

【华为OD机考 统一考试机试C卷】文本统计分析（C+ + Java JavaScript Python）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

2023年11月份，华为官方已经将 华为OD机考：OD统一考试（A卷 / B卷）切换到 OD统一考试（C卷）和 OD统一考试（D卷） 。根据考友反馈：目前抽到的试卷为B卷或C卷/D卷，其中C卷居多，按照之前的经验C卷D卷部分考题会复用A卷/B卷题，博主正积极从考过的同学收集C卷和D卷真题，可以查看下面的真题目录。

真题目录： [华为OD机考机试 真题目录（C卷 + D卷 + B卷 + A卷） + 考点说明](#)

专栏： [2023华为OD机试\(B卷+C卷+D卷\) \(C++JavaJSPy\)](#)

华为OD面试真题精选： [华为OD面试真题精选](#)

在线OJ： [点击立即刷题，模拟真实机考环境](#) [华为OD机考B卷C卷华为OD机考华为OD机考B卷华为OD机试B卷华为OD机试C卷华为OD机考C卷华为OD机考D卷题目华为OD机考C卷/D卷答案华为OD机考C卷/D卷解析](#)
[华为OD机考C卷和D卷真题华为OD机考C卷和D卷题解](#)

题目描述

有一个文件，包含以一定规则写作的文本，请统计文件中包含的文本数量。

规则如下：

- 1. 文本以“;”分隔，最后一条可以没有“;”，但空文本不能算语句，比如“COMMAND A;”只能算一条语句。注意，无字符/空白字符/制表符都算作“空”文本；
- 2. 文本可以跨行，比如下面，是一条文本，而不是三条；

```
1 | COMMAND A
2 | AND
3 | COMMAND B;
```

- 3. 文本支持字符串，字符串为成对的单引号(')或者成对的双引号(")，字符串可能出现用转义字符(\)处理的单双引号("your input is")和转义字符本身，比如

```
1 | COMMAND A "Say \"hello\"";
```

- 4. 支持注释，可以出现在字符串之外的任意位置注释以“--”开头，到换行结束，比如：

```
1 | COMMAND A; --this is comment
2 | COMMAND --comment
3 | A AND COMMAND B;
```

注意字符串内的"--"，不是注释。

输入描述

文本文件

输出描述

包含的文本数量

用例

输入

```
1 | COMMAND TABLE IF EXISTS "UNITED STATE";
2 | COMMAND A GREAT (
3 | ID ADSAB,
4 | download_length INTE-GER, -- test
5 | file_name TEXT,
6 | guid TEXT,
7 | mime_type TEXT,
8 | notifica-tionid INTEGER,
9 | original_file_name TEXT,
10 | pause_reason_type INTEGER,
11 | resumable_flag INTEGER,
12 | start_time INTEGER,
13 | state INTEGER,
14 | folder TEXT,
15 | path TEXT,
16 | total_length INTE-GER,
17 | url TEXT
18 | );
```

输出

题意解读

题目要求编写一个程序来统计一个文本文件中包含的文本数量。这里的“文本”指的是符合一定规则的字符串序列。具体规则如下：

1. 文本以分号(;)分隔，最后一条文本可以没有分号结尾。
2. 如果一段文本只包含空白字符（如空格、制表符等），则不算作一条有效文本。例如， "COMMAND A; ;" 中只有一条有效文本。 "COMMAND A; B;" 为两条有效文本。
3. 文本可以跨越多行。也就是说，一个文本的内容可以分布在多个连续的行中，这些行合起来算作一条文本。
4. 文本支持字符串，字符串可以用单引号(')或双引号(")包裹。字符串内部可能包含转义的引号（例如 "Say \"hello\"" ）和转义字符本身（例如 \ ）。
5. 在单引号和双引号的 ; , 无法作为一条文本结束的标志。
6. 支持注释，注释以连续的两个减号(--)开头，并且一直延续到当前行的末尾。注释只能出现在字符串之外的位置。在字符串内的减号不算作注释的开始。在注释后面的 ; , 无法作为一条文本结束的标志。
7. 单引号和双引号内的注释失效

```
1 | COMMAND A; --this is comment
2 | COMMAND -comment
3 | A AND COMMAND B;
4 | 上面文本数为2
5 |
6 | COMMAND A --this is ; comment;
7 | COMMAND -comment
8 | A AND COMMAND B;
9 | 文本数为1
```

