



# Taller de Programación



# AGENDA

Evolución de Arquitecturas

Conceptos de Concurrencia

Ejemplos



# NUESTRA VIDA – Hoy...



NAVEGADORES



SAMARTPHONE

SISTEMAS  
OPERATIVOS



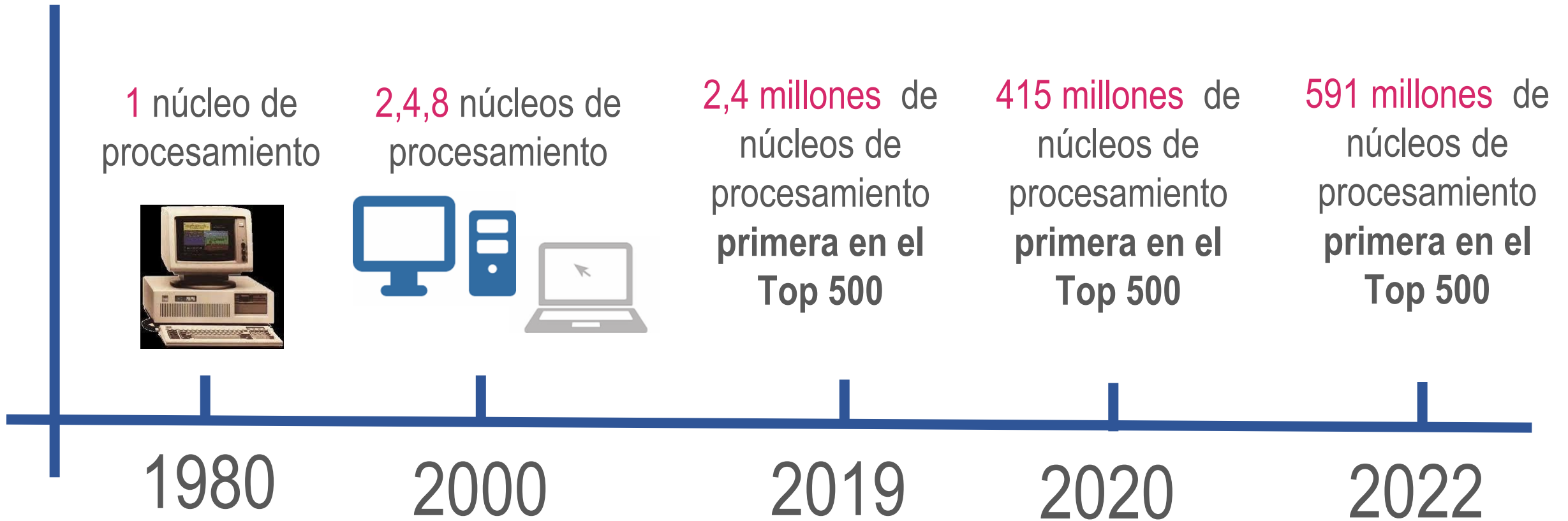
CUENTAS  
BANCARIAS



*Qué características  
comunes hay en  
estos ejemplos?*



# Evolución de las Arquitecturas

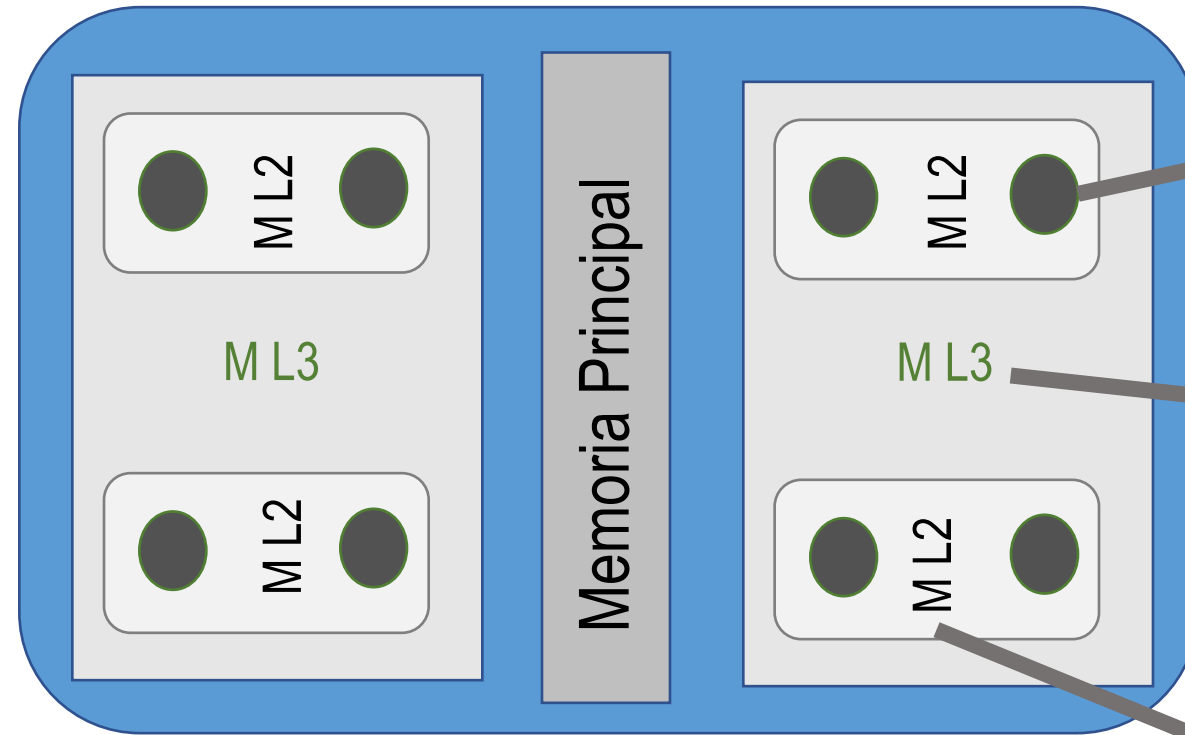


**Cómo es un procesador con más de un núcleo?**



# Evolución de las Arquitecturas

8  
NÚCLEOS



NÚCLEO

MEMORIA  
CACHE  
(nivel3)

MEMORIA  
CACHE  
(nivel2)

VELOCIDAD



MEMORIA LOCAL  
MEMORIA CACHE (nivel 2)  
MEMORIA CACHE (nivel 3)  
MEMORIA PRINCIPAL

CAPACIDAD





# CONCURRENCIA



Un programa concurrente se divide en tareas (2 o más), las cuales se ejecutan al mismo tiempo y realizan acciones para cumplir un objetivo común. Para esto pueden: compartir recursos, coordinarse y cooperar.

## CARACTERISTICAS

Concepto clave en la Ciencia de la Computación

Cambios en HARDWARE y SOFTWARE

## CONCEPTOS

**COMUNICACIÓN**  
**SINCRONIZACION**



# CONCURRENCIA - Ejemplos

Supongamos que una pareja Paula y Juan comparten una cuenta bancaria.



En algún momento ambos salen a sus trabajos y deciden detenerse en un cajero para extraer 1000 pesos

Si en la cuenta hay 50000 pesos es de esperar que después de las dos extracciones queden 48000.

## CUENTA BANCARIA



**Podría ocurrir que ambos accedan a la cuenta en el mismo instante**

## CONCURRENCIA



# CONCURRENCIA - Ejemplos

CUENTA BANCARIA: **saldo** VARIABLE COMPARTIDA



Integrante 1:

```
{  
  ingresa la clave  
  saldo:= saldo - 1000;  
}
```



¿Cómo se protege  
la variable saldo?

Integrante 2:

```
{  
  ingresa la clave  
  saldo:= saldo - 1000;  
}
```



Cualquier lenguaje que brinde  
conurrencia debe proveer  
mecanismos para **comunicar** y  
**sincronizar** procesos.



En este caso quiero **proteger** el acceso a la  
variable compartida (dos procesos no  
accedan al mismo tiempo, sincronicen)

Semáforos (P y V)

Monitores

Pasaje de Mensajes





# CONCURRENCIA - Ejemplos

CUENTA BANCARIA: **saldo** VARIABLE COMPARTIDA




Integrante 1:

```
{  
  P(saldo)  
  
  ingresa clave  
  
  saldo:= saldo - 1000;  
  
  V(saldo)  
}
```

¿Cómo funciona?

Integrante 2:



```
{  
  P(saldo)  
  
  ingresa clave  
  
  saldo:= saldo - 1000;  
  
  V(saldo)  
}
```



¿Este código puede ser más eficiente?



# CONCURRENCIA - Ejemplos

**CUENTA BANCARIA:** **saldo** VARIABLE COMPARTIDA



Integrante 1:

```
{
  ingresar clave
  P(saldo)
  saldo := saldo - 1000;
  V(saldo)
}
```



¿Cómo funciona?

Integrante 2:

```
{
  ingresa clave
  P(saldo)
  saldo := saldo - 1000;
  V(saldo)
}
```



¿Alcanza si hago el cambio en uno de los dos integrantes?



# CONCURRENCIA - Ejemplos



En un programa existen 3 procesos, un arreglo de longitud M y un valor N y se quiere calcular cuántas veces aparece el valor N en el arreglo.



cont

V



Proceso 1

Proceso 2

Proceso 3

```
Proceso 1:
{inf:=...; sup:= ...;
  P(cont)
  for i:= inf to sup do
    if v[i] = N then
      cont:= cont + 1;
  V(cont)
}
```

```
Proceso 2:
{inf:=...; sup:= ...;
  P(cont)
  for i:= inf to sup do
    if v[i] = N then
      cont:= cont + 1;
  V(cont)
}
```

```
Proceso 3:
{inf:=...; sup:= ...;
  P(cont)
  for i:= inf to sup do
    if v[i] = N then
      cont:= cont + 1;
  V(cont)
}
```

¿cómo se puede mejorar?



# PROGRAMA CONCURRENTE - Características

## Programa Secuencial

```
<html><head><title>
<meta name="description" content="HTML Tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="all"><@import "/us.css";</style>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/opensearch+xml" title="HTML 5.0" href="http://www.html5tutorial.com/opensearch.xml">
</head>
<script src="/scripts.js" type="text/javascript"></script>
<style type="text/css">
</style>
</html>
```



## Programa Concurrente

```
<html><head><title>
<meta name="description" content="HTML Tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="all"><@import "/us.css";</style>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/opensearch+xml" title="HTML 5.0" href="http://www.html5tutorial.com/opensearch.xml">
</head>
<script src="/scripts.js" type="text/javascript"></script>
<style type="text/css">
</style>
</html>
```

```
<html><head><title>
<meta name="description" content="HTML Tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="all"><@import "/us.css";</style>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/opensearch+xml" title="HTML 5.0" href="http://www.html5tutorial.com/opensearch.xml">
</head>
<script src="/scripts.js" type="text/javascript"></script>
<style type="text/css">
</style>
</html>
```



## Programa Paralelo

```
<html><head><title>
<meta name="description" content="HTML Tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="all"><@import "/us.css";</style>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/opensearch+xml" title="HTML 5.0" href="http://www.html5tutorial.com/opensearch.xml">
</head>
<script src="/scripts.js" type="text/javascript"></script>
<style type="text/css">
</style>
</html>
```



```
<html><head><title>
<meta name="description" content="HTML Tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="_top">
<style type="text/css" media="all"><@import "/us.css";</style>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/opensearch+xml" title="HTML 5.0" href="http://www.html5tutorial.com/opensearch.xml">
</head>
<script src="/scripts.js" type="text/javascript"></script>
<style type="text/css">
</style>
</html>
```





# PROGRAMA CONCURRENTE - Características

## Programa Concurrente

```
<meta name="description" content="HTML tutorial">  
<meta name="author" content="Andrew">  
<meta name="copyright" content="2000-2011 and beyond...">  
<meta name="robots" content="all">  
<meta name="viewport" content="width=768">  
<base target="_top">
```

```
<style type="text/css" media="all">  
<link rel="stylesheet" type="text/css" href="/us.css" />  
<link rel="stylesheet" type="text/css" href="/print.css" media="print" />  
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico" />  
<link rel="search" type="application/x-sparql-query" href="/opensearch.html" />  
<script />  
</script>
```

**COMUNICACIÓN**

**SINCRONIZACIÓN**





# PROGRAMA CONCURRENTE - Comunicación

## Programa Concurrente

```
<html><html><title>  
<meta name="description" content="HTML tutorial">  
<meta name="author" content="Andrew">  
<meta name="copyright" content="2000-2011 and beyond...">  
<meta name="robots" content="all">  
<meta name="viewport" content="width=780">  
<base target="">
```

```
<link target="_top">  
<style type="text/css" media="e"><link href="/us.css"/></style>  
<link rel="stylesheet" type="te"><link href="/print.css" media="e">  
<link rel="shortcut icon" type="e"><link href="/favicon.ico">  
<link rel="search" type="applie"><link href="/search.xml">  
<script>  
</script>
```



proceso 1



proceso 2



COMUNICACION

PASAJE DE  
MENSAJES

MEMORIA  
COMPARTIDA



# PROGRAMA CONCURRENTE - Comunicación

## Programa Concurrente

```
<meta name="description" content="HTML tutorial">  
<meta name="author" content="Andrew">  
<meta name="copyright" content="2000-2011 and beyond...">  
<meta name="robots" content="all">  
<meta name="viewport" content="width=780">  
<base target="" src="">
```

```
<link target="_top">  
<style type="text/css" media="e"> <link href="/us.css" /></style>  
<link rel="stylesheet" type="text/css" href="/print.css" media="e">  
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">  
<link rel="search" type="application/opensearch+xml" title="HTML Search" href="/search.xml">  
</script>
```



Forma de un mensaje

Origen  
Destino  
Contenido

## PASAJE DE MENSAJES

ENVIAR  
RECIBIR

Es necesario establecer un canal (lógico o físico) para transmitir información entre procesos.

