**CCF 2013-2016年第二题题解及源码**

[**CCF201312-2 ISBN号码（100分）**](http://blog.csdn.net/tigerisland45/article/details/54782244)

|  |  |
| --- | --- |
| 试题编号： | 201312-2 |
| 试题名称： | ISBN号码 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　每一本正式出版的图书都有一个ISBN号码与之对应，ISBN码包括9位数字、1位识别码和3位分隔符，其规定格式如“x-xxx-xxxxx-x”，其中符号“-”是分隔符（键盘上的减号），最后一位是识别码，例如0-670-82162-4就是一个标准的ISBN码。ISBN码的首位数字表示书籍的出版语言，例如0代表英语；第一个分隔符“-”之后的三位数字代表出版社，例如670代表维京出版社；第二个分隔之后的五位数字代表该书在出版社的编号；最后一位为识别码。 　　识别码的计算方法如下： 　　首位数字乘以1加上次位数字乘以2……以此类推，用所得的结果mod 11，所得的余数即为识别码，如果余数为10，则识别码为大写字母X。例如ISBN号码0-670-82162-4中的识别码4是这样得到的：对067082162这9个数字，从左至右，分别乘以1，2，…，9，再求和，即0×1+6×2+……+2×9=158，然后取158 mod 11的结果4作为识别码。 　　编写程序判断输入的ISBN号码中识别码是否正确，如果正确，则仅输出“Right”；如果错误，则输出是正确的ISBN号码。  **输入格式**  　　输入只有一行，是一个字符序列，表示一本书的ISBN号码（保证输入符合ISBN号码的格式要求）。  **输出格式**  　　输出一行，假如输入的ISBN号码的识别码正确，那么输出“Right”，否则，按照规定的格式，输出正确的ISBN号码（包括分隔符“-”）。  **样例输入**  0-670-82162-4  **样例输出**  Right  **样例输入**  0-670-82162-0  **样例输出**  0-670-82162-4 |

**问题链接：**[CCF201312试题](http://blog.csdn.net/tigerisland45/article/details/54429702" \t "_blank)。

**问题描述**：根据输入的ISBN号码，计算识别码并且输出结果。

**问题分析**：这是一个字符串和字符处理的问题，是一个编程基础问题，包括字符与整数之间相互转换和计算。

**程序说明**：需要了解string类的使用。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54782244) [copy](http://blog.csdn.net/tigerisland45/article/details/54782244)

1. /\* CCF201312-2 ISBN号码 \*/
3. #include <iostream>
4. #include <string>
6. **using** **namespace** std;
8. **int** main()
9. {
10. string s;
12. // 输入数据
13. cin >> s;
15. // 计算识别码
16. **int** sum = 0, j = 1;
17. **for**(**int** i=0; i<(**int**)s.length()-1; i++) {
18. **if**(s[i] != '-')
19. sum += (s[i] - '0') \* j++;
20. }
22. // 识别码转字符
23. **char** c;
24. c = sum % 11;
25. **if**(c == 10)
26. c = 'X';
27. **else**
28. c = c + '0';
30. // 判断识别码、校正和输出
31. **if**(c == s[s.length()-1])
32. cout << "Right" << endl;
33. **else** {
34. s[s.length()-1] = c;
35. cout << s << endl;
36. }
38. **return** 0;
39. }

[**CCF201403-2 窗口（100分）**](http://blog.csdn.net/tigerisland45/article/details/54782238)

.

|  |  |
| --- | --- |
| 试题编号： | 201403-2 |
| 试题名称： | 窗口 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在某图形操作系统中,有 N 个窗口,每个窗口都是一个两边与坐标轴分别平行的矩形区域。窗口的边界上的点也属于该窗口。窗口之间有层次的区别,在多于一个窗口重叠的区域里,只会显示位于顶层的窗口里的内容。 　　当你点击屏幕上一个点的时候,你就选择了处于被点击位置的最顶层窗口,并且这个窗口就会被移到所有窗口的最顶层,而剩余的窗口的层次顺序不变。如果你点击的位置不属于任何窗口,则系统会忽略你这次点击。 　　现在我们希望你写一个程序模拟点击窗口的过程。  **输入格式**  　　输入的第一行有两个正整数,即 N 和 M。(1 ≤ N ≤ 10,1 ≤ M ≤ 10) 　　接下来 N 行按照从最下层到最顶层的顺序给出 N 个窗口的位置。 每行包含四个非负整数 x1, y1, x2, y2,表示该窗口的一对顶点坐标分别为 (x1, y1) 和 (x2, y2)。保证 x1 < x2,y1 2。 　　接下来 M 行每行包含两个非负整数 x, y,表示一次鼠标点击的坐标。 　　题目中涉及到的所有点和矩形的顶点的 x, y 坐标分别不超过 2559 和　　1439。  **输出格式**  　　输出包括 M 行,每一行表示一次鼠标点击的结果。如果该次鼠标点击选择了一个窗口,则输出这个窗口的编号(窗口按照输入中的顺序从 1 编号到 N);如果没有,则输出"IGNORED"(不含双引号)。  **样例输入**  3 4 0 0 4 4 1 1 5 5 2 2 6 6 1 1 0 0 4 4 0 5  **样例输出**  2 1 1 IGNORED  **样例说明**  　　第一次点击的位置同时属于第 1 和第 2 个窗口,但是由于第 2 个窗口在上面,它被选择并且被置于顶层。 　　第二次点击的位置只属于第 1 个窗口,因此该次点击选择了此窗口并将其置于顶层。现在的三个窗口的层次关系与初始状态恰好相反了。 　　第三次点击的位置同时属于三个窗口的范围,但是由于现在第 1 个窗口处于顶层,它被选择。 　　最后点击的 (0, 5) 不属于任何窗口。 |

