Esquema Chechu

**programa MarfiaCecilia**

**procesos**

**areas**

**robots**  
 robot  
 variables  
   
 comenzar

fin

robot  
 variables  
   
 comenzar

fin

{--- PROGRAMA PRINCIPAL ---}  
**variables**

**comenzar**

AsignarArea (, )

AsignarArea (, )

AsignarArea (, )

AsignarArea (, )

AsignarArea (, )

Iniciar ()

Iniciar ()

Iniciar ()

**fin**

Algoritmos

{POSICIONA EL ROBOT PARA LA IZQUIERDA}

proceso izquierda

comenzar

repetir 3

derecha

fin

{JUNTA UNA FLOR. SI NO HAY ACTUALIZA UN BOOLEANO}

proceso juntarFlor (ES hayFlor: boolean)

comenzar

si (HayFlorEnLaEsquina)

tomarFlor

sino

hayFlor := F

fin

{JUNTA UN PAPEL. SI NO HAY ACTUALIZA UN BOOLEANO}

proceso juntarPapel (ES hayPapel: boolean)

comenzar

si (HayPapelEnLaEsquina)

tomarPapel

sino

hayPapel := F

fin

{JUNTA TODAS LAS FLORES DE UNA ESQUINA}

proceso juntarFlores (ES cantF: numero)

comenzar

mientras (HayFlorEnLaEsquina)

tomarFlor

cantF := cantF + 1

fin

{JUNTA TODOS LOS PAPELES DE UNA ESQUINA}

proceso juntarPapeles (ES cantP: numero)

comenzar

mientras (HayPapelEnLaEsquina)

tomarPapel

cantP := cantP + 1

fin

{RECORRE UNA AVENIDA}

proceso recorrerAvenida (ES cantF: numero)

comenzar

juntarFlores (cantF)

repetir 99

mover

juntarFlores (cantF)

fin

{RECORRE UNA CALLE}

proceso recorrerCalle (ES cantF: numero)

comenzar

juntarFlores (cantF)

derecha {porque se mueve en avenidas}

repetir 99

mover

juntarFlores (cantF)

fin

{GENERA UN RECTANGULO, EL ROBOT VA JUNTANDO FLORES}

proceso hacerRectangulo (E base: numero; E altura: numero)

comenzar

repetir 2

repetir altura

mientras (HayFlorEnLaEsquina)

tomarFlor

mover

derecha

repetir base

mientras (HayFlorEnLaEsquina)

tomarFlor

mover

derecha

fin

{ENVIA LOS IDENTIFICADORES A LOS ROBOTS}

proceso enviarId

comenzar

EnviarMensaje (1, r1)

EnviarMensaje (2, r2)

EnviarMensaje (3, r3)

EnviarMensaje (4, r4)

fin

{ENVIA A LOS ROBOTS SI PUEDEN FINALIZAR SU EJECUCION}

proceso enviarEstadoPrograma (E end: boolean)

comenzar

EnviarMensaje (end, r1)

EnviarMensaje (end, r2)

EnviarMensaje (end, r3)

EnviarMensaje (end, r4)

fin

{PROCESA UNA ESQUINA COMPARTIDA}

proceso procesarEsquinaCompartida (E x: numero; E y: numero; E cantF: numero; ES cantP: numero)

comenzar

BloquearEsquina (x, y)

Pos (x, y)

juntarObjetos (cantF, cantP)

mover

LiberarEsquina (x, y)

fin

{DEJA LAS FLORES ACUMULADAS UNA A LA VEZ EN UNA ESQ COMPARTIDA}

proceso dejarFlores (E cantF: numero; E avInicio: numero; E caInicio: numero)

comenzar

repetir cantF

BloquearEsquina (xy, xy)

Pos (xy, xy)

depositarFlor

Pos (avInicio, caInicio)

LiberarEsquina (xy, xy)

fin

{DEJA LOS PAPELES ACUMULADOS UNO A LA VEZ EN UNA ESQ COMPARTIDA}

proceso dejarPapeles (E cantP: numero; E avInicio: numero; E caInicio: numero)

comenzar

repetir cantP

BloquearEsquina (xy, xy)

Pos (xy, xy)

depositarPapel

Pos (avInicio, caInicio)

