第一次编程练习报告

姓名：张洋　　学号：2111460　　班级：信安二班

一、编程练习 1——Eratosthenes 筛法打印 1 000 000 内所有素数及个数

```
#include<iostream>
using namespace std;

const int max_number = 1000000; //求素数的范围
int* p = new int[max_number + 1]; //存储该范围内所有数的数组

void eratosthenes(int* p) { //Eratosthenes筛选法
    for (int i = 2; i <= sqrt(max_number); i++)
        for (int j = 2 * i; j <= max_number; j += i) //将i的倍数都赋成0
            p[j] = 0;
}

int main() {
    for (int i = 0; i <= max_number; i++) //初始化数组
        p[i] = i;
    p[1] = 0;
    eratosthenes(p);
    int count = 0;
    for (int i = 2; i <= max_number; i++) { //遍历数组
        if (p[i] != 0) { //输出不为0的数(即素数)
            cout << p[i] << ",";
            count++;
        }
    }
    cout << endl;
    cout << "Total:" << count;
    return 0;
}
```

- **说明部分：**

要得到自然数 n 以内的全部素数，必须把不大于根号 n 的所有素数的倍数剔除，剩下的就是素数。

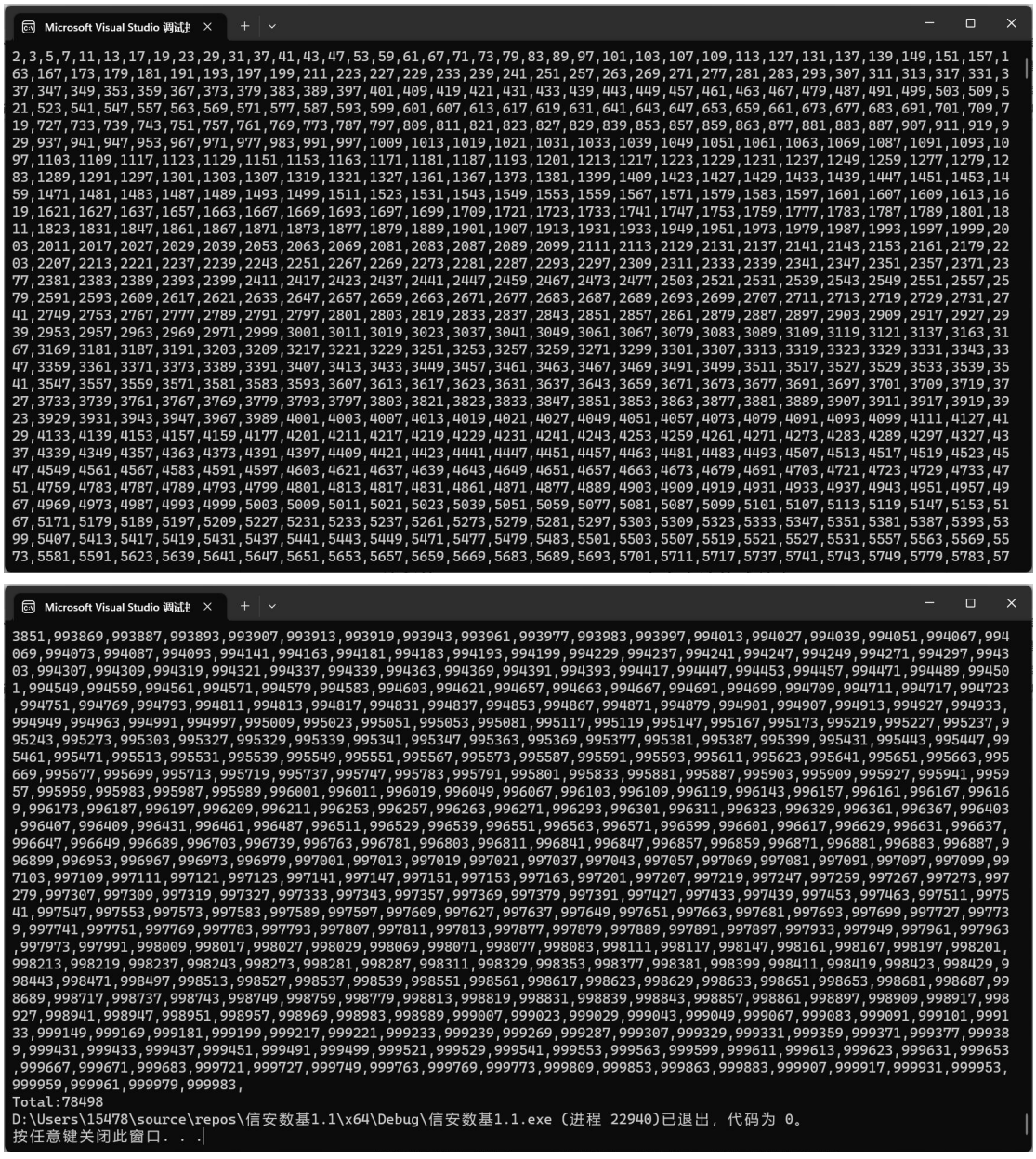
算法流程：

Step1:将[2,1000000]区间排成一列

Step2:标出表中第一个数，筛去其所有的倍数。

不断重复第二步，剩余的就是[2,1000000]中所有的素数。

- **运行示例：**



- **其他：**

a. 对比筛法与普通算法的性能差异

Eratosthenes筛法的时间复杂度为O(nlog_{log_n})，普通算法的时间复杂度是O(nlog_n)

b. 递归调用该算法求更大范围素数进行优化

```
#include <iostream>
using namespace std;
int fun(int n, int i) {
    //相当于for循环的那一环节
    if (i * i > n) {//在√n的范围内搜索
        return 1;
    }
    if (n % i == 0) {//取模为0 return
        return 0;
    }
    fun(n, i + 1);//递归
}
int isprim(int n) {//判断是不是素数，大于2且fun函数返回的值为1，则为素数
    return n >= 2 && fun(n, 2);
}

int main() {
    int count = 0;
    for (int i = 1; i <= 1000000; i++) {
        if (isprim(i)) {
            cout << i << ",";
            count++;
        }
    }
    cout << count;