También el lenguaje debe proveer un protocolo adecuado.

Para que la comunicación sea efectiva los procesos deben “saber” cuándo tienen mensajes para leer y cuando deben transmitir mensajes.



# PROGRAMA CONCURRENTE - Comunicación

## Programa Concurrente

## MEMORIA COMPARTIDA

**BLOQUEAR  
DESBLOQUEAR**

```
<!-- HTML header -->
<meta name="description" content="HTML tutorial">
<meta name="author" content="Andrew">
<meta name="copyright" content="2000-2011 and beyond...">
<meta name="robots" content="all">
<meta name="viewport" content="width=780">
<base target="">
```

```
<script type="text/javascript">
</script>
<script type="text/css" media="all">
</script>
<link rel="stylesheet" type="text/css" href="/print.css" media="print">
<link rel="shortcut icon" type="image/x-icon" href="/favicon.ico">
<link rel="search" type="application/x-sparql+xml" href="/search.xml">
</script>
```

Recurso  
Compartido

LIBRE?

si

no

Bloqueo  
Uso  
Libero

Los procesos intercambian información sobre la memoria compartida o actúan coordinadamente sobre datos residentes en ella.

Lógicamente no pueden operar simultáneamente sobre la memoria compartida, lo que obliga a bloquear y liberar el acceso a la memoria.

La solución más elemental es una variable de control que habilite o no el acceso de un proceso a la memoria compartida.

**Cómo utilizamos CMRE?**





# Taller de Programación



# AGENDA



## Ambiente CMRE



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

Información de la ciudad

Cómo aplica CMRE los conceptos de concurrencia?

Código ejemplo

Información de cada robot

Ciudad para la ejecución

R-Info | Versión 3.0

Elemento: Flores Avenida: \* Calle: \*

Cantidad: 0 Agregar

ROBOTS

Robot	Flores	Papeles	Color
-------	--------	---------	-------

Miniatura...

```
1. {Bienvenidos al entorno CMRE.
2. Lo siguiente es un código de ejemplo que implementa un
3. proceso que recibe un número de avenida como parámetro,
4. se posiciona en esa avenida y la recorre.}
5.
6. programa ejemplo
7. procesos
8. proceso recorrerAvenida(E numAv: numero)
9. comenzar
10. Pos(numAv, 1)
11. repetir 99
12. mover
```

ROBOTS EN EJECUCION



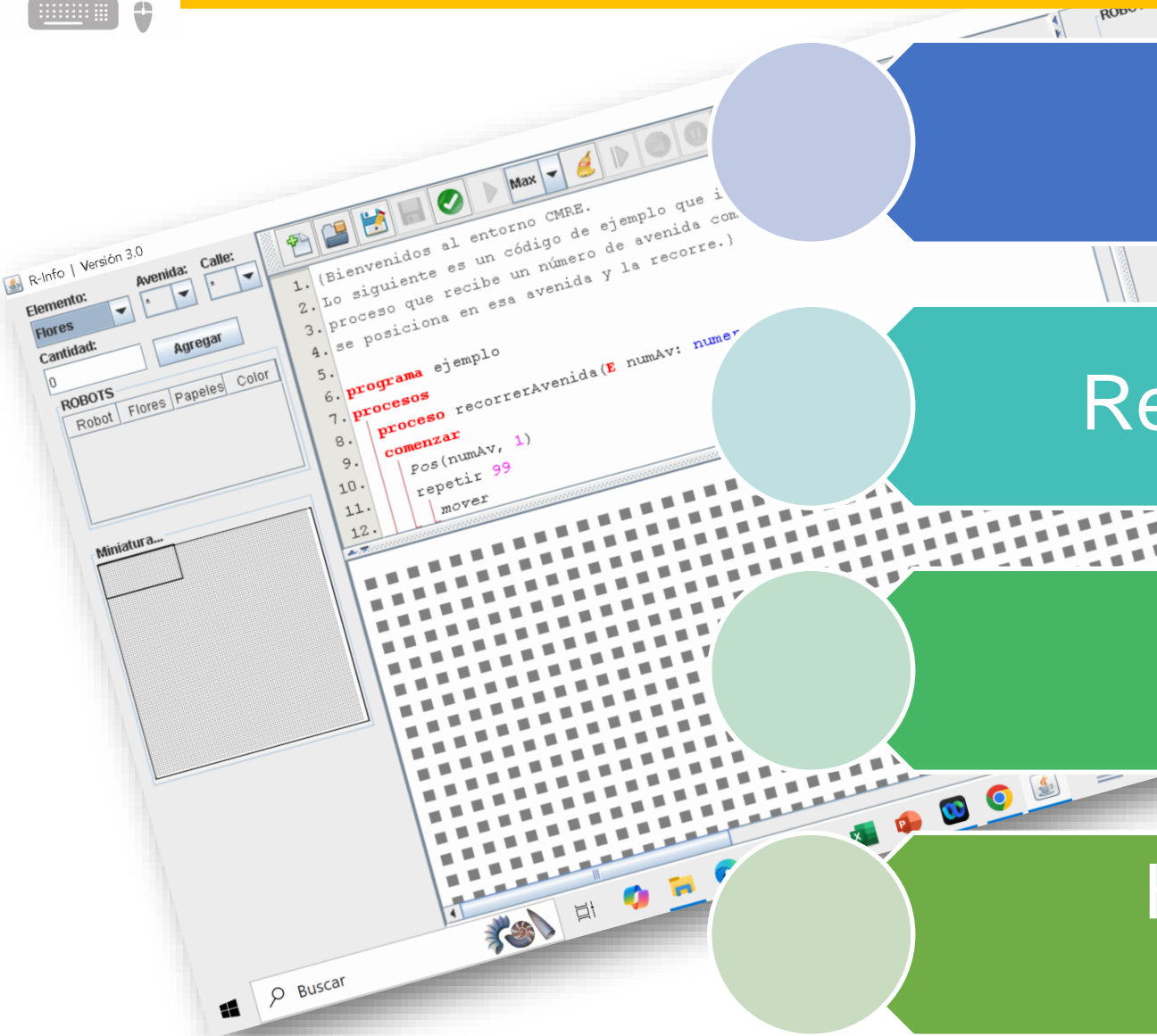
# PROGRAMA CONCURRENTE – AMBIENTE CMRE

Comunicación

Recursos Compartidos

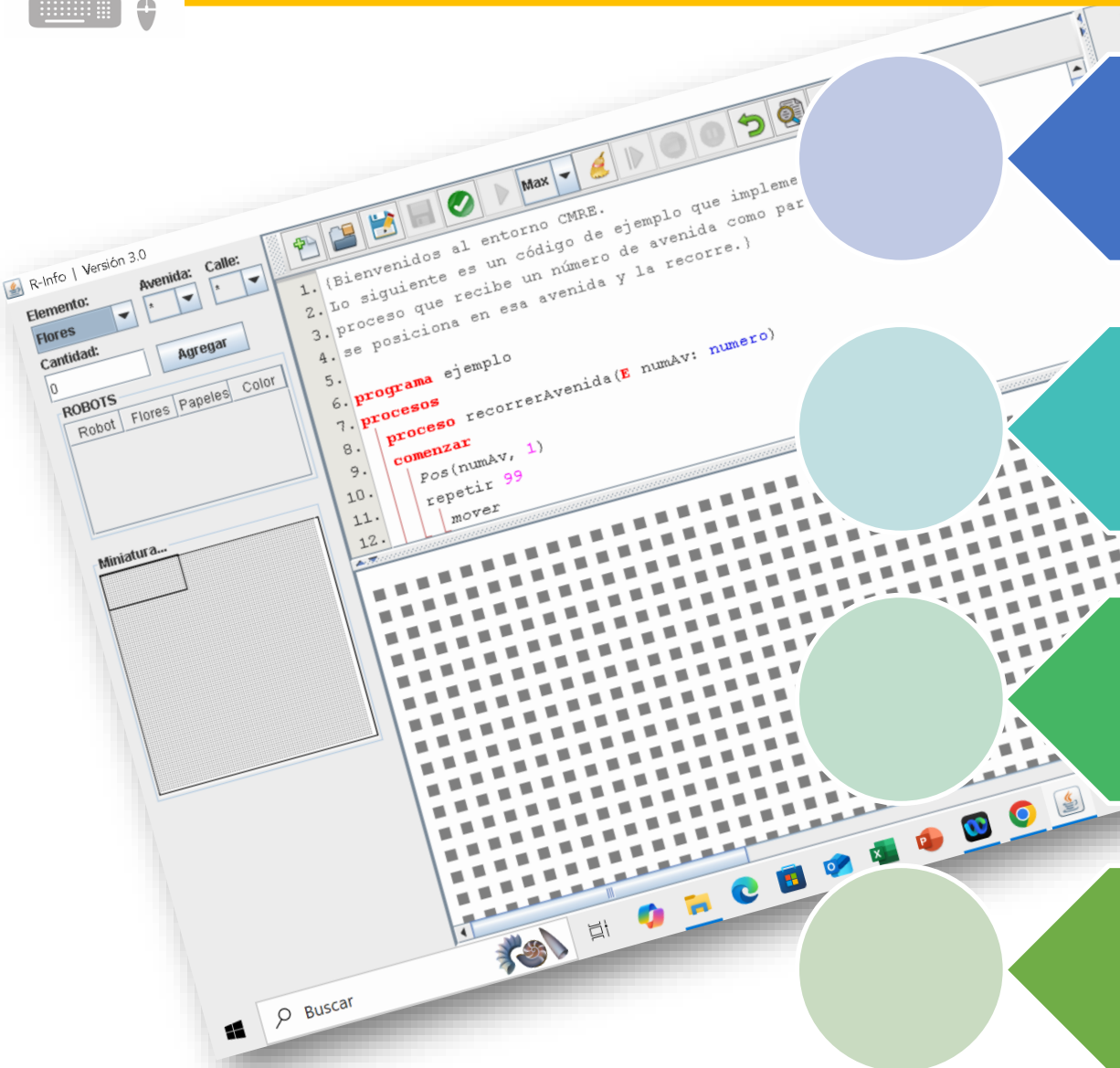
Sincronización

Heterogeneidad de  
procesadores





# PROGRAMA CONCURRENTE – AMBIENTE CMRE



ROBOTS: se permite declarar más de un robot

AREAS: existen distintos tipos de áreas (privadas, compartidas, parcialmente compartidas)

COMUNICACIÓN: permite el intercambio de mensajes entre robots

SINCRONIZACIÓN: permite bloquear y desbloquear recursos compartidos (esquinas)



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

## ESTRUCTURA DE UN PROGRAMA

**programa** ejemplo

**procesos**

proceso recorrerAvenida(E numAv: numero)

comenzar

Pos(numAv, 1)

...

fin

**areas**

ciudad: AreaC (1,1,100,100)

**robots**

robot robot1

comenzar

recorrerAvenida(1)

fin

**variables**

R\_info: robot1

**comenzar**

AsignarArea(R\_info, ciudad)

Iniciar(R\_info, 1,1)

**fin**

Procesos  
necesarios

Areas  
necesarias

Tipo de robots  
que ejecutarán  
el programa

Variables robots  
(mínimo 1 por tipo  
declarado)

Asignación de  
área/áreas de  
ejecución de los  
robots declarados y  
esquina de comienzo



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

```
proceso recorrerAvenida(E numAv: numero)
```

```
comenzar
```

```
    Pos(numAv, 1)
```

```
    ...
```

```
fin
```

