Programación

Clase 5

Alternativa Condicional

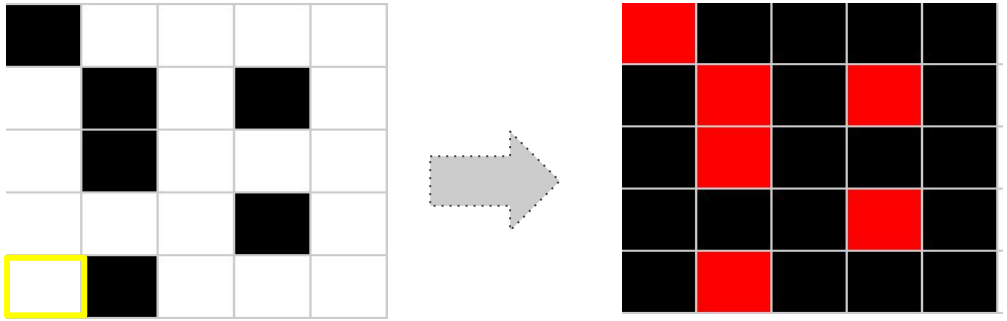
Universidad Nacional de Quilmes

Pensemos el siguiente problema

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

El cabezal comienza en la esquina inferior izquierda.

A continuación se muestra un tablero de ejemplo, pero tener en cuenta que ***no sabemos a priori cuáles son las celdas pintadas de negro*** *en el tablero con estado inicial*.



Estado inicial del tablero

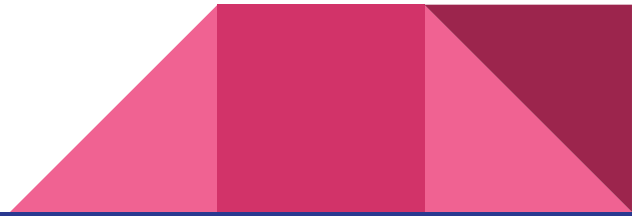
Estado final del tablero

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.



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

Aquí irá una instrucción del tipo condición que veremos más adelante

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

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 o no.

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.

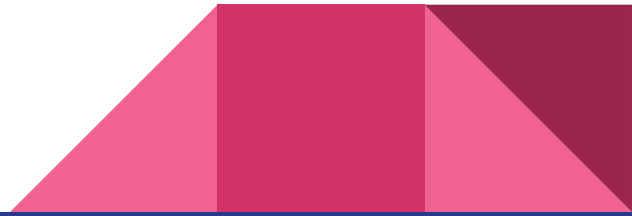
```
procedimiento PintarColumna () {
/*…*/
    repetir 4 veces {
        PintarCelda()
        MoverArriba
    }
    PintarCelda()
}
```

Condiciones

Definición

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 - 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
- **estaVacia**?: Denota Verdadero si la celda está vacía, Falso en cualquier otro caso

Volviendo al ejemplo simple

Reemplazamos con la primitiva correspondiente

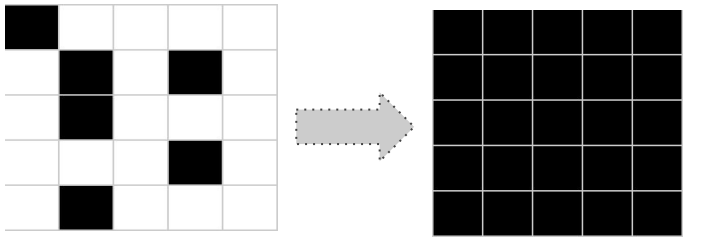
```
procedimiento PintarCelda() {
    si (estaPintadaDeNegro?) entonces {
        PintarRojo
    } sino {
        PintarNegro
    }
}
```

Alternativa Condicional
Forma acotada

Analicemos el siguiente código

Se necesita pintar todas las celdas del tablero 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.