}
```

c. 求更大的素数（如2^{512}数量级）该方法是否适用？会引入哪些新的问题？

不适用。（1）时间和存储空间：由于 Eratosthenes 筛法需要存储所有被筛出的数字，对于比较大的数字，很容易占用大量的计算机内存；另外，由于计算量比较大，Eratosthenes 筛法也需要很长的时间来

完成。（2）不可分割性：由于 Eratosthenes 筛法中的部分运算是不可分割的，因此在某些情况下会导致不能正确处理结果。

二、编程练习 2——编写程序计算最大公因数和最小公倍数

```
#include<iostream>
using namespace std;
void swap(int& a, int& b) {
    int tmp = a;
    a = b;
    b = tmp;
}
int gcd(int a, int b) { //利用辗转相除法求最大公因数
    if (a < b) //a为较小的那个数
        swap(a, b); //交换ab,让a为较大的数
    int r = a % b; //余数
    while (r != 0) { //当余数不为0时
        a = b;
        b = r;
        r = a % b;
    }
    return b;
}
int lcm(int a, int b) { //最小公倍数
    return a * b / gcd(a, b);
}
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;
}
```

- **说明部分：**

辗转相除法，是求最大公因数的一种方法。它的具体做法是：用较小数除较大数，再用出现的余数（第一余数）去除除数，再用出现的余数（第二余数）去除第一余数，如此反复，直到最后余数是 0 为

止。最后的除数就是这两个数的最大公因数。

求出最大公因数后，通过公式求出最小公倍数，最小公倍数=两数乘积/最大公因数。

➢ **运行示例：**

三、编程练习 3——编写程序实现算术基本定理

```
#include<iostream>
using namespace std;
void deposition(int n) { //求素因子分解
    for (int i = 2; i <= n; i++) { //遍历
        int count = 0; //count次幂
        while (n % i == 0) { //计算i对应的count
            count++; //次数+1
            n /= i; //除掉因子
        }
        if (count != 0) { //输出分解式
            cout<< i << "^" << count;
            if (n != 1)
                cout << "*";
            else break;
        }
    }
}
int main() {
    int n;
    cout << "Please input n(n>0):";
    cin >> n;
    cout << n << "=";
    deposition(n);
}
```

➢ **说明部分：**

求素因子分解的过程：从 2 开始遍历，如果 n 取模 i 为 0，说明 i

整除 n，再用 n 除以 i，直至不能整除，计算出 i 的个数。以此类推求出素因子的值以及个数。

- **运行示例：**