MAT1184 EID1: PROGRAMACIÓN DE PROPOSICIONES LÓGICAS

Mellado.O, Revillod.L, Sube.I

Lenguaje de programación Java:

Java es un lenguaje netamente orientado a objetos, o sea que para comenzar debemos definir una clase y dentro de ella se va a escribir el método Main, que es donde se escribe todo el código. En java para definir un arreglo se debe utilizar la siguiente sintaxis:
Tipodedato [ ] NombreDelArreglo = "new boolean[tamaño]" donde "tamaño" debe ser un número entero positivo.

Sintaxis en conectivos lógicos:

Los Operadores lógicos son los operadores que funcionan con valores booleanos, los valores son verdaderos o falsos.

- **Negación:** Para este conectivo se utiliza " ! ". Donde el carácter debe escribirse antes de la proposición.
  Ejemplo: $\sim$p se representa como !p
- **Conjunción:** Para este conectivo se utiliza " && ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\wedge$ q se representa como p && q
- **Disyunción Inclusiva:** Para este conectivo se utiliza " || ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\vee$ q se representa como p || q
- **Disyunción Exclusiva:** Para este conectivo se utiliza " != ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\veebar$ q se representa como p != q
- **Condicional:** Para este conectivo se utilizan " !, || ". En este caso el caracter " ! " se debe anteponer a la primera proposición, mientras que el segundo " || " debe encontrarse entre ambas proposiciones.
  Ejemplo: p $\rightarrow$ q se representa como !p || q
- **Bicondicional:** Para este conectivo se utilizan " == ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\leftrightarrow$ q se representa como p == q

Código:

Terminos y funciones utilizadas :
“Conectivos Lógicos” corresponde a la clase principal del script.
“public static void main(String [ ] args)” corresponde al metodo Main del programa. La función “System.out.println”, cumple la tarea de imprimir la información indicada por pantalla. Los términos “\n“, que representa un salto de línea y “\t“ una tabulación.

```
public class ConectivosLogicos //Se define la clase principal del script
{
    public static void main(String []args)
    {
// En java para definir un arreglo se debe utilizar la siguiente sintaxis:
// Tipodedato[] NombreDelArreglo = "new boolean[tamaño]"
//    donde tamaño tiene que ser un entero positivo
        boolean[] bool = new boolean[2];
        bool[0] = true;
        bool[1] = false;
        //Asignacion de indices desde el [0], [1] ;
        // donde [0] corresponde a True y [1] a False

        //Negacion
        //Para este conectivo lógico se utiliza  " ! "
        //Ejemplo: ~p se representa !p
        System.out.println("Negación: p ~p");
        System.out.println("p\t~p");
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            boolean P = !p;
            //ciclo for
            //se analizan los valores de p y q
            System.out.println(p + "\t" + P);
            //Se muestran los resultados por pantalla
        }
        //Conjuncion
        //Para este conectivo lógico se utiliza  " && "
        //Ejemplo: p & q se representa p && q
        System.out.println("\nConjunción: p AND q");
        System.out.println("p\tq\tp AND q");
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            for (boolean q: bool) {
                boolean P = p && q;
                //ciclo for
                //se analizan los valores de p y q
                System.out.println(p+"\t"+q+"\t"+P);
                //Se muestran los resultados por pantalla
            }
        }
        //Disyunción inclusiva
        //Para este conectivo lógico se utiliza  " || "
        //Ejemplo: p v q se representa p || q
        System.out.println("\nDisyunción inclusiva: p OR q");
        System.out.println("p\tq\tp OR q");
```

```
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            for (boolean q: bool) {
                boolean P = p || q;
                //ciclo for
                //se analizan los valores de p y q
                System.out.println(p+"\t"+q+"\t"+P);
                //Se muestran los resultados por pantalla
            }
        }
        //Disyunción exclusiva
        //Para este conectivo lógico se utiliza  " != ".
        System.out.println("\nDisyunción exclusiva: p XOR q");
        System.out.println("p\tq\tp XOR q");
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            for (boolean q: bool) {
                boolean P = p != q;
                //ciclo for
                //se analizan los valores de p y q
                System.out.println(p+"\t"+q+"\t"+P);
                //Se muestran los resultados por pantalla
            }
        }
        //Condicional
        //Para este conectivo lógico se utilizan  " !, || "
        System.out.println("\nCondicional: p -> q");
        System.out.println("p\tq\tp -> q");
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            for (boolean q: bool) {
                boolean P = !p || q;
        //ciclo for
        //se analizan los valores de p y q
                System.out.println(p+"\t"+q+"\t"+P);
                //Se muestran los resultados por pantalla
            }
        }
        //Bicondicional
        //Para este conectivo lógico se utilizan  " == "
        System.out.println("\nBicondicional: p <-> q");
        System.out.println("p\tq\tp <-> q");
        //Se imprimen las proposiciones
        for (boolean p: bool) {
            for (boolean q: bool) {
                boolean P = p == q;
        //ciclo for
        //se analizan los valores de p y q
                System.out.println(p+"\t"+q+"\t"+P);
        //Se muestran los resultados por pantalla
            }
        }
}
```

```
}
```

