# 第4次编程练习报告

姓名：孙蕗 学号：2112060 班级：信安一班

##### **编程练习——求解最小原根，基于最小原根构造指数表**

* **源码部分：**

#include <iostream>

#include<iomanip>

#include<vector>

#include<cmath>

using namespace std;

int Euler(int n) //直接求欧拉函数的值

{

int rea = n;

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

if (n % i == 0)//第一次找到的必为素因子

{

rea = rea - rea / i;

while (n % i == 0) {

n = n / i;//把该素因子全部约掉

}

}

if (n > 1)

rea = rea - rea / n;

return rea;

}

bool Isprime(int n)//素数判断

{

if (n <= 1)//<=1的都不是素数

{

return false;

}

else if (n == 2)//2是素数

{

return true;

}

else

{

for (int i = 2; i < n; i++)//n%i==0，不是素数

{

if (n % i == 0)

{

return false;

}

}

return true;

}

}

int sqrt\_multi(int a, int n, int m)//平方乘算法求解a^n(mod m)

{

//变成二进制

int bin[100];

int t = n;

int count = 0;

for (int i = 0; t > 0; i++)//计算

{

bin[i] = t % 2;

t /= 2;

count++;

}

for (int i = 0; i < count / 2; i++)//逆序

{

int temp;

temp = bin[i];

bin[i] = bin[count - i - 1];

bin[count - i - 1] = temp;

}

long long int b = 1;

long long int c;

for (int i = 0; i < count; i++)

{

if (bin[i] == 1)

{

c = b \* b \* a - (b \* b \* a) / m \* m;

b = c;

}

else

{

c = b \* b - (b \* b) / m \* m;

b = c;

}

}

return b;

}

int main()

{

int n;

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

cin >> n;

int f = Euler(n);//欧拉函数φ(n)

vector<int>factor;//用来存储因子

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

{

if (f % i == 0)//能整除，为φ(n)的一个因子

{

if (Isprime(i))//如果i是素因子

{

factor.push\_back(f/i);//把f/i加进vector

}

}

}

int root;

/\* for (int i = 0; i < factor.size(); i++)

{

cout << factor[i] << " ";

}

cout << endl;\*/

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

{

int flag = 1;

for (int j = 0; j < factor.size(); j++)

{

if (sqrt\_multi(i, factor[j], n) == 1)//平方乘计算i^factor[j]%n，如果等于1的话，i不是n的原根

{

flag = 0;//不是原根

}

}

if (flag == 1)

{

root = i;

break;

}

}

cout <<"The min primitive root of "<<n<<": g=" << root << endl;

cout << "The ind\_table of " << n << " based on g=" << root << " is: " << endl;

int\* array = new int[n];//建立一个数组，用来存储root^i%n

for (int i = 0; i < n; i++)

{

array[i] = sqrt\_multi(root, i, n);

}

array[0] = 1;

int matrix[300][11];

for (int i = 0; i <= n / 10 + 1; i++)

{

for (int j = 0; j <= 10; j++)

{

matrix[i][j] == -1;//初始化

}

}

for (int i = 1; i <= 10; i++)

{

matrix[0][i] = i - 1;//第0行标记列的信息

}

for (int i = 1; i <= n / 10 + 1; i++)//第0列标记行的信息

{

matrix[i][0] = i - 1;

}

for (int i = 1; i <= n / 10 + 1; i++)

{

for (int j = 1; j <= 10; j++)

{

int temp = matrix[i][0] \* 10 + matrix[0][j];//matrix[i][j]，在array[n]与matrix[i][j]相等的元素，matrix[i][j]表示的含义是，root^(matrix[i][j])%n=matrix[0][j]+matrix[i][0]\*10

int k;

for (k = 0; k < n; k++)

{

if (array[k] == temp)//matrix[i][j]，在array[n]与matrix[i][j]相等的元素，matrix[i][j]表示的含义是，root^(matrix[i][j])%n=matrix[0][j]+matrix[i][0]\*10

{

matrix[i][j] = k;

break;

}

}

if (k == n)

{

matrix[i][j] = -1;//没找到

}

}

}

matrix[0][0] = -1;

for (int i = 0; i <= n / 10 + 1; i++)

{

for (int j = 0; j <= 10; j++)

{

if (matrix[i][j] != -1)

{

cout << std::right << setw(5) << matrix[i][j] << " ";//右对齐输出

}

else

{

cout << std::right << setw(5) <<"-" << " ";

}

}

cout << endl;

}

return 0;

}

* **说明部分：**

根据定理4.2.12，设m是大于2的整数，的所有不同的素因子是q1,q2,…qs，则与m互素的正整数g是m的一个原根的充要条件是



先求出n的欧拉函数，然后找出的素因子q1,q2,…qs。计算。找出满足条件的最小正整数，即为最小原根，最后基于小原根构造指数表。

* **运行示例：**

