【华为OD机考 统一考试机试C卷】 数据单元的变量替换（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。 可以先继续刷B卷， C卷和D卷的题目会放在现在大家购买的专栏内，不需要重新购买，请大家放心。

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

将一个csv格式的数据文件中包含有单元格引用的内容替换为对应单元格内容的实际值。

Comma seprated values（CSV）逗号分隔值，csv格式的数据文件使用逗号作为分隔符将各单位的内容进行分隔。

输入描述

1. 输入只有一行数据，用逗号分隔每个单元格，行尾没有逗号。最多26个单元格，对应编号A-Z。
2. 每个单元格的内容包含字母和数字，以及使用<>分隔的单元格引用，例如：表示引用第一个单元的值。
3. 每个单元格的内容，在替换前和替换后均不超过100个字符。
4. 引用单元格的位置不受限制，运行排在后面的单元格被排在前面的单元格引用。
5. 不存在循环引用的情况，比如下面这种场景是不存在的：
   A单元格：aCd**8u**
   B单元格：kAydzqo
6. 不存在多重<>的情况，一个单元格只能引用一个其他单元格。比如下面这种场景是不存在的：
   A单元格：aCd8u
   B单元格：kAydzqo
   C单元格：y**<>d**

输出描述

输出所有单元格展开的内容，单元格之间用逗号分隔。处理过程中出现错误时，输出字符串“-1”表示出错。

示例1

输入

```
1,2<A>00
```

输出

```
1，2100
```

说明
第二个单元中有对A单元的引用，A单元格的值为1，替换时，将A单元的内容替代的位置，并和其他内容合并。

示例2

输入

```
<B>12,1
```

输出

```
112，1
```

说明
第一个单元中有对B单元的引用，B单元格的值为1，替换时，将第二个数据单元的内容替代**的位置，并和其他内容合并。**

示例3

输入

```
<B<12,1
```

输出

```
-1
```

说明
第一个单元中有错误的单元格引用方式，输出-1

解题思路

1. 首先，程序从用户那里读取一行输入字符串。
2. 然后，程序使用逗号作为分隔符，将输入字符串分割成多个单元格。
3. 接着，程序创建一个map（在Python中称为字典），将每个单元格的标签（从"A"开始）和内容存储在其中。
4. 然后，程序初始化一个空的结果字符串，用于存储处理后的单元格内容。
5. 接下来，程序遍历每个单元格。对于每个单元格，程序首先找到"<“和”>"的位置，这两个字符是引用的标记。
6. 如果单元格不包含"<“和”>"，那么程序会将单元格的内容添加到结果字符串中。
7. 如果单元格包含"<“或”>“，但格式不正确（例如，”<“和”>"的位置不正确，或者它们之间没有恰好一个字符），那么程序会打印-1并退出。
8. 如果单元格包含正确的引用，那么程序会将引用替换为实际的单元格值，并将处理后的单元格内容添加到结果字符串中。
9. 最后，程序打印处理后的结果字符串，但是会删除末尾的逗号。

C++

```cpp
#include <iostream>
#include <string>
#include <map>
#include <sstream>
using namespace std;
int main() {
    // 从用户那里读取一行输入
    string input_str;
    getline(cin, input_str);

    // 通过逗号分割输入字符串
    istringstream ss(input_str);
```

```
    string token;
    map<char, string> cells;
    int i = 0;
    while(getline(ss, token, ',')) {
        // 将每个单元格的内容存储在map中
        cells['A' + i] = token;
        i++;
    }

    // 初始化结果字符串
    string result = "";

    // 遍历输入字符串中的每个单元格
    for (const auto& cell : cells) {
        // 找到"<"和">"的位置
        int result1 = cell.second.find("<");
        int result2 = cell.second.find(">");

        // 如果单元格不包含"<"和">"，将其添加到结果字符串中
        if (result1 == string::npos && result2 == string::npos) {
            result += cell.second + ",";
        }
        // 如果单元格包含"<"或">"，但格式不正确，打印-1并退出程序
        else if (result1 == string::npos || result2 == string::npos || result1 > result2 || result2 - result1 != 2) {
            cout << -1;
            return 0;
        }
        // 如果单元格包含正确的引用，将引用替换为实际的单元格值，并将处理后的单元格添加到结果字符串中
        else {
            result += cell.second.substr(0, result1) + cells[cell.second[result1 + 1]] + cell.second.substr(result2 + 1) + ",";
        }
    }

    // 打印处理后的结果字符串，删除末尾的逗号
    cout << result.substr(0, result.length() - 1);

    return 0;
}
```

