x+1
循环次数	**3**	**4**
$q(x)$	1	$-x+1$
$r(x)$	-1	0
$u(x)$	$x+1$	x^2-x-1
$v(x)$	$-x^3-x^2+3x+2$	$-x^4+x^3+4x^2-4x-1$
$f(x)$	$x-1$	-1
$g(x)$	-1	0
$u_2(x)$	$-x$	$x+1$
$u_1(x)$	$x+1$	x^2-x-1
$v_2(x)$	x^3-3x+1	$-x^3-x^2+3x+2$
$v_1(x)$	$-x^3-x^2+3x+2$	$-x^4+x^3+4x^2-4x-1$
循环次数	**$g(x)=0$ 终止循环**	
$q(x)$		
$r(x)$		
$u(x)$		
$v(x)$	输出：	
$f(x)$	$d(x)\leftarrow f(x)=-1=1$；	
$g(x)$	$u(x)\leftarrow u_2(x)=x+1$；	
$u_2(x)$	$v(x)\leftarrow v_2(x)=-x^3-x^2+3x+2$	
$u_1(x)$		
$v_2(x)$		
$v_1(x)$		

4、 运行截图

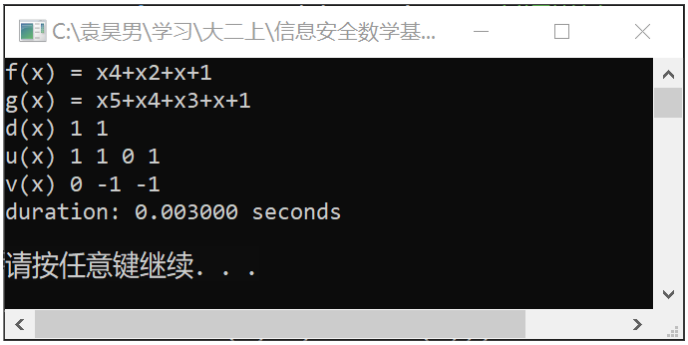
(1) 测试用例 1



注：系数 $-2\ 0\ 1$ 表示 $d(x)$ 的表达式为 $d(x)=-2\cdot x^0+0\cdot x+1\cdot x^2=x^2-2$，下同。

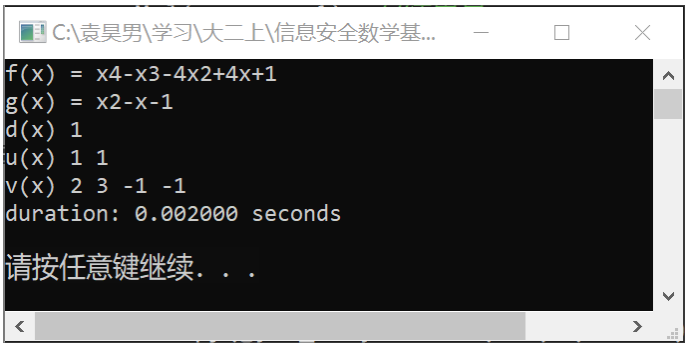
说明：因为 $d(x)\neq 1$，因此 $g(x)\bmod(f(x))$ 无逆元。结果与手工计算验证正确。

(2) 测试用例 2



说明：因为 $d(x)\neq 1$，因此 $g(x)\bmod(f(x))$ 无逆元。结果与手工计算验证正确。

(3) 测试用例 3



说明：因为 $d(x)=1$，因此 $g(x)\bmod(f(x))$ 逆元 $g^{-1}(x)\bmod(f(x))=v(x)$ $v(x)=-x^3-x^2+3x+2$。结果与手工计算验证正确。

九、 总结及心得体会：

因为在上实验课前就完成了本次实验内容，因此本次完成的模多项式求逆算

法是基于实数域的，与老师要求的二进制域稍有不同，但实现的重要算法如扩展的欧几里得算法大同小异。由于多项式域部分老师在课堂上教授课时较少，因此一开始独自完成该实验对我来说难度较大。我首先学习了本实验的实验原理，然后用了几个例子对其在草稿纸上进行扩展欧几里得算法的演算，让我熟悉、掌握了该算法的思路与执行过程，其中要用到一些多项式的加、减、乘、除运算，结合平时在纸上对多项式做长除法的运算思路编写了对应的四则运算算法，但此部分算法针对不同的多项式可能会出现不能预料的错误，因此在“修补”四则运算算法错误方面花了很长的时间，由于一开始就没有把算法设计的非常好，导致代码显得非常杂乱和冗余，这也提醒自己以后要先思考、再编码，不要急于求成。

