信息学奥赛笔记16

二维数组和动态二维数组。

二维数组

一维数组表示的是一个线，一段线性的，排列在一排的，相同类型的数据。

现在我们把一维数组看成是一个单元，如果我们把这个单元纵向的，拉成一排呢?

是不是一排的一维数组就变成了很多排，他们这个整体就变成了一个面，这就是二维数组。

所以二维数组的组成基本单元就是一维数组。

而这个一维数组的组成单元就是数据类型`int, double, char`之类的。

矩阵

矩阵(Matrix)表示一个由数字或者字符按照一个长方形阵列组成的集合。

矩阵的每一行的元素是相同的，每一列的行数也是相同的。

静态二维数组

想要定义一个静态数组，我们应该采用如下的方式:

```
int a[行数][列数];

//例如:
int a[5][6];
//表示一个5行6列的矩阵
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

那么，我们想要访问这个矩阵左上角这个元素，该怎么办呢?

我们要清楚一件事，a[5][6]需要拆解成a[5]，其中每个元素是int [6];

也就是这个数组有5个元素，其中每个元素是一个长度为6的数组，那么这个6就指的是**列**。

所以我们想要访问这个数组左上角的元素应该是第一个数组的第一个元素，也就是a[0][0]。

静态二维数组的输入输出

那么问题从一维的线性数组的读入变成了一个矩阵，我们就得同时知道这个矩阵的行和列。

```
//二维数组的输入
for (int i = 0; i < m; i++) {  //先遍历行
```

```
    for (int j = 0; j < n; j++) {  //再遍历列
        cin >> a[i][j];
    }
}

//二维数组的输出
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        cout << a[i][j] << " ";
    }
    cout << endl;//输出完一行后换行，最终呈现出矩阵的模样
}
```

动态二维数组

前面我们也说了，二维数组就是一个数组里套一个数组，所以定义一个二维的动态数组，就是在类型里套了一个一维的动态数组

```
vector<vector<int> > a;
// 定义一个0行0列的二维数组a

vector<vector<int> > a(5);
// 定义了一个5行0列的二维数组a

vector<vector<int> > a(5, vector<int>(6));
// 定义了一个5行6列的二维数组a
```

那假如给定了你一个未知长宽的动态二维数组，你该怎么求他的行和列呢?

```
//给定一个未知行列的动态数组b;
int m = b.size(), n = b[0].size();

// m就是b的行，  n就是b的列
//因为n是第0行数组的长度，就是一共的列数
```

二维数组的修改

题目描述

输入两个正整数n, m，表示数组a的行列，再将第x行y列的数修改为z，请输出修改后的数组

输入格式

第一行两个正整数n, m，表示数组a的行列。

接下来n行m列，表示数组的值。

接下来一行有3个整数x, y, z。表示需要将第x行y列的值修改为z。

输出格式

n行m列整数，表示修改后的数组。

样例 #1

样例输入 #1

```
3 3
1 4 7
2 5 8
3 6 9
1 1 100
```

样例输出 #1

```
1 4 7
2 100 8
3 6 9
```

提示

对于100%的数据，有$1 \leq n, m, x, y \leq 3000, 1 \leq a_i, z \leq 10^6$。

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, x, y, z;
    cin >> n >> m;
    vector<vector<int> > grid(n, vector<int> (m));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> grid[i][j];
        }
    }
    cin >> x >> y >> z;
    grid[x][y] = z;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cout << grid[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

二维数组的查询

题目描述

输入两个正整数n, m，表示数组a的行列，请输出第x行y列的值。

输入格式

第一行两个正整数n, m，表示数组a的行列。

接下来n行m列，表示数组的值。

接下来一行有2个整数x, y。表示需要查询第x行y列的值。

输出格式

一个整数，表示答案。

样例 #1

样例输入 #1

```
3 3
1 4 7
2 5 8
3 6 9
1 1
```

样例输出 #1

```
5
```

提示

对于100%的数据，有$1 \leq n, m, x, y \leq 3000, 1 \leq a_i, x \leq 10^6$。

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, x, y, z;
    cin >> n >> m;
    vector<vector<int> > grid(n, vector<int> (m));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> grid[i][j];
        }
    }
    cin >> x >> y;
    cout << grid[x][y];
    return 0;
}
```