备注：这里博主默认是注释【后面】的 ; , 是无法作为文本结束的标志。但是有读者提出是不是默认带注释的那一行的分号（即那一行分号的【前面】和【后面】）都无法作为文本结束的标志？机考时如果不是100%通过率，可以试试：带注释的那一行的分号都无法作为文本结束的标志

解题思路

1. 遍历每一行：

- 遍历累积的文本中的每一个字符，使用一个计数器来跟踪文本的数量。
- 使用两个布尔变量 `inString` 和 `inComment` 来分别跟踪当前位置是否在字符串或注释内部。

2. 处理注释：

- 如果当前字符和下一个字符都是减号 `-`，并且不在字符串内，则标记为注释开始。
- 在注释内部，忽略所有字符直到遇到换行符，然后标记注释结束。

3. 处理字符串：

- 如果遇到单引号或双引号，并且不在字符串内，标记为字符串开始，并记录使用的分隔符。
- 在字符串内部，如果再次遇到相同的分隔符，检查是否为转义字符（即是否有连续的两个相同分隔符）。
- 如果不是转义字符，则标记字符串结束。

4. 计数文本：

- 如果遇到分号，并且不在字符串内，且当前文本不为空（即至少有一个非空白字符），则增加计数器，并标记当前文本为空。
- 如果遇到非空白字符，并且不在字符串内，标记当前文本为非空。

5. 处理最后一个文本：

- 遍历结束后，如果最后一个文本没有闭合的分号，且不为空，则增加计数器。

C++

```
1  #include <iostream>
2  #include <string>
3
4  // 统计文本中的文本数量
5  int countTexts(const std::string& input) {
6      // 初始化计数器
7      int count = 0;
8      // 标记是否在字符串内部
9      bool inString = false;
10     // 标记是否在注释内部
11     bool inComment = false;
12     // 记录字符串分隔符
13     char stringDelimiter = 0;
14     // 标记当前是否为空文本（即没有遇到非空白字符）
15     bool isEmpty = true;
16 }
```

```
17 // 遍历输入文本的每个字符
18 for (size_t i = 0; i < input.length(); ++i) {
19     // 当前字符
20     char c = input[i];
21     // 下一个字符 (如果存在)
22     char nextChar = (i + 1 < input.length()) ? input[i + 1] : '\0';
23
24     // 如果在注释中
25     if (inComment) {
26         // 如果遇到换行符, 则注释结束
27         if (c == '\n') {
28             inComment = false;
29         }
30         continue;
31     }
32
33     // 如果遇到连续的两个减号, 并且不在字符串内, 则进入注释状态
34     if (c == '-' && nextChar == '-' && !inString) {
35         inComment = true;
36         i++; // 跳过下一个减号
37         continue;
38     }
39
40     // 如果遇到单引号或双引号, 并且不在字符串内, 则进入字符串状态
41     if ((c == '\'' || c == '\"') && !inString) {
42         inString = true;
43         stringDelimiter = c;
44         isEmpty = false;
45         continue;
46     }
47
48     // 如果在字符串内, 并且遇到了相同的分隔符, 则检查是否为转义
49     if (c == stringDelimiter && inString) {
50         if (nextChar == stringDelimiter) {
51             i++; // 跳过转义的引号
52         } else {
53             inString = false; // 字符串结束
54         }
55         continue;
56     }
57 }
```

```
58
59 // 如果遇到分号, 并且不在字符串内, 则增加计数器
60 if (c == ';' && !inString) {
61     if (!isEmpty) {
62         count++;
63         isEmpty = true;
64     }
65     continue;
66 }
67
68 // 如果遇到非空白字符, 并且不在字符串内, 则标记为非空文本
69 if (!isspace(c) && !inString) {
70     isEmpty = false;
71 }
72 }
73
74 // 如果最后一个文本没有闭合的分号, 则增加计数器
75 if (!isEmpty) {
76     count++; // 最后一个文本没有闭合分号
77 }
78
79 return count;
80 }
81 // 主函数
82 int main() {
83     // 创建字符串用于获取用户输入
84     std::string input;
85     // 获取用户输入直到EOF
86     for (std::string line; std::getline(std::cin, line);) {
87         // 将读取的每一行追加到input字符串, 并添加换行符
88         input += line + "\n";
89     }
90     // 输出文本统计结果
91     std::cout << countTexts(input) << std::endl;
92     return 0;
93 }
```

