

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 n 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 n ($1 \leq n \leq 50$)，以输入 0 表示结束。

输出描述：每个测试用例输出 n 行，每行包括 n 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:

```

Microsoft Visual Studio Debug Console
Input: 1
1
Input: 2
1 2
4 3
Input: 3
1 2 3
8 9 4
7 6 5
Input: 4
1 2 3 4
12 13 14 5
11 16 15 6
10 9 8 7
Input: 5
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
Input: 6
1 2 3 4 5 6
20 21 22 23 24 7
19 32 33 34 25 8
18 31 36 35 26 9
17 30 29 28 27 10
16 15 14 13 12 11
Input: 7
1 2 3 4 5 6 7
24 25 26 27 28 29 8
23 40 41 42 43 30 9
22 39 48 49 44 31 10
21 38 47 46 45 32 11
20 37 36 35 34 33 12
19 18 17 16 15 14 13
Input: 8
1 2 3 4 5 6 7 8
28 29 30 31 32 33 34 9
27 48 49 50 51 52 35 10
26 47 60 61 62 53 36 11
25 46 59 64 63 54 37 12
24 45 58 57 56 55 38 13
23 44 43 42 41 40 39 14
22 21 20 19 18 17 16 15
Input: 9
1 2 3 4 5 6 7 8 9
32 33 34 35 36 37 38 39 10
31 56 57 58 59 60 61 40 11
30 55 72 73 74 75 62 41 12
29 54 71 80 81 76 63 42 13
28 53 70 79 78 77 64 43 14
27 52 69 68 67 66 65 44 15
26 51 50 49 48 47 46 45 16
25 24 23 22 21 20 19 18 17
Input: 0
F:\PostGraduate\Code\The Design And Analysis Of Algorithms\2.1\Debug\2.1.exe (process 20832) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

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 x ,

小明同学定义出两个函数 $f(x)$ 和 $g(x)$, $f(x)$ 表示把 x 这个数用十进制写出后各数位上的数字之和, 例如 $f(123) = 1 + 2 + 3 = 6$; $g(x)$ 表示把 x 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 $g(123) = 1 + 1 + 1 + 1 + 0 + 1 + 1 = 6$ 。小明同学发现对于一些正整数 x 满足 $f(x) = g(x)$, 他把这种数称为幸运数, 现在他想知道小于等于 n 的幸运数有多少个?

输入描述: 每组数据输入一个数 $n(n \leq 100\,000)$ 。

输出描述: 每组数据输出一行, 小于等于 n 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

```
21
1 20 21
func1: 21: 3
1 20 21
func2: 21: 3
50
1 20 21
func1: 50: 3
1 20 21
func2: 50: 3
100
1 20 21
func1: 100: 3
1 20 21
func2: 100: 3
200
1 20 21 122 123
func1: 200: 5
1 20 21 122 123
func2: 200: 5
500
1 20 21 122 123 202 203 222 223 230 231 302 303 410 411
func1: 500: 15
1 20 21 122 123 202 203 222 223 230 231 302 303 410 411
func2: 500: 15
0
F:\PostGraduate\Code\The Design And Analysis Of Algorithms\2.2\Debug\2.2.exe (process 18864) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

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 $n(1 \leq n \leq 50)$, 第 2 行为序列中的 n 个数 $item[i]$ ($1 \leq item[i] \leq 1000$), 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:

```

4
1 1 1 3
count: 2
6
2 4 5 8 9 2
count: 1
0
Press any key to continue . . . _

```

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 n 时可以走 n 步。求玩家走到第 n 步 ($n \leq$ 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 $n(1 \leq n \leq 6)$ 。

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 2^{n-1}

```

#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:

```

Input: 1
Output: 1
Input: 2
Output: 2
Input: 3
Output: 4
Input: 4
Output: 8
Input: 5
Output: 16
Input: 6
Output: 32
Input: 7
Output: 64
Input: 0
F:\PostGraduate\Code\The Design And Analysis Of Algorithms\2_4\Debug\2-4.exe (process 16772) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```