**areas**

```
ciudad: AreaC (1,1,100,100)
```

**robots**

```
robot robot1
```

```
comenzar
```

```
    recorrerAvenida(1)
```

```
fin
```

**variables**

```
R_info: robot1
```

**comenzar**

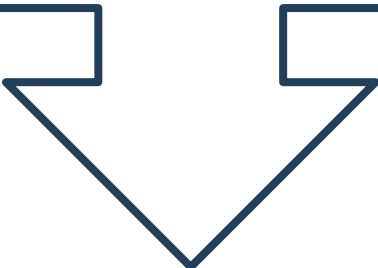
```
    AsignarArea(R_info, ciudad)
```

```
    Iniciar(R_info, 1,1)
```

**fin**

```
proceso nombre (ES flores:numero; E valor:boolean)
variables
    nombre : tipo

comenzar
    //código del proceso
fin
```





# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

```
proceso recorrerAvenida(E numAv: numero)
```

```
comenzar
```

```
    Pos(numAv, 1)
```

```
    ...
```

```
fin
```

**areas**

```
ciudad: AreaC (1,1,100,100)
```

**robots**

```
robot robot1
```

```
comenzar
```

```
    recorrerAvenida(1)
```

```
fin
```

**variables**

```
R_info: robot1
```

**comenzar**

```
AsignarArea(R_info, ciudad)
```

```
Iniciar(R_info, 1,1)
```

**fin**

ciudad1: **areaC**(1,1,10,10)  
ciudad2: **areaP**(15,15,20,20)  
ciudad3: **areaPC**(30,32,50,51)

areaC: área compartida (pueden ser asignados todos los robots declarados)

areaP: área privada (puede ser asignado sólo un robot de los declarados)

areaPC: área parcialmente compartida (debe ser asignado más de un robot pero NO todos los robots declarados)





# PROGRAMA CONCURRENTE – AMBIENTE CMRE

R-Info | Versión 3.0

Elemento: Flores Avenida: Calle: Cantidad: 0 Agregar

Robot	Flores	Papeles	Color
R_info	0	0	Red
R_dos	0	0	Blue

Miniatura...

```
14. areas
15. ciudad: AreaC (1,1,10,10)
16. ciudad2: AreaP (15,1,20,10)
17. ciudad3: AreaPC (25,1,33,10)
18. robots
19. robot robot1
20. comenzar
21. | recorrerAvenida(1)
22. fin
23. variables
24. | R_info: robot1
25. | R_dos: robot1
26. comenzar
```

Área privada

Área compartida

Área parcialmente compartida

Robots declarados

ROBOTS EN EJECUCION

R\_info Pos: (00,00)  
Bolsa Esquina  
F P F P  
00 00 00 00  
No iniciado

R\_dos Pos: (00,00)  
Bolsa Esquina  
F P F P  
00 00 00 00  
No iniciado



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

```
proceso recorrerAvenida(E numAv: numero)
comenzar
  Pos(numAv, 1)
  ...
fin
```

**areas**

```
ciudad: AreaC (1,1,100,100)
```

**robots**

```
robot tipo1
comenzar
  ...
fin
```

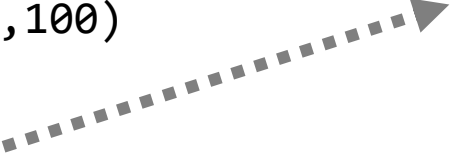
**variables**

```
r1: tipo1
```

**comenzar**

```
AsignarArea(R_info, ciudad)
Iniciar(R_info, 1,1)
```

**fin**



```
robot tipo1
variables
  ...
comenzar
  // Código del robot tipo1
fin
```



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

```
proceso recorrerAvenida(E numAv: numero)
comenzar
  Pos(numAv, 1)
  ...
fin
```

**areas**

```
ciudad: AreaC (1,1,100,100)
```

**robots**

```
robot tipo1
comenzar
  ...
fin
```

**variables**

```
r1: tipo1
r2: tipo1
```

**comenzar**

```
....
```

**fin**

Puedo definir más de un  
robot del mismo tipo. Ambos  
ejecutarán el mismo código.



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

...

**areas**

ciudad: AreaC (1,1,100,100)

**robots**

robot tipo1

comenzar

...

fin

robot tipo2

comenzar

...

fin

**variables**

r1: tipo1

r2: tipo2

**comenzar**

....

**fin**

Puedo definir más de un tipo robot (tipo1 y tipo2). Cada variable robot ejecutará el código correspondiente a su tipo

Obviamente el código a ejecutar en cada tipo de robot debe ser diferente.



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

**procesos**

...

**areas**

ciudadMitad: AreaC (1,1,50,50)

miLugar: AreaP (60,60,85,87)

**robots**

robot tipo1

comenzar

...

fin

**variables**

r1: tipo1

r2: tipo1

**comenzar**

AsignarArea(r1,ciudadMitad)

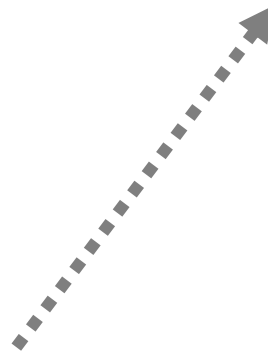
AsignarArea(r2,miLugar)

iniciar(r1, 5, 5)

iniciar(r2, 62, 63)

**fin**

Cada variable robot declarada  
debe tener una esquina de  
inicio y al menos un área  
asignada.





# PROGRAMA CONCURRENTE – AMBIENTE CMRE

**programa** ejemplo

...

**areas**

ciudadMitad: AreaC (1,1,10,10)

miLugar: AreaP(22,1,25,10)

**robots**

robot tipo1

comenzar

...

fin

**variables**

r1: tipo1

r2: tipo1

**comenzar**

AsignarArea(r1,ciudadMitad)

AsignarArea(r2,ciudadMitad)

AsignarArea(r2,miLugar)

iniciar(r1, 3, 3)

iniciar(r2, 23, 5)

**fin**

Un robot puede tener más  
de un área asignada



# PROGRAMA CONCURRENTE – AMBIENTE CMRE

programa uno  
areas

```
ciudadMitad: AreaC (1,1,10,10)  
miLugar: AreaP(22,1,25,10)
```

robots

```
robot robot1  
comenzar
```

```
    repetir 5  
        mover
```

```
fin
```

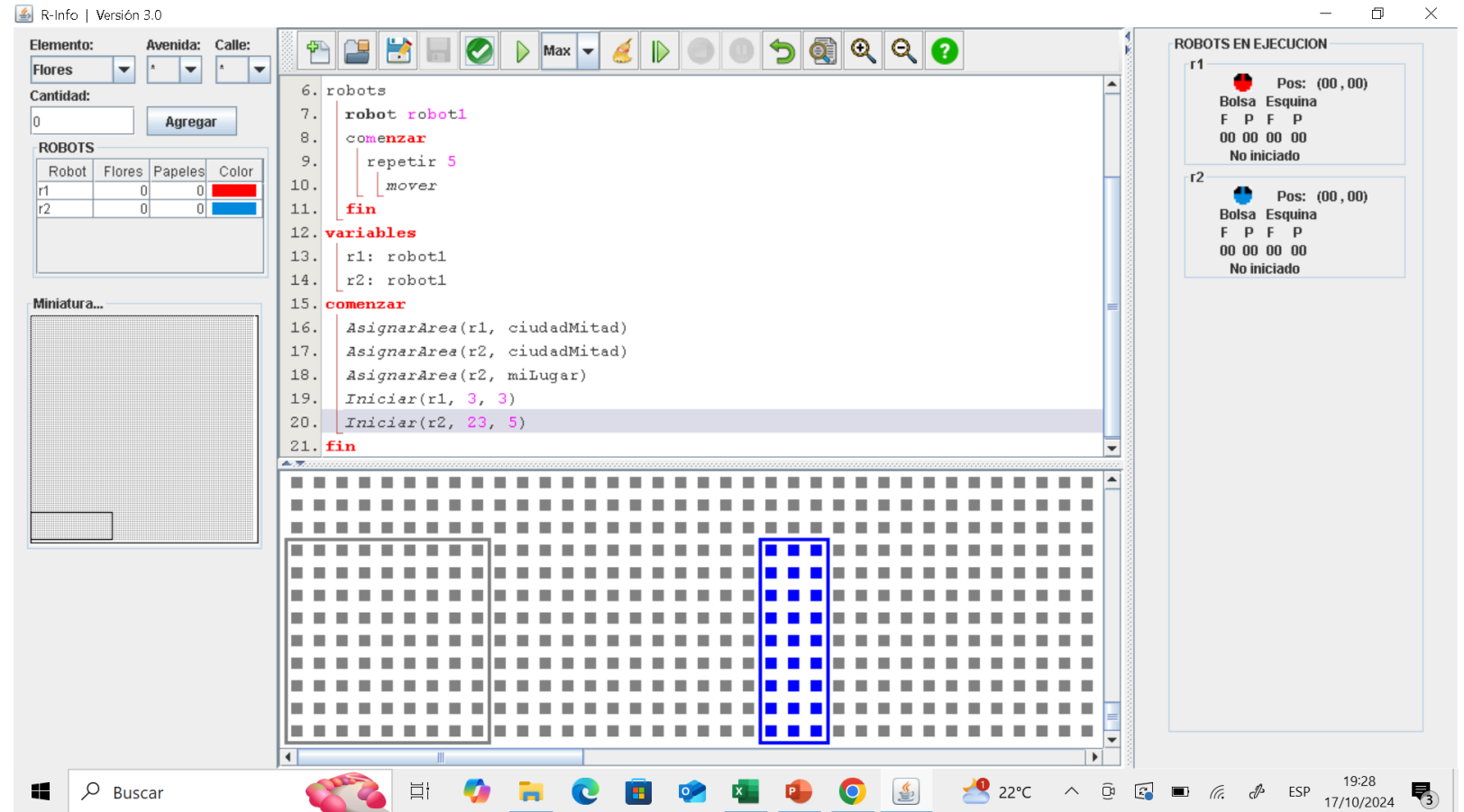
variables

```
r1: robot1  
r2: robot1
```

comenzar

```
AsignarArea(r1, ciudadMitad)  
AsignarArea(r2, ciudadMitad)  
AsignarArea(r2, miLugar)  
Iniciar(r1, 3, 3)  
Iniciar(r2, 23, 5)
```

```
fin
```





# EJERCICIOS PARA ANALIZAR

programa Ejercicio uno

**areas**

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

comenzar

mover

Pos (20,20)

fin

robot tipo2

comenzar

repetir 5

mover

Pos(20,20)

fin

**variables**

robot1: tipo1

robot2: tipo2

**comenzar**

...

**fin**

Dos robots **NO** pueden situarse  
en ningún momento del programa  
en la misma esquina al mismo  
tiempo.





# EJERCICIOS PARA ANALIZAR

programa Ejercicio uno

**areas**

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

comenzar

mover

Pos (20,20)

Pos(40,40)

fin

robot tipo2

comenzar

repetir 5

mover

Pos(20,20)

fin

**variables**

robot1: tipo1

robot2: tipo2

**comenzar**

...

**fin**

**NO** se puede suponer el orden en el cual se ejecutarán las instrucciones entre diferentes robots.



# EJERCICIOS PARA ANALIZAR

**Ejercicio:** Realice un programa donde un robot recorra el perímetro de un rectángulo de un tamaño 5 (alto) x 3 (ancho) juntando flores de todas las esquinas.

Al finalizar el robot debe informar las flores juntadas. Inicialmente el robot se encuentra en la esquina (2,2).

