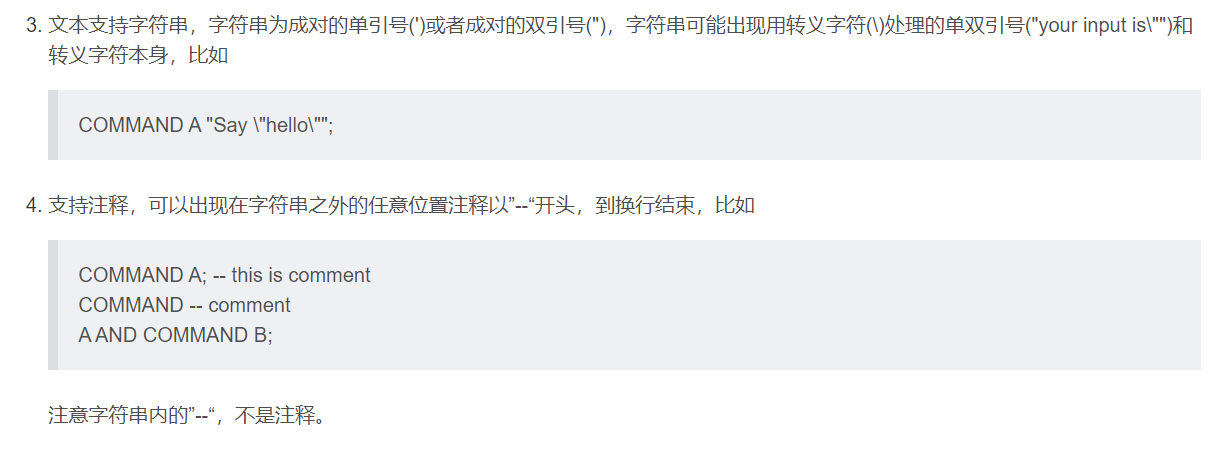
# **E卷-文本统计分析[200分]（ Java | Python3 | C++ | C语言 | JsNode | Go ）**







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,

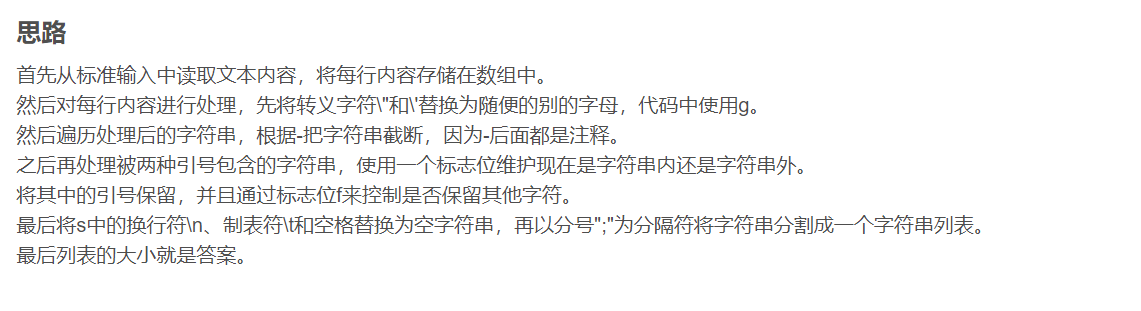
total\\_length INTE-GER,

url TEXT

);



2





import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in); // 创建一个扫描器读取输入

List<String> res = new

ArrayList<>(); // 初始化一个列表用于存储每行输入的文本

while (scanner.hasNextLine()) {

res.add(scanner.nextLine()); // 读取每一行输入并添加到res列表中

}

for (int i = 0; i < res.size(); i++) {

String tmp = res.get(i).replace("\\\"",

"g"); // 替换转义双引号为字符'g'

tmp = tmp.replace("\\\'", "g"); // 替换转义单引号为字符'g'

String now = ""; // 临时字符串，用于存储当前行的有效内容

for (char j : tmp.toCharArray()) {

if (j == '-') { // 如果遇到'-'，说明可能是注释，停止读取

break;

}

now += j; // 将非注释部分字符加入now

}

res.set(i, now); // 将处理后的now赋值回res的当前行

}

String s = ""; // 用于存储所有有效文本

boolean f = false; // 标志位，用于判断当前是否在引号内

for (String i : res) { // 遍历处理后的每一行

for (char j : i.toCharArray()) { // 遍历当前行的每个字符

if (j == '\"' || j == '\'') { // 如果遇到引号

s += j; // 引号加入s

f = !f; // 切换标志位

continue;

}

if (!f) { // 如果不在引号内

s += j; // 非引号内容加入s

}

}

}

s = s.replace("\n", ""); // 删除换行符

s = s.replace("\t", ""); // 删除制表符

s = s.replace(" ", ""); // 删除空格

String[] t = s.split(";"); // 按分号拆分字符串，得到文本段数组

int cnt = 0; // 初始化计数器

for (String i : t) { // 遍历每个文本段

if (!i.equals("")) { // 如果文本段非空

cnt += 1; // 计数器加一

}

}

System.out.println(cnt); // 输出计数结果

}

}



import sys # 导入sys模块，用于读取标准输入

res = [] # 初始化一个空列表，用于存储每行读入的内容

# 读取标准输入的每一行，并将其加入res列表中

for line in sys.stdin:

res.append(line)

