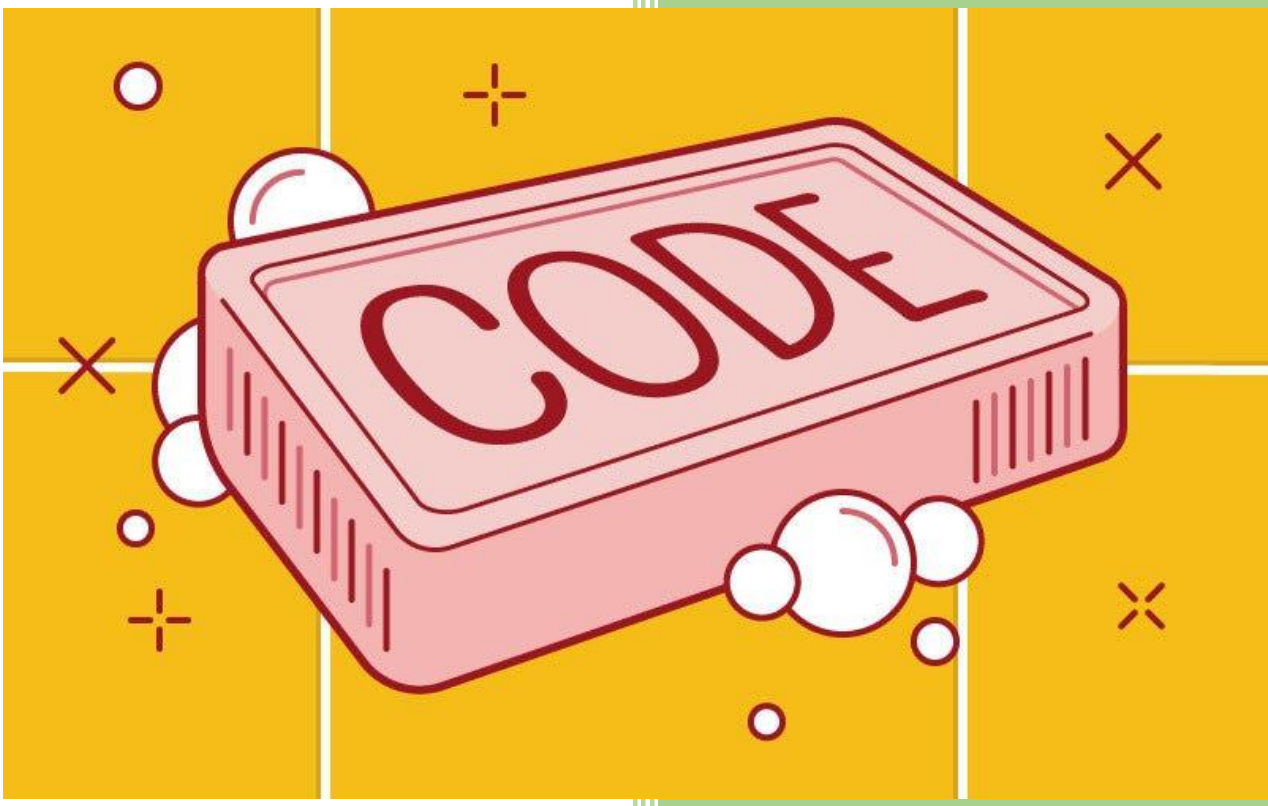
# CLEAN CODE




Daniel Silva Moratilla

12/12/2023

# Contenido Clean Code

# BLOQUE 1. NOMBRES

## 1.1. Usa nombres con significado

Se deben usar nombres con un valor y significado para la comprensión de qué hace esa variable

```
botonPresionado->setText(QString(QChar::fromLatin1(XoO)));
if(XoO == 'X'){
    botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                   "color: white;"
                                   "font: 700 12pt;}");
}
else if (XoO == 'O'){
    botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                   "color: white; "
                                   "font: 700 12pt;}");
}
```

