【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；

```
COMMAND A
AND
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

```
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：

```
COMMAND A; --this is comment
COMMAND --comment
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

```
COMMAND TABLE IF EXISTS "UNITED STATE";
COMMAND A GREAT (
ID ADSAB,
download_length INTE-GER, -- test
file_name TEXT,
guid TEXT,
mime_type TEXT,
notifica-tionid INTEGER,
original_file_name TEXT,
pause_reason_type INTEGER,
resumable_flag INTEGER,
start_time INTEGER,
state INTEGER,
folder TEXT,
path TEXT,
total_length INTE-GER,
url TEXT
);
```

输出

```
1 | 2
```

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

```
COMMAND A; --this is comment
COMMAND -comment
A AND COMMAND B;
上面文本数为2

COMMAND A --this is ; comment;
COMMAND -comment
A AND COMMAND B;
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++

```cpp
#include <iostream>
#include <string>

// 统计文本中的文本数量
int countTexts(const std::string& input) {
    // 初始化计数器
    int count = 0;
    // 标记是否在字符串内部
    bool inString = false;
    // 标记是否在注释内部
    bool inComment = false;
    // 记录字符串分隔符
    char stringDelimiter = 0;
    // 标记当前是否为空文本（即没有遇到非空白字符）
    bool isEmpty = true;

```

```
    // 遍历输入文本的每个字符
    for (size_t i = 0; i < input.length(); ++i) {
        // 当前字符
        char c = input[i];
        // 下一个字符（如果存在）
        char nextChar = (i + 1 < input.length()) ? input[i + 1] : '\0';

        // 如果在注释中
        if (inComment) {
            // 如果遇到换行符，则注释结束
            if (c == '\n') {
                inComment = false;
            }
            continue;
        }

        // 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
        if (c == '-' && nextChar == '-' && !inString) {
            inComment = true;
            i++; // 跳过下一个减号
            continue;
        }

        // 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
        if ((c == '\'' || c == '\"') && !inString) {
            inString = true;
            stringDelimiter = c;
            isEmpty = false;
            continue;
        }

        // 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
        if (c == stringDelimiter && inString) {
            if (nextChar == stringDelimiter) {
                i++; // 跳过转义的引号
            } else {
                inString = false; // 字符串结束
            }
            continue;
        }

```

```
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

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // 创建Scanner对象用于获取用户输入
        Scanner scanner = new Scanner(System.in);
        // 使用StringBuilder来构建整个输入文本
        StringBuilder input = new StringBuilder();
        // 循环读取每一行输入直到没有新的输入
        while (scanner.hasNextLine()) {
            // 将读取的每一行追加到StringBuilder对象，并添加换行符
            input.append(scanner.nextLine()).append("\n");
        }
        // 关闭Scanner对象
        scanner.close();
        // 输出文本统计结果
        System.out.println(countTexts(input.toString()));
    }

    // 统计文本中的文本数量
    private static int countTexts(String input) {
        // 初始化计数器
        int count = 0;
        // 标记是否在字符串内部
        boolean inString = false;
        // 标记是否在注释内部
        boolean inComment = false;
        // 记录字符串分隔符
        char stringDelimiter = 0;
        // 标记当前是否为空文本（即没有遇到非空白字符）
        boolean isEmpty = true;

        // 遍历输入文本的每个字符
        for (int i = 0; i < input.length(); i++) {
            // 当前字符
            char c = input.charAt(i);
            // 下一个字符（如果存在）
            char nextChar = (i + 1 < input.length()) ? input.charAt(i + 1) : '\0';

            // 如果在注释中
```

```
        if (inComment) {
            // 如果遇到换行符，则注释结束
            if (c == '\n') {
                inComment = false;
            }
            continue;
        }

        // 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
        if (c == '-' && nextChar == '-' && !inString) {
            inComment = true;
            continue;
        }

        // 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
        if ((c == '\'' || c == '\"') && !inString) {
            inString = true;
            stringDelimiter = c;
            isEmpty = false;
            continue;
        }

        // 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
        if (c == stringDelimiter && inString) {
            if (nextChar == stringDelimiter) {
                i++; // 跳过转义的引号
            } else {
                inString = false; // 字符串结束
            }
            continue;
        }

        // 如果遇到分号，并且不在字符串内，则增加计数器
        if (c == ';' && !inString) {
            if (!isEmpty) {
                count++;
                isEmpty = true;
            }
            continue;
        }
```

```
            // 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
            if (!Character.isWhitespace(c) && !inString) {
                isEmpty = false;
            }
        }

        // 如果最后一个文本没有闭合的分号，则增加计数器
        if (!isEmpty) {
            count++; // 最后一个文本没有闭合分号
        }

        return count;
    }
}
```