LiberarEsquina (xy, xy)

fin

{ACTUALIZA UN MAXIMO}

proceso actualizarMaximo (ES max: numero; ES idGanador: numero; E cant: numero; E id: numero)

comenzar

si (cant > max)

max := cant

idGanador := id

fin

{ACTUALIZA DOS MAXIMOS}

proceso actualizar2Maximos (ES max1: numero; ES max2: numero; ES idGanador1: numero; ES idGanador2: numero; E cant: numero; E id: numero)

comenzar

si (cant > max1)

max2 := max1

idGanador2 := idGanador1

max1 := cant

idGanador1 := id

sino

si (cant > max2)

max2 := cant

idGanador2 := id

fin

{EL JEFE RECIBE SI LOS ROBOTS FINALIZARON CON SU RECORRIDO}

{PARA EL CUERPO DEL ROBOT}

{SE PUEDE MODULARIZAR PERO EL ENTORNO NO RECIBE MENSAJES CORRECTAMENTE}

repetir xCantidad

RecibirMensaje (id, \*)

si (id = 1)

RecibirMensaje (finalizo, r1)

sino

si (id = 2)

RecibirMensaje (finalizo, r2)

sino

si (id = 3)

RecibirMensaje (finalizo, r3)

sino { si ( id = 4) }

RecibirMensaje (finalizo, r4)

{PARA MODULARIZAR. RECORDA COMENTAR POR QUE NO FUNCIONA}

proceso recibirEstadoRobots (ES id: numero)

variables

finalizo: boolean

comenzar

RecibirMensaje (id, \*)

si (id = 1)

RecibirMensaje (finalizo, r1)

sino

si (id = 2)

RecibirMensaje (finalizo, r2)

sino

si (id = 3)

RecibirMensaje (finalizo, r3)

sino { si (id = 4) }

RecibirMensaje (finalizo, r4)

fin

{AVISA A LOS ROBOTS QUIEN GANO}

proceso enviarGanadores (E id\_1: numero; E id\_2: numero)

comenzar

si (id\_1 = 1) | (id\_2 = 1)

EnviarMensaje (V, r1)

sino

EnviarMensaje (F, r1)

si (id\_1 = 2) | (id\_2 = 2)

EnviarMensaje (V, r2)

sino

EnviarMensaje (F, r2)

si (id\_1 = 3) | (id\_2 = 3)

EnviarMensaje (V, r3)

sino

EnviarMensaje (F, r3)

fin

{LLEVA LA CUENTA DE QUE EQUIPO TERMINA PRIMERO}

proceso determinarGanador (E ultId: numero; ES equipoGanador: numero)

comenzar

si (ultId = 2)

equipoGanador := 1

sino

equipoGanador := 2

fin

{VERIFICA QUE EQUIPO GANO}

proceso determinarGanador (E id: numero; ES equipoGanador: numero)

comenzar

si (id = 1) | (id = 2) {SI EL ULTIMO ID PERTENECE AL EQUIPO 1}

equipoGanador := 2 {EL GANADOR ES EL EQUIPO 2}

sino

equipoGanador := 1

fin

{SI EL ID MAXIMO COINCIDE CON EL PROPIO ES PORQUE GANO}

RecibirMensaje (idMax, robotJefe)

si (id = idMax)

Pos (xy, xy)

depositarFlores (cantF)

{ RANDOM }

Random (variable, inicio, fin)

{GENERA UNA ESQUINA ALEATORIA}

proceso determinarEsquina (E avInicio: numero; E caInicio: numero; ES avRandom: numero; ES caRandom: numero)

variables

av, ca: numero

comenzar

Random (av, 1, 10)

Random (ca, 1, 10)

avRandom := av + avInicio {suma la pos inicial del robot}

caRandom := ca + caInicio

fin

{ENVIA A TODOS LOS ROBOTS LA ESQUINA ALEATORIA}

proceso enviarEsquinaRandom (E avRandom: numero; E calleRandom: numero)

comenzar

EnviarMensaje (avRandom, r1)

EnviarMensaje (calleRandom, r1)

EnviarMensaje (avRandom, r2)

EnviarMensaje (calleRandom, r2)

EnviarMensaje (avRandom, r3)

EnviarMensaje (calleRandom, r3)

EnviarMensaje (avRandom, r4)