**问题链接：**[CCF201403试题](http://blog.csdn.net/tigerisland45/article/details/54430251)。

**问题描述**：n个重叠的窗口，鼠标m次点击屏幕的一个点，输出这些点击所点击到的窗口号。

**问题分析**：这是一个模拟题，模拟鼠标点击屏幕的过程。该问题比实际的多窗口系统要简单。关键在于数据表示，其他要点则是模拟鼠标操作过程的处理。

**程序说明**：

数组win[]用来存储窗口。除了窗口坐标，附加上成员变量winno用来存储窗口号是一种规范的做法，就本问题而言可以不用该变量。

数组point[]用于存储鼠标点击的坐标。

数组order[]用于存储窗口的顺序。这是一种编程技巧，在由于鼠标点击导致窗口顺序改变的时候，就不需要移动窗口坐标等数据，只需要移动这个索引数据即可。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54782238) [copy](http://blog.csdn.net/tigerisland45/article/details/54782238)

1. /\* CCF201403-2 窗口 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. **const** **int** N = 10;
9. **struct** {
10. **int** winno;
11. **int** x1, y1, x2, y2;
12. } win[N];
14. **struct** {
15. **int** x, y;
16. } point[N];
18. **int** order[N];
20. **int** main()
21. {
22. **int** n, m;
24. // 输入数据
25. cin >> n >> m;
26. **for**(**int** i=0; i<n; i++) {
27. win[i].winno = i + 1;
28. cin >> win[i].x1 >> win[i].y1 >> win[i].x2 >> win[i].y2;
29. }
30. **for**(**int** i=0; i<m; i++)
31. cin >> point[i].x >> point[i].y;
33. // 初始化窗口显示顺序
34. **for**(**int** i=0; i<n; i++)
35. order[i] = n - i - 1;
37. // 处理（模拟）鼠标点击
38. **int** winno, temp;
39. **for**(**int** i=0; i<m; i++) {
40. winno = -1;     // 被点击的窗口号，－１表示未被点击
42. // 判断哪个窗口被点击
43. **for**(**int** j=0; j<n; j++) {
44. **if**(win[order[j]].x1 <= point[i].x && point[i].x <= win[order[j]].x2 &&
45. win[order[j]].y1 <= point[i].y && point[i].y <= win[order[j]].y2) {
46. // 得到窗口号
47. winno = win[order[j]].winno;
49. // 将被点击的窗口移到最前端
50. temp = order[j];
51. **for**(**int** k=j; k>0; k--)
52. order[k] = order[k-1];
53. order[0] = temp;
55. **break**;
56. }
57. }
59. // 输出结果
60. **if**(winno == -1)
61. cout << "IGNORED" << endl;
62. **else**
63. cout << winno << endl;
64. }
66. **return** 0;
67. }

[**CCF201409-2 画图（100分）**](http://blog.csdn.net/tigerisland45/article/details/54773758)

.

|  |  |
| --- | --- |
| 试题编号： | 201409-2 |
| 试题名称： | 画图 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在一个定义了直角坐标系的纸上，画一个(x1,y1)到(x2,y2)的矩形指将横坐标范围从x1到x2，纵坐标范围从y1到y2之间的区域涂上颜色。 　　下图给出了一个画了两个矩形的例子。第一个矩形是(1,1) 到(4, 4)，用绿色和紫色表示。第二个矩形是(2, 3)到(6, 5)，用蓝色和紫色表示。图中，一共有15个单位的面积被涂上颜色，其中紫色部分被涂了两次，但在计算面积时只计算一次。在实际的涂色过程中，所有的矩形都涂成统一的颜色，图中显示不同颜色仅为说明方便。 C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center.png 　　给出所有要画的矩形，请问总共有多少个单位的面积被涂上颜色。  **输入格式**  　　输入的第一行包含一个整数n，表示要画的矩形的个数。 　　接下来n行，每行4个非负整数，分别表示要画的矩形的左下角的横坐标与纵坐标，以及右上角的横坐标与纵坐标。  **输出格式**  　　输出一个整数，表示有多少个单位的面积被涂上颜色。  **样例输入**  2 1 1 4 4 2 3 6 5  **样例输出**  15  **评测用例规模与约定**  　　1<=n<=100，0<=横坐标、纵坐标<=100。 |

**问题链接：**[CCF201409试题](http://blog.csdn.net/tigerisland45/article/details/54446624)。

**问题描述**：输入n个矩形的左下角的横坐标与纵坐标，以及右上角的横坐标与纵坐标，求这些矩形所覆盖的面积。

**问题分析**：若干个矩形叠加在一起，要算出其面积，看起来有点复杂。关键是其面积是由若干正方形组成的。计算的方法有两种：

1.将各个正方形标记出来，然后再统计计算其面积。

2.将各个正方形面积累加起来，然后去掉重叠部分。哪些部分重叠也需要进行标记后才能判定。

**程序说明**：本程序采用第2种方法进行计算。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54773758) [copy](http://blog.csdn.net/tigerisland45/article/details/54773758)[在CODE上查看代码片](https://code.csdn.net/snippets/2177670)