Debe modularizar el rectángulo. El rectángulo debe recibir alto y ancho y devolver las flores.

programa Rectangulos

**procesos**

proceso **juntar** (ES flor:numero)

**comenzar**

mientras (HayFlorEnLaEsquina)

tomarFlor

flor:= flor + 1

**fin**

proceso **rectangulo** (E alto:numero;

E ancho:numero;

ES flores:numero)

**comenzar**

flores:= 0

repetir 2

repetir alto

juntar(flores)

mover

derecha

repetir ancho

juntar(flores)

mover

derecha

**fin**



# EJERCICIOS PARA ANALIZAR

programa Rectangulos

**procesos**

proceso **juntar** (ES flor:numero)

**comenzar**

mientras (HayFlorEnLaEsquina)

tomarFlor

flor:= flor + 1

**fin**

proceso **rectangulo** (E alto:numero;E ancho:numero;ES flores:numero)

**comenzar**

flores:= 0

repetir 2

repetir alto

juntar(flores)

mover

derecha

repetir ancho

juntar(flores)

mover

derecha

**fin**

**Modifique el ejercicio  
para que exista otro  
robot que comience en  
(8,8)**

**areas**

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

variables

f:numero

**comenzar**

rectangulo (5,3,f)

Informar (f)

**fin**

**variables**

robot1: tipo1

**comenzar**

AsignarArea(robot1,ciudad)

Iniciar(robot1, 2, 2)

**fin**



# EJERCICIOS PARA ANALIZAR

programa Rectangulos

**procesos**

proceso **juntar** (ES flor:numero)

**comenzar**

mientras (HayFlorEnLaEsquina)

tomarFlor

flor:= flor + 1

**fin**

proceso **rectangulo** (E alto:numero;E ancho:numero;ES flores:numero)

**comenzar**

flores:= 0

repetir 2

repetir alto

juntar(flores)

mover

derecha

repetir ancho

juntar(flores)

mover

derecha

**fin**

**areas**

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

variables

f:numero

**comenzar**

rectangulo (5,3,f)

Informar (f)

**fin**

**variables**

robot1: tipo1

**robot2: tipo1**

**comenzar**

AsignarArea(robot1,ciudad)

**AsignarArea(robot2,ciudad)**

Iniciar(robot1, 2, 2)

**Iniciar(robot2, 8, 8)**

**fin**

Clase 1-2 – Módulo Concurrente



# Taller de Programación



# AGENDA

Ambiente CMRE

COMUNICACION - Pasaje de mensajes

Ejemplos



# EJERCICIO PARA ANALIZAR

Realizar un programa donde existen **dos robots**. El robot 1 debe realizar un rectángulo de 5 (alto) x 3 (ancho) juntando flores y el robot 2 un rectángulo de 8 (alto) x 2 (ancho) juntando flores. El robot 1 inicia en la esquina (1,1) y el robot 2 en la (10,1)

programa Rectangulos

**procesos**

proceso **juntar** (ES flor:numero)

**comenzar**

mientras (HayFlorEnLaEsquina)

tomarFlor

flor:= flor + 1

**fin**

proceso **rectangulo** (E alto:numero;  
E ancho:numero;  
ES flores:numero)

**comenzar**

...

**fin**

**areas**

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

variables

f:numero

**comenzar**

rectangulo (5,3,f)

Informar (f)

**fin**

robot **tipo2**

variables

f:numero

**comenzar**

rectangulo (8,2,f)

Informar (f)

**fin**

**variables**

robot1: tipo1

**robot2: tipo2**

**comenzar**

AsignarArea(robot1,ciudad)

Iniciar(robot1, 1, 2)

**AsignarArea(robot2,ciudad)**

**Iniciar(robot2, 10, 1)**

**fin**

SOLUCION 1



# EJERCICIO PARA ANALIZAR

Realizar un programa donde existen **dos robots**. El robot 1 debe realizar un rectángulo de 5 (alto) x 3 (ancho) juntando flores y el robot 2 un rectángulo de 8 (alto) x 2 (ancho) juntando flores. El robot 1 inicia en la esquina (1,1) y el robot 2 en la (10,1)

programa Rectangulos

procesos

proceso **juntar** (ES flor:numero)

comenzar

mientras (HayFlorEnLaEsquina)

tomarFlor

flor := flor + 1

fin

proceso **rectangulo** (E alto:numero;  
E ancho:numero;  
ES flores:numero)

comenzar

...

fin

areas

ciudad : AreaC(1,1,100,100)

**robots**

robot tipo1

variables

f:numero

comenzar

si (PosAv = 1)

rectangulo (5,3,f)

sino

rectangulo (8,2,f)

Informar (f)

fin

**variables**

robot1: tipo1

**robot2: tipo1**

comenzar

AsignarArea(robot1,ciudad)

Iniciar(robot1, 1, 2)

**AsignarArea(robot2,ciudad)**

**Iniciar(robot2, 10, 1)**

fin

Realizo un tipo de procesos robot que reciba el tamaño del rectángulo a realizar.

SOLUCION 2





# COMUNICACION - Mecanismos

1

- Pasaje de Mensajes

2

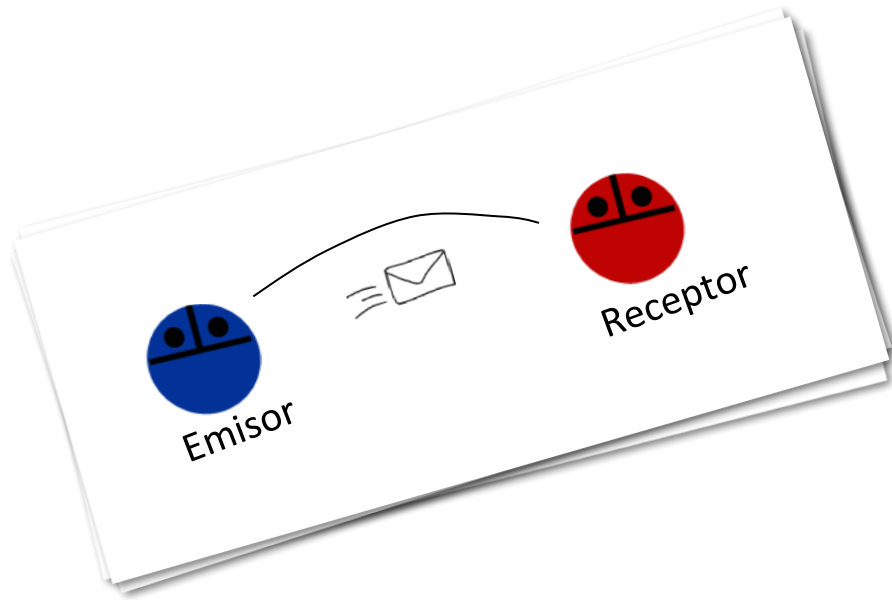
- Memoria Compartida



# COMUNICACIÓN – Pasaje de Mensajes

## OPERACIONES

Enviar Mensaje  
Recibir Mensaje



## FORMAS DE MENSAJES

Sincrónico  
Asincrónico



# COMUNICACIÓN – Pasaje de Mensajes

## PASAJE DE MENSAJES



### ENVÍO DE MENSAJES

Un proceso prepara un mensaje y selecciona uno o varios destinatarios para que lo reciban

### RECEPCION DE MENSAJES

Un proceso recibe un mensaje de un proceso determinado, o puede recibirlo de cualquiera de los procesos con los que interactúa

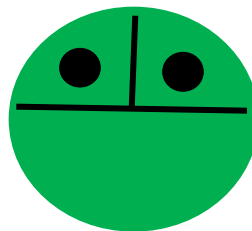
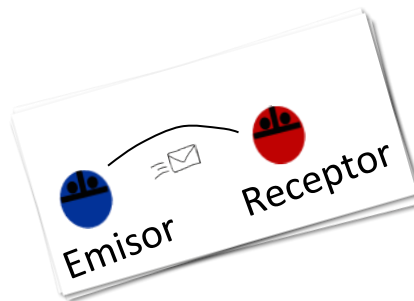


# COMUNICACIÓN – Pasaje de Mensajes

## PASAJE DE MENSAJES - Asincrónico



El proceso que envía/recibe el mensaje **NO** espera que se de la comunicación para continuar su ejecución.



Instrucción 1

Instrucción 2

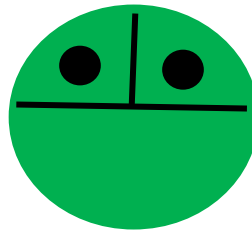
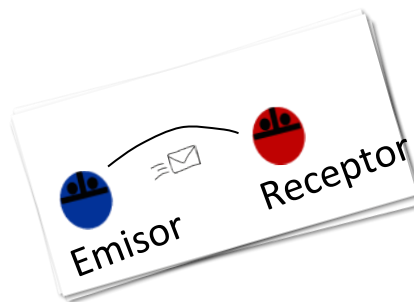
Sentencia de comunicación

Instrucción 3



# COMUNICACIÓN – Pasaje de Mensajes

## PASAJE DE MENSAJES - Sincrónico



El proceso que envía/recibe el mensaje **SI** espera que se de la comunicación para continuar su ejecución.

Instrucción 1

Instrucción 2

Sentencia de comunicación

Instrucción 3



**Cómo es en  
CMRE?**



# Taller de Programación



# AGENDA

Pasaje de mensajes - ENVIO

Ejemplos



# COMUNICACIÓN – Pasaje de Mensajes -CMRE

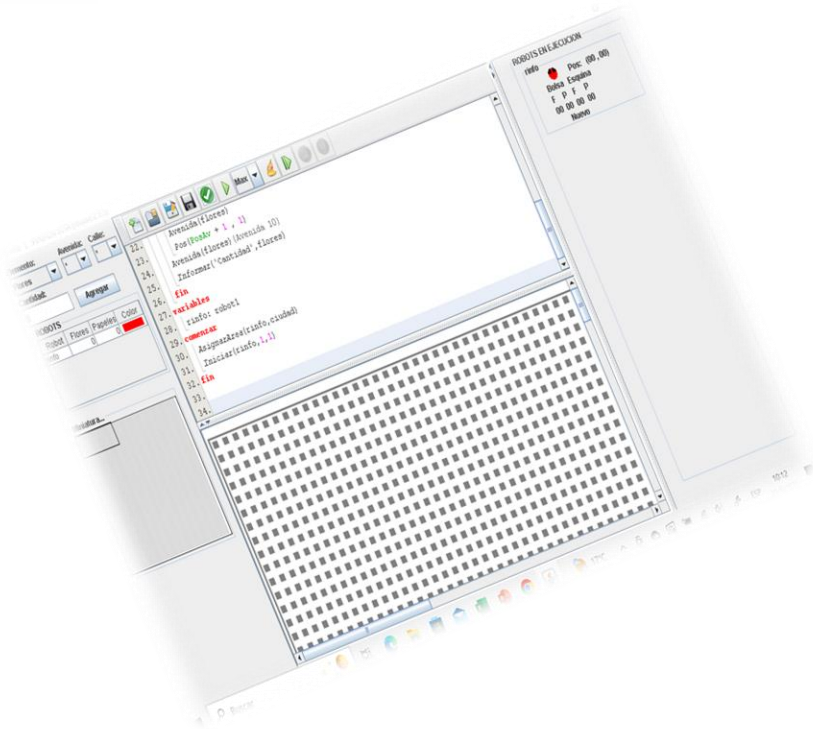
## ENVÍO DE MENSAJES

El envío de mensajes es **asincrónico**, es decir, el robot que envía el mensaje lo hace y sigue procesando sin esperar que el robot receptor lo reciba.

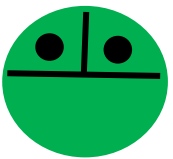
## RECEPCIÓN DE MENSAJES

La recepción de mensajes es **sincrónica**, es decir, el robot que espera un mensaje **NO** sigue procesando hasta que recibe el mensaje.

*Cómo es la sintaxis?*



Instrucción 1



Instrucción 2

Sentencia de comunicación

Instrucción 3



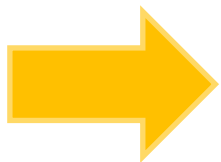


# COMUNICACIÓN – Pasaje de Mensajes - Envío

```
programa ejemploEnvio
areas
  area1: AreaPC(1,1,20,20)
robots
  robot tipo1
  comenzar
  ...
fin

robot tipo2
  comenzar
  ...
fin

variables
  robot1:tipo1
  robot2: tipo1
  robot3:tipo2
```



Supongamos que el **robot 3**, le quiere enviar un mensaje al **robot1** y otro al **robot2**

**EnviarMensaje(valor,variableRobot)**

**EnviarMensaje(variable,variableRobot)**

*Cómo  
escribimos el  
programa?*



# COMUNICACIÓN – Pasaje de Mensajes - Envío

programa envio

**areas**

area1: AreaPC(1,1,20,20)

**Procesos**

....