**MAL**

```
botonPresionado->setText(QString(QChar::fromLatin1(turno)));
if(turno == 'X'){
    botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                   "color: white;"
                                   "font: 700 12pt;}");
}
else if (turno == 'O'){
    botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                   "color: white; "
                                   "font: 700 12pt;}");
}
```

**BIEN**

## 1.2. Usa nombres pronunciables

Los nombres que pongamos deben poder ser leídos fácilmente

```
void MainWindow::chkTble()
{
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < 3; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;
```

**MAL**

```
void MainWindow::checkTable()
{
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < 3; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;
```

**BIEN**

## 1.3. Usa nombres que se puedan buscar

A la hora de buscar donde haces que cosa, siempre se agradece saber el nombre de una variable en lugar de un numero sin aparente significado

```
for (int fila = 0; fila < 3; fila++)
{
    filas[fila] = 0;
    columnas[fila] = 0;
    diagonalLeft = 0;
    diagonalRight = 0;

    for (int columna = 0; columna < 3; columna++)
    {
        filas[fila] += tablero[fila][columna];
        columnas[fila] += tablero[columna][fila];
        diagonalLeft += tablero[columna][columna];
        diagonalRight += tablero[columna][2 - columna];
        /*Aqui se va a comprobar a cuanto equivale cada fila y columna
        con los numeros que he ido asignando en setMatriz() y se va a sumar
        cada numero con la fila, columna y diagonales correspondientes*/
    }
```

**MAL**

```
for (int fila = 0; fila < 3; fila++)
{
    filas[fila] = 0;
    columnas[fila] = 0;
    diagonalLeft = 0;
    diagonalRight = 0;

    for (int columna = 0; columna < 3; columna++)
    {
        filas[fila] += tablero[fila][columna];
        columnas[fila] += tablero[columna][fila];
        diagonalLeft += tablero[columna][columna];
        diagonalRight += tablero[columna][2 - columna];
        /*Aqui se va a comprobar a cuanto equivale cada fila y columna
        con los numeros que he ido asignando en setMatriz() y se va a sumar
        cada numero con la fila, columna y diagonales correspondientes*/
    }
```

**BIEN**

## 1.4. Nombres de clases y métodos

Las clases deben de tener de nombre sustantivos y los métodos o funciones verbos

```
void MainWindow::Winner()
{
    juego = 'W';
    disableAllButtons();
    ui->pushButton->setEnabled(true);

}
```

**MAL**

```
void MainWindow::showWinner()
{
    juego = 'W';
    disableAllButtons();
    ui->pushButton->setEnabled(true);

}
```

**BIEN**

## 1.5. Elegir una palabra por concepto

Si realiza una tarea similar o tiene una responsabilidad parecida en tu código, debería usar el mismo concepto.

```
void MainWindow::enableAllButtons()
{
    for (std::size_t i = 0; i < 9; i++)
    {
        botones[i]->setEnabled(true);
    }
}

void MainWindow::setStartStyleSheet()
{
    for (std::size_t i = 0; i < 9; i++)
    {
        botones[i]->setText("");
        botones[i]->setStyleSheet("");
    }
}
```

**MAL**

```
void MainWindow::enableAllButtons()
{
    for (std::size_t i = 0; i < 9; i++)
    {
        botones[i]->setEnabled(true);
        botones[i]->setText("");
        botones[i]->setStyleSheet("");
    }
}
```

