## 2024 华为软件精英挑战赛

# 复赛任务书

文档版本 v0.1

发布日期 2024-04-01

华为技术有限公司

# 目　录

# 1 更新记录

表1-1 更新记录

| 版本 | 修改说明 | 发布时间 |
| --- | --- | --- |
| 01 | 第一次正式发布 | 2024-04-01 |

# 2 变化点

**船舶及机器人获取方式：购买**

船舶与机器人不再直接生成在地图上，需要选手支付资金在特定的地点进行购买。

**地图地块变化**

引入了新的地块。具体参考后文地图处。

**船只泊靠机制**

船只的移动过程及泊靠过程需要选手通过指令进行路径规划并操控，具体**船只的移动&操控指令请参考后文。**

**机器人机制**

随着地图扩大及新机制的引入，为了可以使选手精力集中在调度策略上，题目中机器人的移动和碰撞限制有所改变，具体变化点请参考后文。

# 3 背景信息

智慧港口是港口建设趋势和发展的方向。 以信息物理系统为框架，通过高新技术的创新应用，使物流供给方和需求方沟通融入集疏运一体化系统；极大提升港口及其相关物流园区对信息的综合处理能力和对相关资源的优化配置能力；智能监管、智能服务、自动装卸成为其主要呈现形式，并能为现代物流业提供高安全、高效率和高品质服务的一类新型港口。“智慧港口”是以现代化基础设施设备为基础，以云计算、大数据、物联网、移动互联网、智能控制等新一代信息技术与港口运输业务深度融合为核心，以港口运输组织服务创新为动力，以完善的体制机制、法律法规、标准规范、发展政策为保障，能够在更高层面上实现港口资源优化配置，在更高境界上满足多层次、敏捷化、高品质港口运输服务要求的，具有生产智能、管理智慧、服务柔性、保障有力等鲜明特征的现代港口运输新业态。


图3-1 华为云智慧港口

在智能港口领域，如何规划多机器人的任务执行以实现最优调度，如何控制泊位和机器人的配合实现最高运输价值等都是非常有价值的算法难题。本次比赛通过软件模拟了智能港口的状态信息，由选手来挑战这些有价值的算法难题。

期待您的精彩解决方案。

# 4 题目

## 4.1 题目介绍

### 题目概述

- **目标：**

  赚取更多的资金。

- **程序操控方式：**

  选手作为运输公司参与运输货物赚取资金，选手程序购买轮船、机器人执行移动、搬运等动作来完成货物递送任务，同时赚取获得利润。在运行结束时，选手拥有的资金数即为最终分数，所获得的资金越高越好。复赛时间为 **15,000 帧**（最多 **5 分钟**）

- **程序交互方式：**

  选手程序通过标准输入和标准输出与判题器进行交互。判题器运行帧率为每秒 50 帧，对于每一帧，判题器都会把场上的实时信息通过标准输入传递给选手程序，同时从选手程序读取机器人的操控指令作用到各个机器人上。每一帧有 1000/50=20ms 的时间，由于判题器需保留 5ms 执行计算来模拟真实场景，故选手程序需要在 15ms 内做出每一帧的决策，如果超过 15ms 未做出决策，则系统将直接忽略这一帧的控制进入下一帧，并且在直到选手程序返回控制指令之前，不会再发送状态数据给选手程序。