EnviarMensaje (calleRandom, r4)

fin

{AVISA AL ROBOT ALEATORIO QUE PUEDE REALIZAR LAS ACCIONES QUE CORRESPONDAN Y AL RESTO QUE NO}

proceso elegirRobotRandom

variables

nro: numero

comenzar

Random (nro, 1, 4) {elige un robot al azar entre los 4 que hay}

{SE ENVIAN A LOS ROBOTS}

si (nro = 1)

EnviarMensaje (V, r1)

EnviarMensaje (F, r2)

EnviarMensaje (F, r3)

EnviarMensaje (F, r4)

sino

si (nro = 2)

EnviarMensaje (V, r2)

EnviarMensaje (F, r3)

EnviarMensaje (F, r4)

EnviarMensaje (F, r1)

sino

si (nro = 3)

EnviarMensaje (V, r3)

EnviarMensaje (F, r4)

EnviarMensaje (F, r1)

EnviarMensaje (F, r2)

sino { si (nro = 4) }

EnviarMensaje (V, r4)

EnviarMensaje (F, r1)

EnviarMensaje (F, r2)

EnviarMensaje (F, r3)

{FIN DE SIES ANIDADOS}

fin

{ PARA HACER UNA ESCALERA: MODULO ESCALON + MODULO ESCALERA }

{HACE UN ESCALON}

proceso escalon (ES cantF: numero; ES cantP: numero)

variables

alto: numero

comenzar

cantF := 0

cantP := 0

Random (alto, 1, 5)

repetir alto

juntarFlores (cantF)

juntarPapeles (cantP)

mover

derecha

mover

fin

{HACE UNA ESCALERA DE 4 ESCALONES}

proceso escalera (...)

variables

cantF, cantP: numero

comenzar

repetir 4

escalon (cantF, cantP)

procesarEscalon (cantF, cantP)

izquierda {queda mirando para arriba}

fin

{EJEMPLO CON CUERPO DE ROBOT HACIENDO EL RECORRIDO “MANUAL”}

procesos

{ENVIA LOS IDENTIFICADORES A LOS ROBOTS}

proceso enviarId

comenzar

EnviarMensaje (1, r1)

EnviarMensaje (2, r2)

fin

{VERIFICA SI HAY UNA FLOR EN LA ESQUINA. SI HAY INCREMENTA UN CONTADOR}

proceso buscarFlor (ES contador: numero)

comenzar

si (HayFlorEnLaEsquina)

contador := contador + 1

fin

{PROCESA "PASOS" CANTIDAD DE ESQUINAS LLEVANDO UN CONTADOR POR ESQUINAS CON AL MENOS UNA FLOR}

proceso procesarEsquinas (E pasos: numero; ES contador: numero)

comenzar

repetir pasos

buscarFlor (contador)

mover

buscarFlor (contador)

fin

{UN ROBOT CORREDOR PROCESA UNA ESQUINA COMPARTIDA}

proceso procesarEsquinaCompartidaCorredor (E x: numero; E y: numero; ES contador: numero)

comenzar

BloquearEsquina (x, y)

Pos (x, y)

buscarFlor (contador)

mover

LiberarEsquina (x, y)

fin

{JUNTA TODAS LAS FLORES DE UNA ESQUINA}

proceso juntarFlores

comenzar

mientras (HayFlorEnLaEsquina)

tomarFlor

fin

{JUNTA TODOS LOS PAPELES DE UNA ESQUINA}

proceso juntarPapeles

comenzar

mientras (HayPapelEnLaEsquina)

tomarPapel

fin

{JUNTA TODOS LOS OBJETOS DE UNA ESQUINA}

proceso juntarObjetos

comenzar

juntarFlores

juntarPapeles

fin

{PROCESA UNA ESQUINA COMPARTIDA}

proceso procesarEsquinaCompartida (E x: numero; E y: numero)

comenzar

BloquearEsquina (x, y)

Pos (x, y)

juntarObjetos

mover

LiberarEsquina (x, y)

fin

{RECORRE "cant" AVENIDAS JUNTANDO TODOS LOS OBJETOS QUE ENCUENTRE}

proceso recorrerAvenidas (E cant: numero)

comenzar

juntarObjetos

repetir cant

mover

juntarObjetos

fin

areas

{AREA JEFE}

areaRobotJefe : AreaP (1, 1, 1, 1)

