【华为OD机考 统一考试机试C卷】机场航班调度（C++ Java JavaScript Python C语言）

华为OD机考:统一考试 C卷 + D卷 + B卷 +A卷

目前在考C卷，经过两个月的收集整理，C卷真题已基本整理完毕
抽到原题的概率为2/3到3/3，也就是最少抽到两道原题。请注意：大家刷完C卷真题，最好要把B卷的真题刷一下，因为C卷的部分真题来自B卷。
另外订阅专栏还可以联系笔者开通在线 OJ 进行刷题，提高刷题效率。
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

题目描述

XX市机场停放了多架飞机，每架飞机都有自己的航班号CA3385，CZ6678，SC6508等，航班号的前2个大写字母(或数字）代表航空公司的缩写，后面4个数字代表航班信息。但是XX市机场只有一条起飞用跑道，调度人员需要安排目前停留在机场的航班有序起飞。为保障航班的有序起飞，调度员首先按照航空公司的缩写（航班号前2个字母）对所有航班进行排序，同一航空公司的航班再按照航班号的后4个数字进行排序最终获得安排好的航班的起飞顺序。请编写一段代码根据输入的航班号信息帮助调度员输出航班的起飞顺序。航空公司缩写排序按照从特殊符号$ & *, 0_9, $_A$Z排序;

输入描述

第一行输入航班信息，多个航班号之间用逗号（“，”）分隔，输入的航班号不超过100个例如：

> CA3385,CZ6678,SC6508,DU7523,HK4456,MK0987
> 备注：航班号为6位长度，后4位为纯数字，不考虑存在后4位重复的场景

输出描述

> CA3385,CZ6678,DU7523,HK4456,MK0987,SC6508

用例1

输入

```
CA3385,CZ6678,SC6508,DU7523,HK4456,MK0987
```

输出

```
CA3385,CZ6678,DU7523,HK4456,MK0987,SC6508
```

说明

> 输入目前停留在该机场的航班号，输出为按照调度排序后输出的有序的航班号

用例2

输入

```
MU1087,CA9908,3U0045,FM1703
```

输出

```
3U0045,CA9908,FM1703,MU1087
```

C++

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <sstream>
using namespace std;
int main() {
    // 读取一行输入，这行输入是由逗号分隔的航班号
    string input;

```

```
    getline(cin, input);

    // 使用逗号将输入的航班号分割为一个vector
    stringstream ss(input);
    string flight;
    vector<string> flights;
    while (getline(ss, flight, ',')) {
        flights.push_back(flight);
    }

    // 对航班号vector进行排序
    sort(flights.begin(), flights.end(), [](const string &s1, const string &s2) {
        // 首先比较航班号的前两个字符（航空公司的缩写）
        int prefixCompare = s1.substr(0, 2).compare(s2.substr(0, 2));
        // 如果航空公司的缩写相同
        if (prefixCompare == 0) {
            // 则比较航班号的后四个字符（航班信息）
            return s1.substr(2) < s2.substr(2);
        } else {
            // 如果航空公司的缩写不同，直接返回比较结果
            return prefixCompare < 0;
        }
    });

    // 输出排序后的航班号
    for (int i = 0; i < flights.size(); i++) {
        cout << flights[i];
        // 如果不是最后一个航班号，输出一个逗号
        if (i != flights.size() - 1) {
            cout << ",";
        }
    }

    return 0;
}
```

Java

```
import java.util.*;

```

```java
public class Main {
    public static void main(String[] args) {
        // 创建一个扫描器来读取用户的输入
        Scanner scanner = new Scanner(System.in);
        // 读取一行输入，这行输入是由逗号分隔的航班号
        String input = scanner.nextLine();
        // 关闭扫描器
        scanner.close();

        // 使用逗号将输入的航班号分割为一个数组
        String[] flights = input.split(",");
        // 对航班号数组进行排序
        Arrays.sort(flights, new Comparator<String>() {
            // 定义比较两个航班号的规则
            public int compare(String s1, String s2) {
                // 首先比较航班号的前两个字符（航空公司的缩写）
                int prefixCompare = s1.substring(0, 2).compareTo(s2.substring(0, 2));
                // 如果航空公司的缩写相同
                if (prefixCompare == 0) {
                    // 则比较航班号的后四个字符（航班信息）
                    return s1.substring(2).compareTo(s2.substring(2));
                } else {
                    // 如果航空公司的缩写不同，直接返回比较结果
                    return prefixCompare;
                }
            }
        });

        // 创建一个StringBuilder来拼接排序后的航班号
        StringBuilder sb = new StringBuilder();
        for (int i = 0; i < flights.length; i++) {
            // 将航班号添加到StringBuilder中
            sb.append(flights[i]);
            // 如果不是最后一个航班号，添加一个逗号
            if (i != flights.length - 1) {
                sb.append(",");
            }
        }

        // 输出排序后的航班号
```

```
        System.out.println(sb.toString());
    }
}
```

javaScript

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

readline.on('line', input => {
    let flights = input.split(',');

    flights.sort((s1, s2) => {
        let order = "$&*0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        for (let i = 0; i < 2; i++) {
            let diff = order.indexOf(s1[i]) - order.indexOf(s2[i]);
            if (diff !== 0) {
                return diff;
            }
        }
        return s1.slice(2) - s2.slice(2);
    });

    console.log(flights.join(','));

    readline.close();
});
```

Python

```python
# 读取一行输入，这行输入是由逗号分隔的航班号
input_str = input()

# 使用逗号将输入的航班号分割为一个列表
flights = input_str.split(',')

# 对航班号列表进行排序

```

```
flights.sort(key=lambda s: (s[:2], s[2:]))

# 输出排序后的航班号
print(','.join(flights))
```

C语言

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// 定义航班号的长度
#define FLIGHT_NUMBER_LENGTH 6
// 定义最大航班数量
#define MAX_FLIGHTS 100

// 航班号比较函数
int compareFlights(const void *a, const void *b) {
    // 比较航空公司的缩写
    int prefixCompare = strncmp((const char *)a, (const char *)b, 2);
    if (prefixCompare == 0) {
        // 如果航空公司的缩写相同，则比较航班信息
        return strncmp((const char *)a + 2, (const char *)b + 2, 4);
    }
    return prefixCompare;
}

int main() {
    char input[FLIGHT_NUMBER_LENGTH * MAX_FLIGHTS + MAX_FLIGHTS]; // 存储输入的航班号
    char flights[MAX_FLIGHTS][FLIGHT_NUMBER_LENGTH + 1]; // 存储分割后的航班号
    int flightCount = 0; // 航班数量

    // 读取一行输入，这行输入是由逗号分隔的航班号
    gets(input);

    // 使用逗号将输入的航班号分割
    char *token = strtok(input, ",");
    while (token != NULL) {
        strcpy(flights[flightCount], token);
        flightCount++;
```

```
        token = strtok(NULL, ",");
    }

    // 对航班号进行排序
    qsort(flights, flightCount, sizeof(flights[0]), compareFlights);

    // 输出排序后的航班号
    for (int i = 0; i < flightCount; i++) {
        printf("%s", flights[i]);
        // 如果不是最后一个航班号，输出一个逗号
        if (i != flightCount - 1) {
            printf(",");
        }
    }

    return 0;
}
```

文章目录


机考真题
华为OD
No No No
CSDN @算法大师