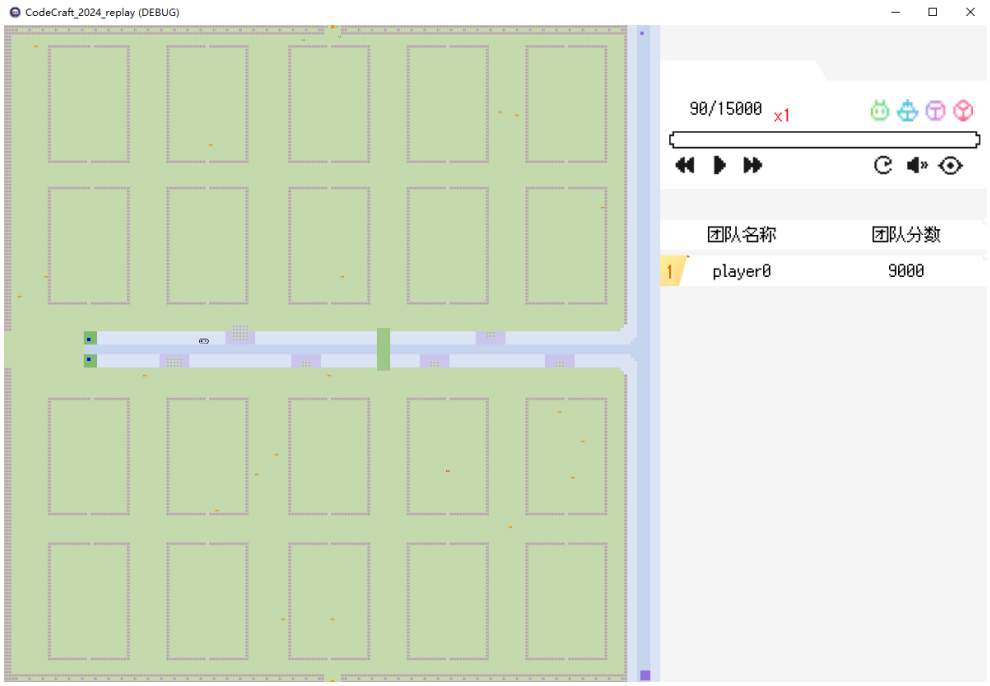
  注意，不具备 50FPS 的程序也可正常运行（例如只处理 10FPS），但该类程序无法及时处理完所有帧数，会导致跳帧现象。

  程序的输入和输出格式请参考输入与输出格式。

- **判题器使用：**

  今年的比赛判题器与数据集完全开放给大家下载，并且做了跨平台设计（Windows/Linux/MacOS），大家可以根据自身习惯选择对应版本下载。但是请注意，比赛平台使用 Linux，因此无论你选择何种平台开发调试，都必须确保你的代码可以在 Linux 下编译运行。

  运行判题器中的 run_simple_demo 可快速运行一个 DEMO，运行界面如下。




## 术语

表4-1 术语

| 名词 | 解释 |
| --- | --- |
| 地图 | 地图是一个 200*200 的封闭区域。分为陆地和海洋两部分。 |
| 坐标系 | 往下为 X 轴正方向，往右为 Y 轴正方向。地图左上角坐标为原点(0,0)，右下角坐标为(199,199)。 |
| 货物 | 货物随机生成在地图上，可由机器人在一处货物处搬运之后放置在泊位上，后续通过船舶运输可产生利润。 |
| 轮船 | 轮船是一个 2*3 的矩形，价格为 8000，轮船有一定的容积，需要停留在泊位上，可以把泊位上的货物运走。 |
| 主航道 | 主航道是一个特殊的海洋地块，在主航道上行驶的船不会发生碰撞，但在该区域行驶的船的速度会变慢。 |
| 主干道 | 主干道是一个特殊的陆地地块，在主干道上移动的机器人不会发生碰撞。 |
| 海陆立体交通地块 | 该地块同时被视为陆地和海洋区域（可想象为“海上立交桥”），机器人和船舶皆可在该类地块中行驶，机器人和船舶之间不会发生碰撞，但机器人之间、船舶之间仍然会发生碰撞。 |
| 泊位 | 泊位代表陆地边缘处可供运输船只停靠的特定区域。泊位可以视为一种特殊的海陆立体交通地块，并同时被视为主干道和主航道，每个泊位有独立的无限容量仓库和装载机器。在同一局游戏中，泊位信息不会发生变化。 |

| 名词 | 解释 |
| --- | --- |
| 靠泊区 | 每个泊位附近会设置一定区域的与之四联通的靠泊区。该靠泊区可被视为一种特殊的海洋主航道地块。每个泊位与靠泊区形成的四连通的连通块中不会包括其他泊位。当船只核心点位于靠泊区或泊位区域内时，该船可以执行靠泊指令，此时该船只将自动停靠到该靠泊区内的对应泊位上。 |
| 机器人 | 机器人是一个占据 1*1 格子的物体，可携带一个货物。价格为 2000，由选手程序进行操控，可以执行**上下左右**四种移动操作，当机器人位于货物生成处时，可以搬运货物，当机器人位于泊位处时，可以放下货物。 |
| 运输 | 指轮船从泊位驶出运输货物产生价值，价值会在运输到达交货点瞬间立即产生（同一帧）。 |
| 交货点 | 可以视为一种特殊的靠泊区，运输船只到达该区域后可立即获取相应运输价值。 |

