Programación

Clase 5

Alternativa Condicional

Universidad Nacional de Quilmes

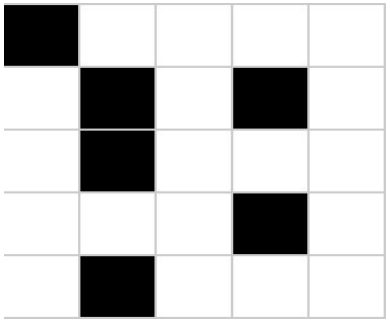
Ejercitamos un poco

Ejercicio

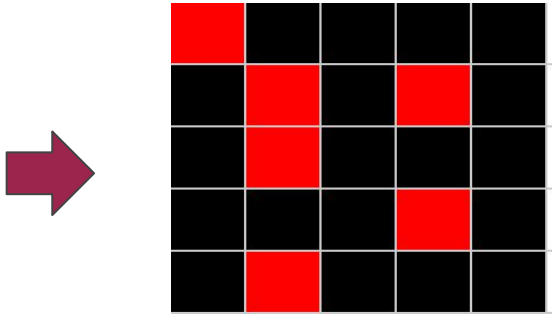
Queremos pintar todas las celdas un tablero de 5x5 de color negro, salvo aquellas que ya estén pintadas de negro, en cuyo caso, queremos pintarlas de rojo.

El cabezal comienza en la esquina inferior izquierda.

Ojo, el tablero mostrado es un ejemplo, ***no sabemos a priori cuales son las celdas pintadas de negro en el tablero inicial***.



Ejemplo de tablero inicial



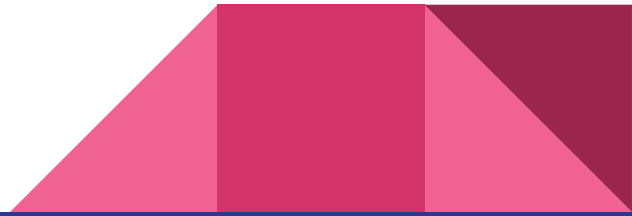
Ejemplo de tablero final

Alternativa Condicional

Alternativa Condicional

La **alternativa condicional** permite elegir en base a una **condición** si un **bloque de código** debe **ejecutarse o no**.

Nos permite elegir dos caminos distintos de acción.



Alternativa Condicional: Sintaxis

En nuestra sintaxis la alternativa condicional se escribe así:

si (*CONDICIÓN)* entonces

BLOQUE SI LA CONDICIÓN SE CUMPLE

sino

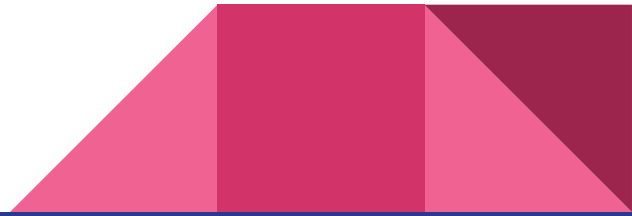
BLOQUE SI LA CONDICIÓN NO SE CUMPLE

Alternativa Condicional: Ejemplo

```
procedimiento PintarCelda() {
    si <La celda está pintada de negro> entonces {
        PintarRojo
    } sino {
        PintarNegro
    }
}
```

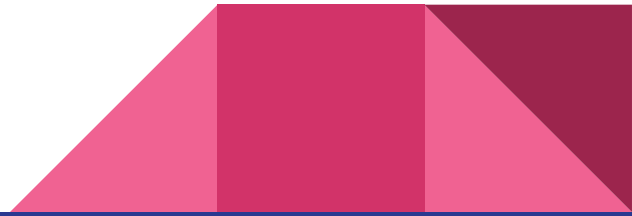
Alternativa Condicional

El código ejecuta el bloque de arriba o el de abajo dependiendo de en qué **estado** esté la **celda actual**, si pintada de negro o no.