Estado inicial del tablero

Estado final del tablero

Como parte del programa contamos con el siguiente procedimiento:

```
procedimiento PintarCelda() {
/**/
      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 PintarCelda() {
/**/
     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 PintarCelda() {
      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:

¿Les resulta familiar?

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

No tenemos bloques en desuso y seguimos cumpliendo con el propósito

Breve repaso de operadores lógicos

Como podemos notar, las condiciones necesarias para evaluar las alternativas, utilizan los operadores lógicos que ya conocemos.

Recordemos cómo funcionan los más utilizados, mediante su 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 disyunción

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

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

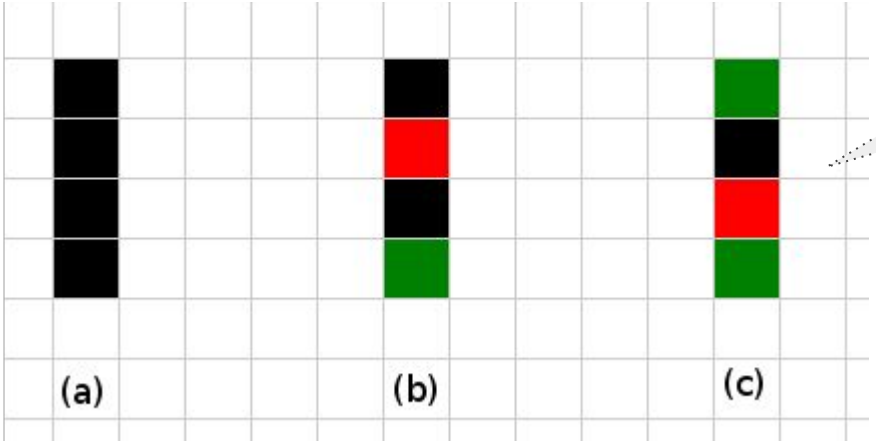
Ahora definir el procedimiento **RecorrerColumna()** que recorra una columna de 4 celdas de alto y pinte de verde la celda actual sólo si está pintada de negro o de rojo

```
procedimiento RecorrerColumna() {
      repetir 3 veces {
            PintarDeVerdeSoloSiEsNegraORoja()
            MoverArriba
      }
            PintarDeVerdeSoloSiEsNegraORoja()
}
```

Ejemplo de tablero inicial

Analicemos mejor

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



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

Resumiendo...

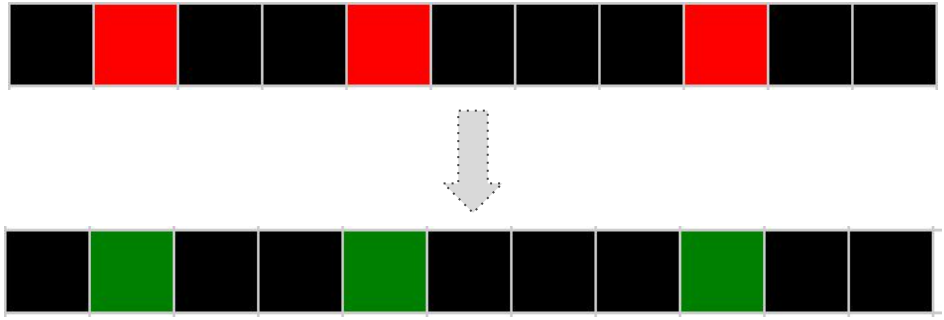
- La alternativa condicional permite elegir en base a una condición
- Las condiciones pueden tomar valores o bien verdaderos, o bien falsos
- En las condiciones podemos utilizar operadores lógicos como negación, conjunción y disyunción
- Utilizar la versión acotada en lugar de dejar bloques vacíos
-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

Ejercicio para precalentar

Enunciado

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.

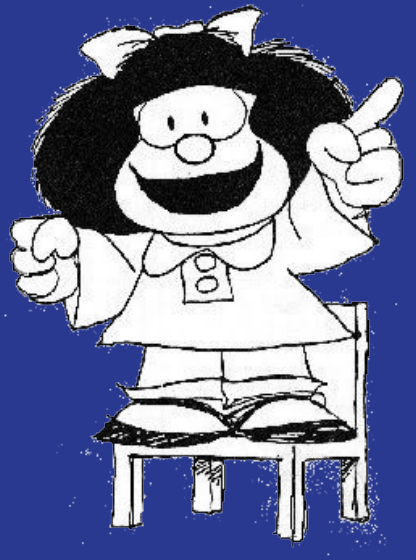
Ejemplo de estado inicial de tablero



Ejemplo de estado final de tablero

Para reflexionar...

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



Programación

Clase 5

Alternativa Condicional

Universidad Nacional de Quilmes