通过本次实验，我掌握了模多项式求逆的算法以及编码，针对测试用例对代码进行了测试，并分析了算法的执行过程，提高了自己对多项式域的理解以及编码能力。

十、 对本实验过程及方法、手段的改进建议：

本实验中多项式的相关内容可以在理论教学时适当增加课时数，且本课程的实验统一安排在期末进行学生完成的时间压力较大。

报告评分：

指导教师签字：

附：完整代码

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAX 100
int c[MAX];
int temp_new[MAX];
int temp2[MAX];
int q[MAX];
int q_new[MAX];
int r[MAX];
int r_new[MAX];
int super = 0;
int flag = 0;
int flag1 = 0;

int u2[MAX];
int u1[MAX];
int v2[MAX];
int v1[MAX];
int u[MAX];
int v[MAX];
int d[MAX];


int Add(int f[], int g[])   //  0用来存放最高项次数
{
    memset(c, 0, sizeof(c));
    int ft = checktimes(f);
    int gt = checktimes(g);
    int ct = (ft > gt ? ft : gt);
    c[ct + 1] = MAX;
    int i;
    for (i = 0; i <= ct; i++)
    {
        if (f[i] != MAX && g[i] != MAX)
            c[i] = f[i] + g[i];
        else if (f[i] == MAX)
            c[i] = g[i];
        else
            c[i] = f[i];
    }
    while (c[ct] == 0)
        if (c[ct] == 0)
        {
            c[ct + 1] = 0;
            c[ct] = MAX;
            ct--;
        }
    return ct;
}

int checktimes(int a[])
{
    int i;
    for (i = 0; i < MAX; i++)
```

```
57.     {
58.         if (a[i] == MAX)
59.         {
60.             if (i == 0)
61.                 return i;
62.             else
63.                 return i - 1;
64.         }
65.     }
66. }
67. 
68. int Sub(int f[], int g[])//f-g
69. {
70.     //  memset(c, 0, sizeof(c));
71.     int ft = checktimes(f);
72.     int gt = checktimes(g);
73.     int i;
74.     for (i = 0; i < gt + 1; i++)
75.         g[i] *= (-1);
76.     int ct = (ft > gt ? ft : gt);
77.     c[ct + 1] = MAX;
78.     for (i = 0; i <= ct; i++)
79.     {
80.         if (f[i] != MAX && g[i] != MAX)
81.             c[i] = f[i] + g[i];
82.         else if (f[i] == MAX)
83.             c[i] = g[i];
84.         else
85.             c[i] = f[i];
86.     }
87.     while (c[ct] == 0)
88.         if (c[ct] == 0 && ct >= 0)
89.         {
90.             c[ct + 1] = 0;
91.             c[ct] = MAX;
92.             ct--;
93.         }
94.         else
95.             ct = 0;
96.     return ct;
97. }
98. 
99. int Mul(int f[], int g[])
100. {
101.     memset(c, 0, sizeof(c));
102.     int ft = checktimes(f);
103.     int gt = checktimes(g);
104.     int i, j;
105.     int ct = ft + gt;
106.     c[ct + 1] = MAX;
107.     for (i = 0; i < ft + 1; i++)
108.     {
109.         for (j = 0; j < gt + 1; j++)
110.         {
111.             c[i + j] = f[i] * g[j] + c[i + j];
112.         }
113. 
114.     }
```

```
115.     return ct;
116. }
117. int checkn(int a[])
118. {
119.     int i = 0;
120.     int count = 0;
121.     int times = checktimes(a);
122.     if (times == 0)
123.         return 0;
124.     else
125.     {
126.         for (i = 0; i <= times; i++)
127.         {
128.             if (a[i] != 0)
129.                 count++;
130.         }
131.     }
132.     return count;
133. }
134. int Div(int f[], int g[])//f÷g
135. {
136.     memset(c, 0, sizeof(c));
137.     memset(r, 0, sizeof(r));
138.     memset(q, 0, sizeof(q));
139. //  memcpy(q, q_new, sizeof(int)*(flag + 2));
140.     int ft = checktimes(f);
141.     int gt = checktimes(g);
142.     int qt;
143.     if (ft < gt)
144.     {
145.         q[super] = 0;
146.         //      q[super] = MAX;
147.         memcpy(r, f, sizeof(int)*(ft + 1));
148.         return 0;
149.     }
150.     else//c 作为每次的余数
151.     {
152.         if (ft == gt && checkn(f) < checkn(g))
153.             return 0;
154.         qt = ft - gt;
155.         super = qt;
156.
157.         q_new[super] = f[ft] / g[gt];
158.         if (flag1 < qt)
159.             flag1 = qt;
160.         q_new[flag1 + 1] = MAX;
161.         q[super + 1] = MAX;
162.         q[super]= f[ft] / g[gt];
163.         int temp[MAX];
164.
165.         do
166.         {
167.             int k = Mul(q, g);
168.             memcpy(temp_new, c, sizeof(int)*(k + 1));
169.             temp_new[k + 1] = MAX;
170.             int ct = Sub(f, temp_new);
171.             int qt = checktimes(q);
172.             flag = qt;
```