1. /\* CCF201409-2 画图 \*/
3. #include <iostream>
4. #include <cstring>
6. **using** **namespace** std;
8. **const** **int** N = 100;
10. **bool** flag[N+1][N+1];
12. **int** main()
13. {
14. **int** n, x1, y1, x2, y2, sum=0;
16. // 变量初始化
17. memset(flag, **false**, **sizeof**(flag));
19. // 输入数据，计算处理
20. cin >> n;
21. **for**(**int** i=1; i<=n; i++) {
22. // 输入数据
23. cin >> x1 >> y1 >> x2 >> y2;
25. // 累加面积
26. sum += (x2 - x1) \* (y2 - y1);
28. // 标记和去除重叠部分
29. **for**(**int** i=x1; i<x2; i++)
30. **for**(**int** j=y1; j<y2; j++) {
31. **if**(flag[i][j])
32. sum--;
33. flag[i][j] = **true**;
34. }
35. }
37. // 输出结果
38. cout << sum << endl;
40. **return** 0;
41. }

[**CCF201412-2 Z字形扫描（100分）**](http://blog.csdn.net/tigerisland45/article/details/54773635)

|  |  |
| --- | --- |
| 试题编号： | 201412-2 |
| 试题名称： | Z字形扫描 |
| 时间限制： | 2.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在图像编码的算法中，需要将一个给定的方形矩阵进行Z字形扫描(Zigzag Scan)。给定一个n×n的矩阵，Z字形扫描的过程如下图所示： C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center(1).png 　　对于下面的4×4的矩阵， 　　1 5 3 9 　　3 7 5 6 　　9 4 6 4 　　7 3 1 3 　　对其进行Z字形扫描后得到长度为16的序列： 　　1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3 　　请实现一个Z字形扫描的程序，给定一个n×n的矩阵，输出对这个矩阵进行Z字形扫描的结果。  **输入格式**  　　输入的第一行包含一个整数n，表示矩阵的大小。 　　输入的第二行到第n+1行每行包含n个正整数，由空格分隔，表示给定的矩阵。  **输出格式**  　　输出一行，包含n×n个整数，由空格分隔，表示输入的矩阵经过Z字形扫描后的结果。  **样例输入**  4 1 5 3 9 3 7 5 6 9 4 6 4 7 3 1 3  **样例输出**  1 5 3 9 7 3 9 5 4 7 3 6 6 4 1 3  **评测用例规模与约定**  　　1≤n≤500，矩阵元素为不超过1000的正整数。 |

**问题链接：**[CCF201412试题](http://blog.csdn.net/tigerisland45/article/details/54446794)。

**问题描述**：给定一个n×n的矩阵，左上角到右下角Z字形扫描输出该矩阵的元素。先输入正整数n，再输入n×n个整数。

**问题分析**：这是一个将二维矩阵元素映射为一维序列的问题。这个问题的关键是，输出一个元素（假设位于<row,col>）后，下一个输出元素的位置如何计算？只要找出一套计算下一个输出元素下标的规则，问题就解决了。

**程序说明**：（略）。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54773635) [copy](http://blog.csdn.net/tigerisland45/article/details/54773635)[在CODE上查看代码片](https://code.csdn.net/snippets/2218148)

1. /\* 201412-2 Z字形扫描 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. **const** **int** EAST = 0;
8. **const** **int** SOUTH = 1;
9. **const** **int** SOUTHWEST = 2;
10. **const** **int** NORTHEAST = 3;
12. **struct** {
13. **int** drow;
14. **int** dcol;
15. } direct[] = {{0, 1}, {1, 0}, {1, -1}, {-1, 1}};
17. **const** **int** N = 500;
19. **int** a[N][N];
21. **int** main()
22. {
23. **int** n;
25. // 输入数据
26. cin >> n;
27. **for**(**int** i=0; i<n; i++)
28. **for**(**int** j=0; j<n; j++)
29. cin >> a[i][j];
31. // Z字形扫描，输出结果
32. **int** row=0, col=0, next=EAST;
33. cout << a[row][col];
34. **while**(row != n - 1 || col != n - 1) {
35. row += direct[next].drow;
36. col += direct[next].dcol;
37. cout << " " << a[row][col];
39. **if**(next == EAST && row == 0)
40. next = SOUTHWEST;
41. **else** **if**(next == EAST && row == n - 1)
42. next = NORTHEAST;
43. **else** **if**(next == SOUTH && col == 0)
44. next = NORTHEAST;
45. **else** **if**(next == SOUTH && col == n - 1)
46. next = SOUTHWEST;
47. **else** **if**(next == SOUTHWEST && row == n - 1)
48. next = EAST;
49. **else** **if**(next == SOUTHWEST && col == 0)
50. next = SOUTH;
51. **else** **if**(next == NORTHEAST && col == n - 1)
52. next = SOUTH;
53. **else** **if**(next == NORTHEAST && row == 0)
54. next = EAST;
55. }
56. cout << endl;
58. **return** 0;
59. }

[**CCF201503-2 数字排序（100分）**](http://blog.csdn.net/tigerisland45/article/details/54772817)

|  |  |
| --- | --- |
| 试题编号： | 201503-2 |
| 试题名称： | 数字排序 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给定n个整数，请统计出每个整数出现的次数，按出现次数从多到少的顺序输出。  **输入格式**  　　输入的第一行包含一个整数n，表示给定数字的个数。 　　第二行包含n个整数，相邻的整数之间用一个空格分隔，表示所给定的整数。  **输出格式**  　　输出多行，每行包含两个整数，分别表示一个给定的整数和它出现的次数。按出现次数递减的顺序输出。如果两个整数出现的次数一样多，则先输出值较小的，然后输出值较大的。  **样例输入**  12 5 2 3 3 1 3 4 2 5 2 3 5  **样例输出**  3 4 2 3 5 3 1 1 4 1  **评测用例规模与约定**  　　1 ≤ n ≤ 1000，给出的数都是不超过1000的非负整数。 |

