**CCF 2013-2016年第三题题解及源码**

[**CCF201312-3 最大的矩形（100分）**](http://blog.csdn.net/tigerisland45/article/details/54834094)

|  |  |
| --- | --- |
| 试题编号： | 201312-3 |
| 试题名称： | 最大的矩形 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在横轴上放了n个相邻的矩形，每个矩形的宽度是1，而第i（1 ≤ i ≤ n）个矩形的高度是hi。这n个矩形构成了一个直方图。例如，下图中六个矩形的高度就分别是3, 1, 6, 5, 2, 3。  　　请找出能放在给定直方图里面积最大的矩形，它的边要与坐标轴平行。对于上面给出的例子，最大矩形如下图所示的阴影部分，面积是10。  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  **输入格式**  　　第一行包含一个整数n，即矩形的数量(1 ≤ n ≤ 1000)。 　　第二行包含n 个整数h1, h2, … , hn，相邻的数之间由空格分隔。(1 ≤ hi ≤ 10000)。hi是第i个矩形的高度。  **输出格式**  　　输出一行，包含一个整数，即给定直方图内的最大矩形的面积。  **样例输入**  6 3 1 6 5 2 3  **样例输出**  10 |

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

**问题描述**：首先输入正整数n，接着输入n个正整数表示直方图的一个高度，计算这些直方图中的最大矩形面积。（详见原问题，点击上面的链接）。

**问题分析**：解决这个问题，一种是用暴力法（枚举法）来解决，任何一个矩形必然始于第i个直方图，终止于第j块直方图（i<=j），从所有这些面积中找出最大矩形面积即可；另外一种办法是对这n个数只看一遍，就算出最大矩形面积，其计算复杂度为O(n)，相比较而言，[**算法**](http://lib.csdn.net/base/datastructure)要复杂一些，还需要付出空间的代价。

**程序说明**：本程序采用暴力法实现，数据用数组来存储。另外一种数据存储方法是使用STL的包装类vector，可以实现动态数据存储，不受限于数据的多少。

**参见：**[CCF201312-3 最大的矩形（解法二）（100分）](http://blog.csdn.net/tigerisland45/article/details/54838099)。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54834094) [copy](http://blog.csdn.net/tigerisland45/article/details/54834094)

1. /\* CCF201312-3 最大的矩形 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. **const** **int** N = 1000;
8. **int** h[N];
10. **int** main()
11. {
12. **int** n, ans, height, area;
14. // 输入数据
15. cin >> n;
16. **for**(**int** i=0; i<n; i++)
17. cin >> h[i];
19. // 计算最大矩形面积：暴力法（枚举法）
20. ans = 0;
21. **for**(**int** i=0; i<n; i++) {
22. height = h[i];
23. **for**(**int** j=i; j<n; j++) {
24. **if**(h[j] < height)
25. height = h[j];
26. area = (j - i + 1) \* height;
27. **if**(area > ans)
28. ans = area;
29. }
30. }
32. // 输出结果
33. cout << ans << endl;
35. **return** 0;
36. }

[**CCF201312-3 最大的矩形（解法二）（100分）**](http://blog.csdn.net/tigerisland45/article/details/54838099)

|  |  |
| --- | --- |
| 试题编号： | 201312-3 |
| 试题名称： | 最大的矩形 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在横轴上放了n个相邻的矩形，每个矩形的宽度是1，而第i（1 ≤ i ≤ n）个矩形的高度是hi。这n个矩形构成了一个直方图。例如，下图中六个矩形的高度就分别是3, 1, 6, 5, 2, 3。  　　请找出能放在给定直方图里面积最大的矩形，它的边要与坐标轴平行。对于上面给出的例子，最大矩形如下图所示的阴影部分，面积是10。  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  **输入格式**  　　第一行包含一个整数n，即矩形的数量(1 ≤ n ≤ 1000)。 　　第二行包含n 个整数h1, h2, … , hn，相邻的数之间由空格分隔。(1 ≤ hi ≤ 10000)。hi是第i个矩形的高度。  **输出格式**  　　输出一行，包含一个整数，即给定直方图内的最大矩形的面积。  **样例输入**  6 3 1 6 5 2 3  **样例输出**  10 |

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

**问题描述**：首先输入正整数n，接着输入n个正整数表示直方图的一个高度，计算这些直方图中的最大矩形面积。

**问题分析**：解决这个问题，一种是用暴力法（枚举法）来解决，任何一个矩形必然始于第i个直方图，终止于第j块直方图（i<=j），从所有这些面积中找出最大矩形面积即可；另外一种办法是对这n个数只看一遍，就算出最大矩形面积，其计算复杂度为O(n)，相比较而言，[**算法**](http://lib.csdn.net/base/datastructure)要复杂一些，还需要付出空间的代价。仔细观察这两个[**算法**](http://lib.csdn.net/base/datastructure)有关的程序，会发现其实二者之间算法复杂度的差异很小。

**程序说明**：本程序采用后一种方法实现。基本思想是先找到一个逐步递增的面积，即如果Hi<Ｈi+1则最大面积是逐步递增的。这个过程中，将这些Hi放入堆栈中，直到不满足Hi<Ｈi+1为止。这个时候，最大的面积可能是最右边是Hi，由若干块（也可能只有１块）拼成的，从中获得一个最大的面积。出现面积非递增时，则把堆栈中比当前高的直方图弹出，重复上述过程，需要说明的是这不影响高的直方图与其右边连成一片。还有一点就是，所有的直方图的高度Hi>=1，这是一个前提，如果有的直方图高度为0，则这个算法需要另外设计。

**参见：**[CCF201312-3 最大的矩形（100分）](http://blog.csdn.net/tigerisland45/article/details/54834094)。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54838099) [copy](http://blog.csdn.net/tigerisland45/article/details/54838099)

1. /\* CCF201312-3 最大的矩形 \*/
3. #include <iostream>
4. #include <stack>
6. **using** **namespace** std;
8. **const** **int** N = 1000;
9. **int** h[N+1];
11. **int** main()
12. {
13. **int** n, ans=0, area, temp;
15. // 输入数据
16. cin >> n;
17. **for**(**int** i=0; i<n; i++)
18. cin >> h[i];
19. h[n] = 0;
21. // 计算最大矩形面积
22. stack<**int**> s;
23. **for**(**int** i=0; i<=n; i++) {
24. **if** (s.empty() || h[s.top()] < h[i])
25. s.push(i);
26. **else** {
27. temp = s.top();
28. s.pop();            //弹出
29. area = h[temp] \* (s.empty() ? i : i - s.top() - 1);
30. **if** (area > ans)
31. ans = area;
32. --i;
33. }
34. }
36. // 输出结果
37. cout << ans << endl;
39. **return** 0;
40. }

[**CCF201403-3 命令行选项（100分）**](http://blog.csdn.net/tigerisland45/article/details/54974448)

|  |  |
| --- | --- |
| 试题编号： | 201403-3 |
| 试题名称： | 命令行选项 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　请你写一个命令行分析程序,用以分析给定的命令行里包含哪些选项。每个命令行由若干个字符串组成,它们之间恰好由一个空格分隔。这些字符串中的第一个为该命令行工具的名字,由小写字母组成,你的程序不用对它进行处理。在工具名字之后可能会包含若干选项,然后可能会包含一 些不是选项的参数。 　　选项有两类:带参数的选项和不带参数的选项。一个合法的无参数选项的形式是一个减号后面跟单个小写字母,如"-a" 或"-b"。而带参数选项则由两个由空格分隔的字符串构成,前者的格式要求与无参数选项相同,后者则是该选项的参数,是由小写字母,数字和减号组成的非空字符串。 　　该命令行工具的作者提供给你一个格式字符串以指定他的命令行工具需要接受哪些选项。这个字符串由若干小写字母和冒号组成,其中的每个小写字母表示一个该程序接受的选项。如果该小写字母后面紧跟了一个冒号,它就表示一个带参数的选项,否则则为不带参数的选项。例如, "ab:m:" 表示该程序接受三种选项,即"-a"(不带参数),"-b"(带参数), 以及"-m"(带参数)。 　　命令行工具的作者准备了若干条命令行用以测试你的程序。对于每个命令行,你的工具应当一直向后分析。当你的工具遇到某个字符串既不是合法的选项,又不是某个合法选项的参数时,分析就停止。命令行剩余的未分析部分不构成该命令的选项,因此你的程序应当忽略它们。  **输入格式**  　　输入的第一行是一个格式字符串,它至少包含一个字符,且长度不超过 52。格式字符串只包含小写字母和冒号,保证每个小写字母至多出现一次,不会有两个相邻的冒号,也不会以冒号开头。 　　输入的第二行是一个正整数 N(1 ≤ N ≤ 20),表示你需要处理的命令行的个数。 　　接下来有 N 行,每行是一个待处理的命令行,它包括不超过 256 个字符。该命令行一定是若干个由单个空格分隔的字符串构成,每个字符串里只包含小写字母,数字和减号。  **输出格式**  　　输出有 N 行。其中第 i 行以"Case i:" 开始,然后应当有恰好一个空格,然后应当按照字母升序输出该命令行中用到的所有选项的名称,对于带参数的选项,在输出它的名称之后还要输出它的参数。如果一个选项在命令行中出现了多次,只输出一次。如果一个带参数的选项在命令行中出 现了多次,只输出最后一次出现时所带的参数。  **样例输入**  albw:x 4 ls -a -l -a documents -b ls ls -w 10 -x -w 15 ls -a -b -c -d -e -l  **样例输出**  Case 1: -a -l Case 2: Case 3: -w 15 -x Case 4: -a -b |

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

**问题描述**：（参见上文）。

**问题分析**：这是[**操作系统**](http://lib.csdn.net/base/operatingsystem)命令选项处理问题，也是一个字符串处理的问题。

**程序说明**：因为C++的cin输入字符串时，无法输入空串，而C的库函数中已经不建议使用函数gets()，所以自己编写一个读入一行的函数mygetline()，该函数可以输入空行。使用[**C语言**](http://lib.csdn.net/base/c)的库函数strtok()来切割输入的命令行（空格隔开），然后将切割的结果放入字符串向量vector<string>类型的变量中，再用map<string,string>变量对各个选项进行排序，这种做法是比较有效的。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54974448) [copy](http://blog.csdn.net/tigerisland45/article/details/54974448)

