第 4 次编程练习报告

一、编程练习 1——求解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 T)
{
    int q = T;
    int r;
    while (q != 0)
    {
        r = q % 2;
        a.push_back(r);
        q = q / 2;
    }
}
int pfc(int a, int n, int x)
{
    vector<int> b;
    binary(b, n);
    int c = 1;
    for (int i = b.size() - 1; i >= 0; i--)
    {
        c = c * c % x;
        if (b[i])
        {
            c = c * a % x;
        }
    }
    return c;
}
bool sushu(int n)
{
    if (n == 1)
        return false;
```

```
    else
    {
        for (int i = 2; i < n; i++)
        {
            if (n % i == 0)
                return false;
        }
        return true;
    }
}
void fj(int n, vector<int>& res)
{
    int l = 0;
    for (int i = 2; i <= n; i++)
    {
        if (sushu(i))
        {
            if (n % i == 0)
            {
                l++;
                res[i]++;
                int t = n / i;
                m = t;
                fj(t, res);
                break;
            }
        }
    }
    if (!l)
        res[m]++;
}
int oula(int n)
{
    vector<int> vec(10000, 0);
    fj(n, vec);
    int fai = n;
    for (int i = 2; i < vec.size(); i++)
    {
        if (vec[i] != 0)
            fai *= (1 - 1 / double(i));
    }
    return fai;
}
int main()
```

```
{
    cout << "Please inputn(n>0): ";
    int n;
    cin >> n;
    int fai = oula(n);
    int g = 0;
    vector<int> exp;
    vector<int> res(10000);
    fj(fai, res);
    for (int i = 2; i < res.size(); i++)
    {
        if (res[i] != 0)
            exp.push_back(fai / i);
    }
    for (int i = 2; i < n; i++)
    {
        bool flag = 1;
        for (int j = 0; j < exp.size(); j++)
        {
            int t = pfc(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 <= fai - 1; i++)
    {
        int t = pfc(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：

这段代码实现了求一个正整数的最小原根以及它的指数表。下面是代码的详细解释：

1.binary 函数实现将一个十进制数转化为二进制数，存储在 vector<int> a 中。这里的二进制数是逆序存储的，即最低位存储在 a[0]。

2.pfc 函数实现了快速幂算法，。

3.sushu 函数判断一个数是否是素数，如果是素数，返回 true，否则返回 false。

4.fj 函数实现了分解质因数的功能，将一个数分解成若干个素数的积的形式，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。

5.oula 函数计算了欧拉函数值，即小于 n 的正整数中与 n 互质的数的个数。这里利用了欧拉函数的性质。

6.在 main 函数中，首先输入一个正整数 n，然后计算 fai(n)，并将 fai(n)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 fai(n)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，并输出到屏幕上。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，则用-表示。

➢ 运行示例：

```
Please inputn(n>0): 41
The min primitive root of 41: g=6
The ind_table of 41 based on g=6 is:
          0     1     2     3     4     5     6     7     8     9
     0    -     0    26    15    12    22     1    39    38    30
     1    8     3    27    31    25    37    24    33    16     9
     2   34    14    29    36    13     4    17     5    11     7
     3   23    28    10    18    19    21     2    32    35     6
     4   20     -     -     -     -     -     -     -     -     -
请按任意键继续. . .
```