**问题链接：**[CCF201503试题](http://blog.csdn.net/tigerisland45/article/details/54446795)。

**问题描述**：给定n个整数，请统计出每个整数出现的次数，按出现次数从多到少的顺序输出。

**问题分析**：这是一个统计排序的问题，先统计，后排序。一般用数组进行统计和排序，时空上难以获得完美。使用STL的包装类实现是一个好的选择。

**程序说明**：使用map进行统计，使用优先队列进行排序，是本程序的关键。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54772817) [copy](http://blog.csdn.net/tigerisland45/article/details/54772817)[在CODE上查看代码片](https://code.csdn.net/snippets/2151674)

1. /\* CCF201503-2 数字排序 \*/
3. #include <iostream>
4. #include <map>
5. #include <queue>
7. **using** **namespace** std;
9. **struct** node {
10. **int** key, count;
11. **bool** operator < (**const** node& n) **const** {
12. **if**(count == n.count)
13. **return** key > n.key;
14. **else**
15. **return** count < n.count;
16. }
17. };
19. **int** main()
20. {
21. priority\_queue<node> q;
22. map<**int**, **int**> m;
23. **int** n, v;
25. // 输入数据，构建Map
26. cin >> n;
27. **for**(**int** i=0; i<n; i++) {
28. cin >> v;
29. m[v]++;
30. }
32. // 构建优先队列进行排序
33. node keyval;
34. **for**(map<**int**,**int**>::iterator it=m.begin(); it!=m.end(); it++) {
35. keyval.key = it->first;
36. keyval.count = it->second;
37. q.push(keyval);
38. }
40. // 输出结果
41. **while**(!q.empty()) {
42. keyval = q.top();
43. q.pop();
45. cout << keyval.key << " " << keyval.count << endl;
46. }
48. **return** 0;
49. }

[**CCF201509-2 日期计算（100分）**](http://blog.csdn.net/tigerisland45/article/details/54769447)

|  |  |
| --- | --- |
| 试题编号： | 201509-2 |
| 试题名称： | 日期计算 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给定一个年份y和一个整数d，问这一年的第d天是几月几日？ 　　注意闰年的2月有29天。满足下面条件之一的是闰年： 　　1） 年份是4的整数倍，而且不是100的整数倍； 　　2） 年份是400的整数倍。  **输入格式**  　　输入的第一行包含一个整数y，表示年份，年份在1900到2015之间（包含1900和2015）。 　　输入的第二行包含一个整数d，d在1至365之间。  **输出格式**  　　输出两行，每行一个整数，分别表示答案的月份和日期。  **样例输入**  2015 80  **样例输出**  3 21  **样例输入**  2000 40  **样例输出**  2 9 |

**问题链接：**[CCF201509试题](http://blog.csdn.net/tigerisland45/article/details/54446801" \t "_blank)。

**问题描述**：给定一个年份y和一个整数d，问这一年的第d天是几月几日？

**问题分析**：需要注意闰年问题。根据天数d，逐月计算月份，最后算出月份内的日，即采用天数逐月减去１月份天数、二月份天数、...的办法进行计算。

**程序说明**：使用数组days[]存储每月的天数。使用根据年判断闰年的函数isleapyear()调整二月份的天数。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54769447) [copy](http://blog.csdn.net/tigerisland45/article/details/54769447)

1. /\* CCF201509-2 日期计算 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. **int** days[12] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
9. // 闰年计算函数
10. **bool** isleapyear(**int** year) {
11. **if**((year%4 == 0 && year%100 != 0) || year%400 == 0)
12. **return** **true**;
13. **return** **false**;
14. }
16. **int** main()
17. {
18. **int** y, d, month=0, day, i;
20. // 输入数据
21. cin >> y >> d;
23. // 计算月与日
24. **if**(isleapyear(y))
25. days[1]++;
26. i = 0;
27. **while**(d > 0) {
28. **if**(d <= days[i]) {
29. month = i + 1;
30. day = d;
31. **break**;
32. } **else** {
33. d -= days[i];
34. i++;
35. }
36. }
38. // 输出结果
39. cout << month << endl << day << endl;
41. **return** 0;
42. }

[**CCF201512-2 消除类游戏（100分）**](http://blog.csdn.net/tigerisland45/article/details/54565428)

.

