第一次编程作业报告

姓名：王雨萌 - 学号:2311819 - 班级：信安班

编程练习1 - Eratosthenes 筛质数

- **源码部分**

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

vector<int> prime;
bool is_prime[1000010];

void Eratosthenes(int n)
{
    is_prime[0] = is_prime[1] = false;
    for (int i = 2; i <= n; ++i)
        is_prime[i] = true;
    for (int i = 2; i <= n; ++i)
    {
        if (is_prime[i])
        {
            prime.push_back(i);
            if ((long long)i * i > n)
                continue;
            for (int j = i * i; j <= n; j += i)
                // 因为从 2 到 i - 1 的倍数我们之前筛过了，这里直接从 i
                // 的倍数开始，提高了运行速度
                is_prime[j] = false;
        }
    }
}
int main()
{
    cout << "Please input the range: ";
    int range;
    cin >> range;
    cout << "----------------------------------------" << endl;
    Eratosthenes(range);
    for (int i = 2; i <= range; i++)
    {
        if (is_prime[i] == true)
            cout << i << " ";
    }
}
```

- **说明部分**： 我们这样考虑，**对于任意一个大于 1 的正整数 n，那么它的 x 倍就是合数（x > 1）**。利用这个结论，如果我们从小到大考虑每个数，然后同时把当前这个数的所有（比自己大的）倍数记为合数，那么最后我们没有被标记的就是素数。

- **运行示例**：

```
Please input the range:1- 1000000
------------------------------------------
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157
163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331
 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 50
9 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 7
09 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911
919 929 937 941 947 953 967 971 977 983 991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1
093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223 1229 1231 1237 1249 1259 1277
1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399 1409 1423 1427 1429 1433 1439 1447 1451
 1453 1459 1471 1481 1483 1487 1489 1493 1499 1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 160
9 1613 1619 1621 1627 1637 1657 1663 1667 1669 1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 17
89 1801 1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951 1973 1979 1987 1993 1
997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083 2087 2089 2099 2111 2113 2129 2131 2137 2141 2143 2153
2161 2179 2203 2207 2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287 2293 2297 2309 2311 2333 2339 2341 2347 2351
 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417 2423 2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 254
9 2551 2557 2579 2591 2593 2609 2617 2621 2633 2647 2657 2659 2663 2671 2677 2683 2687 2689 2693 2699 2707 2711 2713 27
19 2729 2731 2741 2749 2753 2767 2777 2789 2791 2797 2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897 2903 2
909 2917 2927 2939 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023 3037 3041 3049 3061 3067 3079 3083 3089 3109 3119
3121 3137 3163 3167 3169 3181 3187 3191 3203 3209 3217 3221 3229 3251 3253 3257 3259 3271 3299 3301 3307 3313 3319 3323
 3329 3331 3343 3347 3359 3361 3371 3373 3389 3391 3407 3413 3433 3449 3457 3461 3463 3467 3469 3491 3499 3511 3517 352
7 3529 3533 3539 3541 3547 3557 3559 3571 3581 3583 3593 3607 3613 3617 3623 3631 3637 3643 3659 3671 3673 3677 3691 36
97 3701 3709 3719 3727 3733 3739 3761 3767 3769 3779 3793 3797 3803 3821 3823 3833 3847 3851 3853 3863 3877 3881 3889 3
907 3911 3917 3919 3923 3929 3931 3943 3947 3967 3989 4001 4003 4007 4013 4019 4021 4027 4049 4051 4057 4073 4079 4091
4093 4099 4111 4127 4129 4133 4139 4153 4157 4159 4177 4201 4211 4217 4219 4229 4231 4241 4243 4253 4259 4261 4271 4273
 4283 4289 4297 4327 4337 4339 4349 4357 4363 4373 4391 4397 4409 4421 4423 4441 4447 4451 4457 4463 4481 4483 4493 450
7 4513 4517 4519 4523 4547 4549 4561 4567 4583 4591 4597 4603 4621 4637 4639 4643 4649 4651 4657 4663 4673 4679 4691 47
03 4721 4723 4729 4733 4751 4759 4783 4787 4789 4793 4799 4801 4813 4817 4831 4861 4871 4877 4889 4903 4909 4919 4931 4
933 4937 4943 4951 4957 4967 4969 4973 4987 4993 4999 5003 5009 5011 5021 5023 5039 5051 5059 5077 5081 5087 5099 5101
5107 5113 5119 5147 5153 5167 5171 5179 5189 5197 5209 5227 5231 5233 5237 5261 5273 5279 5281 5297 5303 5309 5323 5333
 5347 5351 5381 5387 5393 5399 5407 5413 5417 5419 5431 5437 5441 5443 5449 5471 5477 5479 5483 5501 5503 5507 5519 552
1 5527 5531 5557 5563 5569 5573 5581 5591 5623 5639 5641 5647 5651 5653 5657 5659 5669 5683 5689 5693 5701 5711 5717 57
37 5741 5743 5749 5779 5783 5791 5801 5807 5813 5821 5827 5839 5843 5849 5851 5857 5861 5867 5869 5879 5881 5897 5903 5
923 5927 5939 5953 5981 5987 6007 6011 6029 6037 6043 6047 6053 6067 6073 6079 6089 6091 6101 6113 6121 6131 6133 6143
6151 6163 6173 6197 6199 6203 6211 6217 6221 6229 6247 6257 6263 6269 6271 6277 6287 6299 6301 6311 6317 6323 6329 6337
 6343 6353 6359 6361 6367 6373 6379 6389 6397 6421 6427 6449 6451 6469 6473 6481 6491 6521 6529 6547 6551 6553 6563 656
9 6571 6577 6581 6599 6607 6619 6637 6653 6659 6661 6673 6679 6689 6691 6701 6703 6709 6719 6733 6737 6761 6763 6779 67
```

```
1579 991603 991607 991619 991621 991633 991643 991651 991663 991693 991703 991717 991723 991733 991741 991751 991777 99
1811 991817 991867 991871 991873 991883 991889 991901 991909 991927 991931 991943 991951 991957 991961 991973 991979 99
1981 991987 991999 992011 992021 992023 992051 992087 992111 992113 992129 992141 992153 992179 992183 992219 992231 99
2249 992263 992267 992269 992281 992309 992317 992357 992359 992363 992371 992393 992417 992429 992437 992441 992449 99
2461 992513 992521 992539 992549 992561 992591 992603 992609 992623 992633 992659 992689 992701 992707 992723 992737 99
2777 992801 992809 992819 992843 992857 992861 992863 992867 992891 992903 992917 992923 992941 992947 992963 992983 99
3001 993011 993037 993049 993053 993079 993103 993107 993121 993137 993169 993197 993199 993203 993211 993217 993233 99
3241 993247 993253 993269 993283 993287 993319 993323 993341 993367 993397 993401 993407 993431 993437 993451 993467 99
3479 993481 993493 993527 993541 993557 993589 993611 993617 993647 993679 993683 993689 993703 993763 993779 993781 99
3793 993821 993823 993827 993841 993851 993869 993887 993893 993907 993913 993919 993943 993961 993977 993983 993997 99
4013 994027 994039 994051 994067 994069 994073 994087 994093 994141 994163 994181 994183 994193 994199 994229 994237 99
4241 994247 994249 994271 994297 994303 994307 994309 994319 994321 994337 994339 994363 994369 994391 994393 994417 99
4447 994453 994457 994471 994489 994501 994549 994559 994561 994571 994579 994583 994603 994621 994657 994663 994667 99
4691 994699 994709 994711 994717 994723 994751 994769 994793 994811 994813 994817 994831 994837 994853 994867 994871 99
4879 994901 994907 994913 994927 994933 994949 994963 994991 994997 995009 995023 995051 995053 995081 995117 995119 99
5147 995167 995173 995219 995227 995237 995243 995273 995303 995327 995329 995339 995341 995347 995363 995369 995377 99
5381 995387 995399 995431 995443 995447 995461 995471 995513 995531 995539 995549 995551 995567 995573 995587 995591 99
5593 995611 995623 995641 995651 995663 995669 995677 995699 995713 995719 995737 995747 995783 995791 995801 995833 99
5881 995887 995903 995909 995927 995941 995957 995959 995983 995987 995989 996001 996011 996019 996049 996067 996103 99
6109 996119 996143 996157 996161 996167 996169 996173 996187 996197 996209 996211 996253 996257 996263 996271 996293 99
6301 996311 996323 996329 996361 996367 996403 996407 996409 996431 996461 996487 996511 996529 996539 996551 996563 99
6571 996599 996601 996617 996629 996631 996637 996647 996649 996689 996703 996739 996763 996781 996803 996811 996841 99
6847 996857 996859 996871 996881 996883 996887 996899 996953 996967 996973 996979 997001 997013 997019 997021 997037 99
7043 997057 997069 997081 997091 997097 997099 997103 997109 997111 997121 997123 997141 997147 997151 997153 997163 99
7201 997207 997219 997247 997259 997267 997273 997279 997307 997309 997319 997327 997333 997343 997357 997369 997379 99
7391 997427 997433 997439 997453 997463 997511 997541 997547 997553 997573 997583 997589 997597 997609 997627 997637 99
7649 997651 997663 997681 997693 997699 997727 997739 997741 997751 997769 997783 997793 997807 997811 997813 997877 99
7879 997889 997891 997897 997933 997949 997961 997963 997973 997991 998009 998017 998027 998029 998069 998071 998077 99
8083 998111 998117 998147 998161 998167 998197 998201 998213 998219 998237 998243 998273 998281 998287 998311 998329 99
8353 998377 998381 998399 998411 998419 998423 998429 998443 998471 998497 998513 998527 998537 998539 998551 998561 99
8617 998623 998629 998633 998651 998653 998681 998687 998689 998717 998737 998743 998749 998759 998779 998813 998819 99
8831 998839 998843 998857 998861 998897 998909 998917 998927 998941 998947 998951 998957 998969 998983 998989 999007 99
9023 999029 999043 999049 999067 999083 999091 999101 999133 999149 999169 999181 999199 999217 999221 999233 999239 99
9269 999287 999307 999329 999331 999359 999371 999377 999389 999431 999433 999437 999451 999491 999499 999521 999529 99
9541 999553 999563 999599 999611 999613 999623 999631 999653 999667 999671 999683 999721 999727 999749 999763 999769 99
9773 999809 999853 999863 999883 999907 999917 999931 999953 999959 999961 999979 999983 Process exited with status 0
```

- **思考**：

!!! Note "对比筛法与普通算法的性能差别" 一般筛法用于单个元素的检验，就是简单地对于一个元素 n 从2至sqrt(n)进行检验能否整除，有一个就不是素数。而我们的Eratosthenes 筛质数，可以快速筛选素数：

```
|方法|时间复杂度|
|:---:|:---:|
|普通方法|$O(n^2)$|
|Eratosthenes筛法|$O(nlogn)$|
```

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;

// gcd是关键，一般来说是辗转相除法
long long Gcd(int a, int b)
{
    if (a < b) // 令a一定大于等于b
    {
        int temp = a;
        a = b;
        b = temp;
    }
    int r = a % b;
    while (r != 0)
    {
        a = b;
        b = r;
        r = a % b;
    }
    return b;
}

// 利用[a,b] = ab/(a,b)
long long Lcm(int &a, int &b)
{
    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;
    return 0;
}
```

- **说明部分**： 求最大公因数和最小公倍数，我们知道公式1$[a,b] = \frac{ab}{(a,b)}$ 所以求解最大公因数是关键，而在计算机程序上比较容易实现的算法 - **辗转相除法**（$O(n)$）可以快速得到两个数的最大公因数，求解最大公因数后带入公式1。

- **运行示例**：
```
argv[0] = '/Users/wangyumeng/000 - 学习/050 - 竞赛/算法code文件/.vscode/Gcd&Lcm'
a = 9876
b = 6789
gcd(a,b) = 3
lcm(a,b) = 22349388
Process exited with status 0
```

编程练习3 - 算术基本定理

- **源码部分**：

```
vector<int> prime;
bool is_prime[1000010];

void Eratosthenes(int n)
{
    //同练习1，具体见可执行代码文件，不重复，为了排版
}

bool is_in[1000010];
int cishu[1000010] = {0};

void depart(int num, int a)
{
    Eratosthenes(a);
    for (int i = 2; i <= a; i++)
    {
        if (is_prime[i])
        {
            if (num % i == 0)
            {
                is_in[i] = true;
                while (num % i == 0)
                {
                    num = num / i;
                    cishu[i]++;
                }
            }
        }
    }
}

int main()
{
    cout << "Please input n(n>0): ";
    int n, half = sqrt(n), cut = 0;cin >> n;
    depart(n, half);cout << n << "=";
    for (int i = 2; i <= half; i++)
```

```
    {
        if (is_in[i])
        {
            if (cut++)
                cout << "*";
            cout << i << "^" << cishu[i];
        }
    }
}
```

- **说明部分**： 本题目稍微有点复杂，我的思路是这样的：为了得到一个数字n的质因数分解，我们先在$(1,\sqrt{n})$范围内**用Eratosthenes筛法寻找**数字n所有可能的质因数，然后判断哪些是数字n的质因数，如果是，计算出质因数的次数。

- **运行示例**：

```
argv[0] = '/Users/wangyumeng/000 - 学习/050 - 竞赛/算法code文件/.vscode/算术基本定理'
Please input n(n>0): 888
888=2^3*3^1*37^1Process exited with status 0
```