Programación

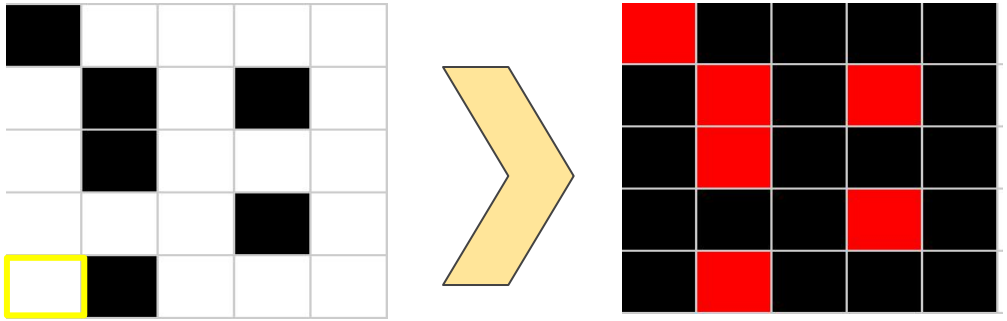
Alternativa Condicional

Universidad Nacional de Quilmes

Pensemos el siguiente problema

A partir de un tablero de un juego de 5x5, se solicitar pintar de color rojo sólo las celdas negras, y pintar de negro aquellas que se encuentren vacías. El cabezal debe iniciar en la esquina inferior izquierda del tablero.

Tener en cuenta que ***no conocemos a priori cuáles son las celdas pintadas de negro*** *en el tablero del juego*. Veamos un ejemplo de una posible situación:



Estado inicial del tablero

Estado final del tablero

Alternativa Condicional

Definición

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 en base a una condición. La alternativa condicional se agrega al lenguaje QDraw como una nueva estructura de control.

Sintaxis

En QDraw, la sintaxis de una alternativa condicional se escribe así:

si (*CONDICIÓN*) **entonces{**

BLOQUE SI LA CONDICIÓN SE CUMPLE (evalúa verdadero)

} sino{

BLOQUE SI LA CONDICIÓN NO SE CUMPLE (evalúa falso)

}

Ejemplo simple

```
procedimiento Marcar() {
/**/
    si (estaPintadaDeNegro?) entonces {
        PintarRojo
    } sino {
        Limpiar
    }
}
```

Primitiva condicional

Se ejecuta el bloque de código superior *(si)* o el inferior *(sino)* dependiendo del **estado** de la **celda actual**. En este caso, si está pintada de negro, entonces la pinta de rojo y en caso contrario, la limpia.

Un ejemplo más completo

Poco a poco vamos enriqueciendo nuestro lenguaje con un repertorio mayor de instrucciones y estructuras de control. De esta manera, podemos sumar herramientas que al combinarse, nos permitan resolver problemas más complejos.

Veamos aquí un ejemplo de cómo podemos utilizar el procedimiento anterior, como parte de un procedimiento más general y ya conocido, que evalúe todas las celdas de una columna.

```
procedimiento MarcarColumna () {
/*…*/
    repetir 4 veces {
        Marcar()
        MoverArriba
    }
    Marcar()
}
```

Condiciones

Definición

Las **condiciones** sólo pueden tomar valores booleanos, o binarios, es decir **verdadero** o bien **falso**.

Estos valores son los que nos van a permitir discernir entre dos alternativas.

Nuevas primitivas al repertorio del lenguaje

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

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

Nuevas primitivas - Sintaxis

Se agregan las siguientes condiciones primitivas:

- **estaPintadaDeNegro?**: Denota Verdadero si la celda está pintada de color **Negro**, Falso en cualquier otro caso
- **estaPintadaDeRojo?:** Denota Verdadero si la celda está pintada de color **Rojo**, Falso en cualquier otro caso
- **estaPintadaDeVerde?**: Denota Verdadero si la celda está pintada de color **Verde**, Falso en cualquier otro caso
- **estaVacia**?: Denota Verdadero si la celda está vacía, Falso en cualquier otro caso

NOTA: QDraw no hace BOOM si se pinta sobre una celda ya pintada.

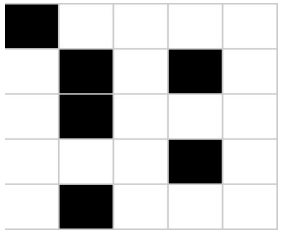
Ejemplo 2

```
procedimiento CambiarColor() {
/**/
    si (estaPintadaDeRojo?) entonces {
        PintarVerde
    }sino {
        PintarNegro
    }
}
```

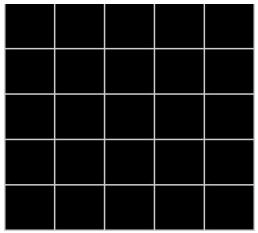
Alternativa Condicional
Forma acotada

Analicemos el siguiente código

Se necesita marcar (pintar) de color negro todas las celdas del cuadrado, ignorando aquellas que ya se encuentren de color negro. El cabezal inicia en la esquina inferior izquierda.