|  |  |
| --- | --- |
| 试题编号： | 201512-2 |
| 试题名称： | 消除类游戏 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　消除类游戏是深受大众欢迎的一种游戏，游戏在一个包含有n行m列的游戏棋盘上进行，棋盘的每一行每一列的方格上放着一个有颜色的棋子，当一行或一列上有连续三个或更多的相同颜色的棋子时，这些棋子都被消除。当有多处可以被消除时，这些地方的棋子将同时被消除。 　　现在给你一个n行m列的棋盘，棋盘中的每一个方格上有一个棋子，请给出经过一次消除后的棋盘。 　　请注意：一个棋子可能在某一行和某一列同时被消除。  **输入格式**  　　输入的第一行包含两个整数n, m，用空格分隔，分别表示棋盘的行数和列数。 　　接下来n行，每行m个整数，用空格分隔，分别表示每一个方格中的棋子的颜色。颜色使用1至9编号。  **输出格式**  　　输出n行，每行m个整数，相邻的整数之间使用一个空格分隔，表示经过一次消除后的棋盘。如果一个方格中的棋子被消除，则对应的方格输出0，否则输出棋子的颜色编号。  **样例输入**  4 5 2 2 3 1 2 3 4 5 1 4 2 3 2 1 3 2 2 2 4 4  **样例输出**  2 2 3 0 2 3 4 5 0 4 2 3 2 0 3 0 0 0 4 4  **样例说明**  　　棋盘中第4列的1和第4行的2可以被消除，其他的方格中的棋子均保留。  **样例输入**  4 5 2 2 3 1 2 3 1 1 1 1 2 3 2 1 3 2 2 3 3 3  **样例输出**  2 2 3 0 2 3 0 0 0 0 2 3 2 0 3 2 2 0 0 0  **样例说明**  　　棋盘中所有的1以及最后一行的3可以被同时消除，其他的方格中的棋子均保留。  **评测用例规模与约定**  　　所有的评测用例满足：1 ≤ n, m ≤ 30。 |

**问题链接：**[CCF201512试题](http://blog.csdn.net/tigerisland45/article/details/54446853)。

**问题描述**：首先输入正整数m和n，分别为矩阵的行和列数，m,n≤30，然后输入m行n列的正整数数据。将m×n矩阵中行或列连续三个元素同值的变为0。

**问题分析**：有各种各样的实现方法。这里采用一种不使用额外存储的做法，需要消除的元素暂时将其值变为负值，等行和列都标记之后，再统一消除（置值）为0，最后输出结果。

**程序说明**：

1.程序中使用了求整数绝对值的数学函数abs()，网上查了一下说使用库math.h有可能有二义性，应该用stdlib.h来替代，有点吓唬人。也许一些简单的函数自己编写更好。

2.玩程序就是玩时间与空间。这个程序使用了一点技巧，标记删除元素时，暂时置为负值。这样做节省了存储空间，但是需要付出一定的时间代价。程序员需要掌握时间与空间的平衡，采用合理的措施。

3.这个程序逻辑上是简洁的，易于阅读。

4.使用函数memset()对数组进行初始化是套路。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54565428) [copy](http://blog.csdn.net/tigerisland45/article/details/54565428)[在CODE上查看代码片](https://code.csdn.net/snippets/2129171)

1. /\* CCF201512-2 消除游戏 \*/
3. #include <iostream>
4. #include <cmath>
6. **using** **namespace** std;
8. **const** **int** N = 30;
10. **int** grid[N][N], n, m;
12. **int** main()
13. {
14. // 输入数据
15. cin >> n >> m;
16. **for**(**int** i=0; i<n; i++)
17. **for**(**int** j=0; j<m; j++) {
18. cin >> grid[i][j];
19. }
21. // 进行行标记（可以消除则置为负）
22. **for**(**int** i=0; i<n; i++)
23. **for**(**int** j=0; j<m-2; j++) {
24. **if**(abs(grid[i][j]) == abs(grid[i][j+1]) && abs(grid[i][j+1]) == abs(grid[i][j+2])) {
25. **if**(grid[i][j] > 0)
26. grid[i][j] = - grid[i][j];
27. **if**(grid[i][j+1] > 0)
28. grid[i][j+1] = - grid[i][j+1];
29. **if**(grid[i][j+2] > 0)
30. grid[i][j+2] = - grid[i][j+2];
31. }
32. }
34. // 进行列标记（可以消除则置为负）
35. **for**(**int** j=0; j<m; j++)
36. **for**(**int** i=0; i<n-2; i++) {
37. **if**(abs(grid[i][j]) == abs(grid[i+1][j]) && abs(grid[i+1][j]) == abs(grid[i+2][j])) {
38. **if**(grid[i][j] > 0)
39. grid[i][j] = - grid[i][j];
40. **if**(grid[i+1][j] > 0)
41. grid[i+1][j] = - grid[i+1][j];
42. **if**(grid[i+2][j] > 0)
43. grid[i+2][j] = - grid[i+2][j];
44. }
45. }
47. // 进行清除（可以消除则置为0）并且输出结果
48. **for**(**int** i=0; i<n; i++) {
49. **for**(**int** j=0; j<m; j++) {
50. **if**(grid[i][j] < 0)
51. grid[i][j] = 0;
52. **if**(j != 0)
53. cout << " ";
54. cout << grid[i][j];
55. }
56. cout << endl;
57. }
59. **return** 0;
60. }

计算绝对值的数学函数abs()有可能有二义性，自己编写了计算绝对值的函数；二维数组采用动态化声明（C99等是支持的）。

改进后100分的C++程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54565428) [copy](http://blog.csdn.net/tigerisland45/article/details/54565428)[在CODE上查看代码片](https://code.csdn.net/snippets/2129171)

