第 4 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——求最小原根并基于最小原根构造指数表

➢ 源码部分：

```
#include <bits/stdc++.h>
using namespace std;
int m;
void binary(vector<int>& a, int temp)
{//实现了十进制转二进制
    int q = temp;
    int r;
    while (q != 0)
    {
        r = q % 2;
        a.push_back(r);
        q = q / 2;
    }
}
int quick_pow(int a, int n, int m) {//是第二章中的练习，实现了平方乘-快速幂算法
    int result = 1;
    while (n > 0) {
        if (n & 1) {
            result *= a;
            result %= m;
        }
        a *= a;
        a %= m;
        n >>= 1;//利用位运算来实现算法
    }
    return result;
}

bool isprime(int a) {//判断一个数是不是素数
    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;
    }
}

void depart_the_num(int n, vector<int>& res)
{//这部分是用递归的方法求解出一个数的质因数分解的结果
    int l = 0;
    for (int i = 2; i <= n; i++)
    {
        if (isprime(i))
        {
            if (n % i == 0)
            {
                l++;
                res[i]++;
                int t = n / i;
                m = t;
                depart_the_num(t, res);
                break;
            }
        }
    }
    if (!l)
        res[m]++;
}
int Euler_totient_function(int n)
{//这部分我们实现了欧拉函数的计算
    vector<int> vec(10000, 0);
    depart_the_num(n, vec);
    int euler = n;
    for (int i = 2; i < vec.size(); i++)
    {
        if (vec[i] != 0)
            euler *= (1 - 1 / double(i));
    }
    return euler;
}
```

```
int main()
{
	cout << "Please input n(n>0): ";
	int n;
	cin >> n;
	int euler = Euler_totient_function(n);
	int g = 0;
	vector<int> exp;
	vector<int> res(10000);
	depart_the_num(euler, res);
	for (int i = 2; i < res.size(); i++)
	{
		if (res[i] != 0)
			exp.push_back(euler / i);
	}
	for (int i = 2; i < n; i++)
	{
		bool flag = 1;
		for (int j = 0; j < exp.size(); j++)
		{
			int t = quick_pow(i, exp[j], n);
			if (t == 1)
			{
				flag = 0;
				break;
			}
		}
		if (flag)
		{
			g = i;
			break;
		}
	}
	cout << "The min primitive root of " << n << ": g=" << g << endl;
	cout << "The ind_table of " << n << " based on g=" << g << " is:" << endl;
	cout << setw(6) << " ";
	for (int i = 0; i < 10; i++)
		cout << setw(6) << i;
	cout << endl;
	int row = n / 10;
	int** table = new int* [row + 1];
	for (int i = 0; i < row + 1; i++)
	{
		table[i] = new int[11];
```

```
            table[i][0] = i;
            for (int j = 1; j < 11; j++)
                table[i][j] = -1;
        }
        for (int i = 0; i <= euler - 1; i++)
        {
            int t = quick_pow(g, i, n);
            int row_num = t / 10;
            int col_num = t % 10;
            table[row_num][col_num + 1] = i;
        }
        for (int i = 0; i < row + 1; i++)
        {
            for (int j = 0; j < 11; j++)
            {
                if (table[i][j] != -1)
                    cout << setw(6) << table[i][j];
                else
                    cout << setw(6) << "-";
            }
            cout << endl;
        }
        system("pause");
        return 0;
}
```

- **说明部分：**

该部分，我们使用多个之前写过的函数来共同实现此次的求最小原根并且输出完整的指数表。

Binary 这一部分实现了从十进制转化为二进制，quick_pow 部分实现了平方乘算法，is_prime 这一部分是用于判断一个数是否是素数，depart_the_num 部分是实现了将一个数分解为素数，并将这些素数的指数存储在 vector<int> res 中，这里使用了递归的方法，对每个质因子进行分解。Euler_totient_function 部分计算了欧拉函数的值。

Main 函数中，首先我们输入一个正整数 n，然后计算他的欧拉函数值，并将其欧拉函数值分解成若干个不同的因子，存储在 vector<int>

exp 中。这里将指数表中的行数设置为 n/10，每行有 10 个元素，因为指数表中的元素是从 0 到 n-1 的所有非重复元素，而 0 可以作为第一行的元素，所以总行数为 n/10+1。然后依次枚举 2 到 n-1 的整数 i，对于每个 i，判断是否是 n 的一个原根，即对于 n 的欧拉函数中的每个因子 d。如果是原根，则将其存储在变量 g 中，并退出循环。最后使用动态数组 int **table 存储指数表，并输出。动态数组的行数为 n/10+1，列数为 11，其中第一列存储行数，第二列到第十一列存储该行的元素。如果某个元素不存在，我们就输出“-”。

- **运行示例：**