## 4.2 运输公司机制

### 4.2.1 资金机制

运输公司的起始资金为 25000，该资金可用于购买机器人或轮船等初始运输设备。选手可通过运输货物获得更多资金，结束时的账户资金将作为各个选手的最终成绩。

### 4.2.2 货物和售价机制

货物将会随机生成在地图的每一个区域，货物有着不同的价值，每个货物会在指定位置停留 **20s(1000 帧)**的时间，若生成后 1000 帧内未被搬运则在地图上消失，选手可以操控机器人从生成位置运送到泊位，由泊位装载货物到轮船，最后由轮船运输至交货点获取运输价值资金，该资金会在轮船运输至交货点的帧时刻完成。

### 4.2.3 运输机制

海上将存在一些特定的交货点，当轮船核心点到达交货点时，该轮船会立即清空所运输的货物，同时当前选手将会收获到对应运输价值资金。

### 4.2.4 泊位机制

泊位上有装卸机器可以将泊位上的货物装载到轮船上，每个泊位有不同的装卸速度，只能容纳一艘船。轮船通过[berth]命令进入泊位并恢复后，装卸机器将会将泊位上放置的货物运到轮船上。轮船有容积，轮船装满后将不能再装载货物到轮船上。泊位的装卸机器会按照选手在泊位上的货物放置顺序进行依次装载。

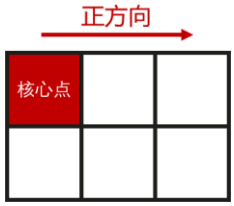
针对泊位上的船执行任何移动命令，命令成功执行时会导致该船只立即离开当前泊位。

## 4.2.5 轮船机制

复赛阶段为轮船提供了四种指令供操控：[rot - 旋转]、[ship - 前进，后退]、[berth - 靠泊]、[dept - 离港]（现实中船舶的操作移动是一套复杂的模型，在该题目中进行了一定的简化处理）。

复赛阶段运输船只在地图上是一个 2×3 的矩形，具有核心点和正方向。当正方向朝右时最左上方（两维坐标最小）的点为核心点，即核心点位于船体左后方。

如果移动后，船体任意部分在不具有主航道属性的海洋地块上与其他船只重合，或者位于没有海洋属性的地块上，则视为目标位置非法，会导致对应指令执行失败。[lboat]指令也适用该判断逻辑。



2*3船体模型示意

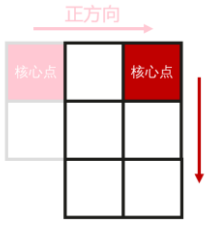
复赛阶段，选手需要使用资金购买方式获取运输轮船。地图上存在多个轮船购买地块，玩家可以通过[lboat x y]指令尝试生成一个核心点在[x,y]、正方向向右的轮船，其中[x,y]必须是轮船购买地块，同时需要支付一定的资金。资金不足以支付时或者该位置无法放置一个船的情况下该操作将会失败，此时不会消耗任何资金。判题器在向选手代码返回的船舶信息中会基于 [lboat]指令的执行情况进行相应数量的运输船只追加。

轮船在一帧中只能执行一条指令。复赛阶段，选手需要严格保证每帧的输出中，每条船只有一条指令。**同一帧内，命令按轮船编号执行（即先租赁的船会先动）。**对处于不可操作（恢复）状态的船的命令将被忽略。关于运输船只的操控命令说明如下：

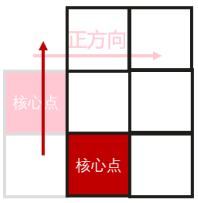
**[dept id] 离开命令：**将第 id 号船位置重置到最近的主航道上，消耗时间约等于两倍的曼哈顿距离。重置过程中，船位置将直接变化为目标位置，需要在一定帧数后恢复为可操作状态。具体位置和恢复时间请以判题器与选手代码的交互信息为准。