1. /\* CCF201512-2 消除游戏 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. **inline** **int** myabs(**int** x)
8. {
9. **return** (x >= 0) ? x : -x;
10. }
12. **int** main()
13. {
14. // 输入数据
15. **int** n, m;
16. cin >> n >> m;
18. **int** grid[n][m];
19. **for**(**int** i=0; i<n; i++)
20. **for**(**int** j=0; j<m; j++) {
21. cin >> grid[i][j];
22. }
24. // 进行行标记（可以消除则置为负）
25. **for**(**int** i=0; i<n; i++)
26. **for**(**int** j=0; j<m-2; j++) {
27. **if**(myabs(grid[i][j]) == myabs(grid[i][j+1]) && myabs(grid[i][j+1]) == myabs(grid[i][j+2])) {
28. **if**(grid[i][j] > 0)
29. grid[i][j] = - grid[i][j];
30. **if**(grid[i][j+1] > 0)
31. grid[i][j+1] = - grid[i][j+1];
32. **if**(grid[i][j+2] > 0)
33. grid[i][j+2] = - grid[i][j+2];
34. }
35. }
37. // 进行列标记（可以消除则置为负）
38. **for**(**int** j=0; j<m; j++)
39. **for**(**int** i=0; i<n-2; i++) {
40. **if**(myabs(grid[i][j]) == myabs(grid[i+1][j]) && myabs(grid[i+1][j]) == myabs(grid[i+2][j])) {
41. **if**(grid[i][j] > 0)
42. grid[i][j] = - grid[i][j];
43. **if**(grid[i+1][j] > 0)
44. grid[i+1][j] = - grid[i+1][j];
45. **if**(grid[i+2][j] > 0)
46. grid[i+2][j] = - grid[i+2][j];
47. }
48. }
50. // 进行清除（可以消除则置为0）并且输出结果
51. **for**(**int** i=0; i<n; i++) {
52. **for**(**int** j=0; j<m; j++) {
53. **if**(grid[i][j] < 0)
54. grid[i][j] = 0;
55. **if**(j != 0)
56. cout << " ";
57. cout << grid[i][j];
58. }
59. cout << endl;
60. }
62. **return** 0;
63. }

[**CCF201604-2 俄罗斯方块（100分）**](http://blog.csdn.net/tigerisland45/article/details/54790942)

|  |  |
| --- | --- |
| 试题编号： | 201604-2 |
| 试题名称： | 俄罗斯方块 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　俄罗斯方块是俄罗斯人阿列克谢·帕基特诺夫发明的一款休闲游戏。 　　游戏在一个15行10列的方格图上进行，方格图上的每一个格子可能已经放置了方块，或者没有放置方块。每一轮，都会有一个新的由4个小方块组成的板块从方格图的上方落下，玩家可以操作板块左右移动放到合适的位置，当板块中某一个方块的下边缘与方格图上的方块上边缘重合或者达到下边界时，板块不再移动，如果此时方格图的某一行全放满了方块，则该行被消除并得分。 　　在这个问题中，你需要写一个程序来模拟板块下落，你不需要处理玩家的操作，也不需要处理消行和得分。 　　具体的，给定一个初始的方格图，以及一个板块的形状和它下落的初始位置，你要给出最终的方格图。  **输入格式**  　　输入的前15行包含初始的方格图，每行包含10个数字，相邻的数字用空格分隔。如果一个数字是0，表示对应的方格中没有方块，如果数字是1，则表示初始的时候有方块。输入保证前4行中的数字都是0。 　　输入的第16至第19行包含新加入的板块的形状，每行包含4个数字，组成了板块图案，同样0表示没方块，1表示有方块。输入保证板块的图案中正好包含4个方块，且4个方块是连在一起的（准确的说，4个方块是四连通的，即给定的板块是俄罗斯方块的标准板块）。 　　第20行包含一个1到7之间的整数，表示板块图案最左边开始的时候是在方格图的哪一列中。注意，这里的板块图案指的是16至19行所输入的板块图案，如果板块图案的最左边一列全是0，则它的左边和实际所表示的板块的左边是不一致的（见样例）  **输出格式**  　　输出15行，每行10个数字，相邻的数字之间用一个空格分隔，表示板块下落后的方格图。注意，你不需要处理最终的消行。  **样例输入**  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 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 3  **样例输出**  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 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 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 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 0 0 0 0 |

**问题链接：**[CCF201604试题](http://blog.csdn.net/tigerisland45/article/details/54447017" \t "_blank)。

**问题描述**：编写一个程序来模拟小方块下落的过程。

**问题分析**：这是一个模拟题，模拟俄罗斯方块游戏中，小方块下落的过程。该问题比实际的俄罗斯方块游戏简单多了。关键在于数据表示，然后就是模拟小方块的下落过程。

**程序说明**：

数组board[]用来存储俄罗斯方块游戏的方格。底边另外放一行1，可以简化处理程序的逻辑。

数组block[]用于存小方块。输入时使用该数组。

数组coords[]用于存储小方块中为1元素的坐标。用这个进行模拟处理可以加快速度。

需要注意下标间的变换问题，略去程序处理逻辑的说明。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54790942) [copy](http://blog.csdn.net/tigerisland45/article/details/54790942)[在CODE上查看代码片](https://code.csdn.net/snippets/2152675)