**robots**

robot tipo1

comenzar

...

fin

robot tipo2

**variables**

x:numero

comenzar

x:= 8

EnviarMensaje (5,robot1)

EnviarMensaje (x,robot2)

fin

**variables**

robot1: tipo1

robot2: tipo1

robot3: tipo2

**comenzar**

AsignarArea(robot1,ciudad)

Iniciar(robot1, 2, 2)

AsignarArea(robot2,ciudad)

Iniciar(robot2, 2, 2)

AsignarArea(robot3,ciudad)

Iniciar(robot3, 2, 2)

**fin**



# COMUNICACIÓN – Pasaje de Mensajes - Envío

## CONSIDERACIONES EN EL ENVIO DE MENSAJES

Puede enviarse un valor (entero o booleano) o una variable

EL envío debe incluir el nombre de una variable robot declarado (no el tipo)

Sólo se puede enviar un valor por mensaje de envío

*Cómo recibe  
un robot un  
mensaje?*



# Taller de Programación



# AGENDA

Pasaje de mensajes - RECEPCION

Ejemplos



# COMUNICACIÓN – Pasaje de Mensajes -CMRE

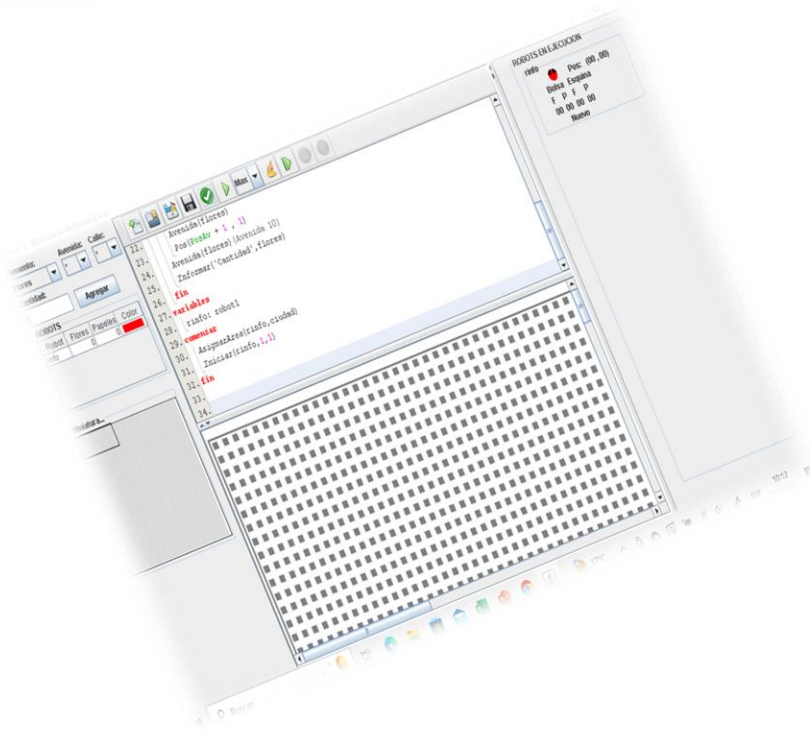
## ENVÍO DE MENSAJES

El envío de mensajes es **asincrónico**, es decir, el robot que envía el mensaje lo hace y sigue procesando sin esperar que el robot receptor lo reciba.

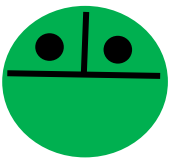
## RECEPCIÓN DE MENSAJES

La recepción de mensajes es **sincrónica**, es decir, el robot que espera un mensaje **NO** sigue procesando hasta que recibe el mensaje.

*Cómo es la sintaxis?*



Instrucción 1



Instrucción 2

Sentencia de comunicación

Instrucción 3



# COMUNICACIÓN – Pasaje de Mensajes - Recepción

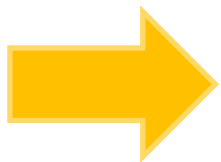
**RecibirMensaje(variable, variableRobot)**

*Cómo  
escribimos el  
programa?*

```
programa ejemploRecepcion
areas
  area1: AreaPC(1,1,20,20)
robots
  robot tipo1
  comenzar
  ...
fin

robot tipo2
  comenzar
  ...
fin
```

```
variables
robot1:tipo1
robot2: tipo1
robot3:tipo2
```



Supongamos que el **robot 1**, y el **robot2** quieren recibir un mensaje del **robot3**



# COMUNICACIÓN – Pasaje de Mensajes - Recepción

programa envio

**areas**

area1: AreaPC(1,1,20,20)

**Procesos**

....

**robots**

robot tipo1

variables

valor:numero

comenzar

**recibirMensaje (valor, robot3)**

fin

robot tipo2

variables

x:numero

comenzar

x:= 8

**EnviarMensaje (5,robot1)**

**EnviarMensaje (x,robot2)**

**fin**

**variables**

robot1: tipo1

robot2: tipo1

robot3: tipo2

**comenzar**

AsignarArea(robot1,ciudad)

Iniciar(robot1, 2, 2)

AsignarArea(robot2,ciudad)

Iniciar(robot2, 2, 2)

AsignarArea(robot3,ciudad)

Iniciar(robot3, 2, 2)

**fin**





# COMUNICACIÓN – Pasaje de Mensajes - Recepción

## CONSIDERACIONES EN LA RECEPCION DE MENSAJES

La recepción es SIEMPRE sobre una variable (entero o booleano)

La recepción SIEMPRE debe incluir el nombre de una variable robot declarado (no el tipo)

*Se puede recibir  
de cualquier  
robot?*



# COMUNICACIÓN – Pasaje de Mensajes – Recepción

Realizar un programa donde existen **dos robots juntadores**. El robot 1 debe realizar un rectángulo de 5 (alto) x 3 (ancho) juntando flores y el robot 2 un rectángulo de 8 (alto) x 2 (ancho) juntando flores. Luego un tercer robot **jefe** debe informar la cantidad de flores juntadas por cada robot. El tamaño de los rectángulos debe enviárselo el robot jefe a los robots juntadores.

## ROBOT juntadores

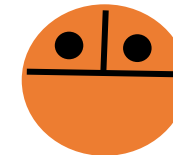


Recibe el alto y ancho del rectángulo

Invoca al proceso rectángulo (le envía el tamaño y le devuelve las flores juntadas)

Envía la cantidad juntada en el rectángulo al **jefe**

## ROBOT jefe



Envía el alto y ancho del rectángulo a cada robot juntador

Recibe la cantidad juntada por el robot 1

Informa la cantidad recibida

Recibe la cantidad juntada por el robot 2

Informa la cantidad recibida



# COMUNICACIÓN – Pasaje de Mensajes – Recepción

```
programa Recepcion
areas
  area1: AreaPC(1,1,20,20)
Procesos
  proceso rectángulo (E alto:numero;
                      E: ancho:numero;
                      ES flores: numero);

  comenzar
  ....
  fin
robots
  robot juntador
  variables
    altura,ancho,f:numero
  comenzar
    RecibirMensaje(altura,robot3)
    RecibirMensaje(ancho,robot3)
    rectángulo (altura,ancho,f)
    EnviarMensaje (f,robot3)
  fin
```

```
robot jefe
variables
  f:numero
comenzar
  EnviarMensaje (5,robot1)
  EnviarMensaje (3,robot1)
  EnviarMensaje (8,robot2)
  EnviarMensaje (2,robot2)
  RecibirMensaje(f,robot1)
  Informar (f)
  RecibirMensaje(f,robot2)
  Informar (f)
fin
```

```
variables
  robot1: juntador
  robot2: juntador
  robot3: jefe
comenzar
  ....
fin
```

¿Qué ocurre si el  
robot 2 termina de  
juntar sus flores  
primero?



# COMUNICACIÓN – Pasaje de Mensajes – Recepción

```
...  
robot jefe  
variables  
  f:numero  
comenzar  
  ....  
  EnviarMensaje (5,robot1)  
  EnviarMensaje (3,robot1)  
  EnviarMensaje (8,robot2)  
  EnviarMensaje (2,robot2)  
  RecibirMensaje(f,*)  
  Informar (f)  
  RecibirMensaje(f,*)  
  Informar (f)  
fin  
variables  
  robot1:juntadores  
  robot2: juntadores  
  robot3: jefe  
comenzar  
  ....  
fin
```

**RecibirMensaje(variable,\*)**

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



# EJERCICIOS PARA ANALIZAR

Realizar un programa donde existen **dos robots juntadores**. El robot 1 debe realizar un rectángulo de 5 (alto) x 3 (ancho) juntando flores y el robot 2 un rectángulo de 8 (alto) x 2 (ancho) juntando flores. Luego un tercer robot **jefe** debe informar la cantidad de flores **TOTALES** juntadas por ambos robots. El tamaño de los rectángulos debe enviárselo el robot jefe a los robots juntadores.

## ROBOT juntadores

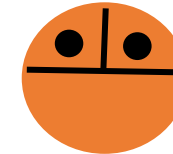


Recibe el alto y ancho del rectángulo

Invoca al proceso rectángulo (le envía el tamaño y le devuelve las flores juntadas)

Envía la cantidad juntada en el rectángulo al **jefe**

## ROBOT jefe



Envía el alto y ancho del rectángulo a cada robot juntador

Repite 2 veces

Recibe la cantidad juntada por algún robot

Suma la cantidad al total

Informa la cantidad recibida



# EJERCICIOS PARA ANALIZAR

```
programa RecepcionIndistinta
```

```
areas
```

```
    area1: AreaPC(1,1,20,20)
```

```
Procesos
```

```
    proceso rectángulo (E alto:numero;  
                        E: ancho:numero;  
                        ES flores: numero);
```

```
        comenzar
```

```
        ....
```

```
    fin
```

```
robots
```

```
    robot juntador
```

```
variables
```

```
    altura,ancho,f:numero
```

```
comenzar
```

```
    RecicibirMensaje(altura,robor3)
```

```
    RecibirMensaje(ancho,robot3)
```

```
    rectángulo (altura,ancho,f)
```

```
    EnviarMensaje (f,robot3)
```

```
fin
```

```
robot jefe
```

```
variables
```

```
    total,f:numero
```

```
comenzar
```

```
    EnviarMensaje (5,robot1)
```

```
    EnviarMensaje (3,robot1)
```

```
    EnviarMensaje (8,robot2)
```

```
    EnviarMensaje (2,robot2)
```

```
    Repetir 2
```

```
        RecibirMensaje(f,*)
```

```
        total:= total + f
```

```
    Informar (total)
```

```
fin
```

```
variables
```

```
    robot1:juntador
```

```
    robot2: juntador
```

```
    robot3: jefe
```

```
comenzar
```

```
    ....
```

```
fin
```

*Y si el jefe quiere  
informar que robot  
juntó mas?*



# EJERCICIOS PARA ANALIZAR

```
programa RecepcionIndistinta
areas
  area1: AreaPC(1,1,20,20)
Procesos
  proceso rectángulo (E alto:numero;
                      E: ancho:numero;
                      ES flores: numero);

  comenzar
  ....
  fin
robots
  robot juntador
  variables
    altura,ancho,f:numero
  comenzar
    RecicibirMensaje(altura,robor3)
    RecibirMensaje(ancho,robot3)
    rectángulo (altura,ancho,f)
    EnviarMensaje (f,robot3)
  fin
```

```
robot jefe
variables
  max,robotMax,f:numero
comenzar
  EnviarMensaje (5,robot1)
  EnviarMensaje (3,robot1)
  EnviarMensaje (8,robot2)
  EnviarMensaje (2,robot2)
  Repetir 2
    RecibirMensaje(f,*)
    si (f > max) entonces
      max:= f
      robotMax:= *
  Informar (robotMax)
  fin
variables
  robot1:juntador
  robot2: juntador
  robot3: jefe
comenzar
  ....
fin
```





# EJERCICIOS PARA ANALIZAR

Realizar un programa donde existen **dos robots juntadores**. El robot 1 debe realizar un rectángulo de 5 (alto) x 3 (ancho) juntando flores y el robot 2 un rectángulo de 8 (alto) x 2 (ancho) juntando flores. Luego un tercer robot **jefe** debe informar que robot juntador juntó la mayor cantidad de flores. El tamaño de los rectángulos debe enviárselo el robot jefe a los robots juntadores.