Java

```java
import java.util.Scanner;
import java.util.HashMap;

public class Main {
    public static void main(String[] args) throws Exception {
        // 创建一个Scanner对象来读取用户的输入
        Scanner in = new Scanner(System.in);
        // 读取一行输入，并将其存储在input_str变量a中
        String input_str = in.nextLine();
        // 使用split方法将输入字符串按照逗号分割，得到一个字符串数组tmp2
        String[] tmp2 = input_str.split(",");

        // 创建一个HashMap，用于存储每个单元格的值
        // 键是单元格的标识符（A-Z），值是单元格的内容
        HashMap<Character, String> cells = new HashMap<>();
        for (int i = 0; i < tmp2.length; i++) {
            // 将每个单元格的内容存储在HashMap中
            cells.put((char) ('A' + i), tmp2[i]);
        }

        // 创建一个StringBuilder对象，用于构建输出的字符串
        StringBuilder result = new StringBuilder();
        // 遍历每个单元格
        for (String cell : tmp2) {
            // 查找"<"和">"的位置
            int result1 = cell.indexOf("<");
            int result2 = cell.indexOf(">");

            // 如果单元格不包含"<"和">"，则直接将其添加到结果字符串中
            if (result1 == -1 && result2 == -1) {
                result.append(cell).append(",");
            }
            // 如果单元格包含"<"或">"，但格式不正确，输出-1并结束程序
            else if (result1 == -1 || result2 == -1 || result1 > result2 || result2 - result1 != 2) {
                // 异常
                System.out.println(-1);
                return;
            }
            // 如果单元格包含正确的引用，将引用替换为实际的单元格值，并将处理后的单元格添加到结果字符串中
            else {
```

```
                // 正常情况替换即可
                result.append(cell, 0, result1)
                        .append(cells.get(cell.charAt(result1 + 1)))
                        .append(cell.substring(result2 + 1))
                        .append(",");
            }
        }

        // 输出处理后的结果字符串，注意删除末尾的逗号
        System.out.println(result.substring(0, result.length() - 1));
    }
}
```

javaScript

```
const readline = require('readline').createInterface({
  input: process.stdin,
  output: process.stdout
});

readline.on('line', input_str => {
  // 通过逗号分割输入字符串
  const tmp2 = input_str.split(",");
  const cells = {};
  for (let i = 0; i < tmp2.length; i++) {
    // 将每个单元格的内容存储在对象中
    cells[String.fromCharCode(65 + i)] = tmp2[i];
  }

  let result = "";
  for (let cell of tmp2) {
    // 找到"<"和">"的位置
    let result1 = cell.indexOf("<");
    let result2 = cell.indexOf(">");

    // 如果单元格不包含"<"和">"，将其添加到结果字符串中
    if (result1 == -1 && result2 == -1) {
      result += cell + ",";
    }
    // 如果单元格包含"<"或">"，但格式不正确，打印-1并退出程序
```

```
    else if (result1 == -1 || result2 == -1 || result1 > result2 || result2 - result1 != 2) {
      console.log(-1);
      process.exit(0);
    }
    // 如果单元格包含正确的引用，将引用替换为实际的单元格值，并将处理后的单元格添加到结果字符串中
    else {
      result += cell.substring(0, result1) + cells[cell.charAt(result1 + 1)] + cell.substring(result2 + 1) + ",";
    }
  }

  // 打印处理后的结果字符串，删除末尾的逗号
  console.log(result.substring(0, result.length - 1));
  readline.close();
});
```

Python

```
input_str = input()

# 通过逗号分割输入字符串
tmp2 = input_str.split(",")

# 创建一个字典来存储每个单元格的内容
cells = {chr(65 + i): tmp2[i] for i in range(len(tmp2))}

# 遍历每个单元格
result = ""
for cell in tmp2:
    # 找到"<"和">"的位置
    result1 = cell.find("<");
    result2 = cell.find(">");

    # 如果单元格不包含"<"和">"，将其添加到结果字符串中
    if result1 == -1 and result2 == -1:
        result += cell + ","
    # 如果单元格包含"<"或">"，但格式不正确，打印-1并退出程序
    elif result1 == -1 or result2 == -1 or result1 > result2 or result2 - result1 != 2:
        print(-1)
        exit(0)
    # 如果单元格包含正确的引用，将引用替换为实际的单元格值，并将处理后的单元格添加到结果字符串中
```

```
    else:
        result += cell[:result1] + cells[cell[result1 + 1]] + cell[result2 + 1:] + ","

# 打印处理后的结果字符串，删除末尾的逗号
print(result[:-1])
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师