增长矩阵

题目描述

输入两个正整数n, m，请输出一个n行m列的矩阵，数字同行依次增加。

输入格式

两个正整数n, m。

输出格式

n行m列的正整数，表示答案。

样例 #1

样例输入 #1

```
3 4
```

样例输出 #1

```
1 2 3 4
5 6 7 8
9 10 11 12
```

提示

对于100%的数据，有$1 \leq n, m \leq 300$。

思路分析

> 对于数组的值，我们需要让他依次增加一个，我们可以使用一个累加器cnt来记录目前存到几了，然后随着数组下标的变化，cnt也随之变化来达到效果。

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main() {
    int n, m, cnt = 1;
    cin >> n >> m;
    vector<vector<int> > grid(n, vector<int>(m));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            grid[i][j] = cnt;
            cnt++;
        }
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cout << grid[i][j] << " ";
        }
```

```
        cout << endl;
    }
    return 0;
}
```

二维数组的逐行反转

题目描述

给定一个$n \times m$的二维数组，将该数组的每行元素反转后输出。

输入格式

第一行两个正整数n, m，表示数组的行列。

接下来n行m列，表示数值的值。

输出格式

n行m列正整数，表示数组逐行反转后的值。

样例 #1

样例输入 #1

```
3 3
1 2 3
4 5 6
7 8 9
```

样例输出 #1

```
3 2 1
6 5 4
9 8 7
```

提示

对于100%的数据，有$1 \leq n, m \leq 10^3$, $1 \leq a_{ij} \leq 10^6$。

思路分析

> 这道题目涉及到了对数组的逐行反转，我们可以使用一维数组的反转技巧，对二维数组每一行反转最后输出即可

```
#include <bits/stdc++.h>
using namespace std;
int main() {
```

```
    int n, m;
    cin >> n >> m;
    vector<vector<int> > a(n, vector<int>(m));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> grid[i][j];
        }
    }
    for (int i = 0; i < n; i++) { // 对二维数组逐行遍历
        reverse(a[i].begin(), a[i].end());  //对于一行来说 a[i]也是一个数组，我们把a[i]反转
    }
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cout << grid[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```

[NOIP2015 提高组] 神奇的幻方

题目背景

NOIp2015 提高组 Day1T1

题目描述

幻方是一种很神奇的 $N \times N$ 矩阵：它由数字 $1, 2, 3, \cdots\cdots, N \times N$ 构成，且每行、每列及两条对角线上的数字之和都相同。

当 N 为奇数时，我们可以通过下方法构建一个幻方：

首先将 1 写在第一行的中间。

之后，按如下方式从小到大依次填写每个数 K $(K = 2, 3, \cdots, N \times N)$：

1. 若 $(K-1)$ 在第一行但不在最后一列，则将 K 填在最后一行，$(K-1)$ 所在列的右一列；
2. 若 $(K-1)$ 在最后一列但不在第一行，则将 K 填在第一列，$(K-1)$ 所在行的上一行；
3. 若 $(K-1)$ 在第一行最后一列，则将 K 填在 $(K-1)$ 的正下方；
4. 若 $(K-1)$ 既不在第一行，也不在最后一列，如果 $(K-1)$ 的右上方还未填数，则将 K 填在 $(K-1)$ 的右上方，否则将 K 填在 $(K-1)$ 的正下方。