**BIEN**

# BLOQUE 2: FUNCIONES

## 2.1. Las funciones deben de ser pequeñas

Funciones de un límite razonable que no sean muy largas con un máximo de 1 o 2 niveles y poco más de 120 caracteres por línea máximo

```
void MainWindow::buttonMethod(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    botonPresionado->setText(QString(QChar::fromLatin1(turno)));
    if(turno == 'X'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                       "color: white;"
                                       "font: 700 12pt;}");
    }
    else if (turno == 'O'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #FF6B00;"
                                       "color: white; "
                                       "font: 700 12pt;}");
    }
    tablero[fila][columna] = (turno == 'X' ? 1 : 5);
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < mNumeroDeFilas; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;

        for (int columna = 0; columna < mNumeroDeColumnas; columna++)
        {
            filas[fila] += tablero[fila][columna];
            columnas[fila] += tablero[columna][fila];
            diagonalLeft += tablero[columna][columna];
            diagonalRight += tablero[columna][2 - columna];
            /*Aqui se va a comprobar a cuanto equivale cada fila y columna
            con los numeros que he ido asignando en setMatriz() y se va a sumar
            cada numero con la fila, columna y diagonales correspondientes*/
        }

        if (filas[fila] == 3 || columnas[fila] == 3 || diagonalLeft == 3 || diagonalRight == 3 ||
            filas[fila] == 15 || columnas[fila] == 15 || diagonalLeft == 15 || diagonalRight == 15)
        {
            showWinner();

        }
        else if (contador == mNumeroMaximoFichas)
        {
            juego = 'N'; //indico que el juego no lo ha ganado ninguno de los jugadores
        }
    }
    switch (juego)
    {
    case 'W':
        ui->label->setText("El ganador es el jugador: " + QString(QChar::fromLatin1(turno)));
        break;
    case 'N':
        ui->label->setText("No ha ganado ninguno de los jugadores");
        ui->pushButton->setEnabled(true);
        break;
    case 'A':
        turno = (turno == 'X' ? 'O' : 'X');
        ui->label->setText("Es el turno del jugador: " + QString(QChar::fromLatin1(turno)));
        break;
    }
    botonPresionado->setEnabled(false);
    /*Esta funcion va a repetirse en todo momento,
    hace que cuando pulse un boton del tablero va a pasar
    por distintas funciones que puede ver más abajo
    sumo 1 al contador para el caso del empate y desactivo
    el boton pulsado para no ser pulsado de nuevo*/
}
```

**MAL**

```
void MainWindow::buttonMethod(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    setIcon(botonPresionado);
    setMatriz(fila, columna);
    checkTable();
    changeTurn();
    botonPresionado->setEnabled(false);

}
```

```
void MainWindow::checkTable()
{
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < mNumeroDeFilas; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;

        for (int columna = 0; columna < mNumeroDeColumnas; columna++)
        {
            filas[fila] += tablero[fila][columna];
            columnas[fila] += tablero[columna][fila];
            diagonalLeft += tablero[columna][columna];
            diagonalRight += tablero[columna][2 - columna];
            /*Aqui se va a comprobar a cuanto equivale cada fila y columna
            con los numeros que he ido asignando en setMatriz() y se va a sumar
            cada numero con la fila, columna y diagonales correspondientes*/
        }

        if (filas[fila] == 3 || columnas[fila] == 3 || diagonalLeft == 3 || diagonalRight == 3 ||
            filas[fila] == 15 || columnas[fila] == 15 || diagonalLeft == 15 || diagonalRight == 15)
        {
            showWinner();

        }
        else if (contador == mNumeroMaximoFichas)
        {
            juego = 'N'; //indico que el juego no lo ha ganado ninguno de los jugadores
        }
    }

}
```

**BIEN**

## 2.2. Haz sólo una cosa

Está relacionado con lo anterior por lo que el propio ejemplo nos sirve, cada función debería de hacer una única cosa y centrarse en hacerla bien

```
void MainWindow::buttonMethod(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    botonPresionado->setText(QString(QChar::fromLatin1(turno)));
    if(turno == 'X'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                       "color: white;"
                                       "font: 700 12pt;}");
    }
    else if (turno == 'O'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                       "color: white; "
                                       "font: 700 12pt;}");
    }
    tablero[fila][columna] = (turno == 'X' ? 1 : 5);
```

