# 第1次编程练习报告

姓名：申宗尚 学号：2213924 班级：信安班

##### 编程练习1——Eratosthenes 筛法

* **源码部分：**
* #include<iostream>  
  #include<cmath>  
  using namespace std;  
  #define maxn 1000000  
  int noprime[maxn];  
  int main()  
  {  
   int cnt = 0;  
   for (int i = 2; i <= sqrt(maxn); i++)  
   {  
   if (noprime[i] == 0)  
   {  
   for (int j = i+i; j < maxn; j += i)  
   {  
   noprime[j] = 1;  
   }  
   }  
   }  
   for (int i = 2; i < maxn; i++)  
   {  
   if (noprime[i] == 0)  
   {  
   cout << i << ",";  
   cnt++;  
   }  
   }  
   cout<<endl;  
   cout << "Total:" << cnt << endl;  
   system("pause");  
   return 0;  
  }
* **说明部分：通过noprime数组记录当前数是否为素数（全局变量初始为0，即均视为素数，随后通过1～sqrt（n）的遍历删掉对应素数的所有倍数），最后通过一次全部遍历，输出所有noprime值仍为0（素数）的编号。在遍历过程中利用cnt变量存储数量，最后通过Total打印出来。**
* **运行示例：**

文本

描述已自动生成

* **其他：**

Q(a):普通算法（依次遍历每个数从1～sqrt（本身），外层n重内层sqrt（n层）），时间复杂度为O（n\*sqrt（n）），Eratosthenes筛法为O（n\*log logn）。

而空间复杂度，Eratosthenes筛法需要一个大小为n的数组标记是否为素数，普通算法只需求当前待测数，故空间复杂度前者为O（n），后者为O（1）

Q(b):在递归调用时，可以对较大的给定n进行开方处理，以求更大范围素数

Q(c):不适用，如Q(a)所言，该算法空间复杂度为Q（n），当素数过大时（如2^512），会导致内存空间溢出等问题。

##### 编程练习2——最大公因数和最小公倍数

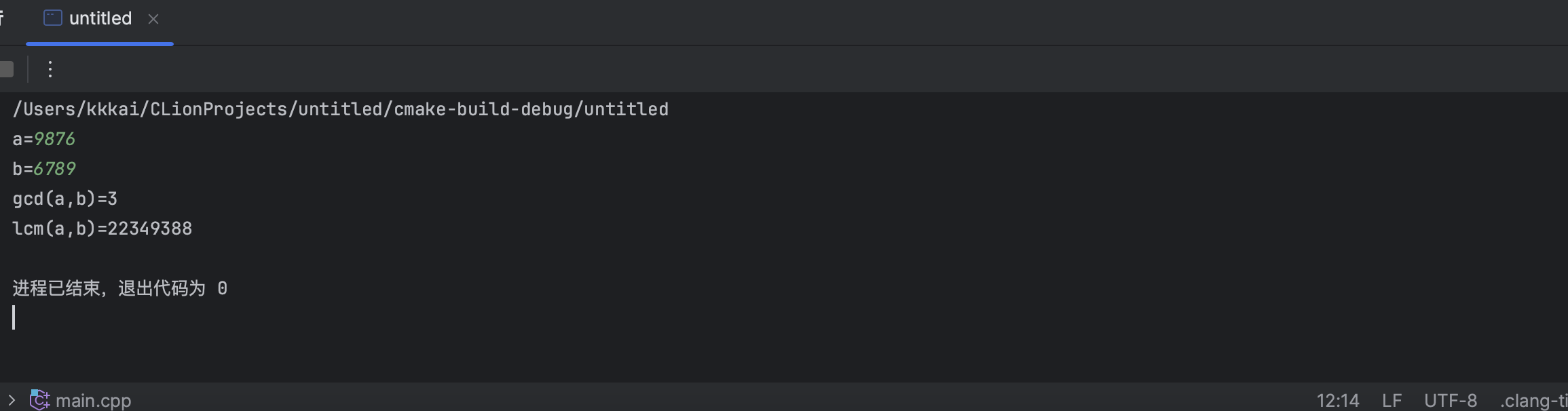
**源码部分：**

#include<iostream>  
using namespace std;  
int gcd(int x, int y)  
{  
 int r = y;  
 while (x % y != 0)  
 {  
 r = x % y;  
 x = y;  
 y = r;  
 }  
 return r;  
}  
int lcm(int x, int y)  
{  
 return (x \* y) / gcd(x, y);  
}  
int main() {  
 int a, b;  
 cout << "a=";  
 cin >> a;  
 cout << "b=";  
 cin >> b;  
 cout << "gcd(a,b)=" << gcd(a, b) << endl;  
 cout << "lcm(a,b)=" << lcm(a, b) << endl;

system("pause");  
 return 0;  
}

**说明部分:通过辗转相除法（原理是被除数与除数的gcd与除数与余数的gcd相同，从而不断迭代，直到不产生新的余数，则最后一个产生的余数就是gcd），而lcm的求法则利用公式：gcd(a,b)\*lcm(a,b)=ab求得。**

**运行示例：**

****

##### 编程练习3——实现算数基本定理

##### 源码部分：

#include<iostream>  
#include<cmath>  
using namespace std;  
#define maxn 1000000  
int noprime[maxn];  
int main()  
{  
 for (int i = 2; i <= sqrt(maxn); i++)  
 {  
 if (noprime[i] == 0)  
 {  
 for (int j = i+i; j < maxn; j += i)  
 {  
 noprime[j] = 1;  
 }  
 }  
 }  
 int a;  
 cin>>a;  
 cout<<a<<'=';  
 int now=1;

int temp=a;  
 for(int i=2;i<=a;i++){  
 if(noprime[i]) continue;  
 else{  
 int cnt=0;  
 while(temp%i==0){  
 cnt++;  
 temp/=i;  
 now\*=i;  
 }  
 if(cnt!=0&&now!=a)  
 cout<<i<<'^'<<cnt<<'\*';  
 if(cnt!=0&&now==a)  
 cout<<i<<'^'<<cnt ;  
 }  
 }  
 return 0;  
}

**说明部分:前半部分代码沿用第一题的Eratosthenes筛法，构造出素数集，为后面的判断做准备。在后面的遍历中，遍历1～N中的所有素数，判断是否能被当前的i整除，如果可以，说明i是一个素因子，计算i的次数，通过cnt储存，然后打出，如果不行 则说明i不是一个素因子 判断下一个素数。其中now代表当前已经找到的所有素因子乘积，用来控制’\*’号的打出（当now=a时，说明已经找全了，可以不用再打星号，同时可以结束）**

**运行示例：**