**[berth id] 靠泊命令：**在第 id 号船核心点位于靠泊区或泊位时，执行该命令，该船只会被移动到泊位上，泊位从此刻开始将被视为有船停靠状态。在该指令执行过程中，当前船只位置将立即更新为目标位置，但需要在若干帧后（约等于两倍的曼哈顿距离）才会恢复为可操作状态，恢复为可操作状态后才能装载货物。当靠泊区对应的泊位内有船时，该命令会失败。

**[rot id d] 转向命令：**使得第 id 号船尝试沿着 d 表示的方向转向。d 的合法取值包括 0 和 1。0 表示顺时针方向，1 表示逆时针方向。以顺时针旋转为例，旋转时正方向会瞬时针变化 90 度，船体的**原左后方**部分将会移动到原左前方位置。逆时针同理，船体的原右后方部分将会移动到原右前方位置。如果旋转后船体位置不合法，如和其他轮船（在非主航道地块）、陆地重叠或超出地图范围，则会导致指令执行失败，此时不产生任何影响。



船体顺时针旋转示意



船体逆时针旋转示意

**[ship id] 航行命令：**操作第 id 号船尝试向正方向移动 1 格。如果移动后位置不合法（如碰撞），该船只则不会移动。

对于[rot]和[ship]命令，如果移动后船体有任何部分位于具有主航道属性的地块中，则会进入 1 帧的恢复状态（即 2 帧移动一个单位）。

### 4.2.6 机器人机制

复赛阶段，选手需要通过购买获取机器人。地图上存在多个机器人购买地块，玩家可以通过[lbot x y]指令尝试在[x,y]生成一个的机器人，其中[x,y]必须是机器人购买地块，同时需要支付一定的资金。资金不足以支付的情况下该操作会失败，不产生任何影响。判题器在向选手代码返回的机器人信息中会基于成功执行的[lbot]指令数量追加输出。

机器人有上下左右四个方向的移动，每帧可以移动一格，如果机器人碰撞到了墙壁或海（即机器人的移动目标位置和墙壁重合）或者两个机器人相互碰撞（即两个机器人的目标位置重合，或者互相前往对方当前所在位置），则会停在原地不动，当前帧所有指令都不会执行。下一帧即可继续操控机器人 **(复赛取消了碰撞惩罚时间)**。

当机器人和货物坐标重合的帧时间可以使用 get 命令获取该坐标的货物，当机器人处于泊位范围的帧时间可以使用 pull 命令将机器人身上的货物放入泊位。

机器人每帧的动作可以分为移动前动作，移动，移动后动作三个部分。每个帧时刻将同时结算所有机器人的移动前动作，然后继续同时结算所有机器人移动动作、最后同时结算移动后动作。若移动动作产生碰撞，则机器人的一帧的所有动作均失效。若未进行移动动作，所有该帧的动作视为移动前动作。复赛阶段，选手需要严格保证每帧的输出中，每个机器人仅有一个移动指令，移动指令前后只能有各不超过一条其他指令。指令的结算顺序为[按机器人 id 顺序结算移动前动作-移动-按机器人 id 顺序结算移动后动作]