```
            if (ct == 0)
            {
                memcpy(q_new, q, sizeof(int)*(qt + 2));
                memcpy(r_new, c, sizeof(int)*(ct + 2));
                r_new[ct + 1] = MAX;
                return qt;
            }
            else {
                memcpy(r_new, c, sizeof(int)*(ct + 2));
                r_new[ct + 1] = MAX;
                memset(f, 0, sizeof(f));
                memcpy(f, c, sizeof(int)*(ct + 2));

                //          memcpy(q_new, q, sizeof(int)*(qt + 2));

                if (Div(f, g) == 0)
                {
                    return qt;
                }
            }
        } while (checktimes(c) >= gt);
    }
}

int Euclid(int f[], int g[])
{
    u2[0] = 1, u2[1] = MAX;
    int u2t = 1;
    u1[0] = 0, u1[1] = MAX;
    int u1t = 0;
    v2[0] = 0, v2[1] = MAX;
    int v2t = 0;
    v1[0] = 1, v1[1] = MAX;
    int v1t = 1;
    int gt = checktimes(g);
    int gn = checkn(g);
    int ft = checktimes(f);
    int fn = checkn(f);
    while (gt != 0 && gn != 0)
    {
        //  memset(q, 0, sizeof(q));
        //  memset(r, 0, sizeof(r));
//      memset(q_new, 0, sizeof(q_new));
        int qt = Div(f, g);
        int rt = checktimes(r_new);
        Mul(q_new, u1); //结果是 c
        int ut = Sub(u2, c);        //结果是 c
        memcpy(u, c, sizeof(int)*(ut + 2));

        Mul(q_new, v1); //结果是 c
        int vt = Sub(v2, c);
        memcpy(v, c, sizeof(int)*(vt + 2));

        //          memset(v, 0, sizeof(v));
           //   memcpy(v, c, sizeof(int)*(vt + 2));
        memset(f, 0, sizeof(f));
        memcpy(f, g, sizeof(int)*(gt + 2));
        memset(g, 0, sizeof(g));
```

```
        memcpy(g, r_new, sizeof(int)*(rt + 2));

        u1t = checktimes(u1);
        u2t = checktimes(u2);
        v1t = checktimes(v1);
        v2t = checktimes(v2);
        memset(u2, 0, sizeof(u2));
        memcpy(u2, u1, sizeof(int)*(u1t + 2));
        memset(u1, 0, sizeof(u1));
        memcpy(u1, u, sizeof(int)*(ut + 2));
        memset(v2, 0, sizeof(v2));
        memcpy(v2, v1, sizeof(int)*(v1t + 2));
        memset(v1, 0, sizeof(v1));
        memcpy(v1, v, sizeof(int)*(vt + 2));
        gt = checktimes(g);
        gn = checkn(g);
    }
    memset(d, 0, sizeof(d));
    ft = checktimes(f);
    memcpy(d, f, sizeof(int)*(ft + 2));
    memset(u, 0, sizeof(u));
    u2t = checktimes(u2);
    memcpy(u, u2, sizeof(int)*(u2t + 2));
    memset(v, 0, sizeof(v));
    v2t = checktimes(v2);
    memcpy(v, v2, sizeof(int)*(v2t + 2));
    int vt = checktimes(v);
    return vt;
}

int main()
{
    int f[6] = { -2,-4,-1,2,1,MAX };
    int g[6] = { -2,-2,-1,1,1,MAX };
    //int ct = Add(f, g);
    int i = 0;
    //for (i = 0; i < ct+1; i++)
    //  printf("%d", c[i]);
    //ct = Sub(f, g);
    //for (i = 0; i < ct + 1; i++)
    //  printf("%d", c[i]);
    //int ct = Div(f, g);
    int vt = Euclid(f, g);
    for (i = 0; i < vt + 1; i++)
        printf("%d", v[i]);
    system("pause");
    return 0;
}
```