{AREAS DE TRABAJO}

area1Robot1 : AreaP (15, 1, 15, 19)

area2Robot1 : AreaP (15, 21, 15, 46)

area1Robot2 : AreaP (25, 1, 25, 19)

area2Robot2 : AreaP (25, 21, 25, 46)

area1RobotLimpiador : AreaP (1, 20, 14, 20)

area2RobotLimpiador : AreaP (16, 20, 24, 20)

area3RobotLimpiador : AreaP (26, 20, 100, 20)

{AREAS DE CHOQUE}

esquinaCompartida1 : AreaPC (15, 20, 15, 20)

esquinaCompartida2 : AreaPC (25, 20, 25, 20)

robots

robot corredor

variables

av, ca, id, pasos, contador, limite: numero

comenzar

contador := 0

av := PosAv {PARA QUE CADA ROBOT CORREDOR TENGA EN QUE AVENIDA ESTA PARADO}

RecibirMensaje (id, rJefe)

{EPIEZA 1RA PARTE DEL RECORRIDO}

ca := PosCa

limite := 19

Random (pasos, 1, 6)

mientras ((ca + pasos) < limite) {SI CORTA ES PORQUE PosCa + pasos >= 19}

procesarEsquinas (pasos, contador)

Random (pasos, 1, 6)

ca := PosCa

pasos := limite - ca {PARA SABER CUANTOS PASOS DE MAS SE HACEN PARA QUE SE PRODUZCA EL CHOQUE}

procesarEsquinas (pasos, contador) {RECORRE CALLES PENDIENTES APARTE PARA NO CHOCAR}

procesarEsquinaCompartidaCorredor (av, 20, contador) {PROCESA ESQUINA COMPARTIDA - EN ESTE PROCESO SE MUEVEN 2 ESQUINAS}

{HACE 2DA PARTE DEL RECORRIDO - POST ESQ COMPARTIDA}

ca := PosCa

limite := 46

Random (pasos, 1, 6)

mientras ((ca + pasos) < limite)

procesarEsquinas (pasos, contador)

Random (pasos, 1, 6)

ca := PosCa

pasos := limite - ca {PARA SABER CUANTOS PASOS DE MAS SE DEBEN HACER}

procesarEsquinas (pasos, contador) {RECORRE CALLES PENDIENTES}

EnviarMensaje (id, rJefe)

EnviarMensaje (contador, rJefe)

fin

robot limpiador

variables

av: numero

comenzar

derecha

recorrerAvenidas (13) {EL CHOQUE SE PRODUCE EN 15, POR LO TANTO NO PUEDE HACER 14 MOVER}

procesarEsquinaCompartida (15, 20) {PROCESA LA ESQUINA COMPARTIDA (15, 20) - HACE EL MOVER NRO 14}

recorrerAvenidas (8) {8 PORQUE DESP DE LA COMPARTIDA QUEDA EN LA AVENIDA 16 --> 25 - 16 = 9, PERO SIEMPRE HAY QUE CONSIDERAR QUE SE TIENEN QUE DAR CANT-1 PASOS!!!}

procesarEsquinaCompartida (25, 20) {PROCESA LA ESQUINA COMPARTIDA (25, 20)}

recorrerAvenidas (74) {TIENE QUE RECORRER HASTA LA 100. DESP DE LA COMPARTIDA TERMINO EN LA 26. (100 - 26) - 1paso = 73}

{PROBAR CON 74, SI NO ANDA ES PORQ SON 73}

fin

robot jefe

variables

id, contador: numero

comenzar

enviarId

repetir 2

RecibirMensaje (id, \*)

si (id = 1)

RecibirMensaje (contador, r1)

sino

RecibirMensaje (contador, r2)

{FIN DEL REPETIR}

Informar ('ROBOT\_QUE\_TERMINO\_ULTIMO\_', id)

Informar ('ESQUINAS\_RECORRIDAS\_POR\_EL\_ROBOT\_CON\_AL\_MENOS\_UNA\_FLOR\_', contador)

fin

CONCEPTOS

* Cuando tengo 2 o mas robots que hacen lo mismo pero quiero que se comuniquen entre ellos deben ser declarados como robots distintos!!! No pueden ser del mismo tipo.
* Error “NullPointerException” es un problema con una variable no declarada.
* Areas:

AreaP () : Privada

El acceso es para UN solo robot durante el programa.

AreaPC () : Parcialmente compartida

Al menos un robot no entra al área durante el programa.

