




Posibles soluciones a los ejercicios del parcial

Estas son posibles soluciones, no significa que sean la única forma, traté de hacerlas simples y siguiendo las ideas que hemos visto en las teorías y/o explicaciones prácticas para no confundir.



Resolver con **ADA** el siguiente problema. Se debe simular un juego en el que participan 30 jugadores que forman 5 grupos de 6 personas. Al llegar cada jugador debe buscar las instrucciones y el grupo al que pertenece en un cofre de cemento privado para cada uno; para esto deben usar un único martillo gigante de a uno a la vez y de acuerdo al orden de llegada. Luego se debe juntar con el resto de los integrantes de su grupo y los 6 juntos realizan las acciones que indican sus instrucciones. Cuando un grupo termina su juego le avisa a un Coordinador que le indica en qué orden terminó el grupo. **Nota:** maximizar la concurrencia; suponer que existe una función *Jugar()* que simula que los 6 integrantes de un grupo están jugando juntos; suponga que existe una función *Romper(grupo)* que simula cuando un jugador está rompiendo su cofre con el martillo y le retorna el grupo al que pertenece.

Procedure ParcialADA is

```
task type jugador;  
task type grupo is  
  entry llegada;  
  entry resultado(pos: OUT integer);  
end grupo;  
task martillo is  
  entry usarMartillo (grupo: OUT integer);  
end martillo;  
task coordinador is  
  entry posicion (pos: OUT integer);  
end coordinador;
```

```
arrJugadores: array (1..30) of jugador;  
arrGrupos: array (1..5) of grupo;
```


```
task body grupo is  
  posicionFinal: integer;  
begin  
  for i in 1..6 loop  
    accept llegada;  
  end loop;  
  Jugar();  
  coordinador.posicion(posicionFinal);  
  for i in 1..6 loop  
    accept resultado(pos: OUT integer) do  
      pos := posicionFinal;  
    end resultado;  
  end loop;  
end grupo;
```

```
task body persona is  
  idG, posFin: integer;  
begin  
  martillo.usarMartillo(idG);  
  arrGrupos(idG).llegada;  
  arrGrupos(idG).resultado(posFin);  
end persona;
```

