# 第1次编程练习报告

姓名：胡博浩 学号：2212998 班级：信息安全

##### 编程练习1——使用Eratosthenes筛法打印1,000,000 内所有素数的个数

* **源码部分：**

#include<iostream>

#include<math.h>

using namespace std;

long long sum = 0;

void find(long long n)

{

bool\* num = new bool[n+1];

for (long long i = 2; i <n+1; i++)

num[i] = true;//初始化

for (long long i = 2; i <= sqrt(n); i++) {

if (num[i]) {

for (long long j = i\*i; j <= n; j+=i) {

num[j] = false;//将素数的倍数舍去

}

}

}

for (long long i = 2; i < n+1; i++) {

if (num[i]) {

sum++;

cout << i << ",";//输出

}

}

cout << endl;

}

void main() {

long long n;

cout << "Please input the range: 1-";

cin >> n;

find(n);

cout << "Total: " << sum << endl;

system("pause");

}

* **说明部分：**

使用一个布尔数组num来表示每个数是否为素数，然后从2开始，将其所有的倍数标记为合数。

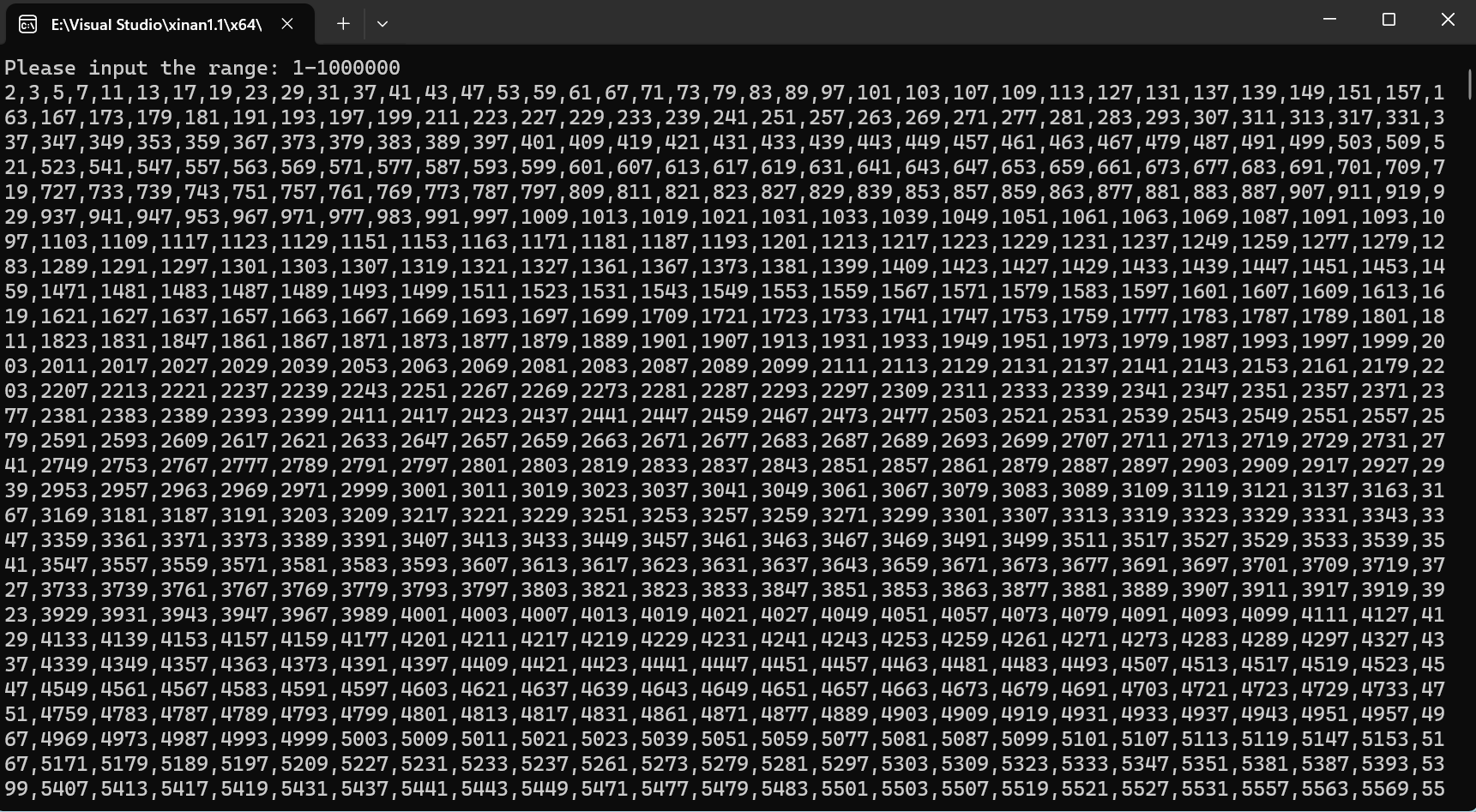
代码的主要逻辑：

创建一个布尔数组num，用于标记是否为素数。初始化所有数为素数（true）。

从2开始遍历到sqrt(n)，对于每个素数i，将其所有的倍数标记为合数（false）。

最后，遍历数组，输出所有素数，并计算素数的总数。

* **运行示例：**

* **其他：**

a. 对比筛法与普通算法的性能差异：

筛法通过标记合数的方式减少了计算量，因此对于大范围内的素数查找，筛法更为有效。

普通算法（如试除法）则是逐个判断每个数是否为素数，因此其时间复杂度较高，特别是在大范围内的素数查找时，性能会明显低于筛法。

普通算法为O（n\*sqrt（n）），Eratosthenes筛法为O（n\*log logn）

b. 递归调用该算法求更大范围素数进行优化：

递归调用可以在超出单次调用范围时分割成多个子问题，然后分别求解，最终将结果合并。

这种优化可以通过逐步增大每次调用的范围来求解更大范围内的素数，但需要注意内存使用情况和递归深度。

c. 求更大的素数该方法是否适用？会引入哪些新的问题？

当求解更大数量级的素数时，筛法的效率会受到影响，很多数字会被重复筛去，如6被2筛去，又会被3筛去等等，操作重复。

同时，由于它需要存储大量的标记数组，随着范围的增大，内存消耗会急剧增加。

因此，对于极大范围的素数求解，筛法可能不太适用，可能需要更高效的算法或者分布式计算等方法来解决。

##### 二、编程练习2——计算最大公因数和最小公倍数

* **源码部分：**