AreaC () : Compartida

Todos los robots entran al área al menos una vez en el programa.

* *EnviarMensaje(variable, robotDeDestino)* {envía el mensaje y **sigue ejecutando las líneas de código** - no se queda trabado mi ciela ;) }
* *RecibirMensaje(variableDondeGuardoElDato, robotOrigenDelMensaje)* {lo recibo del robot específico}
* RecibirMensaje(variableDondeGuardoElDato, \*) {espero recibir un mensaje de cualquier robot, el que llegue primero, **hasta que no llegue el mensaje no puedo seguir ejecutando el código**}
* La cantidad de mensajes enviados debe ser la misma que de mensajes recibidos.
* Evitar que los robots se queden trabados en los RecibirMensaje (maximizamos la concurrencia :D)
* Uso del \* (es como decir “cualquier robot”).
* Cuando lo uso “no tengo manera de saber cuál es el robot que envió el mensaje”, por eso debo identificarlos previamente:
* Pasitos

Para el **robot trabajador**:

1. RecibirMensaje(id,robotJefe) {recibo el identificador enviado por el jefe}
2. Hago lo que tengo que hacer {tomar flores o papeles, depositar, dar unas vueltitas}
3. EnviarMensaje(id,robotJefe) {PRIMERO MANDO EL ID}
4. EnviarMensaje(cantFlores, robotJefe) {le mando p. ej. la cantidad de flores que junté}

Para el **robot jefe / fiscalizador / coordinador**

1. enviarId {si es como pusimos arriba que son varios robots, sino directamente...}

EnviarMensaje (1, r1) // y así sucesivamente según la cantidad de robots que hay...

2)RecibirMensaje(id,\*){recibo el id del robot}

en el caso de ser varios el recibirEstadoRobots que está declarado más arriba. IMPORTANTE: recordar que RecibirMensaje(cantFlores,\*) -p. ej. ESTÁ MAL porque no sabríamos de quién son esas flores juntadas.

si (id = 1)

RecibirMensaje (finalizo, r1)

sino

si (id = 2)

RecibirMensaje (finalizo, r2)

así es como hay que hacerlo, sucesivamente según la cantidad de robots…

Hago un ejemplo general con pseudocódigo:

robot tipo1

recibo el identificador

junto las flores/papeles o lo que pito tenga que hacer

mando el id al robot jefe/fiscalizador/coordinador (que recibí al principio)

mando la cantidad de lo que junté (flores, papeles)

robot tipoJefe

envia el identificador a cada robot

recibo el id del que termino primero (conviene usar el \* para que no se quede esperando toda una vida)

identifico al robot que quiere mandar (esto es lo de los si - sino)

recibo el dato (por ej. cantidad de flores/papeles)

proceso el dato (digamos que, como sabemos de quién es el dato podríamos calcular un máximo o algo así).

Bloqueo de esquinas

*BloquearEsquina(avABloquear,caABloquear)*{si la esquina no está bloqueada, se bloquea y **SIGUE SU EJECUCIÓN**}

//si la esquina ya estaba bloqueada, el proceso “queda esperando” para poder ejecutar la instrucción de bloqueo hasta que la esquina NO esté bloqueada. Hay que garantizar que alguien la va a desbloquear.

*LiberarEsquina(avADesbloquear,caADesbloquear)* {Si la esquina está bloqueada se desbloquea y el proceso **SIGUE SU EJECUCIÓN**.

//Si la esquina no estaba bloqueada, no hace nada, pero no estaría bien tampoco porque no debería desbloquearse algo que no fue previamente bloqueado.

Debe entonces haber la misma cantidad de bloquear que liberar esquina.

P. ej.

Quiero ir a la esquina (10,10) y otro robot puede acceder a esta esquina también.

Arranco desde la (2,2)

comienzo

me guardo la esquina inicial -en este caso la (2,2)-

Bloqueeo la esquina(10,10)

me posiciono en la esquina (10,10)

hago lo que tenga que hacer

vuelvo a mi posicion de origen (2,2)

una vez que me aseguré ya no estar en la esquina bloqueada… la libero.