Estado inicial del cuadrado



Estado final del cuadrado

Como parte del programa contamos con el siguiente procedimiento:

```
procedimiento MarcarVacia() {
/**/
     si (estaVacia?) entonces {
          PintarNegro
     } sino {

     }
}
```

¿Notan algo raro?

Forma Acotada - Caso I

¡Exacto! No tiene sentido que haya un bloque de código vacío.

Por lo tanto, cuando necesitemos accionar mediante sólo una alternativa, el bloque de código que no se utiliza lo debemos **eliminar por completo**, simplificando así el código y su lectura. En este caso, la alternativa eliminada es aquella cuya condición **evalúa falso** (*bloque del **"sino"***). Quedando como resultado el siguiente código:

```
procedimiento MarcarVacia() {
/**/
     si (estaVacia?) entonces {
          PintarNegro
     }
}
```

Forma Acotada - Caso II

Ahora ¿qué pasa si la alternativa en desuso es aquella cuya condición **evalúa verdadero** (*bloque del **"si"***)?

```
procedimiento MarcarNegro() {
/**/
      si (estaPintadaDeNegro?) entonces {

      } sino {
            PintarNegro
      }
}
```

Mmm...

En este caso no se puede simplemente eliminar el bloque como en la situación anterior. Lo que conviene hacer, es cambiar la pregunta **negando la condición original**, e invirtiendo así la situación.

¿Les resulta familiar?

```
procedimiento MarcarNegro() {
/**/
si (¬estaPintadaDeNegro?) entonces {
            PintarNegro
      }
}
```

No tenemos bloques en desuso y seguimos cumpliendo con el propósito

Breve repaso de conectivas lógicas (operadores)

Como podemos notar, las condiciones para evaluar las alternativas, utilizan las conectivas lógicas que ya conocemos. Las cuales también se denominan: conectores lógicos. **Recordemos cómo funcionan aquellas que utilizaremos para programar. Repasemos cada tabla de verdad:**

Negación (¬)

p	¬p
V	F
F	V

Conjunción (^)

p	q	p ∧ q
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción (v)

p	q	p ∨ q
V	V	V
V	F	V
F	V	V
F	F	F

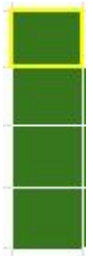
Veamos un ejemplo aplicando la disyunción

```
procedimiento MarcarVerde() {
/*PROPÓSITO: marcar de verde la celda que sea negra o roja. */
      si(estaPintadaDeNegro? ∨ estaPintadaDeRojo?) entonces {
            PintarVerde
      }
 }
```

Notar que al no haber desplazamientos, el procedimiento no tiene precondición. ¡No hay peligro de BOOM!

```
procedimiento MarcarColumna() {
/*PROPÓSITO: marcar de verde la columna cuyas celdas sean negra o roja. El cabezal finaliza en el extremo superior de la columna.
PRECONDICIÓN: El cabezal inicia en la base de la columna, y debe haber 3 celdas hacia arriba.*/
      repetir 3 veces {
            MarcarVerde()
            MoverArriba
      }
      MarcarVerde()
}
```

Ejemplo inicial

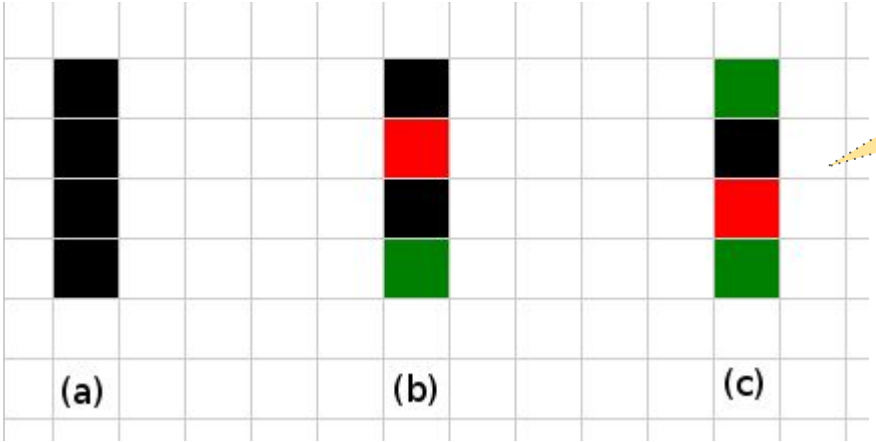


Ejemplo final

Analicemos mejor

Evaluar el procedimiento **MarcarColumnas()** para las siguientes situaciones.
Responder a la pregunta planteada:

¿Cómo queda el estado final de las columnas en cada caso, luego de haberse ejecutado el procedimiento?



