

第 1 次编程练习报告

姓名：陆皓喆 学号：2211044 班级：信息安全

一、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; //用于记录素数的个数
    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：

```
Microsoft Visual Studio 调试
069 994073 994087 994093 994141 994163 994181 994183 994193 994199 994229 994237 994241 994247 994249 994271 994297 9943
03 994307 994309 994319 994321 994337 994339 994363 994369 994391 994393 994417 994447 994453 994457 994471 994489 99450
1 994549 994559 994561 994571 994579 994583 994603 994621 994657 994663 994667 994691 994699 994709 994711 994717 994723
994751 994769 994793 994811 994813 994817 994831 994837 994853 994867 994871 994879 994901 994907 994913 994927 994933
994949 994963 994991 994997 995009 995023 995051 995053 995081 995117 995119 995147 995167 995173 995219 995227 995237 9
95243 995273 995303 995327 995329 995339 995341 995347 995363 995369 995377 995381 995387 995399 995431 995443 995447 99
5461 995471 995513 995531 995539 995549 995551 995567 995573 995587 995591 995593 995611 995623 995641 995651 995663 995
669 995677 995699 995713 995719 995737 995747 995783 995791 995801 995833 995881 995887 995903 995909 995927 995941 9959
57 995959 995983 995987 995989 996001 996011 996019 996049 996067 996103 996109 996119 996143 996157 996161 996167 99616
9 996173 996187 996197 996209 996211 996253 996257 996263 996271 996293 996301 996311 996323 996329 996361 996367 996403
996407 996409 996431 996461 996487 996511 996529 996539 996551 996563 996571 996599 996601 996617 996629 996631 996637
996647 996649 996689 996703 996739 996763 996781 996803 996811 996841 996847 996857 996859 996871 996881 996883 996887 9
96899 996953 996967 996973 996979 997001 997013 997019 997021 997037 997043 997057 997069 997081 997091 997097 997099 99
7103 997109 997111 997121 997123 997141 997147 997151 997153 997163 997201 997207 997219 997247 997259 997267 997273 997
279 997307 997309 997319 997327 997333 997343 997357 997369 997379 997391 997427 997433 997439 997453 997463 997511 9975
41 997547 997553 997573 997583 997589 997597 997609 997627 997637 997649 997651 997663 997681 997693 997699 997727 99773
9 997741 997751 997769 997783 997793 997807 997811 997813 997877 997879 997889 997891 997897 997933 997949 997961 997963
997973 997991 998009 998017 998027 998029 998069 998071 998077 998083 998111 998117 998147 998161 998167 998197 998201
998213 998219 998237 998243 998273 998281 998287 998311 998329 998353 998377 998381 998399 998411 998419 998423 998429 9
98443 998471 998497 998513 998527 998537 998539 998551 998561 998617 998623 998629 998633 998651 998653 998681 998687 99
8689 998717 998737 998743 998749 998759 998779 998813 998819 998831 998839 998843 998857 998861 998897 998909 998917 998
927 998941 998947 998951 998957 998969 998983 998989 999007 999023 999029 999043 999049 999067 999083 999091 999101 9991
33 999149 999169 999181 999199 999217 999221 999233 999239 999269 999287 999307 999329 999331 999359 999371 999377 99938
9 999431 999433 999437 999451 999491 999499 999521 999529 999541 999553 999563 999599 999611 999613 999623 999631 999653
999667 999671 999683 999721 999727 999749 999763 999769 999773 999809 999853 999863 999883 999907 999917 999931 999953
999959 999961 999979 999983
Total:78498
E:\???\??\???\???\???\???\C++\??\homework1_1\x64\Debug\homework1_1.exe (?? 30252)???,??? 0?
?????????. . .
```

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 $O(n\log(\log(n)))$ ，而常规的筛选方法的时间复杂度为 $O(n\sqrt{n})$ 。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 2^{512} 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。

二、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

$$[a,b] = \frac{ab}{(a,b)}$$

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：



```
E:\学学\本科\大二下\信息安 x + v  
a=6789  
b=9876  
gcd(a,b)=3  
lcm(a,b)=22349388  
Press any key to continue . . . |  
  
E:\学学\本科\大二下\信息安 x + v  
a=1  
b=1  
gcd(a,b)=1  
lcm(a,b)=1  
Press any key to continue . . . |
```

三、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。

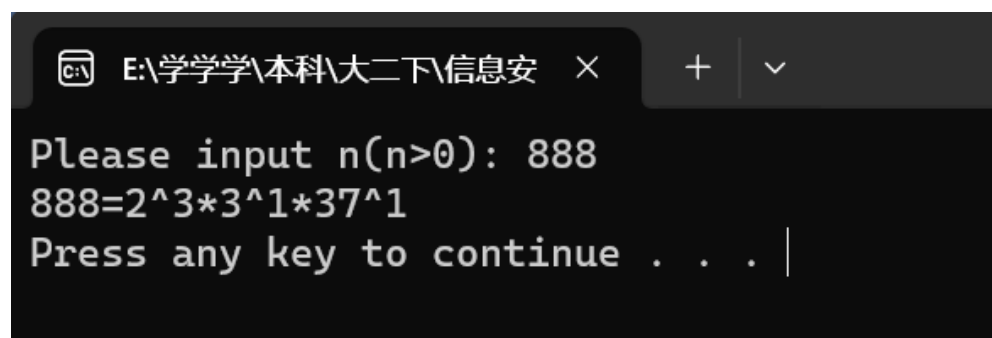
最后完成分解后，程序进行遍历，输出最终的结果。

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：



```
E:\学学学\本科\大二下\信息安
Please input n(n>0): 888
888=2^3*3^1*37^1
Press any key to continue . . . |
```