

【华为OD机考 统一考试机试C卷】手机App防沉迷系统（C++ Java JavaScript Python C语言）

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](#)[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

智能手机方便了我们生活的同时，也侵占了我们不少的时间。“手机App防沉迷系统”能够让我们每天合理地规划手机App使用时间，在正确的时间做正确的事。

它的大概原理是这样的：

1. 在一天24小时内，可以注册每个App的允许使用时段
2. 一个时间段只能使用一个App
3. App有优先级，数值越高，优先级越高。注册使用时段时，如果高优先级的App时间和低优先级的时段有冲突，则系统会自动注销低优先级的时段，如果App的优先级相同，则后添加的App不能注册。

请编程实现，根据输入数据注册App，并根据输入的时间点，返回时间点使用的App名称，如果该时间点没有注册任何App，请返回字符串“NA”。

输入描述

第一行表示注册的App数量 N (N ≤ 100)

第二部分包括 N 行，每行表示一条App注册数据

最后一行输入一个时间点，程序即返回该时间点使用的App

```
2
App1 1 09:00 10:00
App2 2 11:00 11:30
09:30
```

数据说明如下：

- 1. N行注册数据以空格分隔，四项数依次表示：App名称、优先级、起始时间、结束时间
- 2. 优先级1~5，数字越大，优先级越高
- 3. 时间格式 HH:MM，小时和分钟都是两位，不足两位前面补0
- 4. 起始时间需小于结束时间，否则注册不上
- 5. 注册信息中的时间段包含起始时间点，不包含结束时间点

输出描述

输出一个字符串，表示App名称，或NA表示空闲时间

用例2

输入

```
1 | 1
2 | App1 1 09:00 10:00
3 | 09:30
```

输出

```
1 | App1
```

说明

App1注册在9点到10点间，9点半可用的应用名是App1

用例2

输入

1	2
2	App1 1 09:00 10:00
3	App2 2 09:10 09:30
4	09:20

输出

1	App2
---	------

说明

ApP1和App2的时段有冲突，App2优先级高，注册App2之后，App1自动注销，因此输出App2。

用例3

输入

1	2
2	App1 1 09:00 10:00
3	App2 2 09:10 09:30
4	09:50

输出

1	NA
---	----

解题思路

1. **定义App类**：首先，我们定义了一个App类来存储每个App的相关信息，包括名称、优先级、起始时间和结束时间。
2. **时间转换**：将时间从"小时:分钟"格式的字符串转换为以分钟为单位的整数。
3. **处理注册的App**：创建另一个ArrayList来存储注册的App。对于每个App，我们检查它的时间段是否与已注册App的时间段重叠。如果有重叠，我们比较它们的优先级。如果当前App的优先级高于已注册的App，则注销低优先级的App。否则，跳过当前App。这样确保了在任何给定时间，只有最高优先级的App被注册。
4. **查询特定时间的App**：读取时间，然后遍历注册的App列表，找到在该时间活跃的App（即查询时间在App的起始时间和结束时间之间）。将此App的名称存储在变量 `appAtTime` 中。