## 4.3 帧的行为结算顺序

在一帧中，会按顺序结算以下行为：

1、 机器人恢复

2、 船舶恢复

3、 货物生成

4、 生成场面信息，输出给选手

5、 读取选手指令

6、 执行机器人指令

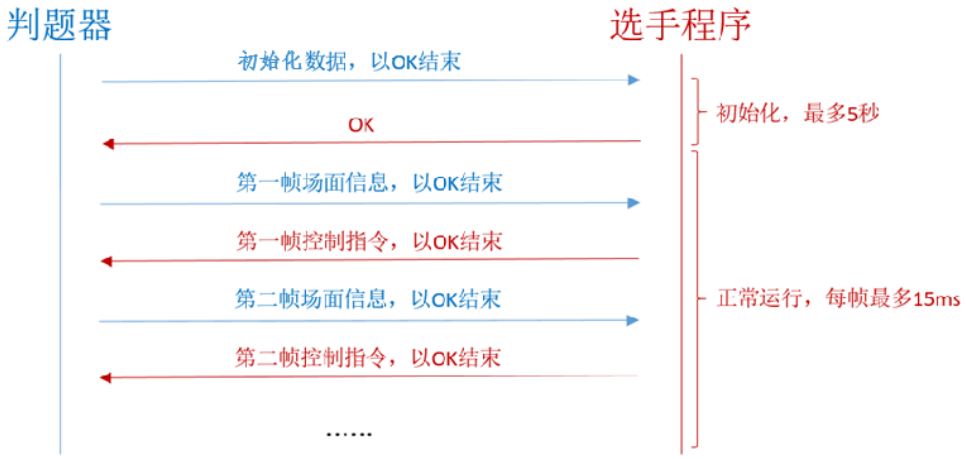
7、 执行船舶指令

8、 泊位装载货物

9、 执行购买指令

# 4.4 输入与输出格式

## 4.4.1 选手程序和判题器交互过程




## 4.4.2 输入格式

### 初始化数据格式介绍

#### 地图

地图数据（可参考 maps/*.txt）是一个 200 行 200 列的字符矩阵。地图数据中每个字符含义如下：

```
'.' : 空地
'>' : 陆地主干道
'*' : 海洋
'~' : 海洋主航道
'#' : 障碍
'R' : 机器人购买地块，同时该地块也是主干道
'S' : 船舶购买地块，同时该地块也是主航道
'B' : 泊位
'K' : 靠泊区
'C' : 海陆立体交通地块
'c' : 海陆立体交通地块，同时为主干道和主航道
'T' : 交货点
```

其余字符均为不合法字符，不会出现在地图数据中。

#### 泊位

每个泊位的信息，由一行包括 4 个数字的行来表示。

每一行四个整数 id, x, y, velocity 表示一个泊位的信息。其中 id 为该泊位的唯一标号，(x,y)表示在该泊位上轮船靠泊时，轮船核心点对应坐标。velocity(1 ≤ velocity ≤ 5)表示该泊位的装载速度，单位是每帧可以装载的货物数。

### 船

一行一个整数 capacity(1 <= capacity <= 100)，表示船的容积，即最多能装的货物数。

## 输入格式

所有数据采用文本格式通过标准输入和标准输出进行交互，数值之间用空格分隔。

- **初始化：**
  - 选手程序初始化时，将输入：
    - 200 行*200 列的字符组成的地图数据
    - 一行泊位数量
    - 泊位数量行的泊位数据
    - 1 行船的容积
  - 然后紧接着一行 OK。
- **每一帧交互：**
  - 第一行输入 2 个整数，表示帧序号（从 1 开始递增）、当前金钱数。
  - 第二行输入 1 个整数，表示场上变化货物的数量 K（0≤新增货物数≤10）。
  - 紧接着 K 行数据，每一行表示一个变化货物，分别由如下所示的数据构成，共计 3 个数字：(金额为 0 表示上一帧被拿取或者该帧消失)

| 名称 | 数据类型 | 说明 |
| --- | --- | --- |
| 坐标 | 2 个整数 x,y | 该货物的坐标 |
| 金额 | 0 或正整数 | 该货物的金额 (<= 1000) |

  - 接下来输入 1 个整数，表示场上机器人的数量 R。
  - 紧接着 R 行数据，每一行表示一个机器人，分别由如下表格中所示的数据构成，每行 4 个数字。

| 名称 | 数据类型 | 说明 |
| --- | --- | --- |
| id | 整数 | • 机器人的 id |
| 是否携带物品 | 整数 | • 0 表示未携带物品。<br>• 1 表示携带物品。 |
| 坐标 | 2 个整数 x,y | • 该机器人的坐标 |

  - 接下来输入 1 个整数，表示场上轮船的数量 B。
  - 紧接着 B 行数据，每一行表示一个轮船，分别由如下表格中所示的数据构成，每行 6 个数字。

| 名称 | 数据类型 | 说明 |
| --- | --- | --- |
| id | 整数 | • 船的 id |
| 携带的货物数 | 整数 | • 携带的货物数量 |
| 坐标 | 2 个整数 x,y | • 该船核心点的坐标 |
| 方向 | 整数 | • 0 到 3 分别对应右、左、上、下。（和机器人移动的方向表示一致） |
| 状态 | 整数 | • 正常行驶状态（状态 0）<br>• 恢复状态（状态 1）<br>• 装载状态（状态 2） |