**---TEORÍA---**

* **Concurrencia**: capacidad de ejecutar varias actividades en paralelo o en simultáneo.
* Un programa concurrente se divide en tareas (2 o mas), las cuales se ejecutan al mismo tiempo y realizan acciones para cumplir un objetivo comun.  
  Para esto pueden compartir recursos, coordinarse y cooperar.
* Comunicacion y Sincronizacion: cada tarea/proceso debe comunicarse y sincronizarse para resolver el problema comun.  
  Para comunicarse los procesos van a tener 2 alternativas:  
  + A traves del ENVIO DE MENSAJES:
    - tiene un **origen** (el que lo manda), un **destino** (a donde se manda), un **contenido**(el mensaje en sí que yo quiero enviar).
    - SIEMPRE es necesario establecer un canal físico o lógico para transmitir información entre procesos.
    - El lenguaje debe proveer un protocolo adecuado (POR EJ: primero va el origen, después el destino, después el contenido).
    - Para que la comunicación sea efectiva tenemos que tener una operación de enviar y una de recibir mensajes, ese envío/ recepción puede ser sincrónico o asincrónico.
    - En cada envío solo se puede mandar un valor, si quiero mandar más de uno tengo que hacer otro envío de mensaje con el otro valor.
    - La cantidad de mensajes enviados debe ser la misma que de mensajes recibidos.
    - La recepción SIEMPRE es sobre una variable.
  + A través de MEMORIA COMPARTIDA (uso de semáforos P y V): dos o más procesos dejan o retiran información en una zona de memoria a la que ambos procesos pueden acceder.
    - Solo un proceso a la vez puede acceder a la variable compartida, el otro va a quedar "dormido" o esperando.
    - Siempre hay que preguntar si esa zona de memoria esta disponible (en R-info lo hacemos automáticamente al BloquearEsquina).
    - DISPONIBLE: se bloquea para que no esté disponible para otro proceso, se hacen las operaciones necesarias y luego se libera la zona.
    - NO DISPONIBLE: el proceso queda preguntando hasta que esa zona se libere.
    - El lenguaje o el SO se va a encargar de despertar al proceso que está dormido (si se tiene más de un proceso dormido es el SO o el programa ejecutado el que va a señalar el proceso con mayor prioridad (es decir, el que se va a despertar primero)).

CONCEPTOS DE AMBIENTE CMRE:

* Recursos compartidos.
* Sincronización (implica que los procesos esperen hasta que otro proceso termine de hacer X cosa con la variable compartida, deben sincronizarse porque tienen que esperar).
* Procesadores heterogéneos.

IMPORTANTE:

* El error “NullPointerException” es un problema con una variable no declarada.
* Cuando tengo 2 o más robots que hacen lo mismo pero quiero que se comuniquen entre ellos deben ser declarados como robots distintos!!! No pueden ser del mismo tipo.

DECLARACIÓN DE ÁREAS:

* Área Compartida: puede posicionarse cualquiera de las variables robot que están declaradas en el programa. **AreaC**
* Área Privada: solo puede posicionarse el robot al que se le asigno esa area (es decir, puede haber un único robot). **AreaP**
* Área Parcialmente Compartida: se declara cuando un subconjunto de robots quieren posicionarse en el mismo lugar. *Debe haber por lo menos un robot al que no este asignada.* **AreaPC**

***UNA ESQUINA NO PUEDE PERTENECER A DOS AREAS AL MISMO TIEMPO. LAS AREAS TIENEN QUE SER DISJUNTAS.***

***UN ROBOT PUEDE ESTAR ASIGNADO A 1 AREA O MAS DEL PROGRAMA***

------------------------------------- CUANDO USAR EL “ \* “ -------------------------------------

Se genera un problema cuando el código del robot jefe tiene que recibir mensajes de otros robots y se establece un orden, porque si el robot jefe recibe un mensaje del robot1 y un mensaje del robot2 y en el código primero se recibe el mensaje del robot1, entonces si el robot2 llega a terminar antes el robot jefe no le va a dar bola! Por ello es que SOLO HAY QUE FORZAR EL ORDEN CUANDO LO PIDE LA CONSIGNA. En caso de que no se pida se debe cambiar la sintaxis de recepción de mensajes de forma tal que en vez de poner RecibirMensaje (variable, robotX), pongamos RecibirMensaje(variable,\*).

Ese asterisco va a permitir que se pueda recibir la información de cualquiera de los robots que están ejecutándose.

{ Cuando se utiliza el \*, eso no implica que en \* esté almacenado el número del robot que hizo el envío }