总结：核心思想是首先读取App信息并转换时间格式，然后在注册阶段处理时间冲突和优先级问题，最后根据查询时间确定哪个App是活跃的，并输出该App的名称。

C++

```
1  #include <iostream>
2  #include <vector>
3  #include <string>
4  #include <sstream>
5
6  // 定义App类, 用于存储App的相关信息
7  class App {
8  public:
9      std::string name; // App名称
10     int priority; // App优先级
11     int startTime; // App允许使用的起始时间 (以分钟为单位)
12     int endTime; // App允许使用的结束时间 (以分钟为单位)
13
14     // App类的构造函数, 用于创建App对象
15     App(std::string name, int priority, int startTime, int endTime)
16         : name(name), priority(priority), startTime(startTime), endTime(endTime) {}
17 };
18
19 // 时间转换函数, 将时间字符串转换为以分钟为单位的整数
20 int convertTime(const std::string& time) {
21     int hours, minutes;
22     char colon;
23 }
```

```

--
24     std::istringstream iss(time);
25     iss >> hours >> colon >> minutes; // 将时间字符串按照":"分割并转换为小时和分钟
26     return hours * 60 + minutes; // 将小时和分钟转换为分钟
27 }
28
29 int main() {
30     int n; // 读取App数量
31     std::cin >> n;
32
33     std::vector<App> apps; // 创建App列表, 用于存储所有App
34     for (int i = 0; i < n; i++) {
35         // 循环读取每个App的信息, 并创建App对象添加到列表中
36         std::string appName;
37         int appPriority, appStartTime, appEndTime;
38         std::string startTimeStr, endTimeStr;
39         std::cin >> appName >> appPriority >> startTimeStr >> endTimeStr;
40         appStartTime = convertTime(startTimeStr);
41         appEndTime = convertTime(endTimeStr);
42         apps.emplace_back(appName, appPriority, appStartTime, appEndTime);
43     }
44
45     std::string queryTimeStr;
46     std::cin >> queryTimeStr;
47     int queryTime = convertTime(queryTimeStr); // 读取查询时间, 并转换为分钟
48     std::string appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
49
50     std::vector<App> registeredApps; // 创建已注册App列表
51     for (const App& app : apps) {
52         if (app.startTime >= app.endTime) continue; // 如果起始时间不小于结束时间, 则跳过
53
54         // 遍历已注册的App列表, 检查时间冲突
55         for (int i = registeredApps.size() - 1; i >= 0; --i) {
56             const App& registered = registeredApps[i];
57             // 如果存在时间冲突
58             if (std::max(app.startTime, registered.startTime) < std::min(app.endTime, registered.endTime)) {
59                 // 如果当前App的优先级高于已注册App的优先级
60                 if (app.priority > registered.priority) {
61                     registeredApps.erase(registeredApps.begin() + i); // 注销低优先级的App
62                 } else {
63                     continue; // 如果优先级不高, 继续检查下一个已注册App
64

```

```

64         }
65     }
66 }
67
68 // 将当前App添加到已注册App列表中
69 registeredApps.push_back(app);
70 }
71
72 // 遍历已注册App列表, 找到查询时间对应的App
73 for (const App& app : registeredApps) {
74     if (queryTime >= app.startTime && queryTime < app.endTime) {
75         appAtTime = app.name; // 更新查询时间对应的App名称
76         break; // 找到后退出循环
77     }
78 }
79
80 std::cout << appAtTime << std::endl; // 输出查询时间对应的App名称
81
82 return 0;
}

```

Java

```

1  import java.util.ArrayList;
2  import java.util.Scanner;
3
4  public class Main {
5      // 定义App类, 用于存储App的相关信息
6      static class App {
7          String name; // App名称
8          int priority; // App优先级
9          int startTime; // App允许使用的起始时间 (以分钟为单位)
10         int endTime; // App允许使用的结束时间 (以分钟为单位)
11
12         // App类的构造函数, 用于创建App对象
13         public App(String name, int priority, int startTime, int endTime) {
14             this.name = name;
15             this.priority = priority;
16             this.startTime = startTime;
17             this.endTime = endTime;
18         }
19     }
20
21     // 主函数
22     public static void main(String[] args) {
23         Scanner scanner = new Scanner(System.in);
24         int n = scanner.nextInt();
25         ArrayList<App> apps = new ArrayList<>();
26         for (int i = 0; i < n; i++) {
27             String name = scanner.next();
28             int priority = scanner.nextInt();
29             int startTime = scanner.nextInt();
30             int endTime = scanner.nextInt();
31             App app = new App(name, priority, startTime, endTime);
32             apps.add(app);
33         }
34
35         int queryTime = scanner.nextInt();
36         String appAtTime = "";
37         for (App app : apps) {
38             if (queryTime >= app.startTime && queryTime < app.endTime) {
39                 appAtTime = app.name;
40                 break;
41             }
42         }
43
44         System.out.println(appAtTime);
45     }
46 }