## ROBOT juntadores

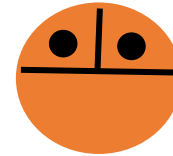


Recibe el alto y ancho del rectángulo

Invoca al proceso rectángulo (le envía el tamaño y le devuelve las flores juntadas)

Envía la cantidad juntada en el rectángulo al **jefe** y **ademas** quien es

## ROBOT jefe



Envía el alto y ancho del rectángulo a cada robot juntador

Repite 2 veces

**Recibe la cantidad juntada por algún robot**

Si la (cantidad es máxima)

Actualiza el máximo y el número de robot máximo

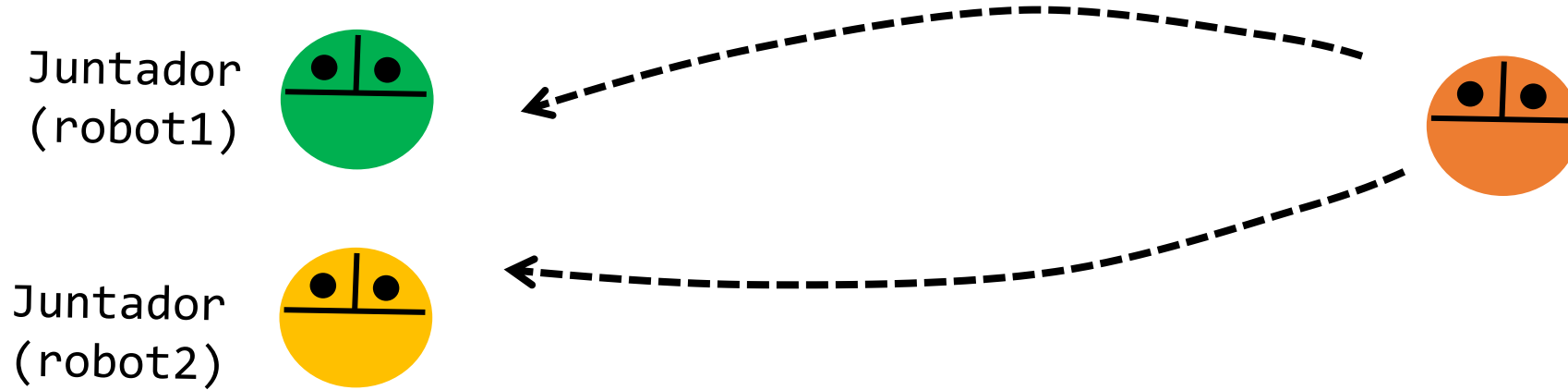
Informa el robot máximo

Los robots  
juntadores  
**NO conocen**  
su  
**identificación**





# COMUNICACIÓN – Pasaje de Mensajes – Paso 1



Juntador (robot1)

```
RecibirMensaje(quienSoy, robot3)
```

Juntador (robot2)

```
RecibirMensaje(quienSoy, robot3)
```

Jefe (robot3)

```
EnviarMensaje(1, robot1)  
EnviarMensaje(2, robot2)
```

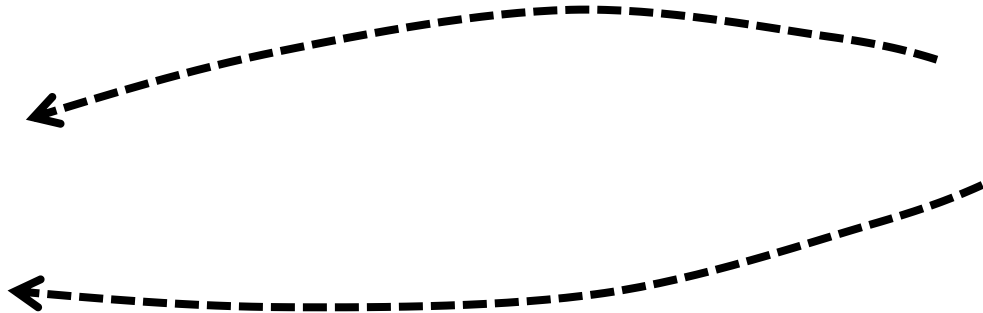
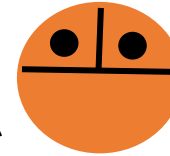


# COMUNICACIÓN – Pasaje de Mensajes – Paso 1

Juntador  
(robot1)



Juntador  
(robot2)



Juntador  
(robot 1)



```
RecibirMensaje(quienSoy, robot3)  
RecibirMensaje(alto, robot3)  
RecibirMensaje(ancho, robot3)  
  
... .
```

Juntador  
(robot2)



```
RecibirMensaje(quienSoy, robot3)  
RecibirMensaje(alto, robot3)  
RecibirMensaje(ancho, robot3)  
  
... .
```

Jefe  
(robot3)

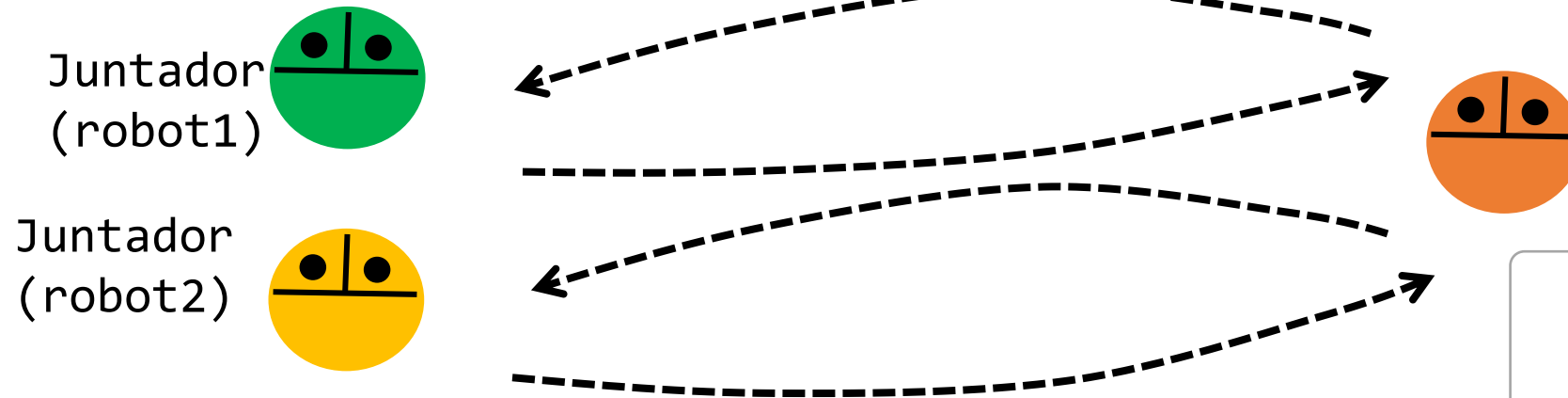


```
EnviarMensaje(1, robot1)  
EnviarMensaje(5, robot1)  
EnviarMensaje(3, robot1)
```

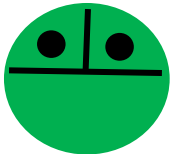
```
EnviarMensaje(2, robot2)  
EnviarMensaje(8, robot2)  
EnviarMensaje(2, robot2)  
  
... .
```



# COMUNICACIÓN – Pasaje de Mensajes – Paso 2



Juntador  
(robot1)



RecibirMensaje(**quienSoy**, robot3)  
....

EnviarMensaje(**quienSoy**, robot3)  
EnviarMensaje(**flores**, robot3)

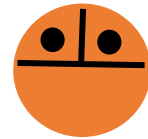
Juntador  
(robot2)



RecibirMensaje(**quienSoy**, robot3)  
....

EnviarMensaje(**quienSoy**, robot3)  
EnviarMensaje(**flores**, robot3)

Jefe  
(robot3)



EnviarMensaje(1, robot1)  
EnviarMensaje(5, robot1)  
EnviarMensaje(3, robot1)

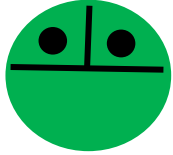
EnviarMensaje(2, robot2)  
EnviarMensaje(8, robot2)  
EnviarMensaje(2, robot2)

RecibirMensaje (quien,\*)  
Si (quien = 1)  
    RecibirMensaje (quien,quien)



# COMUNICACIÓN – Pasaje de Mensajes – TODO

Juntador  
(robot1)



```
RecibirMensaje(quienSoy, robot3)  
...
```

```
EnviarMensaje(quienSoy, robot3)  
EnviarMensaje(flores, robot3)
```

Juntador  
(robot2)



```
RecibirMensaje(quienSoy, robot3)  
...
```

```
EnviarMensaje(quienSoy, robot3)  
EnviarMensaje(flores, robot3)
```

Jefe  
(robot3)



```
EnviarMensaje(1, robot1)  
EnviarMensaje(2, robot2)  
...
```

```
RecibirMensaje(numRobot, *)
```

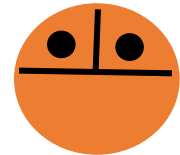
```
Si numRobot = 2
```

```
    RecibirMensaje(valor, robot2)
```

```
Sino
```

```
    RecibirMensaje(valor, robot1)
```

Jefe  
(robot3)



```
EnviarMensaje(1, robot1)  
EnviarMensaje(2, robot2)
```

```
RecibirMensaje(numRobot, *)
```

```
Si * = 2
```

```
    RecibirMensaje(valor, robot2)
```

Jefe  
(robot3)



```
EnviarMensaje(1, robot1)  
EnviarMensaje(2, robot2)
```

```
RecibirMensaje(numRobot, *)
```

```
RecibirMensaje(valor, *)
```



# EJERCICIOS PARA ANALIZAR

```
programa RecepcionMaximo
areas
  area1: AreaPC(1,1,20,20)
Procesos
  proceso rectángulo (E alto:numero;
                      E: ancho:numero;
                      ES flores: numero);

  comenzar
  ....
  fin
robots
  robot juntador
  variables
    altura,ancho,f,quien: numero
  comenzar
    RecicibirMensaje(quien,robor3)
    RecicibirMensaje(altura,robor3)
    RecibirMensaje(ancho,robot3)
    rectángulo (altura,ancho,f)
    EnviarMensaje (quien,robot3)
    EnviarMensaje (f,robot3)
  fin
```

```
robot jefe
variables
  max,rmax,f:numero
comenzar
  max:=0
  EnviarMensaje (1,robot1)
  EnviarMensaje (5,robot1)
  EnviarMensaje (3,robot1)

  EnviarMensaje (2,robot2)
  EnviarMensaje (8,robot2)
  EnviarMensaje (2,robot2)
  Repetir 2
    RecibirMensaje(quien,*)
    Si(quien = 1)
      RecibirMensaje(f,robot1)
    Sino
      RecibirMensaje(f,robot2)
    Si (f> = max)
      max:= f
      rmax:= quien
  Informar (rmax)
fin
```



# Taller de Programación



# AGENDA



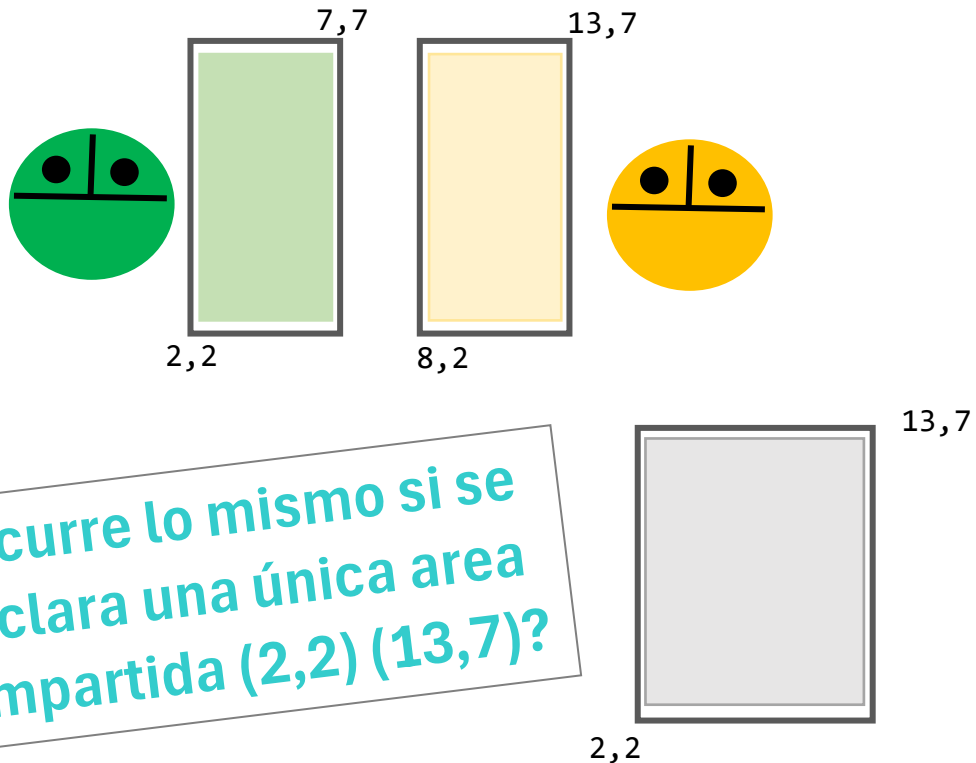
## Mecanismos de Comunicación – MEMORIA COMPARTIDA

### Ejemplos



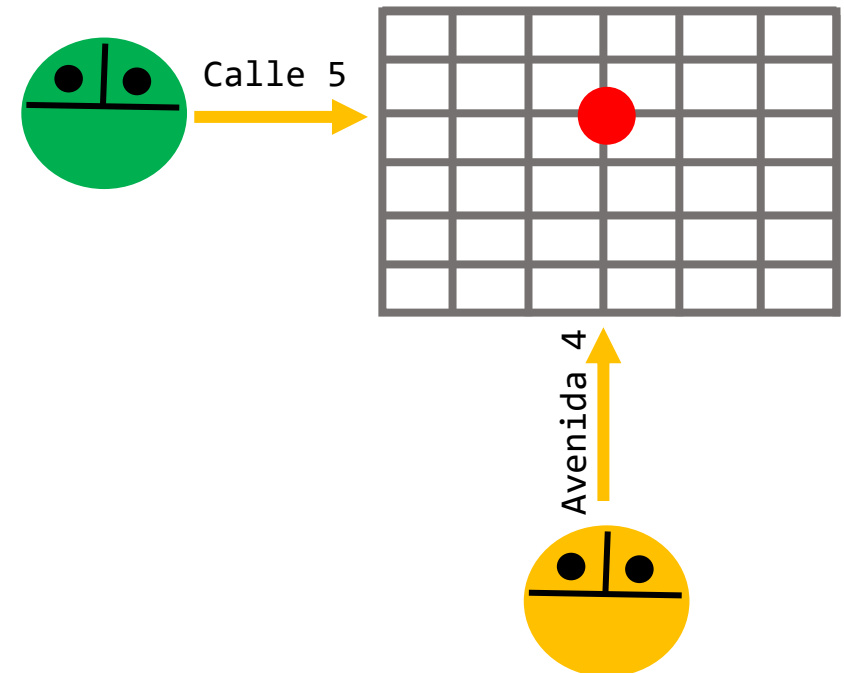
# Mecanismos de Comunicación – MEMORIA COMPARTIDA

Realizar un programa donde existen **dos robots**. El robot 1 trabaja en su área privada delimitada por las esquinas (2,2) y (7,7) contando esquinas vacías y el robot 1 trabaja en su área privada delimitada por las esquinas (8,2) y (13,7) contando esquinas vacías.