Ejecución:


conectivosLogicos.php
ConectivosLogicos.java
buffers
1 public class ConectivosLogicos //Se define la clase principal del script
1 {
2 public static void main(String []args)
3 {
4 // En java para definir un arreglo se debe utilizar la siguiente sintaxis:
5 // Tipodedato[] NombreDelArreglo = "new boolean[tamaño]"
6 // donde tamaño tiene que ser un entero positivo
7 boolean[] bool = new boolean[2];
8 bool[0] = true;
9 bool[1] = false;
10 //Asignacion de indices desde el [0], [1] ;
11 // donde [0] corresponde a True y [1] a False
12
13 //Negacion
14 //Para este conectivo lógico se utiliza " ! "
15 //Ejemplo: ~p se representa !p
16 System.out.println("Negación: p ~p");
17 System.out.println("p\t~p");
18 //Se imprimen las proposiciones
19 for (boolean p: bool) {
20 boolean P = !p;
21 //ciclo for
22 //se analizan los valores de p y q
23 System.out.println(p + "\t" + P);
24 //Se muestran los resultados por pantalla
25 }
26 //Conjuncion
27 //Para este conectivo lógico se utiliza " && "
28 //Ejemplo: p & q se representa p && q
29 System.out.println("\nConjunción: p AND q");
30 System.out.println("p\tq\tp AND q");
31 //Se imprimen las proposiciones
32 for (boolean p: bool) {
33 for (boolean q: bool) {
34 boolean P = p && q;
35 //ciclo for
36 //se analizan los valores de p y q
37 System.out.println(p+"\t"+q+"\t"+P);
38 //Se muestran los resultados por pantalla
NORMAL
ConectivosLogicos.java
java
utf-8[unix]
0% :1/102 :1
[6]trai…
:NERDTreeToggle
oscar@UwU ~/w/u/M/e/final
java ConectivosLogicos
Negación: p ~p
p ~p
true false
false true
Conjunción: p AND q
p q p AND q
true true true
true false false
false true false
false false false
Disyunción inclusiva: p OR q
p q p OR q
true true true
true false true
false true true
false false false
Disyunción exclusiva: p XOR q
p q p XOR q
true true false
true false true
false true true
false false false
Condicional: p -> q
p q p -> q
true true true
true false false
false true true
false false true
Bicondicional: p <-> q
p q p <-> q
true true true
true false false
false true false
false false true
oscar@UwU ~/w/u/M/e/final

Lenguaje de programación php:

Php es un lenguaje orientado al desarrollo web, donde se utilizan etiquetas, para dar inicio al script, "<?php" y otra para finalizar el script "?>". Además, por sintaxis, en este lenguaje las variables deben comenzar con el carácter "$".

Sintaxis en conectivos lógicos:

- **Negación:** Para este conectivo se utiliza " ! ". Donde el carácter debe escribirse antes de la proposición. Ejemplo: $\sim$p se representa !$p
- **Conjunción:** Para este conectivo se utiliza " && ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\wedge$ q se representa $p && $q
- **Disyunción inclusiva:** Para este conectivo se utiliza " || ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\vee$ q se representa $p || $q
- **Disyunción exclusiva:** Para este conectivo se utiliza " != ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\veebar$ q se representa $p != $q
- **Condicional:** Para este conectivo se utiliza " ! " y " || ".En este caso el caracter" ! " se debe anteponer a la primera proposición, mientras que el segundo " || " debe encontrarse entre ambas proposiciones.
  Ejemplo: p $\rightarrow$ q se representa !($p) || $q
- **Bicondicional:** Para este conectivo se utiliza " == ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\leftrightarrow$ q se representa $p == $q

Código:

Terminos y funciones utilizadas:
La función "echo" imprime la información indicada por pantalla. El término "foreach" corresponde a un ciclo. Es una forma de iterar sobre arrays. Además éste sólo funciona sobre arrays y objetos.