```

```

18     }
19 }
20
21 // 主函数
22 public static void main(String[] args) {
23     Scanner sc = new Scanner(System.in); // 创建Scanner对象, 用于读取标准输入
24     int n = sc.nextInt(); // 读取App数量
25
26     ArrayList<App> apps = new ArrayList<>(); // 创建App列表, 用于存储所有App
27     for (int i = 0; i < n; i++) {
28         // 循环读取每个App的信息, 并创建App对象添加到列表中
29         String appName = sc.next();
30         int appPriority = sc.nextInt();
31         int appStartTime = convertTime(sc.next());
32         int appEndTime = convertTime(sc.next());
33         apps.add(new App(appName, appPriority, appStartTime, appEndTime));
34     }
35
36     int queryTime = convertTime(sc.next()); // 读取查询时间, 并转换为分钟
37     String appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
38
39     // 创建已注册App列表
40     ArrayList<App> registeredApps = new ArrayList<>();
41     for (App app : apps) {
42         if (app.startTime >= app.endTime) continue; // 如果起始时间不小于结束时间, 则跳过
43
44         // 遍历已注册的App列表, 检查时间冲突
45         for (int i = registeredApps.size() - 1; i >= 0; i--) {
46             App registered = registeredApps.get(i);
47             // 如果存在时间冲突
48             if (Math.max(app.startTime, registered.startTime) < Math.min(app.endTime, registered.endTime)) {
49                 // 如果当前App的优先级高于已注册App的优先级
50                 if (app.priority > registered.priority) {
51                     registeredApps.remove(i); // 注销低优先级的App
52                 } else {
53                     continue; // 如果优先级不高, 继续检查下一个已注册App
54                 }
55             }
56         }
57     }
58 }

```

```

59         // 将当前App添加到已注册App列表中
60         registeredApps.add(app);
61     }
62
63     // 遍历已注册App列表, 找到查询时间对应的App
64     for (App app : registeredApps) {
65         if (queryTime >= app.startTime && queryTime < app.endTime) {
66             appAtTime = app.name; // 更新查询时间对应的App名称
67             break; // 找到后退出循环
68         }
69     }
70
71     System.out.println(appAtTime); // 输出查询时间对应的App名称
72 }
73
74 // 时间转换函数, 将时间字符串转换为以分钟为单位的整数
75 private static int convertTime(String time) {
76     String[] parts = time.split(":"); // 将时间字符串按照":"分割
77     return Integer.parseInt(parts[0]) * 60 + Integer.parseInt(parts[1]); // 将小时和分钟转换为分钟
78 }
79 }

```

javaScript

```

1  const readline = require('readline');
2
3  // 定义App类, 用于存储App的相关信息
4  class App {
5      constructor(name, priority, startTime, endTime) {
6          this.name = name; // App名称
7          this.priority = priority; // App优先级
8          this.startTime = startTime; // App允许使用的起始时间 (以分钟为单位)
9          this.endTime = endTime; // App允许使用的结束时间 (以分钟为单位)
10     }
11 }
12
13 // 创建readline接口实例
14 const rl = readline.createInterface({
15     input: process.stdin,
16     output: process.stdout
17 });