Alternativa Condicional: Ejemplo

```
procedimiento PintarColumna () {
    repetir 4 veces {
        PintarCelda()
        MoverArrriba
    }
    PintarCelda()
}
```

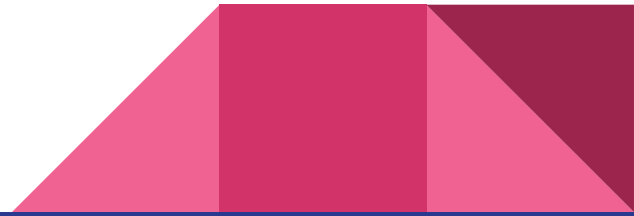


Condiciones

Condiciones

Las **condiciones** pueden tomar valores o bien **verdaderos**, o bien **falsos**.

Son los que nos van a permitir discernir entre dos alternativas.



Nuevas primitivas

Para poder consultar sobre el estado del tablero necesitamos **agregar** a nuestro lenguaje una serie de **nuevas primitivas** que representan esas condiciones.

Vamos a querer consultar por ejemplo, si una celda está pintada de algún color, o si está vacía.

Nuevas primitivas

Adicionamos las siguientes instrucciones:

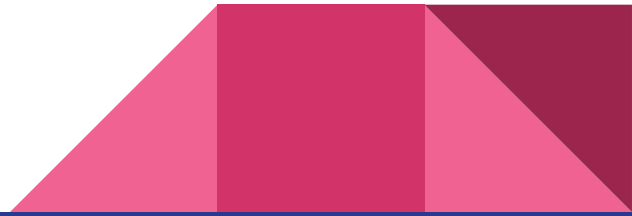
- **estaPintadaDeNegro?** Denota verdadero si la celda está pintada de Negro, falso en otro caso
- e**staPintadaDeRojo?** Denota verdadero si la celda está pintada de Rojo, falso en otro caso
- **estaPintadaDeVerde?** Denota verdadero si la celda está pintada de Verde, falso en otro caso
- **estaVacia**? Denota verdadero si la celda está vacía, falso en otro caso

Ejemplos

```
procedimiento PintarCelda() {
    si (estaPintadaDeNegro?) entonces {
        PintarRojo

    } sino {
        PintarNegro
    }
}
```

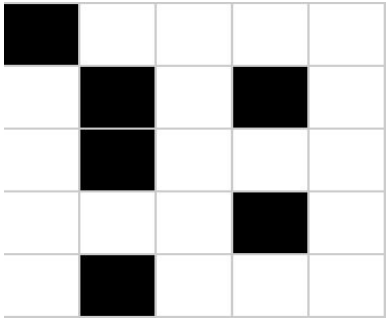
Alternativa Condicional
Forma acotada



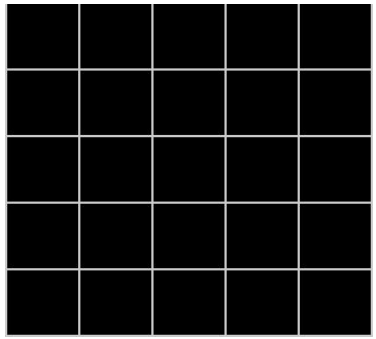
Ejercicio

Queremos pintar todas las celdas un tablero de 5x5 de color negro, salvo aquellas que ya estén pintadas de negro, en cuyo caso, queremos simplemente ignorarlas.

El cabezal comienza en la esquina inferior izquierda.



Ejemplo de tablero inicial



Ejemplo de tablero final

Alternativa Condicional

```
procedimiento PintarCelda() {
    si (estaVacia?) entonces {
        PintarNegro
    } sino {

    }
}
```

Alternativa Condicional: Forma Acotada

Si en el caso del “sino” el bloque de código va a quedar **vacío**, debemos **ignorar** completamente **toda la parte de “sino”**, simplificando el código y la lectura.

