# 递归求解最大公约数GCD

Greatest Common Divisor

1.给定两个整数 a 和 b，如果 b 为 0，则最大公约数为 a。

2.否则，计算 a 除以 b 的余数 r。

3.递归地调用算法，将 b 和 r 作为新的参数，即求 b 和 r 的最大公约数。

4.递归的终止条件是 b=0，此时 a 就是最大公约数。

# 求解最小公倍数LCM

这是因为两个数的乘积等于它们的最大公约数和最小公倍数的乘积。即：

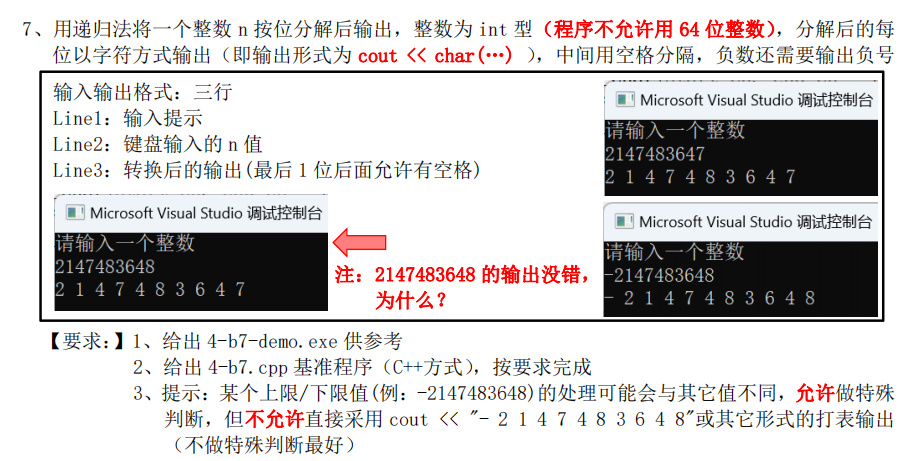
**LCM(a,b)×GCD(a,b)=a×b**

因此，我们可以先计算出两个数的最大公约数，然后用它们的乘积除以最大公约数来得到最小公倍数。

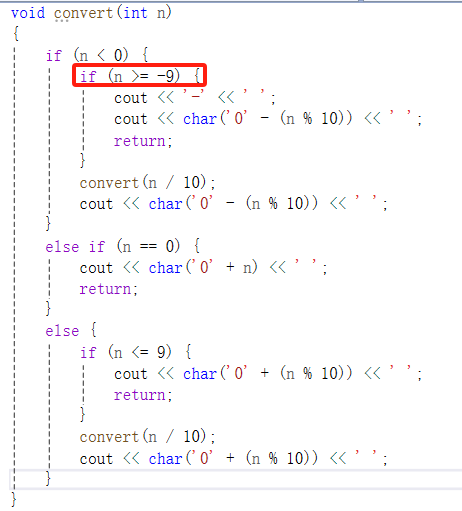
**整数分解——正序**

n%10 --相当于从最后一位开始分解 eg.258%10=8；258/10=25->25%10=5;

怎样正序输出？--调整一下递归的顺序！！



这个容易错！！！因为int型下限是**-**214748364**8**，而上限是2147458364**7**

**这样非常聪明！！（正序输出）**

### 写一个递归函数，求某个十进制正整数是否某个基数的幂

要写一个递归函数来判断一个十进制正整数是否是一个给定基数的幂，我们需要定义一个函数，该函数会**不断将给定的整数除以基数，并检查最终的结果是否为1**。如果在任何一步中，结果不是整数（即有余数），或者最终结果不是1，那么这个数就不是该基数的幂。

**递归打印代替循环**

void fun1(char c, char c1)

{

if (c == c1) {

return;

}

else {

cout << ' ';

fun1(c - 1, c1);

}

}

void fun2(char c)

{

if (c < 'A') {

return;

}

else {

cout << c;;

fun2(c - 1);

}

}

void fun3(char c, char c1)

{

if (c > c1)

{

cout << endl;

return;

}

else {

putchar(c);

fun3(c + 1, c1);

}

}

void fun(char c)

{

if (c > 'Z') //比Z大就结束了

return;

int i;

//分析一行的内容有 空格 降序字符 升序字符 回车组成

fun1('Z', c);//某行的空格数

fun2(c);//降序字符

fun3('B', c);//升序字符，最后带换行

fun(c + 1);//下一个字符

}

int main()

{

fun('A');//从A开始

return 0;

}