```



```
17 });
18
19 // 用于存储输入行的数组
20 const lines = [];
21 // 读取输入
22 rl.on('line', (line) => {
23   lines.push(line);
24 }).on('close', () => {
25   // 当输入完成后开始处理数据
26   processInput(lines);
27 });
28
29 // 处理输入数据的函数
30 function processInput(lines) {
31   const n = parseInt(lines.shift()); // 读取App数量
32   const apps = []; // 创建App列表, 用于存储所有App
33
34   for (let i = 0; i < n; i++) {
35     // 循环读取每个App的信息, 并创建App对象添加到列表中
36     const [appName, appPriority, appStartTime, appEndTime] = lines.shift().split(' ');
37     apps.push(new App(appName, parseInt(appPriority), convertTime(appStartTime), convertTime(appEndTime)));
38   }
39
40   const queryTime = convertTime(lines.shift()); // 读取查询时间, 并转换为分钟
41   let appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
42
43   // 创建已注册App列表
44   const registeredApps = [];
45   for (const app of apps) {
46     if (app.startTime >= app.endTime) continue; // 如果起始时间不小于结束时间, 则跳过
47
48     // 遍历已注册的App列表, 检查时间冲突
49     for (let i = registeredApps.length - 1; i >= 0; i--) {
50       const registered = registeredApps[i];
51       // 如果存在时间冲突
52       if (Math.max(app.startTime, registered.startTime) < Math.min(app.endTime, registered.endTime)) {
53         // 如果当前App的优先级高于已注册App的优先级
54         if (app.priority > registered.priority) {
55           registeredApps.splice(i, 1); // 注销低优先级的App
56         } else {
57
```

```

58         continue; // 如果优先级不高, 继续检查下一个已注册App
59     }
60 }
61 }
62
63 // 将当前App添加到已注册App列表中
64 registeredApps.push(app);
65 }
66
67 // 遍历已注册App列表, 找到查询时间对应的App
68 for (const app of registeredApps) {
69     if (queryTime >= app.startTime && queryTime < app.endTime) {
70         appAtTime = app.name; // 更新查询时间对应的App名称
71         break; // 找到后退出循环
72     }
73 }
74
75 console.log(appAtTime); // 输出查询时间对应的App名称
76 }
77
78 // 时间转换函数, 将时间字符串转换为以分钟为单位的整数
79 function convertTime(time) {
80     const [hours, minutes] = time.split(':').map(Number); // 将时间字符串按照":"分割并转换为数字
81     return hours * 60 + minutes; // 将小时和分钟转换为分钟
82 }

```

Python

```

1 class App:
2     """定义App类, 用于存储App的相关信息"""
3
4     def __init__(self, name, priority, start_time, end_time):
5         self.name = name # App名称
6         self.priority = priority # App优先级
7         self.start_time = start_time # App允许使用的起始时间 (以分钟为单位)
8         self.end_time = end_time # App允许使用的结束时间 (以分钟为单位)
9
10    def convert_time(time_str):
11        """
12        时间转换函数, 将时间字符串转换为以分钟为单位的整数
13    """

```

```

13 :param time_str: 时间字符串, 格式为"小时:分钟"
14 :return: 转换后的分钟数
15 """
16 hours, minutes = map(int, time_str.split(":")) # 将时间字符串按照":"分割并转换为整数
17 return hours * 60 + minutes # 将小时和分钟转换为分钟
18
19 def main():
20     n = int(input()) # 读取App数量
21     apps = [] # 创建App列表, 用于存储所有App
22
23     for _ in range(n):
24         # 循环读取每个App的信息, 并创建App对象添加到列表中
25         app_name, app_priority, app_start_time, app_end_time = input().split()
26         app_priority = int(app_priority)
27         app_start_time = convert_time(app_start_time)
28         app_end_time = convert_time(app_end_time)
29         apps.append(App(app_name, app_priority, app_start_time, app_end_time))
30
31     query_time = convert_time(input()) # 读取查询时间, 并转换为分钟
32     app_at_time = "NA" # 初始化查询时间对应的App名称为"NA"
33
34     # 创建已注册App列表
35     registered_apps = []
36     for app in apps:
37         if app.start_time >= app.end_time:
38             continue # 如果起始时间不小于结束时间, 则跳过
39
40         # 遍历已注册的App列表, 检查时间冲突
41         for i in range(len(registered_apps) - 1, -1, -1):
42             registered = registered_apps[i]
43             # 如果存在时间冲突
44             if max(app.start_time, registered.start_time) < min(app.end_time, registered.end_time):
45                 # 如果当前App的优先级高于已注册App的优先级
46                 if app.priority > registered.priority:
47                     registered_apps.pop(i) # 注销低优先级的App
48             else:
49                 continue # 如果优先级不高, 继续检查下一个已注册App
50
51     # 将当前App添加到已注册App列表中
52     registered_apps.append(app)
53

```

```

54 |
55 |     # 遍历已注册App列表, 找到查询时间对应的App
56 |     for app in registered_apps:
57 |         if query_time >= app.start_time and query_time < app.end_time:
58 |             app_at_time = app.name # 更新查询时间对应的App名称
59 |             break # 找到后退出循环
60 |
61 |     print(app_at_time) # 输出查询时间对应的App名称
62 |
63 | if __name__ == "__main__":
    main()