javaScript

```
const readline = require('readline');

// 创建readline接口实例
const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout,
    terminal: false
});

// 使用字符串来构建整个输入文本
let input = '';

// 事件监听，读取每一行输入
rl.on('line', function(line) {
    // 将读取的每一行追加到input字符串，并添加换行符
    input += line + "\n";
});

// 监听流的结束事件
rl.on('close', function() {
    // 输出文本统计结果
    console.log(countTexts(input));
```

```
});

// 统计文本中的文本数量
function countTexts(input) {
    // 初始化计数器
    let count = 0;
    // 标记是否在字符串内部
    let inString = false;
    // 标记是否在注释内部
    let inComment = false;
    // 记录字符串分隔符
    let stringDelimiter = '';
    // 标记当前是否为空文本（即没有遇到非空白字符）
    let isEmpty = true;

    // 遍历输入文本的每个字符
    for (let i = 0; i < input.length; i++) {
        // 当前字符
        let c = input[i];
        // 下一个字符（如果存在）
        let nextChar = (i + 1 < input.length) ? input[i + 1] : '\0';

        // 如果在注释中
        if (inComment) {
            // 如果遇到换行符，则注释结束
            if (c === '\n') {
                inComment = false;
            }
            continue;
        }

        // 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
        if (c === '-' && nextChar === '-' && !inString) {
            inComment = true;
            i++; // 跳过下一个减号
            continue;
        }

        // 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
        if ((c === '\'' || c === '\"') && !inString) {
```

```
            inString = true;
            stringDelimiter = c;
            isEmpty = false;
            continue;
        }

        // 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
        if (c === stringDelimiter && inString) {
            if (nextChar === stringDelimiter) {
                i++; // 跳过转义的引号
            } else {
                inString = false; // 字符串结束
            }
            continue;
        }

        // 如果遇到分号，并且不在字符串内，则增加计数器
        if (c === ';' && !inString) {
            if (!isEmpty) {
                count++;
                isEmpty = true;
            }
            continue;
        }

        // 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
        if (!/\s/.test(c) && !inString) {
            isEmpty = false;
        }
    }

    // 如果最后一个文本没有闭合的分号，则增加计数器
    if (!isEmpty) {
        count++; // 最后一个文本没有闭合分号
    }

    return count;
}
```

Python

```python
import sys

# 统计文本中的文本数量
def count_texts(input):
    # 初始化计数器
    count = 0
    # 标记是否在字符串内部
    in_string = False
    # 标记是否在注释内部
    in_comment = False
    # 记录字符串分隔符
    string_delimiter = ''
    # 标记当前是否为空文本（即没有遇到非空白字符）
    isEmpty = True

    # 遍历输入文本的每个字符
    for i, c in enumerate(input):
        # 下一个字符（如果存在）
        next_char = input[i + 1] if i + 1 < len(input) else '\0'

        # 如果在注释中
        if in_comment:
            # 如果遇到换行符，则注释结束
            if c == '\n':
                in_comment = False
            continue

        # 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
        if c == '-' and next_char == '-' and not in_string:
            in_comment = True
            i += 1  # 跳过下一个减号
            continue

        # 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
        if (c == '\'' or c == '\"') and not in_string:
            in_string = True
            string_delimiter = c
            isEmpty = False
            continue

```

```
        # 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
        if c == string_delimiter and in_string:
            if next_char == string_delimiter:
                i += 1  # 跳过转义的引号
            else:
                in_string = False  # 字符串结束
            continue

        # 如果遇到分号，并且不在字符串内，则增加计数器
        if c == ';' and not in_string:
            if not isEmpty:
                count += 1
                isEmpty = True
            continue

        # 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
        if not c.isspace() and not in_string:
            isEmpty = False

    # 如果最后一个文本没有闭合的分号，则增加计数器
    if not isEmpty:
        count += 1  # 最后一个文本没有闭合分号

    return count

# 主函数
if __name__ == "__main__":
    # 使用字符串来构建整个输入文本
    input = sys.stdin.read()
    # 输出文本统计结果
    print(count_texts(input))
```

文章目录