El código anterior es entonces equivalente al siguiente.

```
procedimiento PintarCelda () {
    si (estaVacia?)  entonces {
        PintarNegro
    }
}
```

Alternativa Condicional: Forma Acotada

En el caso de que el bloque que nos quede vacío sea el del “entonces”, no se puede acotar directamente.

Lo que puede hacerse es cambiar la condición, **colocando un NO** antes de toda la condición para cambiar en caso.

Por ejemplo, los dos códigos siguientes son equivalentes.

Alternativa Condicional

```
procedimiento PintarSiEsNegro () {
si (estaPintadaDeNegro?)  entonces {

    } sino {
        PintarNegro
    }
}
```

```
procedimiento PintarSiEsNegro () {
si  (¬ estaPintadaDeNegro?) entonces {
        PintarNegro
    }
}
```

Así debemos escribirlo.

¿Resulta familiar ?

“ ¬ “

Se trata del operador lógico de negación, que utilizamos en la unidad anterior.

Repasemos:

p	¬p
V	F
F	V

Conjunción

Repasemos un poco:

p	**q**	**p ∧ q**
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción

Repasemos un poco:

p	**q**	**p ∨ q**
V	V	V
V	F	V
F	V	V
F	F	F

Se podría utilizar en Qdraw?

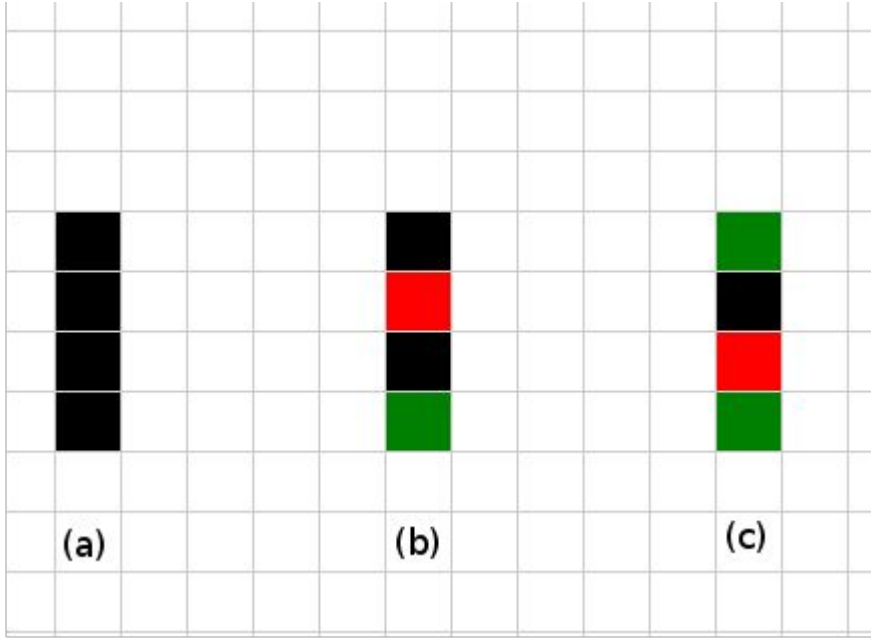
```
procedimiento PintarCeldaDeVerdeSoloSiEsNegraORoja() {
        si  ( estaPintadaDeNegro?   ∨   estaPintadaDeRojo?   )  entonces {
                        PintarVerde

        }
  }

procedimiento RecorrerLinea() {
    repetir 4 veces  {
            PintarCeldaDeVerdeSoloSiEsNegraORoja()
    }
}
```

Analicemos mejor

Evaluar el procedimiento RecorrerLinea() en los siguientes tableros:



Resumiendo

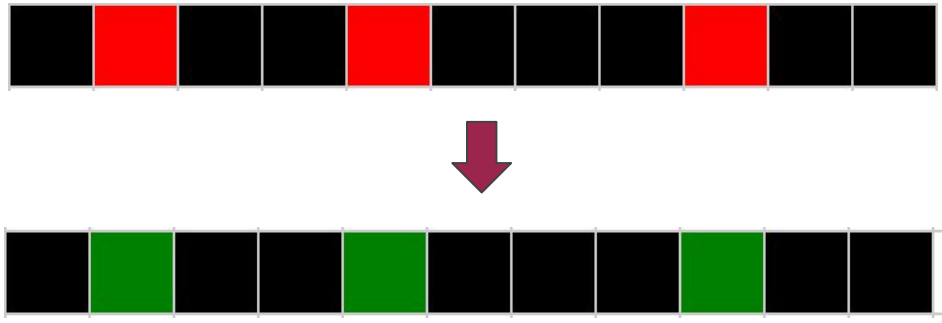
- La alternativa condicional permite elegir en base a una condición.
- Las condiciones pueden tomar valores o bien verdaderos, o bien falsos.
- En las condiciones podemos utilizar operadores lógicos como negación, conjunción y disyunción.
- La alternativa condicional no se puede anidar, al igual que la repetición simple

Ejercitamos

Ejercicio

Dado un tablero que consiste en una fila de celdas pintadas de negro con algunas de ellas pintadas de rojo. Pinte solamente las celdas rojas de color verde.

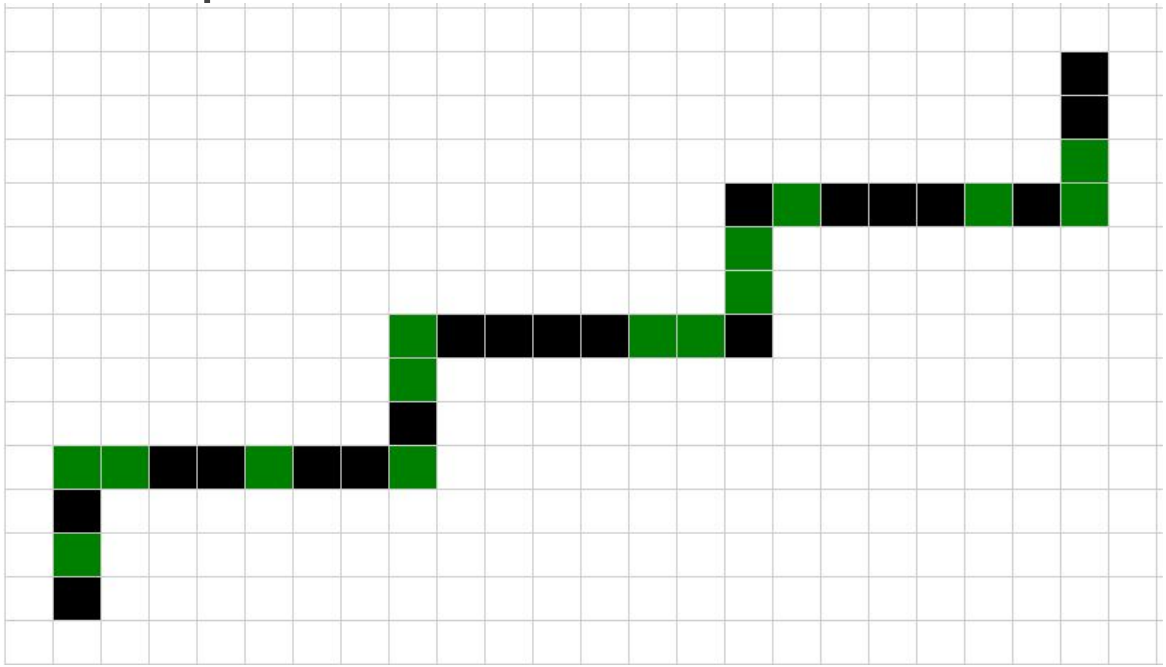
Ejemplo de tablero de inicial.



Ejemplo de tablero de final.

Ejercicio

Recorrer el camino, pintando de color negro todas las celdas verdes que encuentre en el camino.



Programación

Clase 5

Universidad Nacional de Quilmes