```

C语言

```

1 | #include <stdio.h>
2 | #include <stdlib.h>
3 | #include <string.h>
4 |
5 | // 定义App结构体, 用于存储App的相关信息
6 | typedef struct {
7 |     char name[30]; // App名称
8 |     int priority; // App优先级
9 |     int startTime; // App允许使用的起始时间 (以分钟为单位)
10 |    int endTime; // App允许使用的结束时间 (以分钟为单位)
11 | } App;
12 |
13 | // 时间转换函数, 将时间字符串转换为以分钟为单位的整数
14 | int convertTime(char *time) {
15 |     int hours, minutes;
16 |     sscanf(time, "%d:%d", &hours, &minutes); // 将时间字符串按照":"分割并转换为整数
17 |     return hours * 60 + minutes; // 将小时和分钟转换为分钟
18 | }
19 |
20 | int main() {
21 |     int n; // 读取App数量
22 |     scanf("%d", &n);
23 |
24 |     App *apps = (App *)malloc(n * sizeof(App)); // 创建App数组, 用于存储所有App
25 |
26 |     for (int i = 0; i < n; i++) {
27 |

```

```

27 // 循环读取每个App的信息, 并创建App对象添加到数组中
28 scanf("%s %d", apps[i].name, &apps[i].priority);
29 char startTimeStr[6], endTimeStr[6];
30 scanf("%s %s", startTimeStr, endTimeStr);
31 apps[i].startTime = convertTime(startTimeStr);
32 apps[i].endTime = convertTime(endTimeStr);
33 }
34
35 char queryTimeStr[6];
36 scanf("%s", queryTimeStr);
37 int queryTime = convertTime(queryTimeStr); // 读取查询时间, 并转换为分钟
38 char *appAtTime = "NA"; // 初始化查询时间对应的App名称为"NA"
39
40 // 创建已注册App数组和计数器
41 App *registeredApps = (App *)malloc(n * sizeof(App));
42 int registeredCount = 0;
43
44 for (int i = 0; i < n; i++) {
45     if (apps[i].startTime >= apps[i].endTime) continue; // 如果起始时间不小于结束时间, 则跳过
46
47     // 遍历已注册的App数组, 检查时间冲突
48     for (int j = registeredCount - 1; j >= 0; j--) {
49         // 如果存在时间冲突
50         if (apps[i].startTime < registeredApps[j].endTime && apps[i].endTime > registeredApps[j].startTime) {
51             // 如果当前App的优先级高于已注册App的优先级
52             if (apps[i].priority > registeredApps[j].priority) {
53                 // 注销低优先级的App
54                 for (int k = j; k < registeredCount - 1; k++) {
55                     registeredApps[k] = registeredApps[k + 1];
56                 }
57                 registeredCount--; // 减少已注册App的计数
58             } else {
59                 goto continue_outer; // 如果优先级不高, 继续检查下一个已注册App
60             }
61         }
62     }
63
64     // 将当前App添加到已注册App数组中
65     registeredApps[registeredCount++] = apps[i];
66
67

```

```
68     continue_outer: ; // 标签用于跳过当前循环
69 }
70
71 // 遍历已注册App数组, 找到查询时间对应的App
72 for (int i = 0; i < registeredCount; i++) {
73     if (queryTime >= registeredApps[i].startTime && queryTime < registeredApps[i].endTime) {
74         appAtTime = registeredApps[i].name; // 更新查询时间对应的App名称
75         break; // 找到后退出循环
76     }
77 }
78
79 printf("%s\n", appAtTime); // 输出查询时间对应的App名称
80
81 // 释放动态分配的内存
82 free(apps);
83 free(registeredApps);
84
85 return 0;
}
```

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2](#)

[用例2](#)

[用例3](#)

[解题思路](#)

[C++](#)

[Java](#)

[javaScript](#)

[Python](#)

[C语言](#)

机考真题 华为OD



CSDN @算法大师