**MAL**

```
void MainWindow::setIcon(QPushButton* botonPresionado)
{
    botonPresionado->setText(QString(QChar::fromLatin1(turno)));
    if(turno == 'X'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                       "color: white;"
                                       "font: 700 12pt;}");
    }
    else if (turno == 'O'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                       "color: white; "
                                       "font: 700 12pt;}");
    }
    /*Esta funcion me permite cambiar el color del boton presionado
    esto me sirve para mostrar que jugador ha pulsado que boton y cual cuenta
    para las comprobaciones despues.*/

}
```

```
void MainWindow::setMatriz(int fila, int columna)
{
    tablero[fila][columna] = (turno == 'X' ? 1 : 5);
    /*Esta funcion me va a guardar si los botones lo pulsa un jugador o otro
    si lo pulsa la X, me va a guardar un 1 en el boton pulsado y si no, 5.
    Esto sirve para la suma de las posteriores comprobaciones de ganador*/
}
```

**BIEN**

## 2.3. No abuses de los switches/when

Es difícil que una estructura de control haga sólo una cosa. Si los usas debes plantearte si hay una solución o alternativa mejor sin su uso

```
void MainWindow::changeTurn()
{
    switch (juego)
    {
    case 'W':
        ui->label->setText("El ganador es el jugador: " + QString(QChar::fromLatin1(turno)));
        break;
    case 'N':
        ui->label->setText("No ha ganado ninguno de los jugadores");
        ui->pushButton->setEnabled(true);
        break;
    case 'A':
        turno = (turno == 'X' ? 'O' : 'X');
        ui->label->setText("Es el turno del jugador: " + QString(QChar::fromLatin1(turno)));
        break;
    }

}
```

**MAL**

```
void MainWindow::changeTurn()
{
    if (juego == 'W'){
        ui->label->setText("El ganador es el jugador: " + QString(QChar::fromLatin1(turno)));
        return;
    } else if (juego == 'N'){
        ui->label->setText("No ha ganado ninguno de los jugadores");
        ui->pushButton->setEnabled(true);
        return;
    } else if (juego == 'A'){
        turno = (turno == 'X' ? 'O' : 'X');
        ui->label->setText("Es el turno del jugador: " + QString(QChar::fromLatin1(turno)));
    }


}
```

**BIEN**

## 2.4. Numero de argumentos por función

El numero de argumentos es recomendable que sea entre 0 y 3, si son más de 3 deberíamos intentar evitarlo

```
void MainWindow::buttonMethod(int contador, int fila, int columna, QPushButton* botonPresionado)
{
    contador = contador++;
    setIcon(botonPresionado);
    setMatriz(fila, columna);
    checkTable();
    changeTurn();
    botonPresionado->setEnabled(false);
    /*Esta funcion va a repetirse en todo momento,
    hace que cuando pulse un boton del tablero va a pasar
    por distintas funciones que puede ver más abajo
    sumo 1 al contador para el caso del empate y desactivo
    el boton pulsado para no ser pulsado de nuevo*/

}
```

**MAL**

```
void MainWindow::buttonMethod(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    setIcon(botonPresionado);
    setMatriz(fila, columna);
    checkTable();
    changeTurn();
    botonPresionado->setEnabled(false);
    /*Esta funcion va a repetirse en todo momento,
    hace que cuando pulse un boton del tablero va a pasar
    por distintas funciones que puede ver más abajo
    sumo 1 al contador para el caso del empate y desactivo
    el boton pulsado para no ser pulsado de nuevo*/

}
```

**BIEN**

## 2.5. Evita los “FLAG ARGUMENTS”

Intentar sobre todo evitar las funciones booleanas debido a que estas hacen 2 cosas, lo cual choca con la regla de hacer una cosa únicamente

```
void checkTable();

void changeTurn();

void showWinner();

void disableAllButtons();

void enableAllButtons();

void on_pushButton_clicked();

bool isThereWinner();
```

**MAL**

```
void checkTable();

void changeTurn();

void showWinner();

void disableAllButtons();

void enableAllButtons();

void on_pushButton_clicked();

void isThereWinner();
```