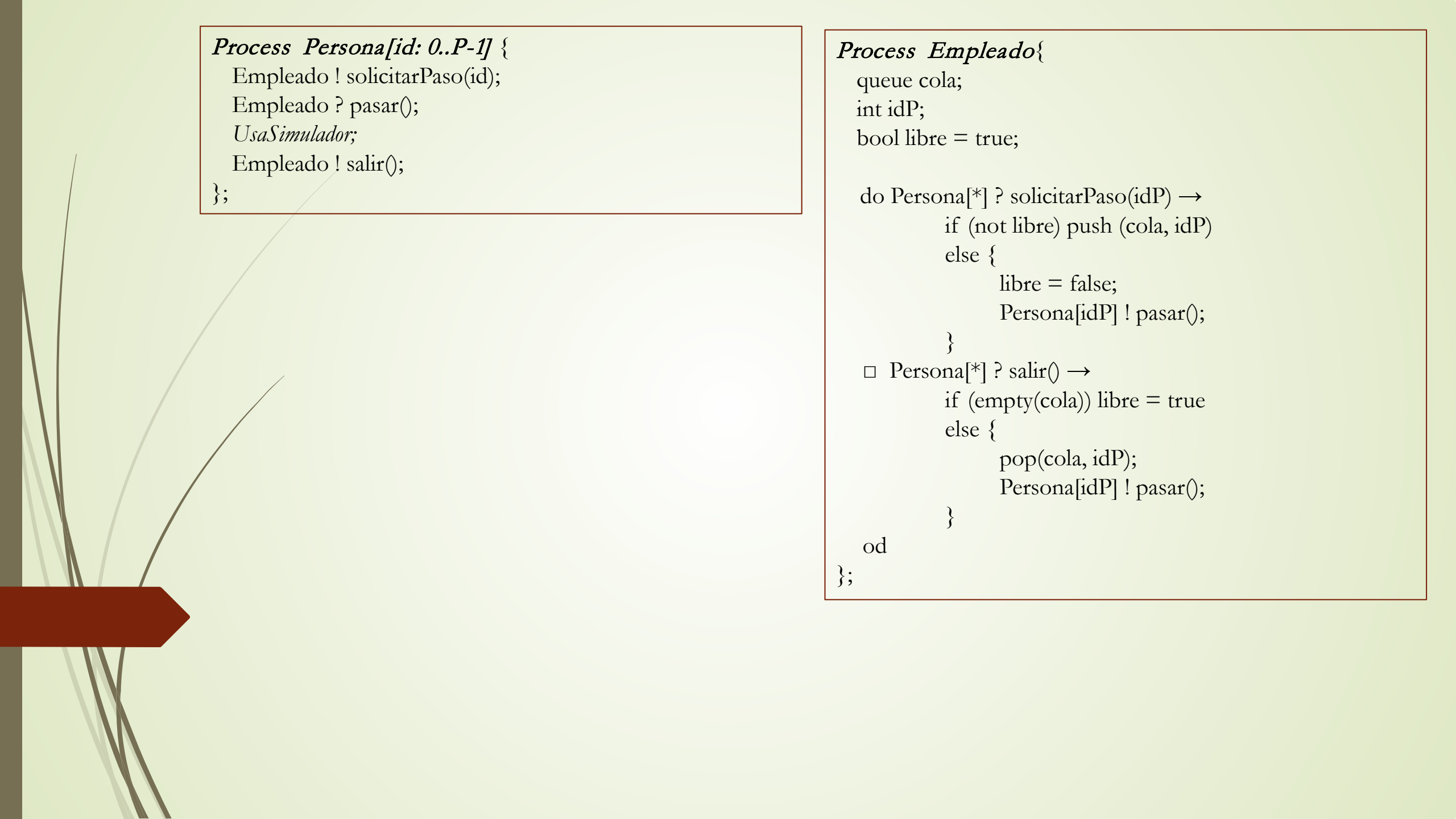
```
task body martillo is  
begin  
  for i in 1..30 loop  
    accept usarMartillo(grupo: OUT integer) do  
      grupo := Romper();  
    end usarMartillo;  
  end loop;  
end martillo;
```

```
task body coordinador is  
  posActual: integer := 1;  
begin  
  for i in 1..5 loop  
    accept posicion (pos: OUT integer) do  
      pos := posActual;  
      posActual := posActual + 1;  
    end posicion;  
  end loop;  
end coordinador;
```

```
Begin  
  null;  
End ParcialADA;
```



Resolver con **PMS (Pasaje de Mensajes SINCRÓNICOS)** el siguiente problema. En una exposición aeronáutica hay un simulador de vuelo (que debe ser usado con exclusión mutua) y un empleado encargado de administrar el uso del mismo. Hay P personas que esperan a que el empleado lo deje acceder al simulador, lo usa por un rato y se retira. El empleado deja usar el simulador a las personas respetando el orden de llegada. **Nota:** cada persona usa sólo una vez el simulador.



```
Process Persona[id: 0..P-1] {  
  Empleado ! solicitarPaso(id);  
  Empleado ? pasar();  
  UsaSimulador;  
  Empleado ! salir();  
};
```

```
Process Empleado{  
  queue cola;  
  int idP;  
  bool libre = true;  
  
  do Persona[*] ? solicitarPaso(idP) →  
    if (not libre) push (cola, idP)  
    else {  
      libre = false;  
      Persona[idP] ! pasar();  
    }  
  □ Persona[*] ? salir() →  
    if (empty(cola)) libre = true  
    else {  
      pop(cola, idP);  
      Persona[idP] ! pasar();  
    }  
  od  
};
```