1. /\* CCF201604-2 俄罗斯方块 \*/
3. #include <iostream>
5. **const** **int** ROW = 15;
6. **const** **int** COL = 10;
7. **const** **int** N = 4;
9. **int** board[ROW+1][COL];
10. **int** block[N][N];
11. **struct** {
12. **int** row, col;
13. } coords[N];
15. **using** **namespace** std;
17. **int** main()
18. {
19. **int** row, col;
21. // 输入数据
22. **for**(**int** i=0; i<ROW; i++)
23. **for**(**int** j=0; j<COL; j++)
24. cin >> board[i][j];
25. **for**(**int** i=0; i<N; i++)
26. **for**(**int** j=0; j<N; j++)
27. cin >> block[i][j];
28. cin >> col;
30. // 底边全放1
31. **for**(**int** j=0; j<COL; j++)
32. board[ROW][j] = 1;
34. // 提取小方块坐标
35. **int** k = 0;
36. **for**(**int** i=N-1; i>=0; i--)
37. **for**(**int** j=0; j<N; j++)
38. **if**(block[i][j] == 1) {
39. coords[k].row = i;
40. coords[k].col = j;
41. k++;
42. }
44. // 模拟小方块落下过程
45. row = 1;
46. col--;
47. **bool** checkflag;
48. **for**(;;) {
49. checkflag = **false**;
51. **for**(**int** i=0; i<N; i++)
52. **if**(board[row + coords[i].row][col + coords[i].col] == 1) {
53. checkflag = **true**;
54. **break**;
55. }
57. **if**(checkflag)
58. **break**;
60. row++;
61. }
62. row--;
64. // 合并小方块到方格
65. **for**(**int** i=0; i<N; i++)
66. board[row + coords[i].row][col + coords[i].col] = 1;
68. // 输出结果
69. **for**(**int** i=0; i<ROW; i++) {
70. **for**(**int** j=0; j<COL; j++) {
71. **if**(j != 0)
72. cout << " ";
73. cout << board[i][j];
74. }
75. cout << endl;
76. }
78. **return** 0;
79. }

[**CCF201609-2 火车购票（100分）**](http://blog.csdn.net/tigerisland45/article/details/54798869)

|  |  |
| --- | --- |
| 试题编号： | 201609-2 |
| 试题名称： | 火车购票 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　请实现一个铁路购票系统的简单座位分配算法，来处理一节车厢的座位分配。 　　假设一节车厢有20排、每一排5个座位。为方便起见，我们用1到100来给所有的座位编号，第一排是1到5号，第二排是6到10号，依次类推，第20排是96到100号。 　　购票时，一个人可能购一张或多张票，最多不超过5张。如果这几张票可以安排在同一排编号相邻的座位，则应该安排在编号最小的相邻座位。否则应该安排在编号最小的几个空座位中（不考虑是否相邻）。 　　假设初始时车票全部未被购买，现在给了一些购票指令，请你处理这些指令。  **输入格式**  　　输入的第一行包含一个整数n，表示购票指令的数量。 　　第二行包含n个整数，每个整数p在1到5之间，表示要购入的票数，相邻的两个数之间使用一个空格分隔。  **输出格式**  　　输出n行，每行对应一条指令的处理结果。 　　对于购票指令p，输出p张车票的编号，按从小到大排序。  **样例输入**  4 2 5 4 2  **样例输出**  1 2 6 7 8 9 10 11 12 13 14 3 4  **样例说明**  　　1) 购2张票，得到座位1、2。 　　2) 购5张票，得到座位6至10。 　　3) 购4张票，得到座位11至14。 　　4) 购2张票，得到座位3、4。  **评测用例规模与约定**  　　对于所有评测用例，1 ≤ n ≤ 100，所有购票数量之和不超过100。 |

**问题链接：**[CCF201609试题](http://blog.csdn.net/tigerisland45/article/details/54447023)。

**问题描述**：实现一个铁路购票系统的简单座位分配[**算法**](http://lib.csdn.net/base/datastructure)，来处理一节车厢的座位分配。

**问题分析**：该问题有以下两个要点：

1.火车票分配算法：采用贪心法，题意中已经叙述的十分清楚。

2.票池余票数据表示：有两种表示，一是使用数组，会导致重复搜索；二是采用动态**[数据结构](http://lib.csdn.net/base/datastructure" \t "_blank" \o "算法与数据结构知识库)**，例如map，处理逻辑可能稍微会复杂一些。

**程序说明**：用map存储票池中的余票数据，可以免去重复搜索余票。程序采用边输入数据边处理的方式。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54798869) [copy](http://blog.csdn.net/tigerisland45/article/details/54798869)[在CODE上查看代码片](https://code.csdn.net/snippets/2152798)

1. /\* CCF201609-2 火车购票 \*/
3. #include <iostream>
4. #include <map>
6. **using** **namespace** std;
8. **const** **int** N = 20;
9. **const** **int** NUM = 5;
11. **int** main()
12. {
13. map<**int**,**int**> m;
14. **int** n, v, start, end;
16. // 初始化车厢票源
17. **for**(**int** i=1; i<=N; i++)
18. m[i] = NUM;
20. // 输入数据，分配车票，输出结果
21. cin >> n;
22. **for**(**int** i=1; i<=n; i++) {
23. // 输入数据：购买火车票张数
24. cin >> v;
26. // 分配车票（同一排分配）并且输出结果
27. **for**(map<**int**,**int**>::iterator it=m.begin(); it!=m.end(); it++) {
28. **if**(it->second >= v) {
29. // 输出结果
30. start = (it->first - 1) \* NUM + 1 + (NUM - it->second);
31. end = start + v - 1;
32. **for**(**int** j=start; j<=end; j++) {
33. **if**(j != start)
34. cout << " ";
35. cout << j;
36. }
37. cout << endl;
39. // 对于票数已经分配完毕的排，删除
40. **if**(it->second == v)
41. m.erase(it);
42. **else**
43. it->second = it->second - v;
45. v = 0;
46. **break**;
47. }
48. }
50. // 分配车票（多排分配）并且输出结果
51. **bool** nofirstflag = **false**;
52. **while**(v > 0) {
53. **for**(map<**int**,**int**>::iterator it=m.begin(); it!=m.end(); it++) {
54. **if**(it->second >= v) {
55. // 输出结果
56. start = (it->first - 1) \* NUM + 1 + (NUM - it->second);
57. end = start + v - 1;
58. **for**(**int** j=start; j<=end; j++) {
59. **if**(nofirstflag)
60. cout << " ";
61. cout << j;
62. nofirstflag = **true**;
63. }
65. // 对于票数已经分配完毕的排，删除
66. **if**(it->second == v)
67. m.erase(it);
68. **else**
69. it->second = it->second - v;
71. v = 0;
72. **break**;
73. } **else** {
74. // 输出结果
75. start = (it->first - 1) \* NUM + 1 + (NUM - it->second);
76. end = start + it->second - 1;
77. **for**(**int** j=start; j<=end; j++) {
78. **if**(nofirstflag)
79. cout << " ";
80. cout << j;
81. nofirstflag = **true**;
82. }
84. v = v - it->second;
86. // 对于票数已经分配完毕的排，删除
87. m.erase(it);
88. }
89. }
90. cout << endl;
91. }
92. }
94. **return** 0;
95. }

