

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ALUMNO:**

**ALDAVERA GALLAGA IVÁN**

**APPLICATION DEVELOPMENT FOR MOBILE DEVICES**

**PROFESOR:**

**ALEJANDRO SIGFRIDO CIFUENTES ÁLVAREZ**

**TÍTULO DEL REPORTE**

**PRIMER EXAMEN**

**FECHA DE REALIZACIÓN:**

**25 – JUNIO – 2020**

**VERSIÓN DEL REPORTE:**

**V1**

**ÍNDICE**

OBJETIVO………………………………………………………………1

CONCEPTOS…………………………………………………………..1

DESARROLLO…………………………………………………………1

LISTADO DE SOFTWARE………………….………………….2

LISTADO DE HARDWARE………………….…………………2

PRUEBAS………………………………………………………………3

CONCLUSIONES……………………………………….…………….6

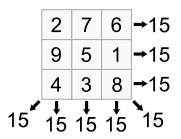
BIBLIOGRAFÍA………………………………………….…………….6

**OBJETIVO**

Desarrollar una aplicación, en la cual se introduzca un número impar y se muestre su cuadrado mágico, junto con las sumas de los números del cuadrado mágico al final de cada fila y columna.

**CONCEPTOS**

Un **cuadrado mágico** es una tabla de grado primario donde se dispone de una serie de números enteros en un cuadrado o matriz de forma tal que la suma de los números por columnas, filas y diagonales principales sea la misma. Usualmente los números empleados para rellenar las casillas son consecutivos, de 1 a n², siendo n el número de columnas y filas del cuadrado mágico.



**DESARROLLO**

Para el desarrollo de este ejercicio, primero se debe entender cuál es el algoritmo que puede generar un cuadrado mágico, el cuál, en el caso de Java vendría a ser el siguiente:

1. **void** generarCuadrado(**int** n) {
2. **int**[][] cuadradoMagico = **new** **int**[n][n];
4. **int** i = n / 2;
5. **int** j = n - 1;
7. **for** (**int** num = 1; num <= n\*n; ) {
8. **if** (i == -1 && j==n) {
9. j = n-2;
10. i = 0;
11. } **else** {
12. **if** (j == n) {
13. j = 0;
14. }
16. **if** (i < 0) {
17. i = n-1;
18. }
19. }
21. **if** (cuadradoMagico[i][j] != 0)  {
22. j -= 2;
23. i++;
24. **continue**;
25. } **else** {
26. cuadradoMagico[i][j] = num++;
27. }
29. j++;  i--;
30. }
31. }

El cual tiene su explicación en lo siguiente:

En cualquier cuadrado mágico, el primer número se almacena en la posición (n / 2, n-1). Deje que esta posición sea (i, j). El siguiente número se almacena en la posición (i-1, j + 1) donde podemos considerar cada fila y columna como una matriz circular, es decir, se envuelven. Seguido de esto se pueden cumplir tres condiciones en este ciclo:

La posición del siguiente número se calcula disminuyendo el número de fila del número anterior en 1, e incrementando el número de la columna del número anterior en 1. En cualquier momento, si la posición de la fila calculada se convierte en -1, se ajustará a n - 1. Del mismo modo, si la posición calculada de la columna se convierte en n, se ajustará a 0.

Si el cuadrado mágico ya contiene un número en la posición calculada, la posición calculada de la columna se reducirá en 2 y la posición calculada de la fila se incrementará en 1.

Si la posición de la fila calculada es -1 y la posición de la columna calculada es n, la nueva posición sería: (0, n-2).

Ahora bien, una vez generada la matriz que contenga el cuadrado mágico, el siguiente reto es representarla en una Activity mediante plantillas, para este caso se eligió crear una tabla que cambia de visibilidad según la matriz esté llena o no, y del número de filas y columnas que se necesiten para representar la matriz. Esto se hizo mediante el código XML del archivo activity\_main.xml y mediante código de la clase .java correspondiente.

También, si se intenta añadir un número par, la aplicación nos responderá diciendo que no es posible introducirlo.

**LISTADO DE SOFTWARE:**

* Java Development Kit 8: Necesario para el desarrollo de aplicaciones Android.
* Android Studio 4.0: IDE para el desarrollo de aplicaciones Android
* Windows 10: Sistema operativo sobre el que corren los anteriores programas.
* Android 10. Sistema Operativo sobre el que corre la aplicación

**LISTADO DE HARDWARE:**

* Laptop HP con GB RAM DDR4 y procesador Intel Core i3
* Dispositivo Móvil Motorola One con 4GB de RAM y procesador SoC Qualcomm Snapdragon 625
* Cable tipo C para la transferencia de datos.

**PRUEBAS**

**N = 1**.

|  |  |
| --- | --- |
|  |  |

**N = 3**

|  |  |
| --- | --- |
|  |  |

**N = 5**

|  |  |
| --- | --- |
|  |  |

**N = 7**

|  |  |
| --- | --- |
|  |  |

**N = 9**

|  |  |
| --- | --- |
|  |  |

**N = 11**

|  |  |
| --- | --- |
|  |  |

**N = 4 (VALOR PAR)**

|  |
| --- |
|  |

**CONCLUSIONES**

Para este ejercicio se tuvo que hacer un uso extensivo de las plantillas, más específicamente de la plantilla tipo Grid, así como de su adecuación a las matrices para poder representar datos. No hubo mucho problema en cuanto al algoritmo, ya que es uno de los algoritmos más estudiados, al tratarse de un algoritmo de tipo Greedy o voraz. Y al menos de manera personal, lo estudié en la amteria de Análisis de Algoritmos en sexto semestre, por lo cual no hubo más necesidad que revisar los apuntes e implemtarlo en el lenguaje de programación Java

**BIBLIOGRAFÍA**

* <https://es.wikipedia.org/wiki/Cuadrado_m%C3%A1gico>