**BIEN**

## 2.6. No generar efectos colaterales

Si una función hace algo que no está representado en el nombre significa que nos oculta información. Los nombres de las funciones deben indicar TODO lo que hace dentro de esa unicidad

```
void MainWindow::buttonMethod(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    setIcon(botonPresionado);
    setMatriz(fila, columna);
    checkTable();
    changeTurn();
    botonPresionado->setEnabled(false);
    /*Esta funcion va a repetirse en todo momento,
    hace que cuando pulse un boton del tablero va a pasar
    por distintas funciones que puede ver más abajo
    sumo 1 al contador para el caso del empate y desactivo
    el boton pulsado para no ser pulsado de nuevo*/

}
```

**MAL**

```
void MainWindow::buttonClicked(int fila, int columna, QPushButton* botonPresionado)
{
    contador++;
    setIcon(botonPresionado);
    setMatriz(fila, columna);
    checkTable();
    changeTurn();
    botonPresionado->setEnabled(false);
    /*Esta funcion va a repetirse en todo momento,
    hace que cuando pulse un boton del tablero va a pasar
    por distintas funciones que puede ver más abajo
    sumo 1 al contador para el caso del empate y desactivo
    el boton pulsado para no ser pulsado de nuevo*/

}
```

**BIEN**

## 2.7. Don’t Repeat Yourself

Repetir código siempre es un problema, siempre es mejor extraer código repetido y realizarlo donde corresponda

```
void MainWindow::disableAllButtons()
{
    botones[1]->setDisabled(true);
    botones[2]->setDisabled(true);
    botones[3]->setDisabled(true);
    botones[4]->setDisabled(true);
    botones[5]->setDisabled(true);
    botones[6]->setDisabled(true);
    botones[7]->setDisabled(true);
    botones[8]->setDisabled(true);
    botones[0]->setDisabled(true);
}
```

**MAL**

```
void MainWindow::disableAllButtons()
{
    for (std::size_t i = 0; i < botones.size(); i++)
    {
        botones[i]->setDisabled(true);
    }
}
```

**BIEN**

# BLOQUE 3: COMENTARIOS

## 3.1. Los comentarios mienten

Si el código asociado a un comentario se modifica, el comentario queda obsoleto, y nadie se enterará.

```
void MainWindow::setIcon(QPushButton* botonPresionado)
{
    botonPresionado->setText(QString(QChar::fromLatin1(turno)));
    if(turno == 'X'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                       "color: white;"
                                       "font: 700 12pt;}");
    }
    else if (turno == 'O'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                       "color: white; "
                                       "font: 700 12pt;}");
    }
    /*Esta funcion me permite cambiar el color del boton presionado
    esto me sirve para mostrar que jugador ha pulsado que boton y cual cuenta
    para las comprobaciones despues.*/

}
```

**MAL**

```
void MainWindow::setIcon(QPushButton* botonPresionado)
{
    botonPresionado->setText(QString(QChar::fromLatin1(turno)));
    if(turno == 'X'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #0D7ADF; "
                                       "color: white;"
                                       "font: 700 12pt;}");
    }
    else if (turno == 'O'){
        botonPresionado->setStyleSheet("QPushButton{ background-color: #FF8B00;"
                                       "color: white; "
                                       "font: 700 12pt;}");
    }

}
```

**BIEN**

## 3.2. Usa código auto explicativo

En lugar de usar comentarios, busca otras alternativas como hacer tu código ser entendido por sí mismo, nombres descriptivos, nombres de funciones detalladas…

```
        else if (contador == mNumeroMaximoFichas)
        {
            juego = 'N'; //indico que el juego no lo ha ganado ninguno de los jugadores
        }
    }

}
```