[**CCF201612-2 工资计算（100分）**](http://blog.csdn.net/tigerisland45/article/details/54799557)

|  |  |
| --- | --- |
| 试题编号： | 201612-2 |
| 试题名称： | 工资计算 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　小明的公司每个月给小明发工资，而小明拿到的工资为交完个人所得税之后的工资。假设他一个月的税前工资（扣除五险一金后、未扣税前的工资）为S元，则他应交的个人所得税按如下公式计算： 　　1） 个人所得税起征点为3500元，若S不超过3500，则不交税，3500元以上的部分才计算个人所得税，令A=S-3500元； 　　2） A中不超过1500元的部分，税率3%； 　　3） A中超过1500元未超过4500元的部分，税率10%； 　　4） A中超过4500元未超过9000元的部分，税率20%； 　　5） A中超过9000元未超过35000元的部分，税率25%； 　　6） A中超过35000元未超过55000元的部分，税率30%； 　　7） A中超过55000元未超过80000元的部分，税率35%； 　　8） A中超过80000元的部分，税率45%； 　　例如，如果小明的税前工资为10000元，则A=10000-3500=6500元，其中不超过1500元部分应缴税1500×3%=45元，超过1500元不超过4500元部分应缴税(4500-1500)×10%=300元，超过4500元部分应缴税(6500-4500)×20%=400元。总共缴税745元，税后所得为9255元。 　　已知小明这个月税后所得为T元，请问他的税前工资S是多少元。  **输入格式**  　　输入的第一行包含一个整数T，表示小明的税后所得。所有评测数据保证小明的税前工资为一个整百的数。  **输出格式**  　　输出一个整数S，表示小明的税前工资。  **样例输入**  9255  **样例输出**  10000  **评测用例规模与约定**  　　对于所有评测用例，1 ≤ T ≤ 100000。 |

**问题链接：**[CCF201612试题](http://blog.csdn.net/tigerisland45/article/details/54456551)。

**问题描述**：工资收入是分段计税的，输入税后工资，求税前工资。（详见原问题，点击上面的链接）。

**问题分析**：根据税前工资算税后工资相对要容易许多，本题正好相反。

从编程方法上来说，一种是将工资段和税率写到程序逻辑中。这种做法修改程序比较难，逻辑也比较复杂。另外一种是查表法，通过查表来计算最后的结果。查表法的优点在于，单有关规定改变时，只需要调整表格，而不需要改变程序逻辑。

该问题原始数据可以建两个表格，一是工资收入段表，二是税率表。根据这两个表可以算出收入范围表，即由实际收入得到最高税率是哪一档的表。进一步的计算就变得简单许多。

**程序说明**：*商业应用开发中，是不允许出现任何金额计算错误的，误差也是不允许的。导致金额误差有多种原因，其中，使用浮点变量是常见的一种，一点要避免使用浮点类型变量，确保金额相关的计算准确无误*。

提交后得100分的**[C++语言](http://lib.csdn.net/base/c" \t "_blank" \o "C语言知识库)**程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54799557) [copy](http://blog.csdn.net/tigerisland45/article/details/54799557)[在CODE上查看代码片](https://code.csdn.net/snippets/2152984)

1. /\* CCF201612-2 工资计算 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. //#define DEBUG
9. **int** salaryrange[] = {3500, 3500+1500, 3500+4500, 3500+9000, 3500+35000, 3500+55000, 3500+80000 };
10. **int** taxrate[] = {3, 10, 20, 25, 30, 35, 45};
11. **const** **int** SIZE = **sizeof**(salaryrange) / **sizeof**(**int**);
13. **int** range[SIZE];
15. **int** main()
16. {
17. **int** t, s;
19. // 计算各种收入范围
20. range[0] = salaryrange[0];
21. **for**(**int** i=1; i<SIZE; i++) {
22. range[i] = range[i-1] + (salaryrange[i] - salaryrange[i-1])
23. - (salaryrange[i] - salaryrange[i-1]) \* taxrate[i-1] / 100;
24. }
26. #ifdef DEBUG
27. **for**(**int** i=0; i<SIZE; i++)
28. cout << range[i] << " ";
29. cout << endl;
30. #endif
32. // 输入数据：
33. cin >> t;
35. // 计算收入范围
36. **int** i;
37. **for**(i=0; i<SIZE; i++)
38. **if**(t <= range[i])
39. **break**;
41. // 计算税前工资
42. **if**(i == 0)
43. s = t;
44. **else** {
45. s = salaryrange[i-1] + (t - range[i-1]) \* 100 / (100 - taxrate[i-1]);
46. }
48. // 输出结果
49. cout << s << endl;
51. **return** 0;
52. }

.