最后，判题器会输出一行 OK，表示所有数据已经写入完毕。

- 示例：合法的一帧输入可能是这样的：

```
8 1000
1
147 65 76
4
0 0 47 150
1 0 47 154
2 0 45 152
3 0 49 152
2
0 0 103 30 0 0
1 0 104 27 0 0
OK
```

- **判题结束：**

  判题结束时，判题器会关闭输入管道，同时选手程序会读到 EOF(end of file)，此时应当退出程序。

## 4.4.3 输出格式

- **初始化：**读入地图数据并完成初始化后，选手程序应当输出一行 OK，告诉判题器已就绪。
- **每一帧交互：**
  - 先进行机器人指令的输出，每行一个指令。
  - 支持的指令如下所示：

| 指令 | 参数 1 | 参数 2 | 说明 |
| --- | --- | --- | --- |
| lbot | x | y | 尝试在[x,y]处购买一个机器人 |
| move | ID | [0,3]之间的整数 | • 0 表示右移一格<br>• 1 表示左移一格<br>• 2 表示上移一格<br>• 3 表示下移一格 |

| 指令 | 参数 1 | 参数 2 | 说明 |
| --- | --- | --- | --- |
| get | ID | 无 | 如机器人在货物生成处且可拾取货物，则取货成功 |
| pull | ID | 无 | 如机器人在泊位处且处于携带货物状态，则放置成功 |

紧接着，每行一个轮船指令。支持的指令如下所示：

| 指令 | 参数 1 | 参数 2 | 说明 |
| --- | --- | --- | --- |
| lboat | x | y | 尝试在[x,y]处购买一个正方向向右的船。 |
| dept | ID | 无 | 尝试将对应船位置重置到主航道上，会导致船进入恢复状态。 |
| berth | ID | 无 | 尝试将对应船靠泊到泊位上，会导致船进入恢复状态。 |
| rot | ID | d<br>值为 0 或 1 | 0 为顺时针旋转。<br>1 为逆时针旋转。 |
| ship | ID | 无 | 向正方向前进 1 格（主航道） |

当你输出完所有指令后，紧跟一行 OK，表示输出结束。例如，一个可能的输出是：

```
lbot 50 50
lboat 60 60
move 3 2
ship 2
rot 1 0
OK
```

**说明**

大多数语言会默认对输出数据做缓冲，因此在输出完一帧控制数据时，你应该主动 **flush** 标准输出以避免判题器读不到数据导致超时。此外，由于标准输出已用于比赛交互，因此不要往标准输出打日志等其他内容，以免造成命令解析错误。平台上没有任何写文件权限，故正式提交版本不要写日志文件，你可以使用 stderr 将日志输出到控制台以方便调试。

## 4.5 异常判定与处理

- **程序异常：**判为 0 分。

  包括：程序崩溃、输出格式错误、指定机器人下标越界、指定船下标越界、指定泊位下标越界、输出数据量超长（1 帧内超 8KB）。

- **逻辑异常及参数错误的命令：**忽略该指令，不扣分。

在无法执行某个指令的时候执行了某个指令，该指令被忽略，不扣分。例如，无法取货、送货的时候执行了对应指令。

- **初始化响应超时**：判为 0 分。

  5 秒内未收到选手的 OK，则退出并返回 0 分。

- **控制帧响应超时**：忽略该帧控制指令

  选手程序没有在 15ms 内返回控制指令，则忽略该帧的控制行为（即：跳帧），直到收到控制指令之前，不会继续给选手程序下发数据。

  **注意**：跳帧后收到的控制数据会作用在当前最新的一帧，例如第 1000 帧下发的数据，选手程序直到 1010 帧才做出响应，那么该控制指令会作用于 1010 帧，此时有可能货物消失，机器人已经不在原来位置，从而导致一些取货、送货指令失败（失败的指令被忽略）。