# 对res中的每一行进行处理

for i in range(len(res)):

# 将行中的转义双引号和单引号替换为普通字符'g'

tmp = res[i].replace('\\"', "g")

tmp = tmp.replace("\\'", "g")

now = "" # 临时字符串，用于存储当前处理的文本

# 遍历当前行中的每个字符，遇到'-'时停止，其他字符加入now

for j in tmp:

if j == "-":

break

now += j

res[i] = now # 将处理后的now赋值回res的当前行

s = "" # 初始化一个空字符串s，用于存储所有处理后的文本

f = 0 # 标记变量，用于判断当前是否在引号内

# 遍历处理后的每一行

for i in res:

for j in i: # 遍历当前行中的每个字符

if j == '"' or j == "'": # 如果遇到双引号或单引号

s += j # 将引号加入s

f ^= 1 # 切换标记变量，进入或退出引号

continue

if not f: # 如果不在引号内

s += j # 将字符加入s

# 删除字符串中的换行符、制表符和空格

s = s.replace("\n", "")

s = s.replace("\t", "")

s = s.replace(" ", "")

# 以分号拆分字符串，得到一个列表t

t = s.split(";")

cnt = 0 # 初始化计数器cnt，用于统计非空文本段的数量

# 遍历拆分后的每个文本段

for i in t:

if i != "": # 如果文本段非空

cnt += 1 # 计数器加一

print(cnt) # 输出计数结果



#include <bits/stdc++.h>

using namespace std;

int main() {

string line;

vector<string> res; // 初始化一个向量用于存储每行输入的文本

// 从标准输入读取每一行，直到EOF

while (getline(cin, line)) {

res.push\_back(line); // 将每行加入res向量

}

// 遍历res向量中的每一行

for (size\_t i = 0; i < res.size(); i++) {

string tmp = res[i];

string::size\_type pos;

// 替换转义双引号为字符'g'

while ((pos = tmp.find("\\\"")) != string::npos)

tmp.replace(pos, 2, "g");

// 替换转义单引号为字符'g'

while ((pos = tmp.find("\\\'")) != string::npos)

tmp.replace(pos, 2, "g");

string now = ""; // 临时字符串，用于存储当前行的有效内容

for (char j : tmp) {

if (j == '-') // 如果遇到'-'，说明可能是注释，停止读取

break;

now += j; // 将非注释部分字符加入now

}

res[i] = now; // 将处理后的now赋值回res的当前行

}

string s = ""; // 用于存储所有有效文本

bool f = false; // 标志位，用于判断当前是否在引号内

// 遍历处理后的每一行

for (string& i : res) {

for (char j : i) { // 遍历当前行的每个字符

if (j == '\"' || j == '\'') { // 如果遇到引号

s += j; // 引号加入s

f = !f; // 切换标志位

continue;

}

if (!f) { // 如果不在引号内

s += j; // 非引号内容加入s

}

}

}

// 删除字符串中的换行符、制表符和空格

s.erase(remove(s.begin(), s.end(), '\n'), s.end()); // 删除换行符

s.erase(remove(s.begin(), s.end(), '\t'), s.end()); // 删除制表符

s.erase(remove(s.begin(), s.end(), ' '), s.end()); // 删除空格

stringstream ss(s); // 使用字符串流进行分割

string item;

int cnt = 0; // 初始化计数器

// 遍历分割得到的每个项

while (getline(ss, item, ';')) {

if (!item.empty()) { // 如果项非空

cnt += 1; // 计数器加一

}

}

cout << cnt << endl; // 输出计数结果

return 0;

}



#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#define MAX\_LINES 1000

#define MAX\_LENGTH 1000

int main() {

char line[MAX\_LENGTH], \*res[MAX\_LINES];

int count = 0;

// 从标准输入读取每一行，直到EOF

while (fgets(line, sizeof(line), stdin) != NULL) {

res[count] = strdup(line);

count++;

}

// 遍历res数组中的每一行

for (int i = 0; i < count; i++) {

char tmp[MAX\_LENGTH];

strcpy(tmp, res[i]);

char\* pos;

// 替换转义双引号为字符'g'

while ((pos = strstr(tmp, "\\\"")) != NULL) {

\*pos = 'g';

memmove(pos + 1, pos + 2, strlen(pos + 2) + 1);

}

// 替换转义单引号为字符'g'

while ((pos = strstr(tmp, "\\\'")) != NULL) {

\*pos = 'g';

memmove(pos + 1, pos + 2, strlen(pos + 2) + 1);

}

char now[MAX\_LENGTH] = "";

// 如果遇到'-'，说明可能是注释，停止读取

for (char\* j = tmp; \*j; j++) {

if (\*j == '-')

break;

strncat(now, j, 1);

