

源代码如下：

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

class Solution {

private:

    //辅助函数

    void arrUpSideDown(vector<int>& arr, const int left, const int right) {

        for (int i = left; i <= (right - left) / 2 + left; i++)

            swap(arr[i], arr[right - i + left]);

    }

    string numVectorToString(const vector<int>arr) {

        string res = "";

        char ch\_buf[10];

        int s = arr.size();

        for (int i = 0; i < s; i++) {

            res += \_itoa(arr[i], ch\_buf, 10);

            res += ' ';

        }

        return res;

    }

    string numVectorToMatrix(const vector<vector<int>>arr) {

        int s1 = arr.size(),//number of rows

            s2 = arr[0].size();//number of cols

        string res;

        char ch\_buf[10];

        for (int i = 0; i < s1; i++) {

            for (int j = 0; j < s2; j++) {

                res += '\t';

                res += \_itoa(arr[i][j], ch\_buf, 10);

            }

            res += '\n';

        }

        return res;

    }

    int factorial(const int n) {

        if (n < 1)

            return 0;

        if (n == 1)

            return 1;

        return factorial(n - 1) \* n;

    }

    void arrRefill(vector<vector<int>>& arr, const int n) {

        vector<int>temp1(n - 1);

        vector<int>temp2(n - 1);

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

            temp1[i] = n + i;

        for (int j = 1; j < n; j++)

            temp2[j - 1] = arr[n - 1][j];

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

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

                if (arr[i][j] >= n) {

                    arr[i][j] = temp1[i];

                    arr[arr[i][j] - 1][j] = i + 1;

                }

                else

                    arr[i + (n - 1)][j] = arr[i][j] + (n - 1);

        for (int i = 0; i < n - 1; i++) {

            for (int j = n; j < 2 \* (n - 1); j++) {

                arr[i][j] = temp1[(1 + i + (j - n)) % (n - 1)];

                arr[arr[i][j] - 1][j] = i + 1;

            }

        }

    }

    void arrRestruct(vector<vector<int>>& arr, const int n) {

        arr.resize(n);

        int s = arr.size();

        for (int i = 0; i < s; i++)

            arr[i].resize(n);

    }

    void arrReturnBack(vector<vector<int>>& arr, const int n) {

        arr.pop\_back();

        int s1 = arr.size(),

            s2 = arr[0].size();

        for (int i = 0; i < s1; i++)

            for (int j = 0; j < s2; j++)

                if (arr[i][j] == n)

                    arr[i][j] = 0;

    }

    void matrixInitial(vector<vector<int>>& arr) {//initialize first line using 1~n

        int s = arr.size();

        for (int i = 0; i < s; i++)

            arr[i][0] = i + 1;

    }

    void matrixCopyPaste(vector<vector<int>>& arr, const int start\_x, const int start\_y, const int n) {

        for (int i = 0; i < n; i++)

            for (int j = 0; j < n; j++) {

                arr[start\_y + n + i][start\_x + n + j] = arr[start\_y + i][start\_x + j];

                arr[start\_y + i][start\_x + n + j] = arr[start\_y + n + i][start\_x + j];

            }

    }

    //循环赛日程表

    void tableRecursion(vector<vector<int>>& arr, const int start\_x, const int start\_y, const int n) {//n refers to the length of the matrix

        if (n < 2)

            return;

        if (n == 2) {

            arr[start\_y][start\_x + 1] = arr[start\_y + 1][start\_x];

            arr[start\_y + 1][start\_x + 1] = arr[start\_y][start\_x];

            return;

        }

        tableRecursion(arr, start\_x, start\_y, n / 2);

        tableRecursion(arr, start\_x, start\_y + n / 2, n / 2);

        matrixCopyPaste(arr, start\_x, start\_y, n / 2);

    }

    void tableRecursionGeneral(vector<vector<int>>& arr, const int start\_x, const int start\_y, const int n) {

        if (n < 2)

            return;

        if (n == 2) {

            arr[start\_y][start\_x + 1] = arr[start\_y + 1][start\_x];

            arr[start\_y + 1][start\_x + 1] = arr[start\_y][start\_x];

            return;

        }

        if ((n / 2) % 2) {

            tableRecursionGeneral(arr, start\_x, start\_y, (n + 1) / 2);

            matrixCopyPaste(arr, start\_x, start\_y, (n + 1) / 2);

            //arrReturnBack(arr, n + 1);

            arrRefill(arr, n + 1);

