



Programación

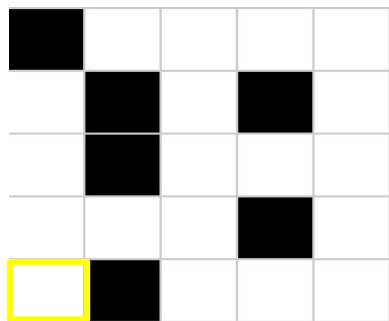
Alternativa Condicional

Universidad Nacional de Quilmes

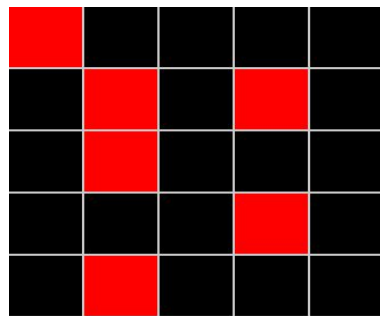
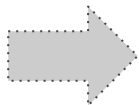
Pensemos el siguiente problema

Dado un cuadrado de 5x5, se solicitará 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 cuadrado.

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



Estado inicial del cuadrado



Estado final del cuadrado



Alternativa Condicional



Definición

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

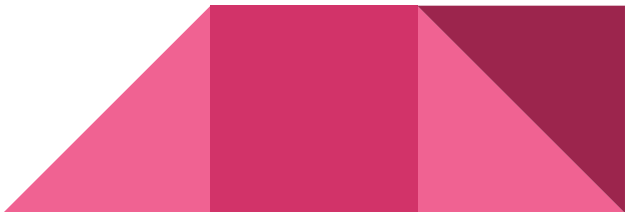
Nos permite elegir dos caminos distintos de acción en base a una condición.



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

Esta primitiva es de tipo condición. Más adelante veremos todas las que contamos en QDraw.

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



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 pinta de rojo y limpia la celda en caso contrario.

Un ejemplo más completo

Poco a poco vamos enriqueciendo nuestro lenguaje con un repertorio mayor de instrucciones y nuevos conceptos. 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, es decir **verdadero** o bien **falso**.

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 dichas condiciones.

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



Nuevas primitivas - Sintaxis

Adicionamos las siguientes instrucciones:

- **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
- **estaVacía?**: 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.



Otro ejemplo

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

Alternativa Condicional

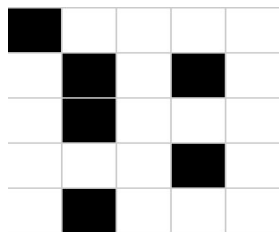
Forma acotada



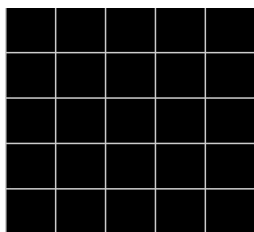
Analicemos el siguiente código

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

Como parte del programa contamos con el siguiente procedimiento:



Estado inicial del cuadrado



Estado final del cuadrado

```
procedimiento MarcarVacía() {  
  /**/  
  si (estaVacía?) 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 MarcarVacía() {  
    /**/  
    si (estaVacía?) 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 aquí, es cambiar la pregunta **negando la condición original**, invirtiendo así la situación.

Quedando como resultado el siguiente código:

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

¿Les resulta familiar?

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 utilizamos para programar.**

Repasemos su tabla de verdad:

Negación (\neg)

p	$\neg p$
V	F
F	V

Conjunción (\wedge)

p	q	$p \wedge q$
V	V	V
V	F	F
F	V	F
F	F	F

Disyunción (\vee)

p	q	$p \vee q$
V	V	V
V	F	V
F	V	V
F	F	F

Veamos un ejemplo aplicando disyunción

Definir un procedimiento que marce en verde la celda actual sólo si está pintada de negro o de rojo

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

Ahora definir el procedimiento **MarcarColumna()** que marque de verde las celdas negras o rojas de la columna.

```
procedimiento MarcarColumna() {  
  /*PROPÓSITO: marcar de verde la columna cuyas celdas  
  sean negra o roja. */  
  repetir 3 veces {  
    MarcarVerde()  
    MoverArriba  
  }  
  MarcarVerde()  
}
```



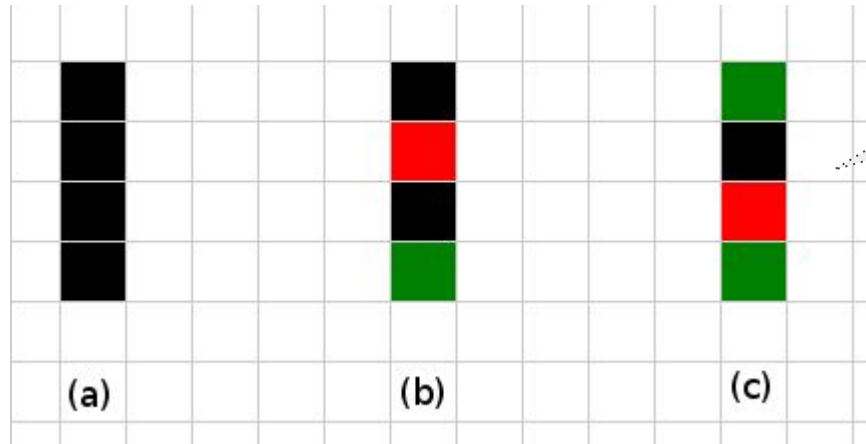
Ejemplo inicial



Ejemplo final

Analicemos mejor

Evaluar el procedimiento **MarcarColumnas()** para los siguientes tableros:



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

Aclaración importante: ¡Precondición!

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

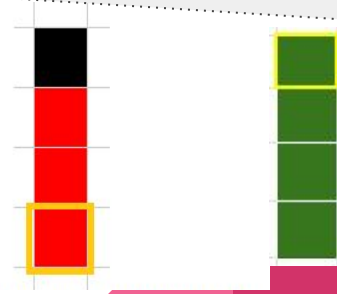
Notar que este procedimiento **no requiere precondición**, dado que no hay desplazamientos que puedan generar BOOM!

```
procedimiento MarcarColumna() {  
  /*PROPÓSITO: marcar de verde la columna cuyas celdas sean negra o roja.  
  El cabezal inicia en el extremo inferior de la columna y finaliza en su  
  extremo superior.
```

PRECONDICIÓN: debe haber 3 celdas hacia arriba del inicio del cabezal.*/

```
  repetir 3 veces {  
    MarcarVerde()  
    MoverArriba  
  }  
  MarcarVerde()  
}
```

En este caso sí nos desplazamos por lo cual es necesario aclarar la cant. de celdas necesarias para evitar el BOOM!



Ejemplo inicial Ejemplo final

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? Y estaPintadaDeRojo?) entonces  
    {  
        PintarVerde  
    }  
}
```



Solución: utilizar la conectiva correspondiente según el propósito.

Condicionales consecutivas: No son buena práctica

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



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

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

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.

Ejemplo de inicial de la fila



Ejemplo final de la fila



Para reflexionar...

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