}

strcpy(res[i], now); // 将处理后的now赋值回res的当前行

}

char s[MAX\_LINES \* MAX\_LENGTH] = "";

int f = 0; // 标志位，用于判断当前是否在引号内

// 遍历处理后的每一行

for (int i = 0; i < count; i++) {

for (char\* j = res[i]; \*j; j++) { // 遍历当前行的每个字符

if (\*j == '\"' || \*j == '\'') { // 如果遇到引号

strncat(s, j, 1); // 引号加入s

f ^= 1; // 切换标志位

continue;

}

if (!f) { // 如果不在引号内

strncat(s, j, 1); // 非引号内容加入s

}

}

}

// 删除字符串中的换行符、制表符和空格

for (char\* p = s; \*p; p++) {

if (\*p == '\n' || \*p == '\t' || \*p == ' ') {

memmove(p, p + 1, strlen(p));

p--;

}

}

int cnt = 0; // 初始化计数器

char\* token = strtok(s, ";"); // 按分号拆分字符串，得到文本段

// 遍历拆分后的每个文本段

while (token) {

if (strlen(token) != 0) { // 如果文本段非空

cnt++; // 计数器加一

}

token = strtok(NULL, ";");

}

printf("%d\n", cnt); // 输出计数结果

// 释放分配的内存

for (int i = 0; i < count; i++) {

free(res[i]);

}

return 0;

}



const readline = require("readline");

const rl = readline.createInterface({

input: process.stdin,

output: process.stdout,

});

const res = []; // 初始化一个数组用于存储每行输入的文本

rl.on("line", (input) => {

res.push(input); // 将每行输入加入res数组

}).on("close", () => {

for (let i = 0; i < res.length; i++) {

let tmp = res[i].replace(/\\"/g, "g"); // 替换转义双引号为字符'g'

tmp = tmp.replace(/\\'/g, "g"); // 替换转义单引号为字符'g'

let now = ""; // 临时字符串，用于存储当前行的有效内容

for (let j = 0; j < tmp.length; j++) {

if (tmp[j] === "-") {

// 如果遇到'-'，说明可能是注释，停止读取

break;

}

now += tmp[j]; // 将非注释部分字符加入now

}

res[i] = now; // 将处理后的now赋值回res的当前行

}

let s = ""; // 用于存储所有有效文本

let f = false; // 标志位，用于判断当前是否在引号内

for (let i = 0; i < res.length; i++) {

// 遍历处理后的每一行

for (let j = 0; j < res[i].length; j++) {

// 遍历当前行的每个字符

if (res[i][j] === '"' || res[i][j] === "'") {

// 如果遇到引号

s += res[i][j]; // 引号加入 s

f = !f; // 切换标志位

continue;

}

if (!f) {

// 如果不在引号内

s += res[i][j]; // 非引号内容加入s

}

}

}

s = s.replace(/\n/g, ""); // 删除换行符

s = s.replace(/\t/g, ""); // 删除制表符

s = s.replace(/ /g, ""); // 删除空格

const t = s.split(";"); // 按分号拆分字符串，得到文本段数组

let cnt = 0; // 初始化计数器

for (let i = 0; i < t.length; i++) {

// 遍历每个文本段

if (t[i] !== "") {

// 如果文本段非空

cnt += 1; // 计数器加一

}

}

console.log(cnt); // 输出计数结果

});



package main

import (

"bufio"

"fmt"

"os"

"strings"

)

func main() {

scanner := bufio.NewScanner(os.Stdin)

var res []string // 初始化一个切片用于存储每行输入的文本

for scanner.Scan() {

res = append(res, scanner.Text()) // 将每行输入加入res切片

}

for i := range res {

tmp := strings.ReplaceAll(res[i], "\\\"", "g") // 替换转义双引号为字符'g'

tmp = strings.ReplaceAll(tmp, "\\'", "g") // 替换转义单引号为字符'g'

now := "" // 临时字符串，用于存储当前行的有效内容

for \_, j := range tmp {

if j == '-' { // 如果遇到'-'，说明可能是注释，停止读取

break

}

now += string(j) // 将非注释部分字符加入now

}

res[i] = now // 将处理后的now赋值回res的当前行

}

s := "" // 用于存储所有有效文本

f := false // 标志位，用于判断当前是否在引号内

for \_, i := range res { // 遍历处理后的每一行

for \_, j := range i { // 遍历当前行的每个字符

if j == '"' || j == '\'' { // 如果遇到引号

s += string(j) // 引号加入s

f = !f // 切换标志位

continue

}

if !f { // 如果不在引号内

s += string(j) // 非引号内容加入s

}

}

}

s = strings.ReplaceAll(s, "\n", "") // 删除换行符

s = strings.ReplaceAll(s, "\t", "") // 删除制表符

s = strings.ReplaceAll(s, " ", "") // 删除空格

t := strings.Split(s, ";") // 按分号拆分字符串，得到文本段切片

cnt := 0 // 初始化计数器

for \_, i := range t { // 遍历每个文本段

if i != "" { // 如果文本段非空

cnt += 1 // 计数器加一

}

}

fmt.Println(cnt) // 输出计数结果

}