¿Ocurre lo mismo si se declara una única área compartida (2,2) (13,7)?

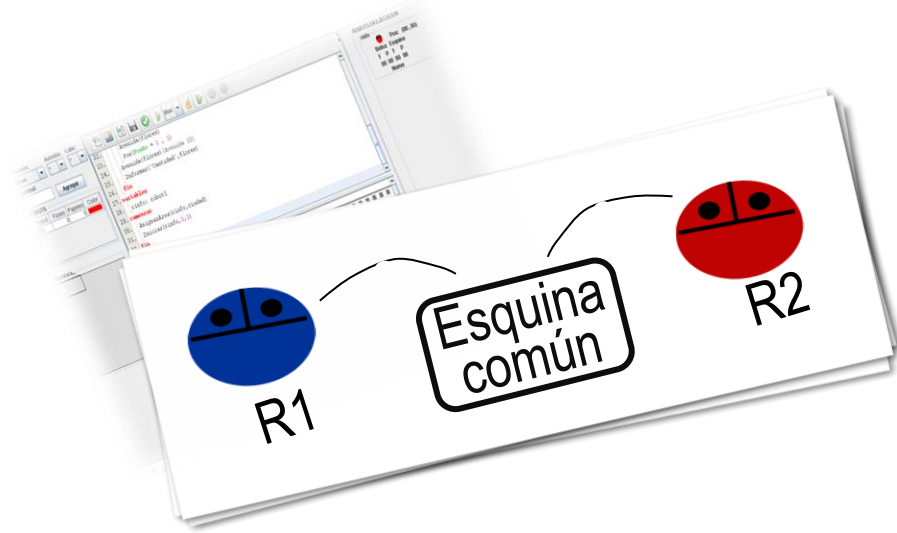
Realizar un programa donde existen **dos robots**. El robot 1 llamado avenida debe recorrer la avenida 10 y juntar las flores que encuentre. El robot 2 llamado calle debe recorrer la calle 5 juntando los papeles que encuentre.







# COMUNICACIÓN – Memoria Compartida



## BLOQUEAR RECURSO

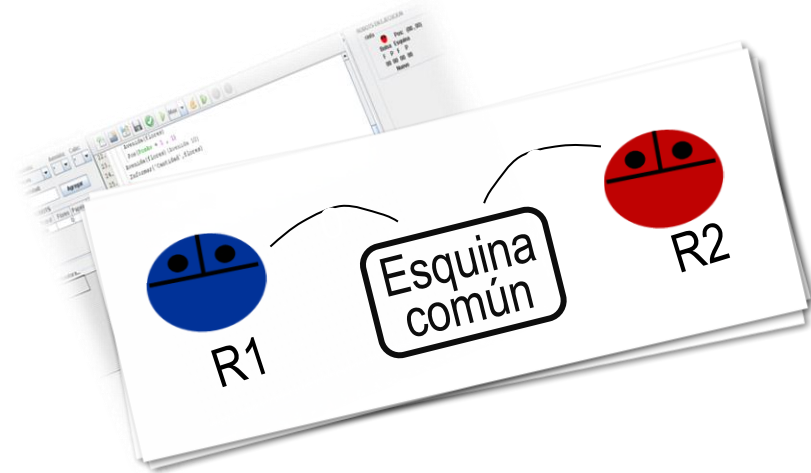
Dado un recurso compartido (por 2 o más procesos) que está **DISPONIBLE** se bloquea ese recurso para que otro proceso no pueda accederlo-

## LIBERAR UN RECURSO

Dado un recurso compartido (por 2 o más procesos) **BLOQUEADO** el programador libera dicho recurso para que cualquier proceso pueda bloquearlo.



# COMUNICACIÓN – Memoria Compartida



¿Cómo es en  
CRME?

## BLOQUEAR RECURSO - CONSIDERACIONES

- Puede realizarlo el programador o el Sistema Operativo
- Sólo se bloquea un recurso libre. Si el recurso ya está bloqueado no se debe intentarse hacerlo.
- Hay que bloquear un recurso cuando puede ser accedido por dos o más procesos de un programa.

## LIBERAR UN RECURSO - CONSIDERACIONES

- Puede realizarlo el programador o el Sistema Operativo
- Sólo se libera un recurso ocupado. Si el recurso no está bloqueado no se debe intentarse hacerlo.
- Hay que liberar un recurso cuando puede ser accedido por dos o más procesos de un programa.



# COMUNICACIÓN – Memoria Compartida

```
programa ejemploBloqueo  
areas
```

```
    area: AreaC(1,1,20,20)
```

```
robots
```

```
    robot tipo1
```

```
    comenzar
```

```
    ...
```

```
fin
```

```
variables
```

```
    robot1:tipo1
```

```
    robot2: tipo1
```

```
comenzar
```

```
...
```

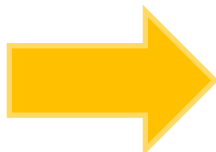
```
fin
```

```
BloquearEsquina(avenida,calle)
```

```
BloquearEsquina(2,8)
```

```
BloquearEsquina(posAv+1,posCa)
```

```
BloquearEsquina(ave,ca)
```



Supongamos que el **robot 1**,  
y el **robot 2** deben acceder  
a la esquina (5,10) en  
algún momento

*Cómo escribo  
el programa?*



# COMUNICACIÓN – Memoria Compartida

```
programa ejemploBloqueo
```

```
areas
```

```
    area: AreaC(1,1,20,20)
```

```
robots
```

```
    robot tipo1
```

```
    comenzar
```

```
        ...
```

```
        BloquearEsquina(5,10)
```

```
        ...
```

```
    fin
```

```
variables
```

```
    robot1: tipo1
```

```
    robot2: tipo1
```

```
comenzar
```

```
    ...
```

```
fin
```

- Si la esquina (5,10) está **desbloqueada**, entonces se marca como bloqueada y el robot continua ejecutando su código.
- Si la esquina (5,10) está **bloqueada**, entonces el robot “queda esperando” hasta que la esquina se libere y pueda ejecutar la instrucción de bloqueo.

**Cómo se libera  
un esquina?**



# COMUNICACIÓN – Memoria Compartida

```
programa ejemploBloqueo
areas
  area1: AreaC(1,1,20,20)
```

```
robots
  robot tipo1
  comenzar
```

```
  ...
```

```
fin
```

```
variables
  robot1:tipo1
  robot2: tipo1
```

```
comenzar
```

```
...
```

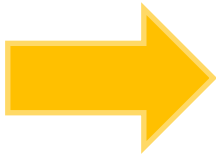
```
fin
```

```
LiberarEsquina(avenida,calle)
```

```
  LiberarEsquina(2,8)
```

```
  LiberarEsquina(posAv+1,posCa)
```

```
  LiberarEsquina(ave,ca)
```



Supongamos que el **robot 1**,  
y el **robot 2** deben liberar  
la esquina (5,10) en algún  
momento (la habían  
bloqueado previamente)

*Cómo escribo  
el programa?*



# COMUNICACIÓN – Memoria Compartida

programa ejemploBloqueo

areas

ciudad: AreaPC(1,1,20,20)

robots

robot tipo1

comenzar

...

BloquearEsquina(5,10)

...

LiberarEsquina(5,10)

...

fin

variables

robot1: tipo1

robot2: tipo1

comenzar

...

fin

Cuando el robot bloquea la esquina (5,10), entonces ningún robot (inclusive él) podrá bloquearla hasta que la misma sea desbloqueada.



# COMUNICACIÓN – MEMORIA COMPARTIDA

## CONSIDERACIONES EN EL BLOQUEO y DESBLOQUEO DE ESQUINAS

Una esquina DEBE ser bloqueada si y sólo si existe la posibilidad que dos o más robots podrían posicionarse en la misma al mismo momento.

Las esquinas deben permanecer bloqueadas el menor tiempo posible que garantice que el programa funcione.

Una esquina bloqueada siempre debe ser desbloqueada en algún momento del programa. Al finalizar el programa NO pueden quedar esquinas bloqueadas.

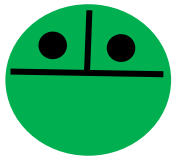
Una esquina DEBE ser bloqueada antes que el robot se posicione en ella.



# EJERCICIO PARA ANALIZAR

Supongamos que tenemos un programa en el cual están declarados dos tipos robots y dos variables robot, una correspondiente a cada tipo. El robot 1 (de tipo 1) inicia su trabajo en la esquina (1,1) camina 5 cuadras y luego se posiciona en la esquina (10,10). El robot2 (de tipo 2) inicia su trabajo en la esquina (2,2) junta las flores de la esquina y luego se posiciona en la esquina (10,10). Ambos robots después de su trabajo vuelven a su esquina original.

## ROBOT tipo1

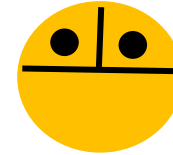


Repetir 5  
mover

Posicionarse en la esquina (10,10)

Posicionarse en la esquina (1,1)

## ROBOT tipo2



Mientras (hay flor en la esquina)  
tomarFlor

Posicionarse en la esquina (10,10)

Posicionarse en la esquina (2,2)





# EJERCICIO PARA ANALIZAR

```
programa ejemploBloqueo
```

```
areas
```

```
    ciudad: AreaC(1,1,20,20)
```

```
robots
```

```
    robot tipo1
```

```
        comenzar
```

```
            repetir 5
```

```
                mover
```

```
                Pos (10,10)
```

```
                Pos (1,1)
```

```
        fin
```

```
    robot tipo2
```

```
        comenzar
```

```
            mientras (HayFlorEnLaEsquina)
```

```
                tomarFlor
```

```
                Pos (10,10)
```

```
                Pos (2,2)
```

```
        fin
```

```
variables
```

```
    robot1: tipo1
```

```
    robot2: tipo2
```

```
comenzar
```

```
    AsignarArea(robot1, ciudad)
```

```
    Iniciar(robot1, 1, 1)
```

```
    AsignarArea(robot2, ciudad)
```

```
    Iniciar(robot2, 2, 2)
```

```
fin
```

Los robots podrían chocarse al querer posicionarse en la esquina (10,10)





# EJERCICIO PARA ANALIZAR

```
programa ejemploBloqueo
areas
  ciudad: AreaC(1,1,20,20)
robots
  robot tipo1
  comenzar
    repetir 5
      mover
      BloquearEsquina (10,10)
      Pos (10,10)
      LiberarrEsquina (10,10)
      Pos (1,1)
  fin
  robot tipo2
  comenzar
    mientras (HayFlorEnLaEsquina)
      tomarFlor
      Pos (10,10)
      Pos (2,2)
```

fin

## variables

```
robot1: tipo1
robot2: tipo2
```

## comenzar

```
AsignarArea(robot1, ciudad)
Iniciar(robot1, 1, 1)
AsignarArea(robot2, ciudad)
Iniciar(robot2, 2, 2)
```

## fin

Los robots podrían chocarse al querer posicionarse en la esquina (10,10)





# EJERCICIO PARA ANALIZAR

```
programa ejemploBloqueo
areas
  ciudad: AreaC(1,1,20,20)
robots
  robot tipo1
  comenzar
    repetir 5
      mover
      BloquearEsquina (10,10)
      Pos (10,10)
      LiberarrEsquina (10,10)
      Pos (1,1)
  fin

  robot tipo2
  comenzar
    mientras (HayFlorEnLaEsquina)
      tomarFlor
      BloquearEsquina (10,10)
      Pos (10,10)
      LiberarrEsquina (10,10)
      Pos (2,2)
  fin
```