Anidaciones: No son buena práctica

```
procedimiento MarcarVerde() {
      si(estaPintadaDeNegro?) entonces {
            si (estaPintadaDeRojo?) entonces {
                  PintarVerde
            }
      }
}
```

¡ANIDAR! No es una buena práctica tampoco en la alternativa condicional. No es legible y es propenso a errores. Además este código no funciona.

```
procedimiento MarcarVerde() {
      si(estaPintadaDeNegro? V estaPintadaDeRojo?) entonces
{
            PintarVerde
      }
   }
```

Solución: utilizar la conectiva correspondiente en base al propósito.

Condicionales consecutivos: No son buena práctica

```
procedimiento MarcarColor() {
      si(estaPintadaDeNegro?) entonces {
            PintarVerde
      }
      si (estaVacia?) entonces {
            PintarRojo
      }
 }
```

¡ALTERNATIVAS CONDICIONALES CONSECUTIVAS! No es una buena práctica al igual que en repeticiones. No es legible y es propenso a errores.

```
procedimiento MarcarColor(){
      si(estaPintadaDeNegro?) entonces {
            PintarVerde
      }sino {
            MarcarRojo()
      }
}
procedimiento MarcarRojo() {
      si (estaVacia?) entonces {
            PintarRojo
      }
}
```

Solución: descomponer el problema. Acá una posible solución, pero todo dependerá del propósito.

Estructuras condicionales anidadas: ¡No!

```
procedimiento MarcarColores() {
      repetir 2 veces {
            si(estaPintadaDeNegro?) entonces {
                  PintarVerde
            }
      }
 }
```

```
procedimiento MarcarColores(){
      si(estaPintadaDeNegro?) entonces {
            repetir 2 veces {
                  MarcarColor()
            }
      }
}
```

```
procedimiento IrACopaArbol(){
      repetir 4 veces {
            repetir 2 veces {
                  MoverArriba
            }
      }
}
```

Anidar diferentes estructuras condicionales tampoco es una buena práctica.
No se entiende qué se desea hacer.

Estructuras condicionales consecutivas: ¡Tampoco!

```
procedimiento MarcarColores() {
      repetir 2 veces {
            MocerArriba
      }
      si(estaPintadaDeNegro?) entonces {
                  PintarVerde
            }
       }
```

```
procedimiento MarcarColores(){
      si(estaPintadaDeNegro?) entonces {
            PintarVerde
      }
      repetir 2 veces {
                  MarcarColor()
            }
      }
}
```

Diferentes estructuras condicionales consecutivas tampoco es una buena práctica. Poco legible, mantenible y propenso a errores.

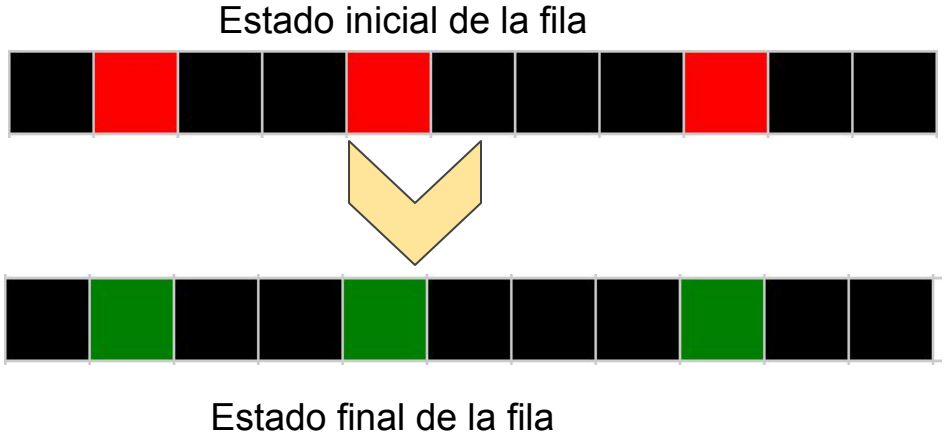
Resumiendo...

- 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
- Utilizar la versión acotada en lugar de dejar bloques vacíos
- La alternativa condicional no se puede anidar ni utilizar de manera consecutiva, al igual que la repetición simple
- Tampoco es legible anidar varias ni distintas estructuras de control, ni colocarlas de manera consecutivas.

Ejercicio para precalentar

Enunciado

Dada una fila de celdas pintadas de negro con algunas de ellas pintadas de rojo. Se solicita marcar (pintar) de verde solamente las celdas rojas.



Para reflexionar...

"Estudiar no es un acto de consumir ideas, sino de crearlas y recrearlas"