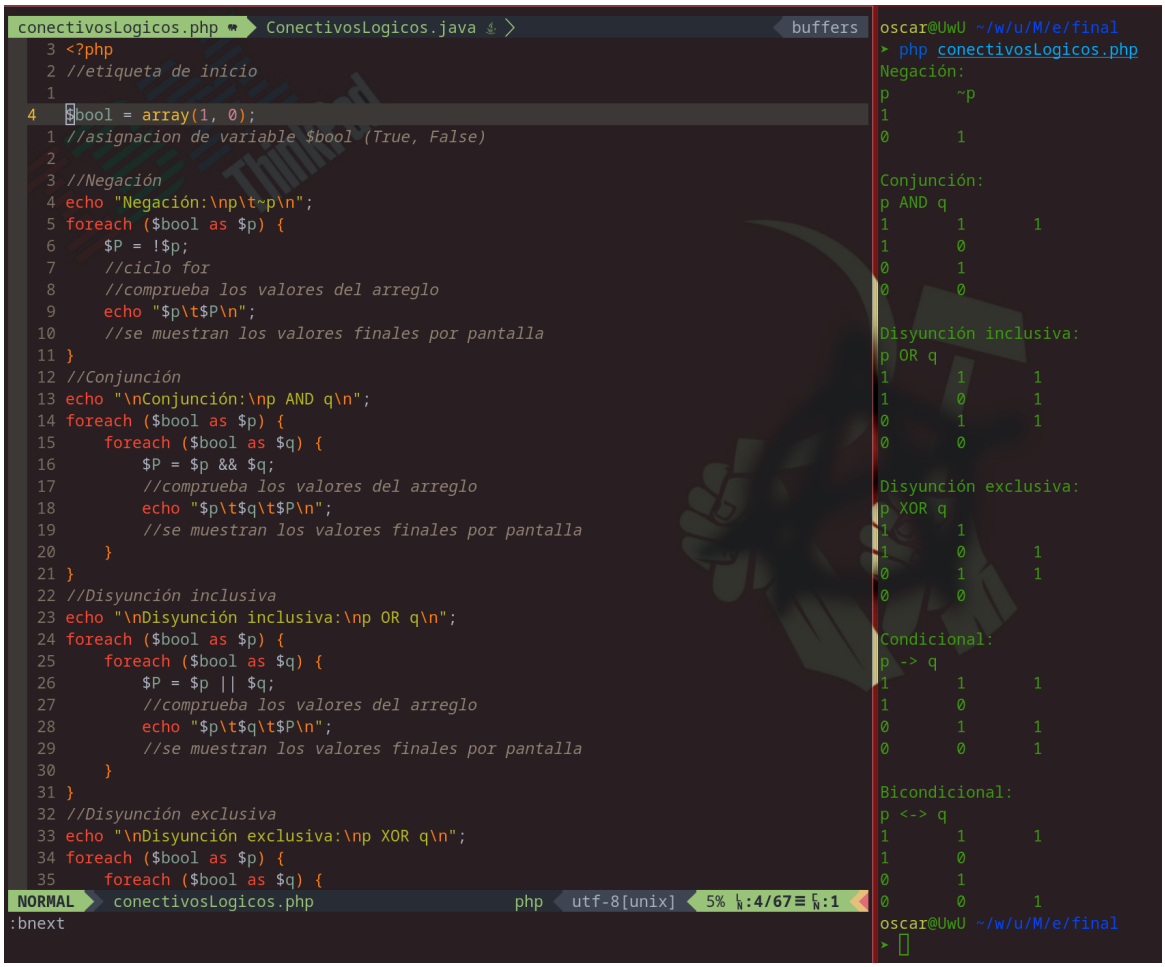
```php
<?php
//etiqueta de inicio

$bool = array(1, 0);
//asignacion de variable $bool (True, False)

//Negación
echo "Negación:\np\t~p\n";
foreach ($bool as $p) {
    $P = !$p;
    //ciclo for
    //comprueba los valores del arreglo
    echo "$p\t$P\n";
    //se muestran los valores finales por pantalla
}
//Conjunción
echo "\nConjunción:\np AND q\n";
foreach ($bool as $p) {
    foreach ($bool as $q) {
        $P = $p && $q;
```

```
        //comprueba los valores del arreglo
        echo "$p\t$q\t$P\n";
        //se muestran los valores finales por pantalla
    }
}
//Disyunción inclusiva
echo "\nDisyunción inclusiva:\np OR q\n";
foreach ($bool as $p) {
    foreach ($bool as $q) {
        $P = $p || $q;
        //comprueba los valores del arreglo
        echo "$p\t$q\t$P\n";
        //se muestran los valores finales por pantalla
    }
}
//Disyunción exclusiva
echo "\nDisyunción exclusiva:\np XOR q\n";
foreach ($bool as $p) {
    foreach ($bool as $q) {
        $P = $p != $q;
        //comprueba los valores del arreglo
        echo "$p\t$q\t$P\n";
        //se muestran los valores finales por pantalla
    }
}
// Condicional
echo "\nCondicional:\np -> q\n";
foreach ($bool as $p) {
    foreach ($bool as $q) {
        $P = !($p) || $q;
        //comprueba los valores del arreglo
        echo "$p\t$q\t$P\n";
        //se muestran los valores finales por pantalla
    }
}
// Bicondicional
echo "\nBicondicional:\np <-> q\n";
foreach ($bool as $p) {
    foreach ($bool as $q) {
        $P = $p == $q;
        //comprueba los valores del arreglo
        echo "$p\t$q\t$P\n";
        //se muestran los valores finales por pantalla
    }
}
//etiqueta de cierre
?>
```

