算法设计与分析第一次作业

2019202243 郭炜

**编程题1. 求解n阶螺旋矩阵问题**

问题描述 创建阶螺旋矩阵并输出。

输入描述：输入包含多个测试用例，每个测试用例为一行，包含一个正整数，以输入0表示结束。

输出描述：每个测试用例输出行，每行包括个整数，整数之间用一个空格分隔。

Code:

#include <iostream>

using namespace std;

const int max = 100;

int res[max][max];

void func(int *n*, int *cur1*, int *len*, int *cur2*)

{

*// cur1:起始数字*

*// cur2:起始坐标*

*//  len:当前子块的大小*

*// 子块大小为1/2 递归结束条件*

*// 子块大小为1*

    if (len == 1)

    {

        res[cur2][cur2] = cur1;

        return;

    }

*// 子块大小为2*

    if (len == 2)

    {

        res[cur2][cur2] = cur1++;

        res[cur2][cur2 + 1] = cur1++;

        res[cur2 + 1][cur2 + 1] = cur1++;

        res[cur2 + 1][cur2] = cur1;

        return;

    }

*// start:当前子块起始数字*

*// x1:当前子块起始横坐标*

*// x2:当前子块结束横坐标*

    int start = cur1;

    int x1 = cur2;

    int x2 = n - cur2 + 1;

    if (len >= 3)

    {

*// 顺时针一圈*

        for (size\_t i = x1; i <= x2; i++)

        {

            res[x1][i] = start++;

        }

        for (size\_t i = x1 + 1; i <= x2; i++)

        {

            res[i][x2] = start++;

        }

        for (size\_t i = x2 - 1; i >= x1; i--)

        {

            res[x2][i] = start++;

        }

        for (size\_t i = x2 - 1; i >= x1 + 1; i--)

        {

            res[i][x1] = start++;

        }

*// 内部子块递归*

        func(n, start, len - 2, cur2 + 1);

    }

}

*// 打印*

void disp(int *cur*)

{

    for (size\_t i = 1; i <= cur; i++)

    {

        for (size\_t j = 1; j <= cur; j++)

        {

            cout << res[i][j];

            if (j != cur)

            {

                cout << " ";

            }

        }

        cout << "\n";

    }

}

int main()

{

    int cur;

    cout << "Input: ";

    while (cin >> cur && cur != 0)

    {

        func(cur,1 , cur, 1);

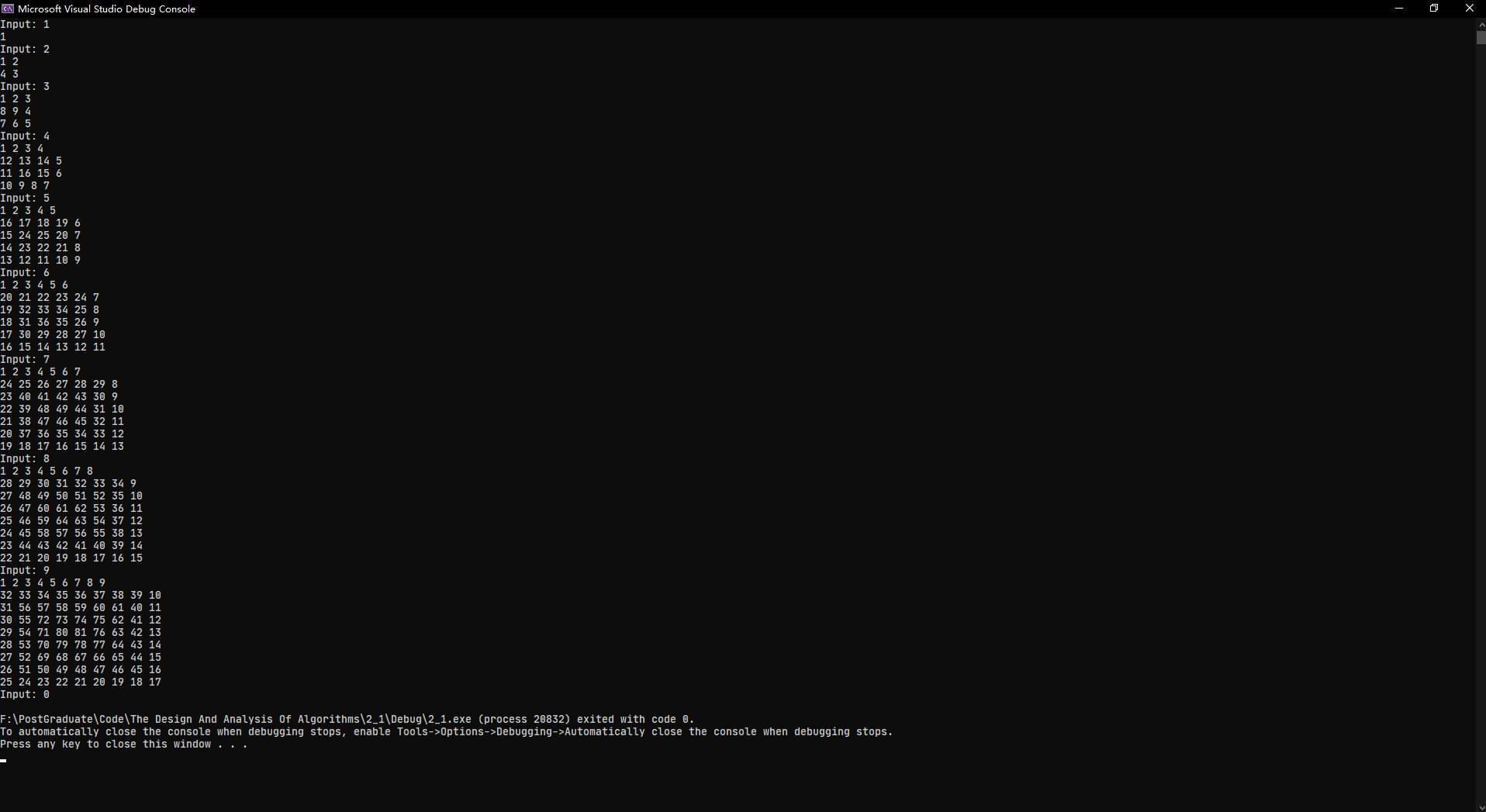
        disp(cur);