Java

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         // 创建Scanner对象用于获取用户输入
6         Scanner scanner = new Scanner(System.in);
7         // 使用StringBuilder来构建整个输入文本
8         StringBuilder input = new StringBuilder();
9         // 循环读取每一行输入直到没有新的输入
10        while (scanner.hasNextLine()) {
11            // 将读取的每一行追加到StringBuilder对象, 并添加换行符
12            input.append(scanner.nextLine()).append("\n");
13        }
14        // 关闭Scanner对象
15        scanner.close();
16        // 输出文本统计结果
17        System.out.println(countTexts(input.toString()));
18    }
19
20    // 统计文本中的文本数量
21    private static int countTexts(String input) {
22        // 初始化计数器
23        int count = 0;
24        // 标记是否在字符串内部
25        boolean inString = false;
26        // 标记是否在注释内部
27        boolean inComment = false;
28        // 记录字符串分隔符
29        char stringDelimiter = 0;
30        // 标记当前是否为空文本 (即没有遇到非空白字符)
31        boolean isEmpty = true;
32
33        // 遍历输入文本的每个字符
34        for (int i = 0; i < input.length(); i++) {
35            // 当前字符
36            char c = input.charAt(i);
37            // 下一个字符 (如果存在)
38            char nextChar = (i + 1 < input.length()) ? input.charAt(i + 1) : '\0';
39
40            // 如果在注释中
41
```



```
41 if (inComment) {
42     // 如果遇到换行符, 则注释结束
43     if (c == '\n') {
44         inComment = false;
45     }
46     continue;
47 }
48
49 // 如果遇到连续的两个减号, 并且不在字符串内, 则进入注释状态
50 if (c == '-' && nextChar == '-' && !inString) {
51     inComment = true;
52     continue;
53 }
54
55 // 如果遇到单引号或双引号, 并且不在字符串内, 则进入字符串状态
56 if ((c == '\'' || c == '\"') && !inString) {
57     inString = true;
58     stringDelimiter = c;
59     isEmpty = false;
60     continue;
61 }
62
63 // 如果在字符串内, 并且遇到了相同的分隔符, 则检查是否为转义
64 if (c == stringDelimiter && inString) {
65     if (nextChar == stringDelimiter) {
66         i++; // 跳过转义的引号
67     } else {
68         inString = false; // 字符串结束
69     }
70     continue;
71 }
72
73 // 如果遇到分号, 并且不在字符串内, 则增加计数器
74 if (c == ';' && !inString) {
75     if (!isEmpty) {
76         count++;
77         isEmpty = true;
78     }
79     continue;
80 }
81 }
```

```
82
83     // 如果遇到非空白字符, 并且不在字符串内, 则标记为非空文本
84     if (!Character.isWhitespace(c) && !inString) {
85         isEmpty = false;
86     }
87 }
88
89 // 如果最后一个文本没有闭合的分号, 则增加计数器
90 if (!isEmpty) {
91     count++; // 最后一个文本没有闭合分号
92 }
93
94 return count;
95 }
```

javaScript

```
1  const readline = require('readline');
2
3  // 创建readline接口实例
4  const rl = readline.createInterface({
5      input: process.stdin,
6      output: process.stdout,
7      terminal: false
8  });
9
10 // 使用字符串来构建整个输入文本
11 let input = '';
12
13 // 事件监听, 读取每一行输入
14 rl.on('line', function(line) {
15     // 将读取的每一行追加到input字符串, 并添加换行符
16     input += line + "\n";
17 });
18
19 // 监听流的结束事件
20 rl.on('close', function() {
21     // 输出文本统计结果
22     console.log(countTexts(input));
23 })
```

