# 赛程安排矩阵生成

## 应承轩 201785071 电计1701

## 动机

我认为老师课上介绍的非递归写法看起来很别扭，并且老师讲起来也费劲，因此我重写了一个递归版本的程序。程序经过测试，符合预期。

## 算法代码

**#define N 8**

**int table[N][N];**

**struct pt {**

**int x;**

**int y;**

**};**

**void assignmat(struct pt src\_pt, struct pt tar\_pt, int sz) {**

**for (int i = 0; i < sz; ++i) {**

**for (int j = 0; j < sz; ++j) {**

**table[tar\_pt.y + i][tar\_pt.x + j] = table[src\_pt.y + i][src\_pt.x + j];**

**}**

**}**

**}**

**void solve(int rows2cover) {**

**if (rows2cover == 1) {**

**for (int i = 0; i < N; ++i) {**

**table[0][i] = i + 1;**

**}**

**return;**

**}**

**int sz = rows2cover / 2;**

**solve(sz);**

**for (int x1 = 0; x1 < N; x1 += rows2cover) {**

**struct pt b1 = {x1, 0};**

**struct pt b2 = {x1 + sz, 0};**

**struct pt b3 = {x1, sz};**

**struct pt b4 = {x1 + sz, sz};**

**assignmat(b1, b4, sz);**

**assignmat(b2, b3, sz);**

**}**

**}**

**void pt\_Board\_() {**

**for (int i = 0; i < N; ++i) {**

**for (int j = 0; j < N; ++j) {**

**cout << table[i][j] << ' ';**

**}**

**cout << endl;**

**}**

**cout << endl;**

**}**

**void tournament\_arrangement() {**

**solve(N);**

**pt\_Board\_();**

**}**

## 运行结果

**N=1**

**1**

**N=2**

**1 2**

**2 1**

**N=4**

**1 2 3 4**

**2 1 4 3**

**3 4 1 2**

**4 3 2 1**

**N=8**

**1 2 3 4 5 6 7 8**

**2 1 4 3 6 5 8 7**

**3 4 1 2 7 8 5 6**

**4 3 2 1 8 7 6 5**

**5 6 7 8 1 2 3 4**

**6 5 8 7 2 1 4 3**

**7 8 5 6 3 4 1 2**

**8 7 6 5 4 3 2 1**

**N=16**

**1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16**

**2 1 4 3 6 5 8 7 10 9 12 11 14 13 16 15**

**3 4 1 2 7 8 5 6 11 12 9 10 15 16 13 14**

**4 3 2 1 8 7 6 5 12 11 10 9 16 15 14 13**

**5 6 7 8 1 2 3 4 13 14 15 16 9 10 11 12**

**6 5 8 7 2 1 4 3 14 13 16 15 10 9 12 11**

**7 8 5 6 3 4 1 2 15 16 13 14 11 12 9 10**

**8 7 6 5 4 3 2 1 16 15 14 13 12 11 10 9**

**9 10 11 12 13 14 15 16 1 2 3 4 5 6 7 8**

**10 9 12 11 14 13 16 15 2 1 4 3 6 5 8 7**

**11 12 9 10 15 16 13 14 3 4 1 2 7 8 5 6**

**12 11 10 9 16 15 14 13 4 3 2 1 8 7 6 5**

**13 14 15 16 9 10 11 12 5 6 7 8 1 2 3 4**

**14 13 16 15 10 9 12 11 6 5 8 7 2 1 4 3**

**15 16 13 14 11 12 9 10 7 8 5 6 3 4 1 2**

**16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1**