        cout << "Input: ";

    }

}

Result:



**编程题2. 求解幸运数问题**

问题描述 小明同学在学习了不同的进制之后用一些数字做起了游戏。小明同学知道，在日常生活中最常用的是十进制数，而计算机中的二进制数也很常用。现在对于一个数字，小明同学定义出两个函数和，表示把这个数用十进制写出后各数位上的数字之和，例如；表示把这个数用二进制写出后各数位上的数字之和，例如123的二进制表示为1111011，那么。小明同学发现对于一些正整数满足，他把这种数称为幸运数，现在他想知道小于等于的幸运数有多少个？

输入描述：每组数据输入一个数。

输出描述：每组数据输出一行，小于等于的幸运数个数。

Code:

#include <iostream>

using namespace std;

*// 迭代*

int myGet(int *t*, int *z*)

{

    int res = 0;

    while (t != 0)

    {

        res += t % z;

        t /= z;

    }

    return res;

}

void func1(int *n*)

{

    int count = 0;

    for (size\_t i = 1; i <= n; i++)

    {

        if (myGet(i, 10) == myGet(i, 2))

        {

            count += 1;

            cout << i << " ";

        }

    }

    cout << endl;

    cout << "func1: " << n << ": " << count << endl;

}

*// 递归求二进制*

int getB(int *n*)

{

    if (n == 1)

    {

        return 1;

    }

    else

    {

        return n % 2 + getB(n / 2);

    }

}

*// 递归求十进制*

int getT(int *n*)

{

    if (n < 10)

    {

        return n;

    }

    else

    {

        return n % 10 + getT(n / 10);

    }

}

void func2(int *n*)

{

    int count = 0;

    for (size\_t i = 1; i <= n; i++)

    {

        if (getB(i) == getT(i))

        {

            count += 1;

            cout << i << " ";

        }

    }

    cout << endl;

    cout << "func2: " << n << ": " << count << endl;

}

int main()

{

    int n;

    while (cin >> n && n != 0)

    {

        func1(n);