**MAL**

```
else if (contador == mNumeroMaximoFichas)
{
    juego = "No Win";
}
```

**BIEN**

## 3.3. A veces los comentarios son necesarios

En casos donde el código es complejo, trabajas con una API/framework o tienes algún problema externo, puede ser necesario usar comentarios

```
void MainWindow::checkTable()
{
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < mNumeroDeFilas; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;

        for (int columna = 0; columna < mNumeroDeColumnas; columna++)
        {
            filas[fila] += tablero[fila][columna];
            columnas[fila] += tablero[columna][fila];
            diagonalLeft += tablero[columna][columna];
            diagonalRight += tablero[columna][2 - columna];

        }

        if (filas[fila] == 3 || columnas[fila] == 3 || diagonalLeft == 3 || diagonalRight == 3 ||
            filas[fila] == 15 || columnas[fila] == 15 || diagonalLeft == 15 || diagonalRight == 15)
        {
            showWinner();

        }
        else if (contador == mNumeroMaximoFichas)
        {
            juego = "No Win";
        }
    }

}
```

**MAL**

```
void MainWindow::checkTable()
{
    int filas [3];
    int columnas [3];
    int diagonalLeft;
    int diagonalRight;

    for (int fila = 0; fila < mNumeroDeFilas; fila++)
    {
        filas[fila] = 0;
        columnas[fila] = 0;
        diagonalLeft = 0;
        diagonalRight = 0;

        for (int columna = 0; columna < mNumeroDeColumnas; columna++)
        {
            filas[fila] += tablero[fila][columna];
            columnas[fila] += tablero[columna][fila];
            diagonalLeft += tablero[columna][columna];
            diagonalRight += tablero[columna][2 - columna];
            /*Aqui se va a comprobar a cuanto equivale cada fila y columna
            con los numeros que he ido asignando en setMatriz() y se va a sumar
            cada numero con la fila, columna y diagonales correspondientes
            como le indicamos que la X vale 1 y la O vale 5, si la suma
            de toda la fila/columna/diagonal es 3, significa que gana la X
            si la suma vale 15, habrán ganado las O.
            Si no vale ninguna de esas opciones, significará que no hay ganador*/
        }

        if (filas[fila] == 3 || columnas[fila] == 3 || diagonalLeft == 3 || diagonalRight == 3 ||
            filas[fila] == 15 || columnas[fila] == 15 || diagonalLeft == 15 || diagonalRight == 15)
        {
            showWinner();

        }
        else if (contador == mNumeroMaximoFichas)
        {
            juego = "No Win";
        }
    }

}
```

**BIEN**

## 3.4. Los comentarios dicen qué hace el código, no como se hace

Los comentarios donde dices paso a paso qué se está realizando no deberían existir. Un comentario se usa para explicar por qué de cierta decisión y algo que no sea evidente a simple vista

```
for (int columna = 0; columna < mNumeroDeColumnas; columna++)
{
    filas[fila] += tablero[fila][columna];
    columnas[fila] += tablero[columna][fila];
    diagonalLeft += tablero[columna][columna];
    diagonalRight += tablero[columna][2 - columna];
    /*Aqui se va a comprobar a cuanto equivale cada fila y columna
    con los numeros que he ido asignando en setMatriz() y se va a sumar
    cada numero con la fila, columna y diagonales correspondientes
    como le indicamos que la X vale 1 y la O vale 5, si la suma
    de toda la fila/columna/diagonal es 3, significa que gana la X
    si la suma vale 15, habrán ganado las O.
    Si no vale ninguna de esas opciones, significará que no hay ganador*/
}
```

**MAL**

```
for (int columna = 0; columna < mNumeroDeColumnas; columna++)
{
    filas[fila] += tablero[fila][columna];
    columnas[fila] += tablero[columna][fila];
    diagonalLeft += tablero[columna][columna];
    diagonalRight += tablero[columna][2 - columna];
    /*Se hace de esta forma debido a ser lo más simple posible
    con operaciones sencillas y lógica donde se emplean diversos
    elementos realizados en metodos anteriores*/

}
```

**BIEN**