```
23 });
24
25 // 统计文本中的文本数量
26 function countTexts(input) {
27     // 初始化计数器
28     let count = 0;
29     // 标记是否在字符串内部
30     let inString = false;
31     // 标记是否在注释内部
32     let inComment = false;
33     // 记录字符串分隔符
34     let stringDelimiter = '';
35     // 标记当前是否为空文本 (即没有遇到非空白字符)
36     let isEmpty = true;
37
38     // 遍历输入文本的每个字符
39     for (let i = 0; i < input.length; i++) {
40         // 当前字符
41         let c = input[i];
42         // 下一个字符 (如果存在)
43         let nextChar = (i + 1 < input.length) ? input[i + 1] : '\0';
44
45         // 如果在注释中
46         if (inComment) {
47             // 如果遇到换行符, 则注释结束
48             if (c === '\n') {
49                 inComment = false;
50             }
51             continue;
52         }
53
54         // 如果遇到连续的两个减号, 并且不在字符串内, 则进入注释状态
55         if (c === '-' && nextChar === '-' && !inString) {
56             inComment = true;
57             i++; // 跳过下一个减号
58             continue;
59         }
60
61         // 如果遇到单引号或双引号, 并且不在字符串内, 则进入字符串状态
62         if ((c === '\'' || c === '\"') && !inString) {
63             // 进入字符串状态
64             inString = true;
65             stringDelimiter = c;
66             continue;
67         }
68
69         // 如果不是空白字符, 则计数
70         if (c !== ' ' && c !== '\n' && c !== '\0') {
71             count++;
72             isEmpty = false;
73         }
74     }
75
76     // 如果为空文本, 则返回0
77     if (isEmpty) {
78         return 0;
79     }
80
81     // 返回统计结果
82     return count;
83 }
```

```
64         inString = true;
65         stringDelimiter = c;
66         isEmpty = false;
67         continue;
68     }
69
70     // 如果在字符串内, 并且遇到了相同的分隔符, 则检查是否为转义
71     if (c === stringDelimiter && inString) {
72         if (nextChar === stringDelimiter) {
73             i++; // 跳过转义的引号
74         } else {
75             inString = false; // 字符串结束
76         }
77         continue;
78     }
79
80     // 如果遇到分号, 并且不在字符串内, 则增加计数器
81     if (c === ';' && !inString) {
82         if (!isEmpty) {
83             count++;
84             isEmpty = true;
85         }
86         continue;
87     }
88
89     // 如果遇到非空白字符, 并且不在字符串内, 则标记为非空文本
90     if (!/\s/.test(c) && !inString) {
91         isEmpty = false;
92     }
93 }
94
95 // 如果最后一个文本没有闭合的分号, 则增加计数器
96 if (!isEmpty) {
97     count++; // 最后一个文本没有闭合分号
98 }
99
100 return count;
}
```

Python

```
1 import sys
2
3 # 统计文本中的文本数量
4 def count_texts(input):
5     # 初始化计数器
6     count = 0
7     # 标记是否在字符串内部
8     in_string = False
9     # 标记是否在注释内部
10    in_comment = False
11    # 记录字符串分隔符
12    string_delimiter = ''
13    # 标记当前是否为空文本 (即没有遇到非空白字符)
14    isEmpty = True
15
16    # 遍历输入文本的每个字符
17    for i, c in enumerate(input):
18        # 下一个字符 (如果存在)
19        next_char = input[i + 1] if i + 1 < len(input) else '\0'
20
21        # 如果在注释中
22        if in_comment:
23            # 如果遇到换行符, 则注释结束
24            if c == '\n':
25                in_comment = False
26                continue
27
28        # 如果遇到连续的两个减号, 并且不在字符串内, 则进入注释状态
29        if c == '-' and next_char == '-' and not in_string:
30            in_comment = True
31            i += 1 # 跳过下一个减号
32            continue
33
34        # 如果遇到单引号或双引号, 并且不在字符串内, 则进入字符串状态
35        if (c == '\'' or c == '\"') and not in_string:
36            in_string = True
37            string_delimiter = c
38            isEmpty = False
39            continue
40
41
```

```
41 # 如果在字符串内，并且遇到了相同的分隔符，则检查是否为转义
42
43 if c == string_delimiter and in_string:
44     if next_char == string_delimiter:
45         i += 1 # 跳过转义的引号
46     else:
47         in_string = False # 字符串结束
48         continue
49
50 # 如果遇到分号，并且不在字符串内，则增加计数器
51 if c == ';' and not in_string:
52     if not isEmpty:
53         count += 1
54         isEmpty = True
55     continue
56
57 # 如果遇到非空白字符，并且不在字符串内，则标记为非空文本
58 if not c.isspace() and not in_string:
59     isEmpty = False
60
61 # 如果最后一个文本没有闭合的分号，则增加计数器
62 if not isEmpty:
63     count += 1 # 最后一个文本没有闭合分号
64
65 return count
66
67 # 主函数
68 if __name__ == "__main__":
69     # 使用字符串来构建整个输入文本
70     input = sys.stdin.read()
71     # 输出文本统计结果
72     print(count_texts(input))
```

文章目录

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

题目描述

输入描述

输出描述

用例

题意解读

解题思路

C++

Java

JavaScript

Python

机考真题 华为OD



CSDN @算法大师