## variables

```
robot1:tipo1
robot2: tipo2
```

## comenzar

```
AsignarArea(robot1,ciudad)
Iniciar(robot1, 1, 1)
AsignarArea(robot2,ciudad)
Iniciar(robot2, 2, 2)
```

## fin

**Los robots podrían chocarse al querer posicionarse en la esquina (10,10)**





# EJERCICIO PARA ANALIZAR

```
programa ejemploBloqueo
areas
  ciudad: AreaC(1,1,20,20)
robots
  robot tipo1
  comenzar
    BloquearEsquina (10,10)
    repetir 5
      mover
    Pos (10,10)
    Pos (1,1)
    LiberarrEsquina (10,10)
  fin
  robot tipo2
  comenzar
    BloquearEsquina (10,10)
    mientras (HayFlorEnLaEsquina)
      tomarFlor
    Pos (10,10)
    Pos (2,2)
    LiberarrEsquina (10,10)
  fin
```

## variables

```
robot1:tipo1
robot2: tipo2
```

## comenzar

```
AsignarArea(robot1,ciudad)
Iniciar(robot1, 1, 1)
AsignarArea(robot2,ciudad)
Iniciar(robot2, 2, 2)
```

## fin



**Los robots deben  
bloquear los recursos el  
mínimo tiempo necesario**



# COMUNICACIÓN – Memoria Compartida

**ROBOT robot1**



Realizar código seguro

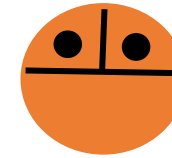
Bloquear la esquina compartida

Realizar la tarea en esa esquina

Moverme a una esquina segura

Liberar la esquina compartida

**ROBOT robot2**



Realizar código seguro

Bloquear la esquina compartida

Realizar la tarea en esa esquina

Moverme a una esquina segura

Liberar la esquina compartida





# EJERCICIO PARA ANALIZAR

```
programa ejemploBloqueo
areas
  ciudad: AreaC(1,1,20,20)

robots
  robot tipo1
  comenzar
    repetir 5
      mover
      BloquearEsquina (10,10)
      Pos (10,10)
      Pos (1,1)
      LiberarrEsquina (10,10)
  fin
```

```
robot tipo2
  comenzar
    mientras (HayFlorEnLaEsquina)
      tomarFlor
      BloquearEsquina (10,10)
      Pos (10,10)
      Pos (2,2)
      LiberarrEsquina (10,10)
  fin
variables
  robot1: tipo1
  robot2: tipo2

comenzar
  AsignarArea(robot1, ciudad)
  Iniciar(robot1, 1, 1)
  AsignarArea(robot2, ciudad)
  Iniciar(robot2, 2, 2)
fin
```



# Taller de Programación



# AGENDA



Algoritmos y arquitecturas concurrentes

Ejemplos del mundo real





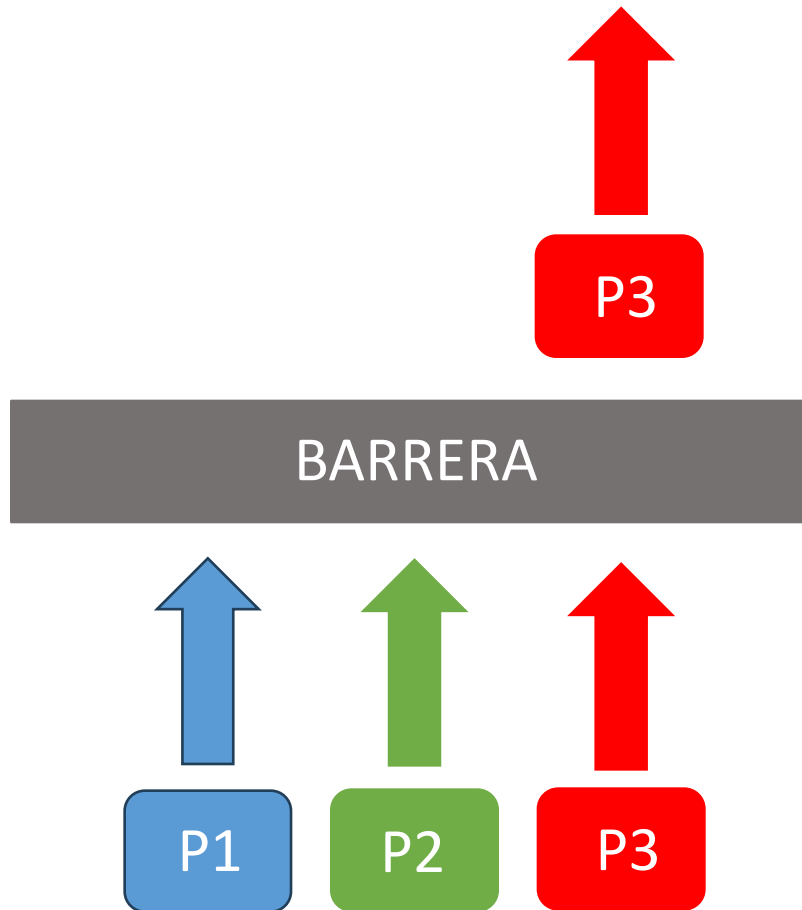
# Qué conceptos vimos hasta ahora



**PROBLEMAS  
REALES**



# Tipos de problemas reales – Sincronización por barrera



## CARACTERISTICAS

Múltiples procesos se ejecutan concurrentemente, hasta que llegan a un punto especial, llamado **barrera**.

Los procesos que llegan a la barrera deben detenerse y esperar que **todos** los procesos.

Cuando **todos** los procesos alcanzan la barrera, podrán retomar su actividad (hasta finalizar o hasta alcanzar la próxima barrera).

Para esto los procesos deben avisar que llegaron.

## PROBLEMAS REALES

La vacunación por COVID.  
Un partido de paddle.



# Tipos de problemas reales – Passing the baton

## CARACTERISTICAS

Múltiples procesos se ejecutan en concurrente.

Sólo un proceso a la vez, el que posee el testigo (baton), se mantiene activo.

Cuando el proceso activo completa su tarea, entrega el baton a otro proceso. El proceso que entregó el baton queda a la espera hasta recibirlo nuevamente.

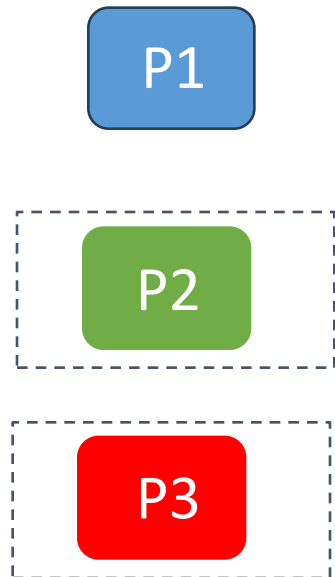
El proceso que recibió el baton completa su ejecución. Al completar su tarea, pasará el baton a otro proceso.

Para esto los procesos deben tener una forma de comunicarse con el otro proceso.

## PROBLEMAS REALES

La producción de un producto.

Una carrera de postas.





# Tipos de problemas reales – Productor/consumidor

**Productor**



**Consumidor**



## CARACTERISTICAS

Existen dos tipos de procesos:

Productores: trabajan para generar algún recurso y almacenarlo en un espacio compartido.

Consumidores: utilizan los recursos generados por los productores para realizar su trabajo.

Para esto los procesos deben coordinar donde almacenan los datos los productores, cuando saben los consumidores que hay datos.

## PROBLEMAS REALES

Corrección de parciales

Cualquier sistema de producción



# Tipos de problemas reales – Servidor/Cliente

## CARACTERISTICAS

Los procesos se agrupan en dos categorías:

**Servidores:** permanecen inactivos hasta que un cliente les solicita algo. Cuando reciben una solicitud, realizan su tarea, entregan el resultado y vuelven a “dormir” hasta que otro cliente los despierte

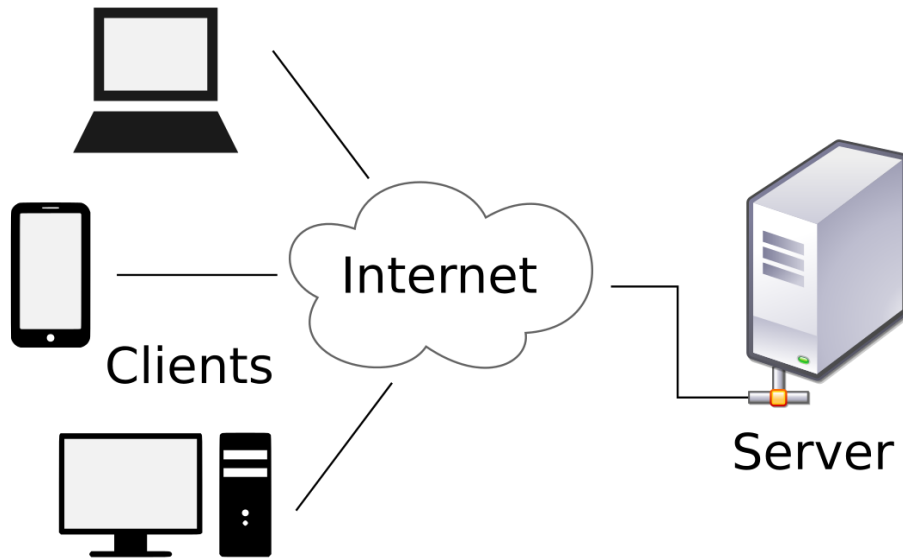
**Clients:** realizan su trabajo de manera independiente, hasta que requieren algo de un servidor. Entonces realizan una solicitud a un proceso servidor, y esperan hasta que recibir la respuesta. Cuando esto sucede, el cliente continúa su trabajo.

Para esto los procesos cliente deben realizar sus pedidos y el servidor debe administrar como los atiende.

## PROBLEMAS REALES

Cualquier navegador

Cualquier empresa





# Tipos de problemas reales – Master/Slave

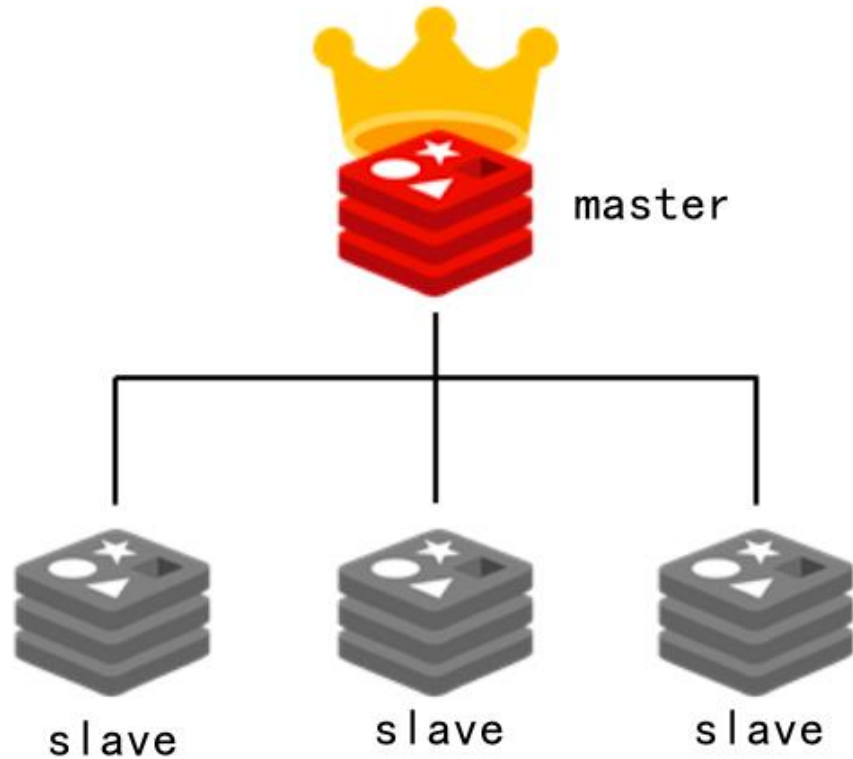
## CARACTERISTICAS

Los procesos se agrupan en dos categorías:

Maestro: deriva tareas a otros procesos (trabajadores)

Esclavos: realizan la tarea solicitada y envían el resultado al jefe, quedando a la espera de la siguiente tarea

Para esto el proceso jefe determina cuantos trabajadores necesita, cómo les reparte la tarea, cómo recibe los resultados.



## PROBLEMAS REALES

Buscar valores en un arreglo

Remisería