Ejecución:


conectivosLogicos.php
ConectivosLogicos.java
buffers
3 <?php
2 //etiqueta de inicio
1
4 $bool = array(1, 0);
1 //asignacion de variable $bool (True, False)
2
3 //Negación
4 echo "Negación:\np\t~p\n";
5 foreach ($bool as $p) {
6 $P = !$p;
7 //ciclo for
8 //comprueba los valores del arreglo
9 echo "$p\t$P\n";
10 //se muestran los valores finales por pantalla
11 }
12 //Conjunción
13 echo "\nConjunción:\np AND q\n";
14 foreach ($bool as $p) {
15 foreach ($bool as $q) {
16 $P = $p && $q;
17 //comprueba los valores del arreglo
18 echo "$p\t$q\t$P\n";
19 //se muestran los valores finales por pantalla
20 }
21 }
22 //Disyunción inclusiva
23 echo "\nDisyunción inclusiva:\np OR q\n";
24 foreach ($bool as $p) {
25 foreach ($bool as $q) {
26 $P = $p || $q;
27 //comprueba los valores del arreglo
28 echo "$p\t$q\t$P\n";
29 //se muestran los valores finales por pantalla
30 }
31 }
32 //Disyunción exclusiva
33 echo "\nDisyunción exclusiva:\np XOR q\n";
34 foreach ($bool as $p) {
35 foreach ($bool as $q) {
NORMAL
conectivosLogicos.php
php
utf-8[unix]
5% ℕ:4/67≡ ℕ:1
:bnext
oscar@UwU ~/w/u/M/e/final
▸ php conectivosLogicos.php
Negación:
p ~p
1
0 1
Conjunción:
p AND q
1 1 1
1 0
0 1
0 0
Disyunción inclusiva:
p OR q
1 1 1
1 0 1
0 1 1
0 0
Disyunción exclusiva:
p XOR q
1 1
1 0 1
0 1 1
0 0
Condicional:
p -> q
1 1 1
1 0
0 1 1
0 0 1
Bicondicional:
p <-> q
1 1 1
1 0
0 1
0 0 1
oscar@UwU ~/w/u/M/e/final
▸

Lenguaje de programación R:

R es un entorno y lenguaje de programación enfocado al análisis estadístico, utilizado en su mayoría en la investigación científica, debido a su simple sintaxis y la facilidad para generar gráficos. Se pueden crear diferentes variables con nombres iguales, pero alternando mayúsculas y minúsculas. Para agrupar expresiones, se puede utilizar: punto y coma, paréntesis y llaves.