现给定 N，请按上述方法构造 $N \times N$ 的幻方。

输入格式

一个正整数 N，即幻方的大小。

输出格式

共 N 行，每行 N 个整数，即按上述方法构造出的 $N \times N$ 的幻方，相邻两个整数之间用单空格隔开。

样例 #1

样例输入 #1

```
3
```

样例输出 #1

```
8 1 6
3 5 7
4 9 2
```

样例 #2

样例输入 #2

```
25
```

样例输出 #2

```
327 354 381 408 435 462 489 516 543 570 597 624 1 28 55 82 109 136 163 190 217 244 271 298 325
353 380 407 434 461 488 515 542 569 596 623 25 27 54 81 108 135 162 189 216 243 270 297 324 326
379 406 433 460 487 514 541 568 595 622 24 26 53 80 107 134 161 188 215 242 269 296 323 350 352
405 432 459 486 513 540 567 594 621 23 50 52 79 106 133 160 187 214 241 268 295 322 349 351 378
431 458 485 512 539 566 593 620 22 49 51 78 105 132 159 186 213 240 267 294 321 348 375 377 404
457 484 511 538 565 592 619 21 48 75 77 104 131 158 185 212 239 266 293 320 347 374 376 403 430
483 510 537 564 591 618 20 47 74 76 103 130 157 184 211 238 265 292 319 346 373 400 402 429 456
509 536 563 590 617 19 46 73 100 102 129 156 183 210 237 264 291 318 345 372 399 401 428 455 482
535 562 589 616 18 45 72 99 101 128 155 182 209 236 263 290 317 344 371 398 425 427 454 481 508
561 588 615 17 44 71 98 125 127 154 181 208 235 262 289 316 343 370 397 424 426 453 480 507 534
587 614 16 43 70 97 124 126 153 180 207 234 261 288 315 342 369 396 423 450 452 479 506 533 560
613 15 42 69 96 123 150 152 179 206 233 260 287 314 341 368 395 422 449 451 478 505 532 559 586
14 41 68 95 122 149 151 178 205 232 259 286 313 340 367 394 421 448 475 477 504 531 558 585 612
40 67 94 121 148 175 177 204 231 258 285 312 339 366 393 420 447 474 476 503 530 557 584 611 13
66 93 120 147 174 176 203 230 257 284 311 338 365 392 419 446 473 500 502 529 556 583 610 12 39
92 119 146 173 200 202 229 256 283 310 337 364 391 418 445 472 499 501 528 555 582 609 11 38 65
118 145 172 199 201 228 255 282 309 336 363 390 417 444 471 498 525 527 554 581 608 10 37 64 91
```

```
144 171 198 225 227 254 281 308 335 362 389 416 443 470 497 524 526 553 580 607 9 36 63 90 117
170 197 224 226 253 280 307 334 361 388 415 442 469 496 523 550 552 579 606 8 35 62 89 116 143
196 223 250 252 279 306 333 360 387 414 441 468 495 522 549 551 578 605 7 34 61 88 115 142 169
222 249 251 278 305 332 359 386 413 440 467 494 521 548 575 577 604 6 33 60 87 114 141 168 195
248 275 277 304 331 358 385 412 439 466 493 520 547 574 576 603 5 32 59 86 113 140 167 194 221
274 276 303 330 357 384 411 438 465 492 519 546 573 600 602 4 31 58 85 112 139 166 193 220 247
300 302 329 356 383 410 437 464 491 518 545 572 599 601 3 30 57 84 111 138 165 192 219 246 273
301 328 355 382 409 436 463 490 517 544 571 598 625 2 29 56 83 110 137 164 191 218 245 272 299
```

提示

对于 100% 的数据，对于全部数据，$1 \leq N \leq 39$ 且 N 为奇数。

思路分析

> 本题只需要按照题目的思路模拟即可，那模拟的过程中我们发现数组的下标并不是按照遍历顺序做的，而是按照它给定的规则一直变换，那么有没有什么是能够遍历的量呢？是不是我们正在填的数，是从1开始，一直增长到n * n的，也就是说，我们只需要按照填的数开始遍历即可。然后用两个变量来存储一下目前应该把数填在哪，所以真正需要我们关注的问题就是，在每个循环的时候第一步填数，第二步移动坐标，这样就可以完成了。

```cpp
#include<iostream>
using namespace std;
int a[40][40];
int n;
int main() {
    int n;
    cin >> n;
    int x = 0, y = n / 2;
    for (int i = 1; i <= n * n; i++) {
        a[x][y] = i;
        if (x == 0 && y != n - 1) {          //条件1
            x = n - 1, y++;
        }
        else if (y == n - 1 && x != 0) {     //条件2
            y = 0, x--;
        }
        else if (x == 0 && y == n - 1) {     //条件3
            x++;
        }
        else if (x != 0 && y != n - 1) {     //条件4
            if (a[x - 1][y + 1] == 0) {
                x--, y++;
            } else {
                x++;
```

```
            }
        }
    }
    for (int i = 0; i < n; i++) {              //二维数组输出
        for (int j = 0; j < n; j++) {
            cout << a[i][j] << " ";
        }
        cout << endl;
    }
    return 0;
}
```