#include<iostream>

using namespace std;

int gcd(int a, int b)

{

a = abs(a);//取绝对值

b = abs(b);

int r = b;

while (r != 0) {

r = a % b;

a = b;

b = r;

}

return a;

}

void main() {

int a, b;

cout << "a=";

cin >> a;

cout << "b=";

cin >> b;

int x = gcd(a, b);

int y = a \* b / x;

cout << "gcd(a,b)=" << x << endl;

cout << "lcm(a,b)=" << y << endl;

system("pause");

}

* **说明部分：**

此代码利用辗转相除法求最大公因子，然后利用最小公倍数的计算式，调用前面已求得的最大公因子得到最小公倍数。

主要逻辑如下：

1.将两个整数a和b中较大的数赋值给a，较小的数赋值给b。

2.计算a除以b的余数，将余数赋值给r。

3.如果r等于0，则b就是最大公约数。

4.如果r不等于0，则将b赋值给a，将r赋值给b，然后重复步骤2和3，直到r等于0。

通过这个过程，最终得到的b就是a和b的最大公约数。

* **运行示例：**



##### 三、编程练习3——实现算术基本定理

* **源码部分：**

#include <iostream>

#include <map>

#include <math.h>

using namespace std;

void find(int n) {

cout << n << "=";

if (n == 1) {

cout << "1" << endl;

return;

}

map<int, int> factors;//使用map存储质因数及其指数

for (int i = 2; i <= sqrt(n); i++) {

while (n % i == 0) {

factors[i]++;//记录质因数及其指数

n /= i;

}

}

//如果n仍然大于1，则说明它本身就是质数

if (n > 1) {

factors[n]++;

}

auto it = factors.begin();

while (it != factors.end()) {

cout << it->first;//输出

cout << "^" << it->second;

it++;

if (it != factors.end()) {

cout << "\*";

}

}

cout << endl;

}

void main() {

int n;

cout << "Please input n(n>0): ";

cin >> n;

find(n);

system("pause");

}

* **说明部分：**

利用质因数分解的方法，即将一个正整数依次除以素数，直到商为1为止，得到的一系列素数就是该正整数的质因数。然后使用 map<int, int>数据结构来存储每个质因数及其对应的指数。

* **运行示例：**