Sintaxis en conectivos lógicos:

- **Negación:** Para este conectivo se utiliza " ! ". Donde el carácter debe escribirse antes de la proposición.
  Ejemplo: ∼p se representa !p
- **Conjunción:** Para este conectivo se utiliza " & ". Donde el carácter debe escribirse antes de la proposición.
  Ejemplo: p ∧ q se representa p & q
- **Disyunción inclusiva:** Para este conectivo se utiliza " | ". Donde el carácter debe escribirse antes de la proposición.
  Ejemplo: p ∨ q se representa p | q
- **Disyunción exclusiva:** Para este conectivo se utiliza " != ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p $\veebar$ q se representa p != q
- **Condicional:** Para este conectivo se utiliza " ! " y " | ". En este caso el caracter" ! " se debe anteponer a la primera proposición, mientras que el segundo " | " debe encontrarse entre ambas proposiciones.
  Ejemplo: p → q se representa !p | q
- **Bicondicional:** Para este conectivo se utiliza " == ". Los caracteres deben posicionarse entre ambas proposiciones.
  Ejemplo: p ↔ q se representa p == q

Código:

Terminos y funciones utilizadas:
Funciones "print" y "cat" que cumplen la tarea de mostrar información por pantalla. Print tiene una sintaxis más parecida a la función print de python y cat se parece más a la función printf en el lenguaje c. Bucle "for" el cual es un ciclo iterativo y su complemento "in" con el cual se verifican los elementos de un determinado arreglo. El operador "<-", que cumple la función de asignar valores a una variable/arreglo.

```
# Conectivos Logicos R

bool <- c(T, F) #Asignacion de arreglo(True,False)

print("Negación: p ~p")
cat("p\t~p\n")
for (p in bool) {
#ciclo for que verifica los booleanos del arreglo
    P <- !p
    cat(p,"\t", P, "\n")
#Se imprimen los resultados por pantalla
}
#Conjución
#Para este conectivo logico se utiliza "&"
cat("\n")
print("Conjunción: p AND q")
cat("p\tq\tp AND q\n")
```

```
for (p in bool) {
    for (q in bool) {
#ciclo for que verifica los booleanos del arreglo
        P <- p & q
        cat(p,"\t",q,"\t", P, "\n")
#Se imprimen los resultados por pantalla
    }
}
#Disyucion inclusiva
#Para este conectivo logico se utiliza "|"
cat("\n")
print("Disyunción inclusiva: p OR q")
cat("p\tq\tp OR q\n")
for (p in bool) {
    for (q in bool) {
#ciclo for que verifica los booleanos del arreglo
        P <- p | q
        cat(p,"\t",q,"\t", P, "\n")
#Se imprimen los resultados por pantalla
    }
}
#Disyucion exclusiva
#Para este conectivo logico se utiliza "!="
cat("\n")
print("Disyunción exclusiva: p XOR q")
cat("p\tq\tp XOR q\n")
for (p in bool) {
    for (q in bool) {
#ciclo for que verifica los booleanos del arreglo
        P <- p != q
        cat(p,"\t",q,"\t", P, "\n")
#Se imprimen los resultados por pantalla
    }
}
#Condicional
#Para este conectivo logico se utiliza "!" y "|"
cat("\n")
print("Condicional: p -> q")
cat("p\tq\tp -> q\n")
for (p in bool) {
    for (q in bool) {
#ciclo for que verifica los booleanos del arreglo
        P <- !p | q
        cat(p,"\t",q,"\t", P, "\n")
#Se imprimen los resultados por pantalla
    }
}
#Bicondicional
#Para este conectivo logico se utiliza "=="
cat("\n")
print("Bicondicional: p <-> q")
cat("p\tq\tp <-> q\n")
for (p in bool) {
```

```r
    for (q in bool) {
#ciclo for que verifica los booleanos del arreglo
        P <- p == q
        cat(p,"\t",q,"\t", P, "\n")
#Se imprimen los resultados por pantalla
    }
}
```

Ejecución: