# 第4次编程练习报告

姓名：齐明杰 学号：2113997 班级：信安二班

##### 编程练习4——求解最小原根并构造指数表

* **源码部分：**
* #include <bits/stdc++.h>
* #define MAXN 100000
* **using** **namespace** std;
* **int** ind\_table[MAXN], ind\_val[MAXN];
* **bool** vis[MAXN];
* **int** elm[MAXN], lens;
* **void** Eratosthenes(**int** n){
* memset(vis, **false**, 4\*(n+1));
* lens = 0;
* **for**(**int** i=2;i<=n;i++){
* **if**(!vis[i]){
* **if**(n%i==0){
* elm[lens] = i;
* **while**(n%i==0) n/=i;
* lens++;
* **if**(n==1) **break**;
* }
* **for**(**int** j = 2\*i;j <= n;j += i) vis[j] = **true**;
* }
* }
* }
* **int** Quick\_mod(**int** a,**int** b,**int** c){
* a%=c;
* **long** **long** ans=1, base=a;
* **while**(b>0){
* **if**(b&1) ans=(ans\*base)%c;
* base=(base\*base)%c;
* b>>=1;
* }
* **return** ans;
* }
* **int** Euler(**int** n){
* Eratosthenes(n);
* **int** ans = n;
* **for**(**int** i=0;i<lens;i++) ans = (ans / elm[i]) \* (elm[i] - 1);
* **return** ans;
* }
* **int** get\_g(**int** n){
* **int** phi = Euler(n);
* Eratosthenes(phi);
* **for**(**int** i=2;i<=phi;i++){
* **bool** flag = **true**;
* **for**(**int** j=0;j<lens;j++){
* **if**(Quick\_mod(i, phi / elm[j], n) == 1){
* flag = **false**;
* **break**;
* }
* }
* **if**(flag) **return** i;
* }
* **return** -1;
* }
* **void** get\_table(**int** g, **int** n){
* memset(ind\_table, -1, 4\*(n+1));
* ind\_val[0] = 1;
* ind\_table[1] = 0;
* **for**(**int** i=1;i<Euler(n);i++){
* ind\_val[i] = (ind\_val[i-1] \* g) % n;
* ind\_table[ind\_val[i]] = i;
* }
* }
* **void** print\_table(**int** n){
* **int** y = n / 10;
* printf("     ");
* **for**(**int** i=0;i<10;i++) printf("%5d", i);
* printf("\n");
* **for**(**int** i=0;i<=y;i++){
* printf("%5d", i);
* **for**(**int** j=0;j<10;j++){
* **int** val = ind\_table[i\*10+j];
* **if**(val==-1 || 10\*i+j>=n) printf("%5c", '-');
* **else** printf("%5d", val);
* }
* printf("\n");
* }
* }
* **int** main(){
* printf("Please input n(n>0): ");
* **int** n, g;
* scanf("%d", &n);
* g = get\_g(n);
* printf("The min primitive root if %d: g=%d\n", n, g);
* printf("The ind\_table if %d based on g=%d is:\n", n, g);
* get\_table(g, n);
* print\_table(n);
* system("pause");
* **return** 0;
* }
* **说明部分：**

**代码结构与功能说明**：

1、Eratosthenes 函数：实现埃拉托斯特尼筛选法，找出正整数 n 的素数因子。通过遍历小于等于 n 的整数 i（i 从 2 开始），如果当前整数 i 未被标记为合数，则表示 i 是一个素数。接着，将所有 i 的倍数标记为合数，并检查 n 是否能被 i 整除。如果可以，则将 i 添加到素数因子列表中，同时更新 n 的值。遍历完成后，素数因子列表即为所求。

2、Quick\_mod 函数：实现快速幂取模算法，用于计算模幂。通过不断地平方计算底数 a，将指数 b 二进制拆分，对每一个二进制位为 1 的部分，将对应的平方结果相乘，并对结果取模。最终得到的结果即为所求模幂。

3、Euler 函数：计算给定正整数 n 的欧拉函数值。首先调用 Eratosthenes 函数找出 n 的所有素数因子，然后根据欧拉函数的定义，依次将 n 除以每个素数因子并乘以（素数因子 - 1），最后得到的结果即为欧拉函数值。

4、get\_g 函数：寻找正整数 n 的最小原根。首先计算欧拉函数值，然后遍历 2 到欧拉函数值之间的所有整数 i。对于每个 i，检查是否满足原根的条件：对于所有素数因子 p，i^((phi)/p) % n ≠ 1。如果满足这个条件，则返回当前整数 i 作为最小原根。其中，phi 为 n 的欧拉函数值。

5、get\_table 函数：基于最小原根 g 和正整数 n 构造指数表。通过遍历 0 到欧拉函数值之间的整数 i，计算 (g^i) % n 的结果，并将其作为指数表的键，i 作为对应的值。这样，指数表可以表示为：ind\_table[a] = i，表示 a 的离散对数是 i。

6、print\_table 函数：以易读的格式打印指数表。以表格形式输出指数表，其中行表示十位数，列表示个位数。对于指数表中不存在的值，输出一个 '-' 符号。

7、main 函数：程序的入口点。首先获取用户输入的正整数 n，然后计算其最小原根 g，接着调用 get\_table 函数构造基于最小原根 g 的指数表。最后，使用 print\_table 函数以易读的格式打印指数表，并暂停程序以便用户查看输出结果。

* **运行示例：**