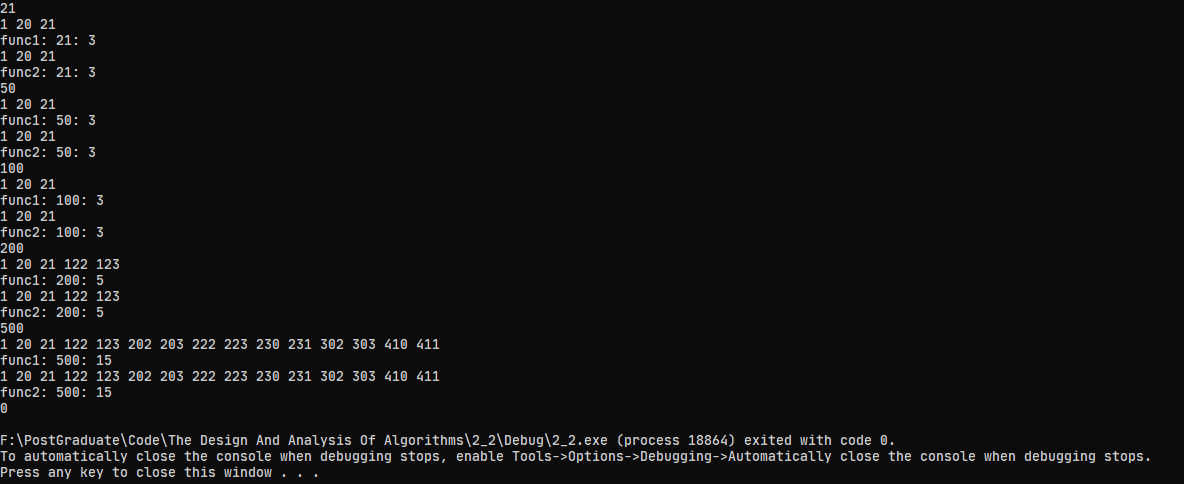
        func2(n);

    }

}

Result:

**递归**和**非递归**方法输出结果相同



**编程题3. 求解回文序列问题**

问题描述 如果一个数字序列逆置后跟原序列是一样的，则称这样的数字序列为回文序列。例如，{1,2,1}、{15,78,78,15}、{11,2,11}是回文序列，而{1,2,2}、{15,78,87,51}、{112,2,11}不是回文序列。现在给出一个数字序列，允许使用一种转换操作：选择任意两个相邻的数，然后从序列中移除这两个数，并将这两个数的和插入到这两个之前的位置（只插入一个和）。

对于所给序列求出最少需要多少次操作可以将其变成回文序列。

输入描述：输入为两行，第1行为序列长度 ，第2行为序列中的个整数,以空格分隔。

输出描述：输出一个数，表示最少需要的转换次数。

Code:

VS2019里面全局变量在函数中调用需要加域标识符::，不然会报错，存在二义性的问题。

#include <iostream>

#include <vector>

using namespace std;

int count = 0;

int func(vector<int> *v*, int *start*, int *end*)

{

    if (start >= end)

    {

        return -1;

    }

    else if (v.at(start) == v.at(end))

    {

        func(v, start + 1, end - 1);

    }

    else if (v.at(start) < v.at(end))

    {

        ::count += 1;

        v.at(start + 1) += v.at(start);

        func(v, start + 1, end);

    }

    else if (v.at(start) > v.at(end))

    {

        ::count += 1;

        v.at(end - 1) += v.at(end);

        func(v, start, end - 1);

    }

    return ::count;

}

int main()

{

    int cur;

    int s;

    while (cin >> s && s != 0)

    {

        vector<int> v;

        for (size\_t i = 0; i < s; i++)

        {

            cin >> cur;

            v.push\_back(cur);

        }

        ::count = 0;

        cout << "count: " << func(v, 0, s - 1) << endl;

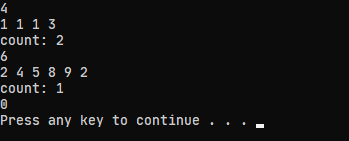
    }

    system("pause");

    return 0;

}

Result:



**编程题4. 求解投骰子游戏问题**

问题描述 玩家根据骰子的点数决定走的步数，即骰子点数为1时可以走一步，点数为2时可以走两步，点数为时可以走步。求玩家走到第步（≤骰子最大点数且投骰子方法唯一）时总共有多少种投骰子的方法。

输入描述：输入包括一个整数。

输出描述：输出一个整数，表示投骰子的方法数。

Code:

这道题找规律，结果为

#include <iostream>

using namespace std;

*// 这道题结果就是2^(n-1)*

int func(int *cur*)

{

    if (cur == 1)

    {

        return 1;

    }

    int count = 1;

    for (size\_t i = 1; i < cur; i++)

    {

        count += func(i);

    }

    return count;

}

void main()

{

    int cur;

    cout << "Input: ";

    while (cin >> cur && cur > 0)

    {

*// int res = func(cur);*

*// cout << res << endl;*

        cout << "Output: " << (1 << (cur - 1)) << endl;

        cout << "Input: ";

    }

}

Result:

