# 第1次编程练习报告

姓名：陆皓喆 学号：2211044 班级：信息安全

##### 编程练习1——Eratosthenes 筛法

* **源码部分：**

#include<iostream>

using namespace std;

bool isprime(int a) {

bool flag = true;

if (a < 2) {

return false;

}

else {

for (int i = 2; i \* i <= a; i++) {

if (a % i == 0) {

flag = false;

}

}

return flag;

}

}

int findprime(int n) {

int count = 0;//用于计录素数的个数

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

for (int i = 1; i <= n; i++) {

num[i] = true;//赋初始值，全部为true，表示都为素数

}

num[1] = false;//在Eratosthenes筛法中不会遍历到1这个数，因此先赋值为false

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

if (isprime(i)) {

for (int j = i + i; j <= n; j += i) {

num[j] = false;//完成了Eratosthenes筛法的主要步骤

}

}

}

for (int i = 1; i <= n; i++) {

if (num[i]) {

count++;//用于计数

cout << i << " ";

}

}

cout << endl;

return count;

}

int main() {

int n;

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

cin >> n;

int count = findprime(n);

cout <<"Total:"<< count << endl;

system("pause");

return 0;

}

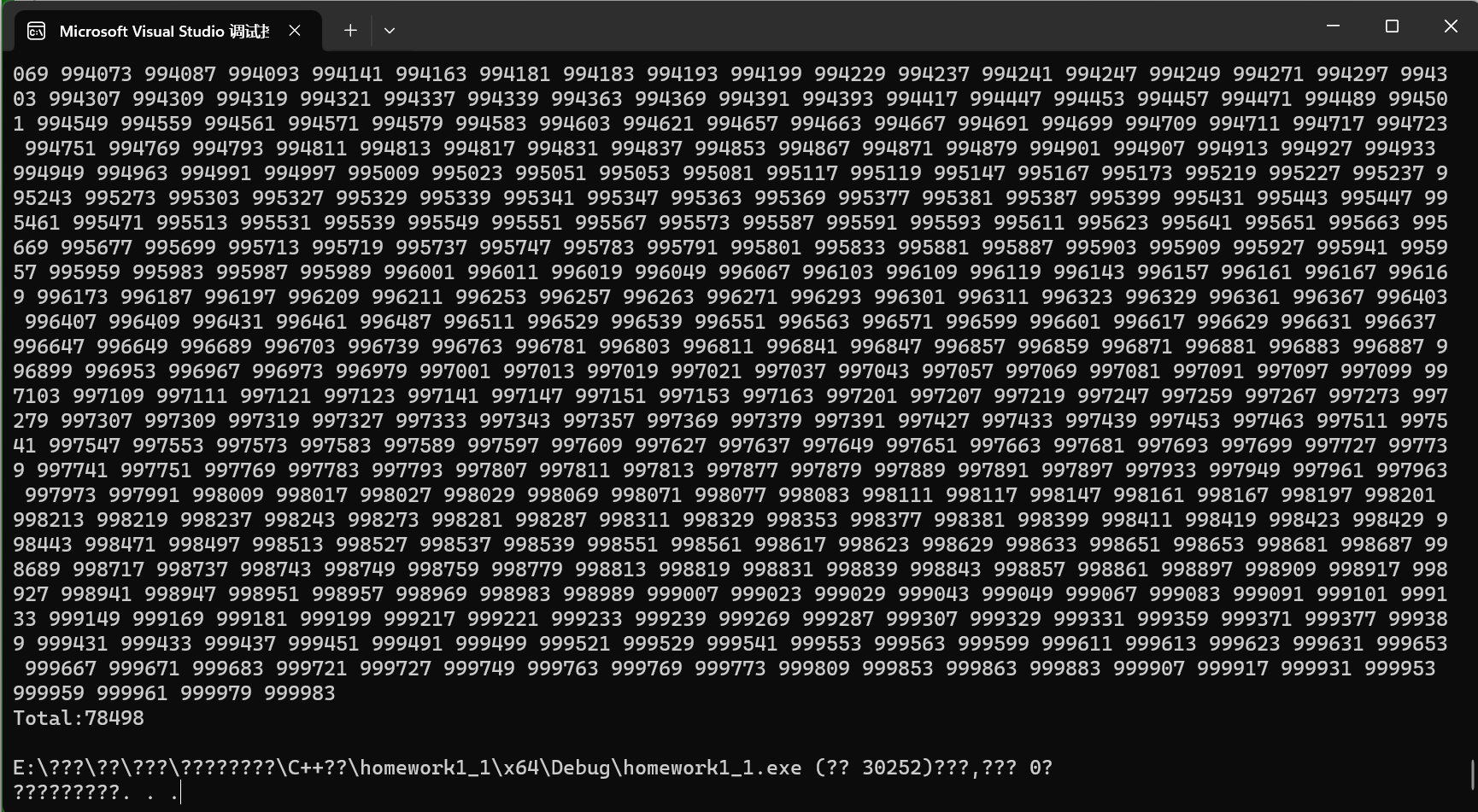
* **说明部分：**

本题要求打印1-1000000的素数，我们使用教材上出现过的Eratosthenes筛法来进行筛选。

首先，我们先编写一个isprime函数来实现判断一个数是不是素数。然后，编写一个findprime函数来实现Eratosthenes筛法。我们新开一个num数组，都赋值为true，表示当前均为素数。然后根据其因子的倍数关系来进行一一的删除（即将num数组的bool值赋值为false）。

