

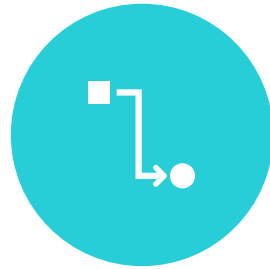
# PROGRAMACIÓN AVANZADA

Ejercicios sobre ciclos paralelos en OpenMP

# CONTENIDO



DEPENDENCIAS



CONDICIONES  
DE BERNSTEIN



EJEMPLOS



EJERCICIOS

# DEPENDENCIAS

Por lo general, existen dos formas de **paralelismo**: de tareas y de datos.

En el paralelismo de **datos** los pasos del algoritmo están estrechamente ligados con los datos.

En estos casos, el uso de **ciclos** suele ser muy efectivo, como en el trabajo con matrices.

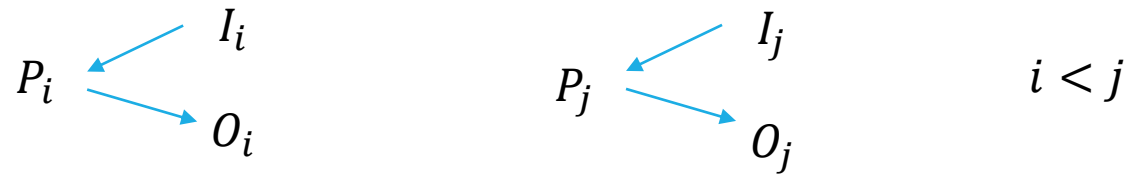
Sin embargo, cuando **implementamos** ciclos, a menudo los valores de una iteración dependen de cálculos que se realizan en otras iteraciones.

Si existe esta **dependencia**, entonces las iteraciones no se pueden paralelizar, por lo que es necesario modificar la implementación.



# CONDICIONES DE BERNSTEIN

Si dos segmentos de **programa**  $P_i, P_j$  tienen variables de entrada y salida (I/O) de la siguiente manera:



Entonces  $P_i, P_j$  se consideran **independientes** si se cumple:

- a)  $I_j \cap O_i = \emptyset \Rightarrow$  “el segundo no necesita el resultado del primero”
- b)  $I_i \cap O_j = \emptyset \Rightarrow$  “el primero no necesita ninguna variable generada por el segundo”
- c)  $O_i \cap O_j = \emptyset \Rightarrow$  “ambos copian sus resultados en variables diferentes”

# EJEMPLO EN CICLOS

En el siguiente **fragmento** de código aparece un ciclo **for** de **n-2** iteraciones.

```
for (i = 1; i < n-1; i++) {  
    x[i+1] = x[i] + x[i-1];  
}
```


Para analizar si hay dependencia entre las **iteraciones**, establecemos cada una como un **segmento** de programa. Seleccionemos entonces los segmentos P1 y P2.

P1 => Entradas: x[0], x[1]. Salidas: x[2]

P2 => Entradas: x[1], x[2]. Salidas: x[3]

Por lo tanto:  $I_2 \cap O_1 \neq \emptyset$ , y podemos decir que las iteraciones son **dependientes**.

Según las condiciones de Bernstein, indica el tipo de dependencias de datos existente entre las distintas iteraciones en los ciclos que se presentan a continuación.



Justifica si se puede eliminar o no esa dependencia de datos, eliminándola en caso de que sea posible.



Además implemente ambos ciclos en paralelo para comparar las soluciones y validar si es correcto paralelizar o no.

## EJERCICIOS:

```
for (i=0; i < n; i++) {  
    a[i] = a[i] + y[i];  
    x = a[i];  
}
```

## EJERCICIO 1

---

```
for (i = n-2; i >= 0; i--) {
```

---

```
    x[i] = x[i] + y[i+1];
```

---

```
    y[i] = y[i] + z[i];
```

---

```
}
```

---

## EJERCICIO 2



**MUCHAS GRACIAS**