            //start here...

            return;

        }

        tableRecursionGeneral(arr, start\_x, start\_y, n / 2);

        tableRecursionGeneral(arr, start\_x, start\_y + n / 2, n / 2);

        matrixCopyPaste(arr, start\_x, start\_y, n / 2);

    }

public:

    //字典序

    int dictionaryIndex(vector<int>arr, const int n) {

        int ans = 0;

        for (int i = 0; i < n; i++) {

            int pos = 0;

            for (int j = i + 1; j < n; j++)

                if (arr[j] < arr[i])

                    pos++;

            ans += pos \* factorial(n - i - 1);

        }

        return ans;

    }

    string dictionaryNextOrder(vector<int>arr, const int n) {

        int s = arr.size();

        int i = s - 2;

        bool flag = false;

        for (; i >= 0; i--) {

            if (arr[i] < arr[i + 1]) {

                flag = true;

                break;

            }

        }

        if (flag) {

            int j = s - 1;

            for (; arr[j] < arr[i] && j >= 0; j--)

                ;

            swap(arr[i], arr[j]);

            arrUpSideDown(arr, i + 1, s - 1);

            return numVectorToString(arr);

        }

        return "error\_overflow";

    }

    //汉诺塔

    void hanoiCore(const int n, const char src, const char tmp, const char dst) {

        if (n == 1) {

            cout << n << ' ' << src << ' ' << dst << endl;

            return;

        }

        hanoiCore(n - 1, src, dst, tmp);

        cout << n << ' ' << src << ' ' << dst << endl;

        hanoiCore(n - 1, tmp, src, dst);

    }

    string tenisTimeTable(vector<vector<int>>& arr, const int n) {

        matrixInitial(arr);

        tableRecursion(arr, 0, 0, n);

        return numVectorToMatrix(arr);

    }

    string tenisTimeTableGeneral(vector<vector<int>>& arr, const int n) {

        if (n % 2) {

            arrRestruct(arr, n + 1);

            matrixInitial(arr);

            tableRecursionGeneral(arr, 0, 0, n + 1);

            arrReturnBack(arr, n + 1);

            return numVectorToMatrix(arr);

        }

        matrixInitial(arr);

        tableRecursionGeneral(arr, 0, 0, n);

        return numVectorToMatrix(arr);

    }

};

class testData {

private:

    //检验是否为2的k次幂

    bool checkPowerOf2(int n) {

        if (n <= 1)

            return false;

        int count = 0;

        while (n) {

            count += (n % 2);

            n /= 2;

        }

        if (count > 1)

            return false;

        return true;

    }

public:

    //检验程序——字典序

    void dicOrder(vector<int>arr, const int n) {

        Solution test;

        cout << "The index of this is: " << endl << test.dictionaryIndex(arr, n) << endl;

        cout << "The next order of this order is: " << endl << test.dictionaryNextOrder(arr, n) << endl;

        cout << endl;

    }

    //检验程序——汉诺塔

    void hanoi(const int n) {

        Solution test;

        cout << "The steps to finish the task is: " << endl;

        test.hanoiCore(n, 'A', 'B', 'C');

        cout << endl;

    }

    //检验程序——循环赛日程表

    void schedule(vector<vector<int>>& arr, const int n) {

        Solution test;

        if (checkPowerOf2(n) == false) {

            cout << "Error, the test data is illegal." << endl;

            return;

        }

        cout << "The schedule of this tenis match is: " << endl << test.tenisTimeTable(arr, n);

        cout << endl;

    }

    void scheduleGeneral(vector<vector<int>>& arr, const int n) {

        Solution test;

        cout << "The schedule of this tenis match is: " << endl << test.tenisTimeTableGeneral(arr, n);

        cout << endl;

    }

};

int main() {

    testData test;

    //test of dictionary order

    if (1) {

        vector<int>arr = { 2,6,4,5,8,1,7,3 };//edit here

        test.dicOrder(arr, arr.size());

    }

    //test of hanoi tower

    if (1) {

        test.hanoi(3);//edit here

    }

    //test of schedule

    if (1) {

        int n = 8;

        vector<vector<int>>arr(n, vector<int>(n, 0));

        test.schedule(arr, n);

    }

    //test of schedule(General)

    if (1) {

        int n = 5;

        vector<vector<int>>arr(n, vector<int>(n, 0));

        test.scheduleGeneral(arr, n);

    }

    return 0;

}