1. /\* CCF201403-3 命令行选项 \*/
3. #include <iostream>
4. #include <cstring>
5. #include <string>
6. #include <vector>
7. #include <map>
9. **using** **namespace** std;
11. **const** **int** N = 256;
13. **char** delimiter[] = " ";
15. **void** spilt(vector<string>& v, **char** \*s, **char** \*t)
16. {
17. **char** \*sp;
19. sp = strtok(s, t);
20. **while**(sp) {
21. v.push\_back(sp);
22. sp = strtok(NULL, t);
23. }
24. }
26. **void** mygetline(**char** \*pc)
27. {
28. **char** c;
30. **while**((c=getchar()) != '\n' && c !=EOF)
31. \*pc++ = c;
32. \*pc = '\0';
33. }
35. **int** main()
36. {
37. string format;
38. **char** s[N+1];
39. **int** n;
41. // 输入数据：格式和整数n
42. cin >> format >> n;
43. getchar();
45. // 输入n行命令行进行处理
46. **for**(**int** i=1; i<=n; i++) {
47. vector<string> sv;
48. map<string, string> m;
50. // 输入命令行
51. mygetline(s);
53. // 切分命令行：命令和各个参数分开
54. spilt(sv, s, delimiter);
56. // 处理各个参数，放入map变量ｍ中
57. **for**(**int** j=1; j<(**int**)sv.size(); j++) {
58. **if**(sv[j].size() == 2 && sv[j][0] == '-') { // 判断是否是选项，选项则处理
59. // 选项在格式中未找到则出错结束
60. **int** pos = format.find(sv[j][1]);
61. **if**(pos == -1)
62. **break**;
64. // 选项未出现过则添加
65. **if**(m.find(sv[j]) == m.end())
66. m[sv[j]] = "";
68. // 更新参数：后出现优先
69. **if**(format[pos+1] == ':' && j+1 < (**int**)sv.size()) {
70. m[sv[j]] = sv[j+1];
71. j++;
72. }
73. } **else**
74. **break**;
75. }
77. // 输出结果
78. cout << "Case " << i << ":";
79. **for**(map<string,string>::iterator iter=m.begin(); iter!=m.end(); iter++) {
80. cout << " " << iter->first;
81. **if**(iter->second != "")
82. cout << " " << iter->second;
83. }
84. cout << endl;
85. }
87. **return** 0;
88. }

[**CCF201409-3 字符串匹配（100分）**](http://blog.csdn.net/tigerisland45/article/details/54920717)

|  |  |
| --- | --- |
| 试题编号： | 201409-3 |
| 试题名称： | 字符串匹配 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给出一个字符串和多行文字，在这些文字中找到字符串出现的那些行。你的程序还需支持大小写敏感选项：当选项打开时，表示同一个字母的大写和小写看作不同的字符；当选项关闭时，表示同一个字母的大写和小写看作相同的字符。  **输入格式**  　　输入的第一行包含一个字符串S，由大小写英文字母组成。 　　第二行包含一个数字，表示大小写敏感的选项，当数字为0时表示大小写不敏感，当数字为1时表示大小写敏感。 　　第三行包含一个整数n，表示给出的文字的行数。 　　接下来n行，每行包含一个字符串，字符串由大小写英文字母组成，不含空格和其他字符。  **输出格式**  　　输出多行，每行包含一个字符串，按出现的顺序依次给出那些包含了字符串S的行。  **样例输入**  Hello 1 5 HelloWorld HiHiHelloHiHi GrepIsAGreatTool HELLO HELLOisNOTHello  **样例输出**  HelloWorld HiHiHelloHiHi HELLOisNOTHello  **样例说明**  　　在上面的样例中，第四个字符串虽然也是Hello，但是大小写不正确。如果将输入的第二行改为0，则第四个字符串应该输出。  **评测用例规模与约定**  　　1<=n<=100，每个字符串的长度不超过100。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个简单的字符串处理问题，关键是对有关字符串处理函数是否熟悉。

**程序说明**：有关字符串的处理，可以用[**C语言**](http://lib.csdn.net/base/c)的函数实现，也可用C++的string变量实现。

**相关链接：**[CCF201409-3 字符串匹配（解法二）（100分）](http://blog.csdn.net/tigerisland45/article/details/55307733)。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54920717) [copy](http://blog.csdn.net/tigerisland45/article/details/54920717)

1. /\* CCF201409-3 字符串匹配 \*/
3. #include <iostream>
4. #include <cstring>
6. **using** **namespace** std;
8. **const** **int** N = 100;
10. **void** mystrlwr(**char** \*ps)
11. {
12. **while**(\*ps) {
13. **if**('A' <= \*ps && \*ps <= 'Z')
14. \*ps += 'a'-'A';
15. ps++;
16. }
17. }
19. **int** main()
20. {
21. **char** key[N+1], s[N+1], lowerkey[N+1], lowers[N+1];
22. **int** option, n;
24. // 输入数据
25. cin >> key >> option >> n;
27. // 获得key的小写字符串，放在变量lowerkey中
28. strcpy(lowerkey, key);
29. mystrlwr(lowerkey);
31. // 循环处理
32. **for**(**int** i=1; i<=n; i++) {
33. cin >> s;
35. **if**(option == 0) {   // 大小写无关
36. strcpy(lowers, s);
37. mystrlwr(lowers);
39. **if**(strstr(lowers, lowerkey))
40. cout << s << endl;
41. } **else** { // option = 1，大小写有关
42. **if**(strstr(s, key))
43. cout << s << endl;
44. }
45. }
46. }

在gcc环境中，不支持函数strlwr()，所以需要自己写一个函数mystrlwr()，在Dev-C++环境中，是支持函数strlwr()的，不需要自己写该函数，上述的函数mystrlwr()可以直接写为strlwr()。

以下这个程序提交后只得了50分，希望哪位帮助改进一下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54920717) [copy](http://blog.csdn.net/tigerisland45/article/details/54920717)

1. /\* CCF201409-3 字符串匹配 \*/
3. #include <iostream>
4. #include <cctype>
5. #include <string>
7. **using** **namespace** std;
9. **int** main()
10. {
11. string key, s, kt, st;
12. **int** option, n;
14. // 输入数据
15. cin >> key >> option >> n;
17. // 获得key的小写字符串，放在变量kt
18. kt = s;
19. **for**(**int** i=0; i<(**int**)kt.size(); i++)
20. **if**(isupper(kt[i]))
21. kt[i] = tolower(kt[i]);
23. // 循环处理
24. **for**(**int** i=1; i<=n; i++) {
25. cin >> s;
27. **if**(option == 0) {   // 大小写无关
28. st = s;
29. **for**(**int** i=0; i<(**int**)st.size(); i++)
30. **if**(isupper(st[i]))
31. st[i] = tolower(st[i]);
33. **int** pos = st.find(kt);
34. **if**(pos >= 0)
35. cout << s << endl;
36. } **else** { // option = 1，大小写有关
37. **int** pos = s.find(key);
38. **if**(pos >= 0)
39. cout << s << endl;
40. }
41. }
43. **return** 0;
44. }

[**CCF201409-3 字符串匹配（解法二）（100分）**](http://blog.csdn.net/tigerisland45/article/details/55307733)

|  |  |
| --- | --- |
| 试题编号： | 201409-3 |
| 试题名称： | 字符串匹配 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　给出一个字符串和多行文字，在这些文字中找到字符串出现的那些行。你的程序还需支持大小写敏感选项：当选项打开时，表示同一个字母的大写和小写看作不同的字符；当选项关闭时，表示同一个字母的大写和小写看作相同的字符。  **输入格式**  　　输入的第一行包含一个字符串S，由大小写英文字母组成。 　　第二行包含一个数字，表示大小写敏感的选项，当数字为0时表示大小写不敏感，当数字为1时表示大小写敏感。 　　第三行包含一个整数n，表示给出的文字的行数。 　　接下来n行，每行包含一个字符串，字符串由大小写英文字母组成，不含空格和其他字符。  **输出格式**  　　输出多行，每行包含一个字符串，按出现的顺序依次给出那些包含了字符串S的行。  **样例输入**  Hello 1 5 HelloWorld HiHiHelloHiHi GrepIsAGreatTool HELLO HELLOisNOTHello  **样例输出**  HelloWorld HiHiHelloHiHi HELLOisNOTHello  **样例说明**  　　在上面的样例中，第四个字符串虽然也是Hello，但是大小写不正确。如果将输入的第二行改为0，则第四个字符串应该输出。  **评测用例规模与约定**  　　1<=n<=100，每个字符串的长度不超过100。 |

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

**问题描述**：参见上文。

**问题分析**：这是一个简单的字符串处理问题，关键是对有关字符串处理函数是否熟悉。这里采用KMP[**算法**](http://lib.csdn.net/base/datastructure)实现。

**程序说明**：有关字符串的处理，这里采用用**[C语言](http://lib.csdn.net/base/c" \o "C语言知识库" \t "_blank)**的函数实现。

**相关链接：**[CCF201409-3 字符串匹配（100分）](http://blog.csdn.net/tigerisland45/article/details/54920717)。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/55307733) [copy](http://blog.csdn.net/tigerisland45/article/details/55307733)

1. /\* CCF201409-3 字符串匹配 \*/
3. #include <iostream>
4. #include <cstring>
6. **using** **namespace** std;
8. // KMP算法：函数getnext()和kmp()
9. **void** getnext(**int** next[], **char** t[])
10. {
11. **int** i=0, j=-1;
12. next[0] = -1;
13. **while**(t[i]!='\0')
14. **if**(j == -1 || t[i] == t[j])
15. next[++i] = ++j;
16. **else**
17. j = next[j];
18. }
20. **int** kmp(**int** next[], **char** s[], **char** t[])
21. {
22. **int** i=0, j=0, len1=strlen(s), len2=strlen(t);
24. **while**(i < len1 && j < len2)
25. {
26. **if**(j == -1 || s[i] == t[j])
27. ++i, ++j;
28. **else**
29. j = next[j];
30. }
32. **if**(j>=len2)
33. **return** i-len2+1;
34. **else**
35. **return** -1;
36. }
38. **const** **int** MAXN = 100;
39. **int** next[MAXN+1];
40. **char** s[MAXN+1], s2[MAXN+1], t[MAXN+1];

43. **int** main()
44. {
45. **int** option, m;
47. cin >> t >> option;
49. **if**(option == 0)
50. **for**(**int** i=0; t[i]; i++)
51. t[i] = tolower(t[i]);
53. getnext(next, t);
55. cin >> m;
56. **while**(m--) {
57. cin >> s;
58. strcpy(s2, s);
60. **if**(option == 0)
61. **for**(**int** i=0; s2[i]; i++)
62. s2[i] = tolower(s2[i]);
64. **if**(kmp(next, s2, t) != -1)
65. cout << s << endl;
66. }
68. **return** 0;
69. }

[**CCF201412-3 集合竞价（100分）**](http://blog.csdn.net/tigerisland45/article/details/54928820)

|  |  |
| --- | --- |
| 试题编号： | 201412-3 |
| 试题名称： | 集合竞价 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　某股票交易所请你编写一个程序，根据开盘前客户提交的订单来确定某特定股票的开盘价和开盘成交量。 　　该程序的输入由很多行构成，每一行为一条记录，记录可能有以下几种： 　　1. buy p s 表示一个购买股票的买单，每手出价为p，购买股数为s。 　　2. sell p s 表示一个出售股票的卖单，每手出价为p，出售股数为s。 　　3. cancel i表示撤销第i行的记录。 　　如果开盘价为p0，则系统可以将所有出价至少为p0的买单和所有出价至多为p0的卖单进行匹配。因此，此时的开盘成交量为出价至少为p0的买单的总股数和所有出价至多为p0的卖单的总股数之间的较小值。 　　你的程序需要确定一个开盘价，使得开盘成交量尽可能地大。如果有多个符合条件的开盘价，你的程序应当输出最高的那一个。  **输入格式**  　　输入数据有任意多行，每一行是一条记录。保证输入合法。股数为不超过108的正整数，出价为精确到恰好小数点后两位的正实数，且不超过10000.00。  **输出格式**  　　你需要输出一行，包含两个数，以一个空格分隔。第一个数是开盘价，第二个是此开盘价下的成交量。开盘价需要精确到小数点后恰好两位。  **样例输入**  buy 9.25 100 buy 8.88 175 sell 9.00 1000 buy 9.00 400 sell 8.92 400 cancel 1 buy 100.00 50  **样例输出**  9.00 450  **评测用例规模与约定**  　　对于100%的数据，输入的行数不超过5000。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个竞价匹配的问题。数据可以存储在结构数组中，但未必是好的方案。使用两个优先队列，一个用于存储购入订单，按价格从大到小排列；另外一个用于存储卖出订单，按价格从小到大排列；然后进行价格的匹配处理。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54928820) [copy](http://blog.csdn.net/tigerisland45/article/details/54928820)

1. /\* CCF201412-3 集合竞价 \*/
3. #include <iostream>
4. #include <queue>
5. #include <cstring>
6. #include <cstdio>
8. **using** **namespace** std;
10. **const** **int** N = 5000;
12. **struct** trading {
13. **int** orderno;
14. **char** t;
15. **float** price;
16. **long** **long** quantity;
17. **bool** operator < (**const** trading& n) **const** {
18. **if**(t == 's')
19. **return** price > n.price;
20. **else** // t == 'b'
21. **return** price < n.price;
22. }
23. };
25. **bool** cancelflag[N+1];
27. **int** main()
28. {
29. trading t;
30. priority\_queue<trading> sell, buy;
31. string strading;
33. // 变量初始化
34. memset(cancelflag, **false**, **sizeof**(cancelflag));
36. // 输入数据
37. **int** no = 0, tno;
39. **while**(cin >> strading) {
40. **if**(strading[0] == 'c') {
41. // 设置交易号
42. no++;
44. // 输入取消的交易号
45. cin >> tno;
47. // 设置取消标志
48. cancelflag[tno] = **true**;
49. } **else** **if**(strading[0] == 'b' || strading[0] == 's') {
50. // 设置交易号
51. t.orderno = ++no;
53. // 输入交易价格和数量
54. cin >> t.price >> t.quantity;
56. // 将交易分别放入买入和卖出的优先队列
57. **if**(strading[0] == 'b') {
58. t.t = strading[0];
59. buy.push(t);
60. } **else** {    // t.trading[0] == 's'
61. t.t = strading[0];
62. sell.push(t);
63. }
64. } **else**
65. **break**;
66. }
68. // 集合竞价处理
69. t.price = 0;
70. t.quantity = 0;
71. trading b, s;
72. **for**(;;) {
73. // 清除被取消的订单（同时将队头放在b和s中）
74. **while**(!buy.empty()) {
75. b = buy.top();
76. **if**(cancelflag[b.orderno])
77. buy.pop();
78. **else**
79. **break**;
80. }
81. **while**(!sell.empty()) {
82. s = sell.top();
83. **if**(cancelflag[s.orderno])
84. sell.pop();
85. **else**
86. **break**;
87. }
89. // 买卖队列只要有一个为空，则处理结束
90. **if**(buy.empty() || sell.empty())
91. **break**;
93. // 集合竞价处理
94. **if**(b.price >= s.price) {
95. t.quantity += min(b.quantity, s.quantity);
96. t.price = b.price;
98. **if**(b.quantity == s.quantity) {
99. buy.pop();
100. sell.pop();
101. } **else** **if**(b.quantity > s.quantity) {
102. b.quantity -= s.quantity;
103. buy.pop();
104. buy.push(b);
105. sell.pop();
106. } **else** {    // b.quantity < s.quantity
107. buy.pop();
108. s.quantity -= b.quantity;
109. sell.pop();
110. sell.push(s);
111. }
112. } **else**
113. **break**;
114. }
116. // 输出结果
117. printf("%.2f", t.price);
118. cout << " " << t.quantity << endl;
120. **return** 0;
121. }
123. /\*
124. 测试样例：
126. sell 8.88 100
127. sell 8.88 175
128. sell 9.00 400
129. buy 8.88 400
130. cancel 1
131. sell 100.00 50
132. 8.88 175
134. buy 9.25 100
135. buy 8.88 175
136. buy 9.00 400
137. sell 8.88 400
138. cancel 1
139. buy 100.00 50
140. 9.00 400
142. buy 9.25 100
143. buy 8.88 175
144. buy 9.00 400
145. sell 8.79 1501
146. cancel 1
147. cancel 2
148. 9.00 400
150. buy 9.25 110
151. buy 8.88 300
152. buy 18.88 200
153. sell 8.88 201
154. sell 9.25 100
155. 9.25 301
156. \*/

[**CCF201503-3 节日（100分）**](http://blog.csdn.net/tigerisland45/article/details/54850565)

|  |  |
| --- | --- |
| 试题编号： | 201503-3 |
| 试题名称： | 节日 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有一类节日的日期并不是固定的，而是以“a月的第b个星期c”的形式定下来的，比如说母亲节就定为每年的五月的第二个星期日。 　　现在，给你a，b，c和y1, y2(1850 ≤ y1, y2 ≤ 2050)，希望你输出从公元y1年到公元y2年间的每年的a月的第b个星期c的日期。 　　提示：关于闰年的规则：年份是400的整数倍时是闰年，否则年份是4的倍数并且不是100的倍数时是闰年，其他年份都不是闰年。例如1900年就不是闰年，而2000年是闰年。 　　为了方便你推算，已知1850年1月1日是星期二。  **输入格式**  　　输入包含恰好一行，有五个整数a, b, c, y1, y2。其中c=1, 2, ……, 6, 7分别表示星期一、二、……、六、日。  **输出格式**  　　对于y1和y2之间的每一个年份，包括y1和y2，按照年份从小到大的顺序输出一行。 　　如果该年的a月第b个星期c确实存在，则以"yyyy/mm/dd"的格式输出，即输出四位数的年份，两位数的月份，两位数的日期，中间用斜杠“/”分隔，位数不足时前补零。 　　如果该年的a月第b个星期c并不存在，则输出"none"（不包含双引号)。  **样例输入**  5 2 7 2014 2015  **样例输出**  2014/05/11 2015/05/10  **评测用例规模与约定**  　　所有评测用例都满足：1 ≤ a ≤ 12，1 ≤ b ≤ 5，1 ≤ c ≤ 7，1850 ≤ y1, y2 ≤ 2050。 |

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

**问题描述**：计算“a月的第b个星期c”形式的日期。（详见原问题，点击上面的链接）。

输入包含恰好一行，有五个整数a, b, c, y1, y2。其中c=1, 2, ……, 6, 7分别表示星期一、二、……、六、日。

对于y1和y2之间的每一个年份，包括y1和y2，按照年份从小到大的顺序输出一行。  
如果该年的a月第b个星期c确实存在，则以"yyyy/mm/dd"的格式输出，即输出四位数的年份，两位数的月份，两位数的日期，中间用斜杠“/”分隔，位数不足时前补零。  
如果该年的a月第b个星期c并不存在，则输出"none"（不包含双引号)。

**问题分析**：需要注意闰年问题，其他都是计算问题。

**程序说明**：（略）

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54850565) [copy](http://blog.csdn.net/tigerisland45/article/details/54850565)

1. /\* CCF201503-3 节日 \*/
3. #include <iostream>
5. **using** **namespace** std;
7. // 月份天数
8. **int** monthdays[2][13] = {{0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31},
9. {0, 31, 29, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31}};
11. // 闰年计算函数
12. **int** leapyear(**int** year) {
13. **if**((year%4 == 0 && year%100 != 0) || year%400 == 0)
14. **return** 1;
15. **return** 0;
16. }
18. **int** main()
19. {
20. **int** a, b, c, y1, y2, weekd, d, y;
22. // 输入数据
23. cin >> a >> b >> c >> y1 >> y2;
25. // 计算1850年到起始年的天数
26. **int** days = 0;
27. **for**(**int** i=1850; i<y1; i++)
28. **if**(leapyear(i) == 1)
29. days += 366;
30. **else**
31. days += 365;
33. // 计算年月日并且输出
34. **for**(**int** i=y1; i<=y2; i++) {
35. // 计算闰年：０为非闰年，1为闰年
36. y = leapyear(i);
38. // 计算从1850年开始，到i年a月1日的天数
39. **int** days2 = days;
40. **for**(**int** i=1; i<a; i++)
41. days2 += monthdays[y][i];
43. // 计算i年a月1日的前一天为星期几
44. weekd = 1 + days2 % 7;
46. // 计算i年的a月第b个星期c是当月的几号
47. d = (b-1) \* 7 + ((weekd >= c) ? (c + 7 - weekd) : (c - weekd));
49. // 输出结果
50. **if**(d > monthdays[y][a])
51. cout << "none" << endl;
52. **else** {
53. cout << i << "/";
54. **if**(a<10)
55. cout << "0";
56. cout << a << "/";
57. **if**(d<10)
58. cout << "0";
59. cout << d << endl;
60. }
62. // 为计算下一年做准备：计算从1850年到下一年开始的天数
63. **if**(leapyear(i) == 1)
64. days += 366;
65. **else**
66. days += 365;
67. }

70. **return** 0;
71. }

# [CCF201509-3 模板生成系统（100分）](http://blog.csdn.net/tigerisland45/article/details/54980592)

|  |  |
| --- | --- |
| 试题编号： | 201509-3 |
| 试题名称： | 模板生成系统 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　成成最近在搭建一个网站，其中一些页面的部分内容来自数据库中不同的数据记录，但是页面的基本结构是相同的。例如，对于展示用户信息的页面，当用户为 Tom 时，网页的源代码是 C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center(3).png  　　而当用户为 Jerry 时，网页的源代码是 C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center(4).png  　　这样的例子在包含动态内容的网站中还有很多。为了简化生成网页的工作，成成觉得他需要引入一套模板生成系统。 　　模板是包含特殊标记的文本。成成用到的模板只包含一种特殊标记，格式为 {{ VAR }}，其中 VAR 是一个变量。该标记在模板生成时会被变量 VAR 的值所替代。例如，如果变量 name = "Tom"，则 {{ name }} 会生成 Tom。具体的规则如下： 　　·变量名由大小写字母、数字和下划线 (\_) 构成，且第一个字符不是数字，长度不超过 16 个字符。 　　·变量名是大小写敏感的，Name 和 name 是两个不同的变量。 　　·变量的值是字符串。 　　·如果标记中的变量没有定义，则生成空串，相当于把标记从模板中删除。 　　·模板不递归生成。也就是说，如果变量的值中包含形如 {{ VAR }} 的内容，不再做进一步的替换。  **输入格式**  　　输入的第一行包含两个整数 m, n，分别表示模板的行数和模板生成时给出的变量个数。 　　接下来 m 行，每行是一个字符串，表示模板。 　　接下来 n 行，每行表示一个变量和它的值，中间用一个空格分隔。值是字符串，用双引号 (") 括起来，内容可包含除双引号以外的任意可打印 ASCII 字符（ASCII 码范围 32, 33, 35-126）。  **输出格式**  　　输出包含若干行，表示模板生成的结果。  **样例输入**  11 2 <!DOCTYPE html> <html> <head> <title>User {{ name }}</title> </head> <body> <h1>{{ name }}</h1> <p>Email: <a href="mailto:{{ email }}">{{ email }}</a></p> <p>Address: {{ address }}</p> </body> </html> name "David Beckham" email "david@beckham.com"  **样例输出**  <!DOCTYPE html> <html> <head> <title>User David Beckham</title> </head> <body> <h1>David Beckham</h1> <p>Email: <a href="mailto:david@beckham.com">david@beckham.com</a></p> <p>Address: </p> </body> </html>  **评测用例规模与约定**  　　0 ≤ m ≤ 100 　　0 ≤ n ≤ 100 　　输入的模板每行长度不超过 80 个字符（不包含换行符）。 　　输入保证模板中所有以 {{ 开始的子串都是合法的标记，开始是两个左大括号和一个空格，然后是变量名，结尾是一个空格和两个右大括号。 　　输入中所有变量的值字符串长度不超过 100 个字符（不包括双引号）。 　　保证输入的所有变量的名字各不相同。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个输入字符串处理的问题，也是一个模板替换问题。

**程序说明**：程序用需要用函数getline()来读入一行，之前都是用自己编写的程序。因为C++的cin输入字符串时，无法输入空串和读入空格，而C的库函数中已经不建议使用函数gets()，所以需要使用该函数。模板的各个行，用字符串向量vector<string>来存储比较有效。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54980592) [copy](http://blog.csdn.net/tigerisland45/article/details/54980592)[在CODE上查看代码片](https://code.csdn.net/snippets/2221740)

1. /\* CCF201509-3 模板生成系统 \*/
3. #include <iostream>
4. #include <string>
5. #include <vector>
6. #include <map>
8. **using** **namespace** std;
10. **int** main()
11. {
12. **int** n, m;
13. vector<string> tv;
14. string key, value, temp;
15. map<string, string> dict;
16. **int** prev, next;
18. // 输入数据
19. cin >> n >> m;
20. getchar();
21. **for**(**int** i=1; i<=n; i++) {
22. getline(cin, temp);
24. tv.push\_back(temp);
25. }
26. **for**(**int** i=1; i<=m; i++) {
27. cin >> key;
28. getline(cin, value);
30. dict[key] = value.substr(2, value.length()-3);
31. }
33. // 替换处理与输出结果
34. **for**(**int** i=0; i<n; i++) {
35. // 替换处理
36. prev = 0;
37. **for**(;;) {
38. **if**((prev = tv[i].find("{{ ", prev)) == (**int**)string::npos)
39. **break**;
40. **if**((next = tv[i].find(" }}", prev)) == (**int**)string::npos)
41. **break**;
43. key = tv[i].substr(prev + 3, next - prev - 3);
44. // 以下两个语句是等价的，后一句会更快
45. //            tv[i].replace(prev, key.length() + 6, dict.count(key) ? dict[key] : "");
46. tv[i].replace(prev, next - prev + 3, dict.count(key) ? dict[key] : "");
48. prev += dict.count(key) ? dict[key].length() : 0;   // 避免重复替换
49. }
51. // 输出结果
52. cout << tv[i] << endl;
53. }
55. **return** 0;
56. }

[**CCF201512-3 画图（100分）**](http://blog.csdn.net/tigerisland45/article/details/54976424)

|  |  |
| --- | --- |
| 试题编号： | 201512-3 |
| 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**  　　用 ASCII 字符来画图是一件有趣的事情，并形成了一门被称为 ASCII Art 的艺术。例如，下图是用 ASCII 字符画出来的 CSPRO 字样。 　　..\_\_\_\_.\_\_\_\_..\_\_\_\_..\_\_\_\_...\_\_\_.. 　　./.\_\_\_/.\_\_\_||..\_.\|..\_.\./.\_.\. 　　|.|...\\_\_\_.\|.|\_).|.|\_).|.|.|.| 　　|.|\_\_\_.\_\_\_).|..\_\_/|..\_.<|.|\_|.| 　　.\\_\_\_\_|\_\_\_\_/|\_|...|\_|.\\_\\\_\_\_/. 　　本题要求编程实现一个用 ASCII 字符来画图的程序，支持以下两种操作： 　　 画线：给出两个端点的坐标，画一条连接这两个端点的线段。简便起见题目保证要画的每条线段都是水平或者竖直的。水平线段用字符 - 来画，竖直线段用字符 | 来画。如果一条水平线段和一条竖直线段在某个位置相交，则相交位置用字符 + 代替。 　　 填充：给出填充的起始位置坐标和需要填充的字符，从起始位置开始，用该字符填充相邻位置，直到遇到画布边缘或已经画好的线段。注意这里的相邻位置只需要考虑上下左右 4 个方向，如下图所示，字符 @ 只和 4 个字符 \* 相邻。 　　.\*. 　　\*@\* 　　.\*.  **输入格式**  　　第1行有三个整数m, n和q。m和n分别表示画布的宽度和高度，以字符为单位。q表示画图操作的个数。 　　第2行至第q + 1行，每行是以下两种形式之一： 　　 0 x1 y1 x2 y2：表示画线段的操作，(x1, y1)和(x2, y2)分别是线段的两端，满足要么x1 = x2 且y1 ≠ y2，要么 y1 = y2 且 x1 ≠ x2。 　　 1 x y c：表示填充操作，(x, y)是起始位置，保证不会落在任何已有的线段上；c 为填充字符，是大小写字母。 　　画布的左下角是坐标为 (0, 0) 的位置，向右为x坐标增大的方向，向上为y坐标增大的方向。这q个操作按照数据给出的顺序依次执行。画布最初时所有位置都是字符 .（小数点）。  **输出格式**  　　输出有n行，每行m个字符，表示依次执行这q个操作后得到的画图结果。  **样例输入**  4 2 3 1 0 0 B 0 1 0 2 0 1 0 0 A  **样例输出**  AAAA A--A  **样例输入**  16 13 9 0 3 1 12 1 0 12 1 12 3 0 12 3 6 3 0 6 3 6 9 0 6 9 12 9 0 12 9 12 11 0 12 11 3 11 0 3 11 3 1 1 4 2 C  **样例输出**  ................ ...+--------+... ...|CCCCCCCC|... ...|CC+-----+... ...|CC|......... ...|CC|......... ...|CC|......... ...|CC|......... ...|CC|......... ...|CC+-----+... ...|CCCCCCCC|... ...+--------+... ................  **评测用例规模与约定**  　　所有的评测用例满足：2 ≤ m, n ≤ 100，0 ≤ q ≤ 100，0 ≤ x < m（x表示输入数据中所有位置的x坐标），0 ≤ y < n（y表示输入数据中所有位置的y坐标）。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个模拟题，看懂题意就不难了。

**程序说明**：坐标的转换需要注意。递归填充时，需要仔细考虑有关条件。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54976424) [copy](http://blog.csdn.net/tigerisland45/article/details/54976424)

1. /\* CCF201512-3 画图 \*/
3. #include <iostream>
4. #include <cstring>
6. **using** **namespace** std;
8. **const** **int** DIRECTSIZE = 4;
9. **struct** direct {
10. **int** dx, dy;
11. } direct[DIRECTSIZE] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
13. **const** **int** N = 100;
15. **char** grid[N+1][N+1];
17. // 递归填充：被填充的点的四周也要填充
18. **void** fill(**int** x, **int** y, **char** c, **int** m, **int** n)
19. {
20. **int** nx, ny;
22. grid[y][x] = c;
24. **for**(**int** i=0; i<DIRECTSIZE; i++) {
25. ny = y + direct[i].dy;
26. nx = x + direct[i].dx;
28. **if**(0 <= nx && nx < m && 0 <= ny && ny < n &&
29. grid[ny][nx] != '|' && grid[ny][nx] != '-' && grid[ny][nx] != '+' &&
30. grid[ny][nx] != c)
31. fill(nx, ny, c, m, n);
32. }
33. }
35. **int** main()
36. {
37. **int** m, n, q, option;
38. **int** x1, y1, x2, y2;
39. **char** c;
40. **int** start, end;
42. // 变量初始化：全部设置为“.”
43. memset(grid, '.', **sizeof**(grid));
45. // 输入数据
46. cin >> m >> n >> q;
48. // 处理q个画图操作
49. **for**(**int** i=1; i<=q; i++) {
50. // 输入形式
51. cin >> option;
53. **if**(option == 0) {
54. // 输入坐标
55. cin >> x1 >> y1 >> x2 >> y2;
57. // 画线
58. **if**(x1 == x2) {
59. start = min(y1, y2);
60. end = max(y1, y2);
61. **for**(**int** j=start; j<=end; j++)
62. **if**(grid[j][x1] == '-' || grid[j][x1] == '+')
63. grid[j][x1] = '+';
64. **else**
65. grid[j][x1] = '|';
66. } **else** **if**(y1 == y2) {
67. start = min(x1, x2);
68. end = max(x1, x2);
69. **for**(**int** j=start; j<=end; j++)
70. **if**(grid[y1][j] == '|' || grid[y1][j] == '+')
71. grid[y1][j] = '+';
72. **else**
73. grid[y1][j] = '-';
74. }
75. } **else** **if**(option == 1) {
76. // 输入填充的坐标和字符
77. cin >> x1 >> y1 >> c;
79. // 递归填充
80. fill(x1, y1, c, m, n);
81. }
82. }
84. // 输出结果
85. **for**(**int** i=n-1; i>=0; i--) {
86. **for**(**int** j=0; j<m; j++)
87. cout << grid[i][j];
88. cout << endl;
89. }
91. **return** 0;
92. }

[**CCF201604-3 路径解析（100分）**](http://blog.csdn.net/tigerisland45/article/details/54966865)

|  |  |
| --- | --- |
| 试题编号： | 201604-3 |
| 试题名称： | 路径解析 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在操作系统中，数据通常以文件的形式存储在文件系统中。文件系统一般采用层次化的组织形式，由目录（或者文件夹）和文件构成，形成一棵树的形状。文件有内容，用于存储数据。目录是容器，可包含文件或其他目录。同一个目录下的所有文件和目录的名字各不相同，不同目录下可以有名字相同的文件或目录。 　　为了指定文件系统中的某个文件，需要用路径来定位。在类 Unix 系统（Linux、Max OS X、FreeBSD等）中，路径由若干部分构成，每个部分是一个目录或者文件的名字，相邻两个部分之间用 / 符号分隔。 　　有一个特殊的目录被称为根目录，是整个文件系统形成的这棵树的根节点，用一个单独的 / 符号表示。在操作系统中，有当前目录的概念，表示用户目前正在工作的目录。根据出发点可以把路径分为两类：         绝对路径：以 / 符号开头，表示从根目录开始构建的路径。　            相对路径：不以 / 符号开头，表示从当前目录开始构建的路径。  　　例如，有一个文件系统的结构如下图所示。在这个文件系统中，有根目录 / 和其他普通目录 d1、d2、d3、d4，以及文件 f1、f2、f3、f1、f4。其中，两个 f1 是同名文件，但在不同的目录下。  　　对于 d4 目录下的 f1 文件，可以用绝对路径 /d2/d4/f1 来指定。如果当前目录是 /d2/d3，这个文件也可以用相对路径 ../d4/f1 来指定，这里 .. 表示上一级目录（注意，根目录的上一级目录是它本身）。还有 . 表示本目录，例如 /d1/./f1 指定的就是 /d1/f1。注意，如果有多个连续的 / 出现，其效果等同于一个 /，例如 /d1///f1 指定的也是 /d1/f1。 　　本题会给出一些路径，要求对于每个路径，给出正规化以后的形式。一个路径经过正规化操作后，其指定的文件不变，但是会变成一个不包含 . 和 .. 的绝对路径，且不包含连续多个 / 符号。如果一个路径以 / 结尾，那么它代表的一定是一个目录，正规化操作要去掉结尾的 /。若这个路径代表根目录，则正规化操作的结果是 /。若路径为空字符串，则正规化操作的结果是当前目录。  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  **输入格式**  　　第一行包含一个整数 P，表示需要进行正规化操作的路径个数。 　　第二行包含一个字符串，表示当前目录。 　　以下 P 行，每行包含一个字符串，表示需要进行正规化操作的路径。  **输出格式**  　　共 P 行，每行一个字符串，表示经过正规化操作后的路径，顺序与输入对应。  **样例输入**  7 /d2/d3 /d2/d4/f1 ../d4/f1 /d1/./f1 /d1///f1 /d1/ /// /d1/../../d2  **样例输出**  /d2/d4/f1 /d2/d4/f1 /d1/f1 /d1/f1 /d1 / /d2  **评测用例规模与约定**  　　1 ≤ P ≤ 10。 　　文件和目录的名字只包含大小写字母、数字和小数点 .、减号 - 以及下划线 \_。 　　不会有文件或目录的名字是 . 或 .. ，它们具有题目描述中给出的特殊含义。 　　输入的所有路径每个长度不超过 1000 个字符。 　　输入的当前目录保证是一个经过正规化操作后的路径。 　　对于前 30% 的测试用例，需要正规化的路径的组成部分不包含 . 和 .. 。 　　对于前 60% 的测试用例，需要正规化的路径都是绝对路径。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个输入字符串处理的问题，可以有各种各样的处理方法。

1.直接用[**C语言**](http://lib.csdn.net/base/c)及其库函数来处理。这是最基本的最高效的（时间上）的方法，逻辑相对会复杂一些。

2.用C++的string类有关的方法（函数）来处理。这种方法编程效率比较高。

3.采用混合方法来处理。既使用C语言的库函数，也使用C++的类库。

这个问题有一个陷阱，就是输入的字符串可能是空串。

**程序说明**：本程序用第3中方法来实现。因为C++的cin输入字符串时，无法输入空串，而C的库函数中已经不建议使用函数gets()，所以自己编写一个读入一行的函数mygetline()，该函数可以输入空行。用“/”来分割字符串，切割字符串的函数使用C语言的库函数strtok()，将分割好的字符串存储在字符串向量vector<string>是一个有效的做法，程序逻辑上比较简洁。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54966865) [copy](http://blog.csdn.net/tigerisland45/article/details/54966865)

1. /\* CCF201604-3 路径解析 \*/
3. #include <iostream>
4. #include <string>
5. #include <vector>
6. #include <cstring>
7. #include <cstdio>
9. **using** **namespace** std;
11. **const** **int** N = 1000;
13. **char** currentpath[N+1];
14. **char** v[N\*2+1], t[N+1];
16. **void** spilt(vector<string>& v, **char** \*s)
17. {
18. **char** \*sp;
20. sp = strtok(s, "/");
21. **while**(sp) {
22. v.push\_back(sp);
23. sp = strtok(NULL, "/");
24. }
25. }
27. **void** process(vector<string>& sv)
28. {
29. **int** delcount = 0;
31. **for**(**int** i=0; i<(**int**)sv.size(); i++) {
32. **if**(sv[i] == "..") {
33. delcount++;
34. sv[i] = "";
35. **for**(**int** j=i-1; j>=0; j--)
36. **if**(sv[j] != "") {
37. delcount++;
38. sv[j] = "";
39. **break**;
40. }
41. } **else** **if**(sv[i] == ".") {
42. delcount++;
43. sv[i] = "";
44. }
45. }
47. **if**((**int**)sv.size() - delcount <= 0)
48. cout << "/" << endl;
49. **else** {
50. **for**(**int** i=0; i<(**int**)sv.size(); i++) {
51. **if**(sv[i] != "" )
52. cout << "/" << sv[i];
53. }
54. cout << endl;
55. }
56. }
58. **void** mygetline(**char** \*pc)
59. {
60. **char** c;
62. **while**((c=getchar()) != '\n' && c !=EOF)
63. \*pc++ = c;
64. \*pc = '\0';
65. }
67. **int** main()
68. {
69. **int** p;
71. // 输入数据：整数p和当前目录
72. cin >> p >> currentpath;
73. getchar();
75. // 输入p个路径进行正规化处理
76. **for**(**int** i=1; i<=p; i++) {
77. vector<string> sv;
79. // 输入路径
80. mygetline(t);
82. // 非根路径处理
83. **if**(t[0] != '/') {
84. strcpy(v, currentpath);
85. strcat(v, "/");
86. strcat(v, t);
87. } **else**
88. strcpy(v, t);
90. // 切分路径
91. spilt(sv, v);
93. // 对输入路径进行正规化处理，并且输出结果
94. process(sv);
95. }
97. **return** 0;
98. }

# [CCF201604-3 路径解析（解法二）（100分）](http://blog.csdn.net/tigerisland45/article/details/54970117)

|  |  |
| --- | --- |
| 试题编号： | 201604-3 |
| 试题名称： | 路径解析 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在操作系统中，数据通常以文件的形式存储在文件系统中。文件系统一般采用层次化的组织形式，由目录（或者文件夹）和文件构成，形成一棵树的形状。文件有内容，用于存储数据。目录是容器，可包含文件或其他目录。同一个目录下的所有文件和目录的名字各不相同，不同目录下可以有名字相同的文件或目录。 　　为了指定文件系统中的某个文件，需要用路径来定位。在类 Unix 系统（Linux、Max OS X、FreeBSD等）中，路径由若干部分构成，每个部分是一个目录或者文件的名字，相邻两个部分之间用 / 符号分隔。 　　有一个特殊的目录被称为根目录，是整个文件系统形成的这棵树的根节点，用一个单独的 / 符号表示。在操作系统中，有当前目录的概念，表示用户目前正在工作的目录。根据出发点可以把路径分为两类：         绝对路径：以 / 符号开头，表示从根目录开始构建的路径。　           相对路径：不以 / 符号开头，表示从当前目录开始构建的路径。  　　例如，有一个文件系统的结构如下图所示。在这个文件系统中，有根目录 / 和其他普通目录 d1、d2、d3、d4，以及文件 f1、f2、f3、f1、f4。其中，两个 f1 是同名文件，但在不同的目录下。 C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center(2).png 　　对于 d4 目录下的 f1 文件，可以用绝对路径 /d2/d4/f1 来指定。如果当前目录是 /d2/d3，这个文件也可以用相对路径 ../d4/f1 来指定，这里 .. 表示上一级目录（注意，根目录的上一级目录是它本身）。还有 . 表示本目录，例如 /d1/./f1 指定的就是 /d1/f1。注意，如果有多个连续的 / 出现，其效果等同于一个 /，例如 /d1///f1 指定的也是 /d1/f1。 　　本题会给出一些路径，要求对于每个路径，给出正规化以后的形式。一个路径经过正规化操作后，其指定的文件不变，但是会变成一个不包含 . 和 .. 的绝对路径，且不包含连续多个 / 符号。如果一个路径以 / 结尾，那么它代表的一定是一个目录，正规化操作要去掉结尾的 /。若这个路径代表根目录，则正规化操作的结果是 /。若路径为空字符串，则正规化操作的结果是当前目录。  **输入格式**  　　第一行包含一个整数 P，表示需要进行正规化操作的路径个数。 　　第二行包含一个字符串，表示当前目录。 　　以下 P 行，每行包含一个字符串，表示需要进行正规化操作的路径。  **输出格式**  　　共 P 行，每行一个字符串，表示经过正规化操作后的路径，顺序与输入对应。  **样例输入**  7 /d2/d3 /d2/d4/f1 ../d4/f1 /d1/./f1 /d1///f1 /d1/ /// /d1/../../d2  **样例输出**  /d2/d4/f1 /d2/d4/f1 /d1/f1 /d1/f1 /d1 / /d2  **评测用例规模与约定**  　　1 ≤ P ≤ 10。 　　文件和目录的名字只包含大小写字母、数字和小数点 .、减号 - 以及下划线 \_。 　　不会有文件或目录的名字是 . 或 .. ，它们具有题目描述中给出的特殊含义。 　　输入的所有路径每个长度不超过 1000 个字符。 　　输入的当前目录保证是一个经过正规化操作后的路径。 　　对于前 30% 的测试用例，需要正规化的路径的组成部分不包含 . 和 .. 。 　　对于前 60% 的测试用例，需要正规化的路径都是绝对路径。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个输入字符串处理的问题，可以有各种各样的处理方法。

1.直接用[**C语言**](http://lib.csdn.net/base/c)及其库函数来处理。这是最基本的最高效的（时间上）的方法，逻辑相对会复杂一些。

2.用C++的string类有关的方法（函数）来处理。这种方法编程效率比较高。

3.采用混合方法来处理。既使用C语言的库函数，也使用C++的类库。

这个问题有一个陷阱，就是输入的字符串可能是空串。

**程序说明**：本程序用第2中方法来实现。因为C++的cin输入字符串时，无法输入空串，而C的库函数中已经不建议使用函数gets()，所以自己编写一个读入一行的函数mygetline()，该函数可以输入空行。使用C++类string的各种方法（函数）也同样可以实现字符串的处理。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54970117) [copy](http://blog.csdn.net/tigerisland45/article/details/54970117)[在CODE上查看代码片](https://code.csdn.net/snippets/2222206)

1. /\* CCF201604-3 路径解析 \*/
3. #include <iostream>
4. #include <string>
6. **using** **namespace** std;
8. **const** **int** N = 1000;
9. **char** s[N+1];
11. **void** mygetline(**char** \*pc)
12. {
13. **char** c;
15. **while**((c=getchar()) != '\n' && c !=EOF)
16. \*pc++ = c;
17. \*pc = '\0';
18. }
20. **int** main()
21. {
22. **int** p, pos;
23. string cp, line;
25. // 输入数据：整数p和当前目录
26. cin >> p >> cp;
27. getchar();
29. // 输入p个路径进行正规化处理
30. **for**(**int** i=1; i<=p; i++) {
31. // 输入路径
32. mygetline(s);
33. line = s;
35. // 非根路径处理
36. **if**(line[0] != '/')
37. line = cp + "/" + line + "/";
39. // 去除多个"/"
40. **while**((pos = line.find("//")) != -1) {
41. **int** count = 2;
42. **while**(line[pos + count] == '/')
43. count++;
44. line.erase(pos, count-1);
45. }
47. // 去除"./"
48. **while**((pos = line.find("/./")) != -1)
49. line.erase(pos + 1, 2);
51. // 去除最后的"/"
52. **if**(line.size() > 1 && line[line.size() - 1] == '/')
53. line.erase(line.size() - 1);
55. // 去除"../"
56. **while**((pos = line.find("/../")) != -1) {
57. **if**(pos == 0)
58. line.erase(pos, 3);
59. **else** {
60. **int** spos;
61. spos = line.rfind("/", pos - 1);
62. line.erase(spos, pos - spos + 3);
63. }
65. **if**(line.size() == 0)
66. line = "/";
67. }
69. // 输出结果
70. cout << line << endl;
71. }
73. **return** 0;
74. }

[**CCF201604-3 路径解析（解法三）（90分）**](http://blog.csdn.net/tigerisland45/article/details/54970771)

|  |  |
| --- | --- |
| 试题编号： | 201604-3 |
| 试题名称： | 路径解析 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　在操作系统中，数据通常以文件的形式存储在文件系统中。文件系统一般采用层次化的组织形式，由目录（或者文件夹）和文件构成，形成一棵树的形状。文件有内容，用于存储数据。目录是容器，可包含文件或其他目录。同一个目录下的所有文件和目录的名字各不相同，不同目录下可以有名字相同的文件或目录。 　　为了指定文件系统中的某个文件，需要用路径来定位。在类 Unix 系统（Linux、Max OS X、FreeBSD等）中，路径由若干部分构成，每个部分是一个目录或者文件的名字，相邻两个部分之间用 / 符号分隔。 　　有一个特殊的目录被称为根目录，是整个文件系统形成的这棵树的根节点，用一个单独的 / 符号表示。在操作系统中，有当前目录的概念，表示用户目前正在工作的目录。根据出发点可以把路径分为两类：         绝对路径：以 / 符号开头，表示从根目录开始构建的路径。　            相对路径：不以 / 符号开头，表示从当前目录开始构建的路径。  　　例如，有一个文件系统的结构如下图所示。在这个文件系统中，有根目录 / 和其他普通目录 d1、d2、d3、d4，以及文件 f1、f2、f3、f1、f4。其中，两个 f1 是同名文件，但在不同的目录下。  　　对于 d4 目录下的 f1 文件，可以用绝对路径 /d2/d4/f1 来指定。如果当前目录是 /d2/d3，这个文件也可以用相对路径 ../d4/f1 来指定，这里 .. 表示上一级目录（注意，根目录的上一级目录是它本身）。还有 . 表示本目录，例如 /d1/./f1 指定的就是 /d1/f1。注意，如果有多个连续的 / 出现，其效果等同于一个 /，例如 /d1///f1 指定的也是 /d1/f1。 　　本题会给出一些路径，要求对于每个路径，给出正规化以后的形式。一个路径经过正规化操作后，其指定的文件不变，但是会变成一个不包含 . 和 .. 的绝对路径，且不包含连续多个 / 符号。如果一个路径以 / 结尾，那么它代表的一定是一个目录，正规化操作要去掉结尾的 /。若这个路径代表根目录，则正规化操作的结果是 /。若路径为空字符串，则正规化操作的结果是当前目录。  C:\Users\XIAOQI~1\AppData\Local\Temp\Center.png  **输入格式**  　　第一行包含一个整数 P，表示需要进行正规化操作的路径个数。 　　第二行包含一个字符串，表示当前目录。 　　以下 P 行，每行包含一个字符串，表示需要进行正规化操作的路径。  **输出格式**  　　共 P 行，每行一个字符串，表示经过正规化操作后的路径，顺序与输入对应。  **样例输入**  7 /d2/d3 /d2/d4/f1 ../d4/f1 /d1/./f1 /d1///f1 /d1/ /// /d1/../../d2  **样例输出**  /d2/d4/f1 /d2/d4/f1 /d1/f1 /d1/f1 /d1 / /d2  **评测用例规模与约定**  　　1 ≤ P ≤ 10。 　　文件和目录的名字只包含大小写字母、数字和小数点 .、减号 - 以及下划线 \_。 　　不会有文件或目录的名字是 . 或 .. ，它们具有题目描述中给出的特殊含义。 　　输入的所有路径每个长度不超过 1000 个字符。 　　输入的当前目录保证是一个经过正规化操作后的路径。 　　对于前 30% 的测试用例，需要正规化的路径的组成部分不包含 . 和 .. 。 　　对于前 60% 的测试用例，需要正规化的路径都是绝对路径。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个输入字符串处理的问题，可以有各种各样的处理方法。

1.直接用[**C语言**](http://lib.csdn.net/base/c)及其库函数来处理。这是最基本的最高效的（时间上）的方法，逻辑相对会复杂一些。

2.用C++的string类有关的方法（函数）来处理。这种方法编程效率比较高。

3.采用混合方法来处理。既使用C语言的库函数，也使用C++的类库。

这个问题有一个陷阱，就是输入的字符串可能是空串。

**程序说明**：本程序用第1中方法来实现。因为C++的cin输入字符串时，无法输入空串，而C的库函数中已经不建议使用函数gets()，所以自己编写一个读入一行的函数mygetline()，该函数可以输入空行。

该程序有个BUG，只得了90分，希望有人帮助解决一下。

提交后得90分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54970771) [copy](http://blog.csdn.net/tigerisland45/article/details/54970771)

1. /\* CCF201604-3 路径解析 \*/
3. #include <iostream>
4. #include <cstdio>
5. #include <cstring>
7. **using** **namespace** std;
9. **const** **int** N = 1000;
11. **char** cp[N+1], s[N+1], ans[N\*2+1];
13. **void** mygetline(**char** \*pc)
14. {
15. **char** c;
17. **while**((c=getchar()) != '\n' && c !=EOF)
18. \*pc++ = c;
19. \*pc = '\0';
20. }
22. **int** main()
23. {
24. **int** p;
26. // 输入数据：整数p
27. scanf("%d", &p);
28. getchar();
30. // 输入数据：当前目录
31. mygetline(cp);
33. // 输入p个路径进行正规化处理
34. **for**(**int** i=1; i<=p; i++) {
35. // 输入路径
36. mygetline(s);
38. // 非根路径处理
39. **if**(s[0] != '/') {
40. strcpy(ans, cp);
41. strcat(ans, "/");
42. strcat(ans, s);
43. } **else**
44. strcpy(ans, s);
46. // 对输入路径进行正规化处理，并且输出结果
47. **char** \*p1, \*p2, \*pt;
48. p1 = p2 = ans;
49. **while**(\*p2) {
50. // 去除多个"/"
51. **if**(\*p2 == '/' && \*(p2+1) == '/')
52. p2++;
54. // 去除"../"
55. **else** **if**(\*p2 == '/' && \*(p2+1) == '.' && \*(p2+2) == '.' && \*(p2+3) == '/') {
56. **if**(p1 == ans)
57. p2 += 3;
58. **else** {
59. pt = p1;
60. **while**(--pt != ans)
61. **if**(\*pt == '/')
62. **break**;
63. p1 = pt;
64. p2 += 3;
65. }
67. // 去除"./"
68. } **else** **if**(\*p2 == '/' && \*(p2+1) == '.' && \*(p2+2) == '/')
69. p2 += 2;
71. // 去除最后的"/"
72. **else** **if**(p2 != ans && \*p2 == '/' && \*(p2+1) == '\0')
73. p2++;
75. // 去除最后的"/."
76. **else** **if**(p2 != ans && \*p2 == '/' && \*(p2+1) == '.' && \*(p2+2) == '\0')
77. p2 += 2;
78. **else** {
79. \*p1++ = \*p2++;
80. }
81. }
82. \*p1 = '\0';
84. // 输出结果
85. printf("%s\n", ans);
86. }
88. **return** 0;
89. }

# CCF201609-3 炉石传说（100分）

|  |  |
| --- | --- |
| 试题编号： | 201609-3 |
| 试题名称： | 炉石传说 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　《炉石传说：魔兽英雄传》（Hearthstone: Heroes of Warcraft，简称炉石传说）是暴雪娱乐开发的一款集换式卡牌游戏（如下图所示）。游戏在一个战斗棋盘上进行，由两名玩家轮流进行操作，本题所使用的炉石传说游戏的简化规则如下： C:\Users\XIAOQI~1\AppData\Local\Temp\enhtmlclip\Center.jpg 　　\* 玩家会控制一些角色，每个角色有自己的生命值和攻击力。当生命值小于等于 0 时，该角色死亡。角色分为英雄和随从。 　　\* 玩家各控制一个英雄，游戏开始时，英雄的生命值为 30，攻击力为 0。当英雄死亡时，游戏结束，英雄未死亡的一方获胜。 　　\* 玩家可在游戏过程中召唤随从。棋盘上每方都有 7 个可用于放置随从的空位，从左到右一字排开，被称为战场。当随从死亡时，它将被从战场上移除。 　　\* 游戏开始后，两位玩家轮流进行操作，每个玩家的连续一组操作称为一个回合。 　　\* 每个回合中，当前玩家可进行零个或者多个以下操作： 　　1) 召唤随从：玩家召唤一个随从进入战场，随从具有指定的生命值和攻击力。 　　2) 随从攻击：玩家控制自己的某个随从攻击对手的英雄或者某个随从。 　　3) 结束回合：玩家声明自己的当前回合结束，游戏将进入对手的回合。该操作一定是一个回合的最后一个操作。 　　\* 当随从攻击时，攻击方和被攻击方会同时对彼此造成等同于自己攻击力的伤害。受到伤害的角色的生命值将会减少，数值等同于受到的伤害。例如，随从 X 的生命值为 HX、攻击力为 AX，随从 Y 的生命值为 HY、攻击力为 AY，如果随从 X 攻击随从 Y，则攻击发生后随从 X 的生命值变为 HX - AY，随从 Y 的生命值变为 HY - AX。攻击发生后，角色的生命值可以为负数。 　　本题将给出一个游戏的过程，要求编写程序模拟该游戏过程并输出最后的局面。  **输入格式**  　　输入第一行是一个整数 n，表示操作的个数。接下来 n 行，每行描述一个操作，格式如下： 　　<action> <arg1> <arg2> ... 　　其中<action>表示操作类型，是一个字符串，共有 3 种：summon表示召唤随从，attack表示随从攻击，end表示结束回合。这 3 种操作的具体格式如下： 　　\* summon <position> <attack> <health>：当前玩家在位置<position>召唤一个生命值为<health>、攻击力为<attack>的随从。其中<position>是一个 1 到 7 的整数，表示召唤的随从出现在战场上的位置，原来该位置及右边的随从都将顺次向右移动一位。 　　\* attack <attacker> <defender>：当前玩家的角色<attacker>攻击对方的角色 <defender>。<attacker>是 1 到 7 的整数，表示发起攻击的本方随从编号，<defender>是 0 到 7 的整数，表示被攻击的对方角色，0 表示攻击对方英雄，1 到 7 表示攻击对方随从的编号。 　　\* end：当前玩家结束本回合。 　　注意：随从的编号会随着游戏的进程发生变化，当召唤一个随从时，玩家指定召唤该随从放入战场的位置，此时，原来该位置及右边的所有随从编号都会增加 1。而当一个随从死亡时，它右边的所有随从编号都会减少 1。任意时刻，战场上的随从总是从1开始连续编号。  **输出格式**  　　输出共 5 行。 　　第 1 行包含一个整数，表示这 n 次操作后（以下称为 T 时刻）游戏的胜负结果，1 表示先手玩家获胜，-1 表示后手玩家获胜，0 表示游戏尚未结束，还没有人获胜。 　　第 2 行包含一个整数，表示 T 时刻先手玩家的英雄的生命值。 　　第 3 行包含若干个整数，第一个整数 p 表示 T 时刻先手玩家在战场上存活的随从个数，之后 p 个整数，分别表示这些随从在 T 时刻的生命值（按照从左往右的顺序）。 　　第 4 行和第 5 行与第 2 行和第 3 行类似，只是将玩家从先手玩家换为后手玩家。  **样例输入**  8 summon 1 3 6 summon 2 4 2 end summon 1 4 5 summon 1 2 1 attack 1 2 end attack 1 1  **样例输出**  0 30 1 2 30 1 2  **样例说明**  　　按照样例输入从第 2 行开始逐行的解释如下： 　　1. 先手玩家在位置 1 召唤一个生命值为 6、攻击力为 3 的随从 A，是本方战场上唯一的随从。 　　2. 先手玩家在位置 2 召唤一个生命值为 2、攻击力为 4 的随从 B，出现在随从 A 的右边。 　　3. 先手玩家回合结束。 　　4. 后手玩家在位置 1 召唤一个生命值为 5、攻击力为 4 的随从 C，是本方战场上唯一的随从。 　　5. 后手玩家在位置 1 召唤一个生命值为 1、攻击力为 2 的随从 D，出现在随从 C 的左边。 　　6. 随从 D 攻击随从 B，双方均死亡。 　　7. 后手玩家回合结束。 　　8. 随从 A 攻击随从 C，双方的生命值都降低至 2。  **评测用例规模与约定**  　　\* 操作的个数0 ≤ n ≤ 1000。 　　\* 随从的初始生命值为 1 到 100 的整数，攻击力为 0 到 100 的整数。 　　\* 保证所有操作均合法，包括但不限于： 　　1) 召唤随从的位置一定是合法的，即如果当前本方战场上有 m 个随从，则召唤随从的位置一定在 1 到 m + 1 之间，其中 1 表示战场最左边的位置，m + 1 表示战场最右边的位置。 　　2) 当本方战场有 7 个随从时，不会再召唤新的随从。 　　3) 发起攻击和被攻击的角色一定存在，发起攻击的角色攻击力大于 0。 　　4) 一方英雄如果死亡，就不再会有后续操作。 　　\* 数据约定： 　　前 20% 的评测用例召唤随从的位置都是战场的最右边。 　　前 40% 的评测用例没有 attack 操作。 　　前 60% 的评测用例不会出现随从死亡的情况。 |

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

**问题描述**：（参见上文）。

**问题分析**：这是一个模拟题，数据表示解决之后一切就简单了。

**程序说明**：程序中，在可阅读性方面做了一定的努力。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/54986242) [copy](http://blog.csdn.net/tigerisland45/article/details/54986242)[在CODE上查看代码片](https://code.csdn.net/snippets/2217945)

1. /\* CCF201609-3 炉石传说 \*/
3. #include <iostream>
4. #include <vector>
5. #include <string>
7. #define OPONENT 1-player
9. **using** **namespace** std;
11. // 红黑双方，红先黑后
12. **const** **int** RED = 0;
13. **const** **int** BLACK = 1;
15. // 状态包括生命值和攻击力
16. **struct** status {
17. **int** life, power;
18. status(**int** l, **int** a){life = l; power = a;}
19. };
21. // 变量v用于存储英雄和随从的状态，v[0]为红方，v[1]为黑方
22. // 向量v[RED]和v[BLACK]中分别存储双方的英雄和随从的状态
23. // v[RED][0]存储红方英雄的状态，v[RED][1]-v[RED][7]存储随从的状态
24. vector<status> v[2];
26. **int** main()
27. {
28. **int** n, player=RED;
29. string action;
30. **int** position, attack, health;
31. **int** attacker, defender;
33. // 变量初始化：英雄在向量v中的位置为０
34. v[RED].push\_back(status(30, 0));
35. v[BLACK].push\_back(status(30, 0));
37. // 输入数据：命令数量n
38. cin >> n;
40. // 输入并且处理n个命令
41. **while**(n--) {
42. cin >> action;
43. **if**(action == "summon") {
44. cin >> position >> attack >> health;
46. v[player].insert(v[player].begin() + position, status(health, attack));
47. } **else** **if**(action == "attack") {
48. cin >> attacker >> defender;
50. // 当随从攻击时，攻击方和被攻击方会同时对彼此造成等同于自己攻击力的伤害。
51. // 受到伤害的角色的生命值将会减少，数值等同于受到的伤害。
52. v[player][attacker].life -= v[OPONENT][defender].power;
53. v[OPONENT][defender].life -= v[player][attacker].power;
55. // 攻击方（随从）生命结束则移除
56. **if**(v[player][attacker].life <= 0)
57. v[player].erase(v[player].begin() + attacker);
59. // 防御方的随从生命结束则移除
60. **if**(defender != 0 && v[OPONENT][defender].life <= 0)
61. v[OPONENT].erase(v[OPONENT].begin() + defender);
63. } **else** **if**(action == "end") {
64. player = OPONENT;
65. }
66. }
68. // 输出结果
69. // 第１行
70. **if**(v[RED][0].life <=0)
71. cout << -1 << endl;
72. **else** **if**(v[BLACK][0].life <=0)
73. cout << 1 << endl;
74. **else**
75. cout << 0 << endl;
76. // 第２行
77. cout << v[RED][0].life << endl;
78. // 第３行
79. cout << v[RED].size() - 1;
80. **for**(**int** i=1; i<(**int**)v[RED].size(); i++)
81. cout << " " << v[RED][i].life;
82. cout << endl;
83. // 第４行
84. cout << v[BLACK][0].life << endl;
85. // 第5行
86. cout << v[BLACK].size() - 1;
87. **for**(**int** i=1; i<(**int**)v[BLACK].size(); i++)
88. cout << " " << v[BLACK][i].life;
89. cout << endl;
91. **return** 0;
92. }

# [CCF201612-3 权限查询（100分）](http://blog.csdn.net/tigerisland45/article/details/56483841)

|  |  |
| --- | --- |
| 试题编号： | 201612-3 |
| 试题名称： | 权限查询 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　授权 (authorization) 是各类业务系统不可缺少的组成部分，系统用户通过授权机制获得系统中各个模块的操作权限。 　　本题中的授权机制是这样设计的：每位用户具有若干角色，每种角色具有若干权限。例如，用户 david 具有 manager 角色，manager 角色有 crm:2 权限，则用户 david 具有 crm:2 权限，也就是 crm 类权限的第 2 等级的权限。 　　具体地，用户名和角色名称都是由小写字母组成的字符串，长度不超过 32。权限分为分等级权限和不分等级权限两大类。分等级权限由权限类名和权限等级构成，中间用冒号“:”分隔。其中权限类名也是由小写字母组成的字符串，长度不超过 32。权限等级是一位数字，从 0 到 9，数字越大表示权限等级越高。系统规定如果用户具有某类某一等级的权限，那么他也将自动具有该类更低等级的权限。例如在上面的例子中，除 crm:2 外，用户 david 也具有 crm:1 和 crm:0 权限。不分等级权限在描述权限时只有权限类名，没有权限等级（也没有用于分隔的冒号）。 　　给出系统中用户、角色和权限的描述信息，你的程序需要回答多个关于用户和权限的查询。查询可分为以下几类： 　　\* 不分等级权限的查询：如果权限本身是不分等级的，则查询时不指定等级，返回是否具有该权限； 　　\* 分等级权限的带等级查询：如果权限本身分等级，查询也带等级，则返回是否具有该类的该等级权限； 　　\* 分等级权限的不带等级查询：如果权限本身分等级，查询不带等级，则返回具有该类权限的等级；如果不具有该类的任何等级权限，则返回“否”。  **输入格式**  　　输入第一行是一个正整数 p，表示不同的权限类别的数量。紧接着的 p 行被称为 P 段，每行一个字符串，描述各个权限。对于分等级权限，格式为 <category>:<level>，其中 <category> 是权限类名，<level> 是该类权限的最高等级。对于不分等级权限，字符串只包含权限类名。 　　接下来一行是一个正整数 r，表示不同的角色数量。紧接着的 r 行被称为 R 段，每行描述一种角色，格式为 　　<role> <s> <privilege 1> <privilege 2> ... <privilege s> 　　其中 <role> 是角色名称，<s> 表示该角色具有多少种权限。后面 <s> 个字符串描述该角色具有的权限，格式同 P 段。 　　接下来一行是一个正整数 u，表示用户数量。紧接着的 u 行被称为 U 段，每行描述一个用户，格式为 　　<user> <t> <role 1> <role 2> ... <role t> 　　其中 <user> 是用户名，<t> 表示该用户具有多少种角色。后面 <t> 个字符串描述该用户具有的角色。 　　接下来一行是一个正整数 q，表示权限查询的数量。紧接着的 q 行被称为 Q 段，每行描述一个授权查询，格式为 <user> <privilege>，表示查询用户 <user> 是否具有 <privilege> 权限。如果查询的权限是分等级权限，则查询中的 <privilege> 可指定等级，表示查询该用户是否具有该等级的权限；也可以不指定等级，表示查询该用户具有该权限的等级。对于不分等级权限，只能查询该用户是否具有该权限，查询中不能指定等级。  **输出格式**  　　输出共 q 行，每行为 false、true，或者一个数字。false 表示相应的用户不具有相应的权限，true 表示相应的用户具有相应的权限。对于分等级权限的不带等级查询，如果具有权限，则结果是一个数字，表示该用户具有该权限的（最高）等级。如果用户不存在，或者查询的权限没有定义，则应该返回 false。  **样例输入**  3 crm:2 git:3 game 4 hr 1 crm:2 it 3 crm:1 git:1 game dev 2 git:3 game qa 1 git:2 3 alice 1 hr bob 2 it qa charlie 1 dev 9 alice game alice crm:2 alice git:0 bob git bob poweroff charlie game charlie crm charlie git:3 malice game  **样例输出**  false true false 2 false true false true false  **样例说明**  　　样例输入描述的场景中，各个用户实际的权限如下： 　　\* 用户 alice 具有 crm:2 权限 　　\* 用户 bob 具有 crm:1、git:2 和 game 权限 　　\* 用户 charlie 具有 git:3 和 game 权限 　　\* 用户 malice 未描述，因此不具有任何权限  **评测用例规模与约定**  　　评测用例规模： 　　\* 1 ≤ p, r, u ≤ 100 　　\* 1 ≤ q ≤ 10 000 　　\* 每个用户具有的角色数不超过 10，每种角色具有的权限种类不超过 10 　　约定： 　　\* 输入保证合法性，包括： 　　1) 角色对应的权限列表（R 段）中的权限都是之前（P 段）出现过的，权限可以重复出现，如果带等级的权限重复出现，以等级最高的为准 　　2) 用户对应的角色列表（U 段）中的角色都是之前（R 段）出现过的，如果多个角色都具有某一分等级权限，以等级最高的为准 　　3) 查询（Q 段）中的用户名和权限类名不保证在之前（U 段和 P 段）出现过 　　\* 前 20% 的评测用例只有一种角色 　　\* 前 50% 的评测用例权限都是不分等级的，查询也都不带等级 |

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

**问题描述**：（参见上文）。

**问题分析**：这个问题是RBAC问题，即基于角色的权限访问问题，是商业应用中必然出现的问题。商业应用中，通常将这些数据存储于[**数据库**](http://lib.csdn.net/base/mysql)中。这个问题只要采用合适的数据结果来存储有关数据即可，查询处理过程相对比较简单。

**程序说明**：程序中，使用了三个结构向量分别用来存储权限、角色和用户。查询用的数据则不需要存储。查询过程根据题意进行处理即可。[**测试**](http://lib.csdn.net/base/softwaretest)样例中，似乎没有使用权限数据，对权限数据不做任何处理也可以的100分，有点怪怪的。

提交后得100分的[**C++语言**](http://lib.csdn.net/base/c)程序如下：

**[cpp]** [view plain](http://blog.csdn.net/tigerisland45/article/details/56483841) [copy](http://blog.csdn.net/tigerisland45/article/details/56483841)

1. /\* CCF201612-3 权限查询 \*/
3. #include <iostream>
4. #include <string>
5. #include <vector>
6. #include <cstdlib>
8. **const** **int** NOVALUE = -1;
9. **const** **int** TRUE = -2;
10. **const** **int** FALSE = -3;
12. **using** **namespace** std;
14. // 权限
15. **struct** \_privilege {
16. string category;
17. **int** level;
18. };
19. vector<\_privilege> privilege;
21. // 角色
22. **struct** \_role {
23. string role;
24. **int** s;
25. vector<\_privilege> privilege;
26. };
27. vector<\_role> role;
29. // 用户
30. **struct** \_user {
31. string user;
32. **int** t;
33. vector<string> role;
34. };
35. vector<\_user> user;
37. string getcategory(string& s)
38. {
39. **int** pos = s.find(":");
41. **if**(pos == (**int**)string::npos)
42. **return** s;
43. **else**
44. **return** s.substr(0, pos);
45. }
47. **int** getlevel(string &s)
48. {
49. **int** pos = s.find(":");
51. **if**(pos == (**int**)string::npos)
52. **return** NOVALUE;
53. **else**
54. **return** atoi(s.substr(pos+1, s.length()-1).c\_str());
55. }
57. **int** privilegematch(\_privilege& p1, \_privilege& p2)
58. {
59. **if**(p1.category != p2.category)
60. **return** FALSE;
61. **else** **if**(p2.level == NOVALUE) {
62. // 不分等级查询
63. **if**(p1.level == NOVALUE)
64. **return** TRUE;
65. **else**
66. **return** p1.level;
67. } **else** {    // p2.level >= 0
68. // 分等级查询
69. **if**(p1.level == NOVALUE)
70. **return** TRUE;
71. **else** {
72. **if**(p1.level >= p2.level)
73. **return** TRUE;
74. **else**
75. **return** FALSE;
76. }
77. }
78. }
80. **int** rolematch(string& rl, \_privilege& prvl)
81. {
82. **int** ans = FALSE;
84. **for**(**int** i=0; i<(**int**)role.size(); i++) {
85. **if**(role[i].role == rl) {
86. **for**(**int** j=0; j<role[i].s; j++) {
87. **int** rt = privilegematch(role[i].privilege[j], prvl);
88. **if**(rt > ans)
89. ans = rt;
90. }
91. }
92. }
94. **return** ans;
95. }
97. **int** query(string& usr, \_privilege& prvl)
98. {
99. **for**(**int** i=0; i<(**int**)user.size(); i++) {
100. **if**(user[i].user == usr) {
101. **int** ans = FALSE;
102. **for**(**int** j=0; j<user[i].t; j++) {
103. **int** rt = rolematch(user[i].role[j], prvl);
104. **if**(rt > ans)
105. ans = rt;
106. }
107. **return** ans;
108. }
109. }
111. **return** FALSE;
112. }
114. **int** main()
115. {
116. **int** p, r, u, q;
118. // 输入权限类别数量
119. cin >> p;
120. // 输入权限类别
121. string c;
122. \_privilege prvl;
123. **for**(**int** i=1; i<=p; i++) {
124. cin >> c;
126. prvl.category = getcategory(c);
127. prvl.level = getlevel(c);
128. privilege.push\_back(prvl);
129. }
131. // 输入角色数量r
132. cin >> r;
133. // 输入角色
134. **for**(**int** i=1; i<=r; i++) {
135. \_role rl;
137. cin >> rl.role >> rl.s;
139. **for**(**int** j=1; j<=rl.s; j++) {
140. cin >> c;
142. prvl.category = getcategory(c);
143. prvl.level = getlevel(c);
144. rl.privilege.push\_back(prvl);
145. }
146. role.push\_back(rl);
147. }
149. // 输入用户数量u
150. cin >> u;
151. // 输入用户
152. **for**(**int** i=1; i<=u; i++) {
153. \_user us;
155. cin >> us.user >> us.t;
157. **for**(**int** j=1; j<=us.t; j++) {
158. cin >> c;
160. us.role.push\_back(c);
161. }
162. user.push\_back(us);
163. }
165. // 输入查询数量q
166. cin >> q;
167. string suser;
168. **for**(**int** i=1; i<=q; i++) {
169. // 查询输入
170. cin >> suser >> c;
172. // 权限分解
173. prvl.category = getcategory(c);
174. prvl.level = getlevel(c);
176. // 查询处理
177. **int** ans = query(suser, prvl);
179. // 输出结果
180. **if**(ans == TRUE)
181. cout << "true" << endl;
182. **else** **if**(ans == FALSE)
183. cout << "false" << endl;
184. **else**
185. cout << ans << endl;
186. }
188. **return** 0;
189. }
191. /\* 这个问题是RBAC问题，即基于角色的权限访问问题，是商业应用中必然出现的问题 \*/