然后对这些数进行一一遍历，若num的bool值仍然是true的话，就说明该数为素数，则将其输出。

* **运行示例：**



可以看出最后求出的总数为78948个。

* **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，而常规的筛选方法的时间复杂度为。因此可以看出，在大范围的素数筛选中，该筛选方法能够很好的在时间上对原算法进行简化。

**b.递归调用该算法求更大范围素数进行优化；**

由于该算法未涉及到递归的方法来进行计算，我们可以使用递归来实现程序的简化，我们可以通过对原数进行开方，从而简化过程。

**c.求更大的素数（如数量级）该方法是否使用？会引入哪些新的问题？**

该方法在更大的素数中将会不再适用。在前面可以看出，虽然该算法的时间复杂度相对来说较小，但是代价是占用了更多的空间。当素数过大时，会导致内存不足而无法进行计算。

##### 编程练习2——计算最大公因数与最小公倍数

* **源码部分：**

#include<iostream>

using namespace std;

int gcd(int a, int b) {

if (a < 0) {

a = -a;

}

if (b < 0) {

b = -b;

}

int temp = a;

while (temp != 0) {

temp = a % b;

a = b;

b = temp;

}

return a;

}

int lcm(int a, int b) {

return a \* b / gcd(a, b);

}

int main() {

int a;

int b;

cout << "a=";

cin >> a;

cout << endl;

cout << "b=";

cin >> b;

cout << endl;

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

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

system("pause");

return 0;

}

* **说明部分：**

首先，我们先将两数都取正，然后我们利用书中的辗转相除法来实现求解两个数的最大公因数。

while (temp != 0) {

temp = a % b;

a = b;

b = temp;

}

以上代码完成了辗转相除的求解过程。

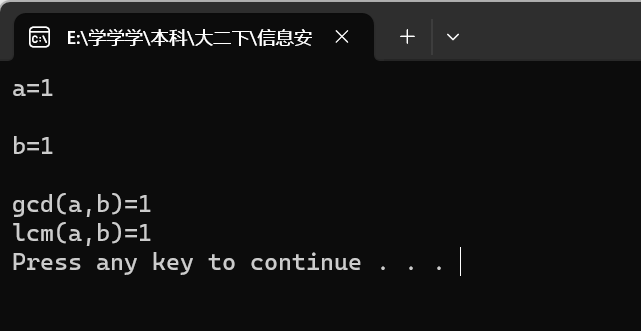
由书中公式



可以知道，在得到了gcd之后，实际上最小公倍数就是两数相乘除以两数的最大公因数。

所以我们就完成了两个数的最大公因数与最小公倍数的求解。

* **运行示例：**

##### 编程练习3——实现算术基本定理

* **源码部分：**

#include<iostream>

using namespace std;

bool isprime(int n)

{

if (n < 2)return false;

bool flag = true;

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

{

if (n % i == 0)

{

flag = false;

break;

}

}

return flag;

}

void suanshufenjie(int n) {

if (n == 1) {

cout << "1^1" << endl; return;

}

if (isprime(n)) {

cout << n << "^1"; cout << endl;

}

else {

int count = 0; bool temp = true;

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

if (n % i == 0) {

while (!(n % i)){

count++;

n = n / i;

}

if (!temp) {

cout << "\*";

}

temp = false;

}

cout << i<<"^" << count ;

if (n == 1) {

cout << endl; return;

}

count = 0;

if (isprime(n)) {

cout << "\*";

cout << n << "^1"; cout << endl; return;

}

}

}

}

int main() {

int n;

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

cin>>n;

cout << n<<"=";

suanshufenjie(n);

system("pause");

return 0;

}

* **说明部分：**

我们使用迭代的方式来找出最小的质因数，并且一步步的除下来，将原数分解开来。

最后完成分解后，程序进行遍历，输出最终的结果。

需要注意的是，如果输入的是数字1的话，需要手动的操作输出“1=1^1”。另外，在遇到一个因子时，需要使用while函数来进行连续除法判定，我在程序中使用了count来进行计数，并且每一次计算之后都会清零，重新进行计数。

如果除的时候发现该数为素数的时候，就可以直接执行输出。

* **运行示例：**

