**Informe Desafío 1 – Juan Pablo Avendaño y Freddy Castaño.**

**Análisis del problema.**

El desafío pide que se realice un algoritmo capaz de entregar al menos una configuración de candado que cumpla con la regla entregada, además de eso en cada configuración va a haber solo matrices cuadradas impares y el ángulo de rotación entre un estado y el siguiente va a ser de 90°.   
  
Teniendo muy presente las consideraciones iniciales nos resultó facil idear una solución a este problema, primero que todo notamos que rotar las matrices completamente con el fin de pasar de un estado a otro era totalmente innecesario, lo que se hizo fue desarrollar una función que devolviera una celda dada según el estado en que se encontrara la matriz, es decir, si se ingresa como celda deseada ‘[4,1]’ , es decir fila 4 y columna 1, y como estado ponemos el 3, la funcion devolverá el número que se encontraría en la celda dada si se hubiera rotado la matriz hasta el estado 3. Como la función que permitía rotar la matriz completamente era un requisito entonces decidimos implementar esta función con fines netamente de experiencia de usuario, es decir, solamente se implementó este requisito para mostrarle al usuario la matriz.

Continuando con la solución fue evidente para nosotros que en la regla iba a estar el tamaño minimo de cada una de las matrices que fueran a componer el candado, por lo tanto si en la regla para abrir el candado se especificaba la celda ‘[4,3]’, se tomaría el número mayor ‘n’ entre las dos posiciones de la celda y en el caso de que fuera par, entonces se crearía una matriz de orden n+1, si el número es impar, entonces se crearía una matriz de orden n, si la celda corresponde a la posicion ‘[1,1]’, entonces se creará una matriz de orden 3, también se hace la verificación de que la celda no corresponda al centro de la matriz.

En cuanto a la validación de la regla se refiere definimos que ésta será invalida si después de la posicion de la celda deseada se encuentran números diferentes a ‘1’, ‘0’ y ‘-1’, o bien si directamente no existen números. La cantidad de matrices que se va a crear para la configuración del candado será la cantidad de elementos almacenados en la estructura de datos donde se guardará la regla (en este caso usamos un arreglo lineal de enteros alojado en el heap).

A continuación, consideramos pertinente crear una estructura que nos permitiera almacenar las matrices creadas, la utilidad de este componente la verá más adelante.   
  
En cuanto a generación de la configuración se refiere nosotros consideramos un algoritmo que se encargara de encontrar una combinación válida entre dos matrices que cumpliera una regla dada (por regla nos referimos a que una celda sea mayor, menor o igual a otra). La rotación de ambas matrices hasta encontrar una combinación valida es una de las características de este algoritmo, si no se encuentra tal combinación simplemente retornará un valor false, si la combinación es válida entonces se retornará true y además de eso se editará un arreglo de enteros donde la posición 0 indica el estado de la matriz uno, y la posicion 1 indica el estado de la matriz dos, cabe recalcar que el algoritmo permite dejar una o ambas matrices estáticas, es decir que no van a rotar, esto es útil para la generación completa de la configuración.

Para concluir la solución del problema notamos que la primera comparación entre matrices que fueran a componer mi candado podría tener cualquier orden y estado de rotación, pero las siguientes comparaciones deberían tener una matriz estática, es decir, con estado invariable, y este estado sería el estado de la segunda matriz a comparar en la comparación anterior, a continuación, se da un ejemplo para que visualice mejor el concepto:

Tenemos una estructura de 4 matrices con una regla general dada, para la primera combinación se obtuvo un resultado donde la primera matriz debía tener el estado 2 y la segunda el estado neutral para cumplir el primer criterio de la regla. Seguido a eso debe comparar la segunda matriz con la tercera, pero en este caso la segunda matriz debe conservar el mismo estado que se obtuvo en la primera comparación, y así mismo pasa con la tercera y cuarta comparación. Si este aspecto no es tomado en cuenta el resultado final será con toda seguridad erróneo.

Finalmente, se recorre toda la estructura de matrices haciendo las comparaciones necesarias entre pares de matrices, cuando se finalice la comparación entre dos matrices, la segunda de ellas se pondrá estática, es decir no se alterará su orden o estado de rotación, esto con el fin de que las matrices sigan cumpliendo los criterios de comparación que se cumplieron en las comparaciones anteriores.

Finalmente se creará una matriz donde se van a almacenar los resultados obtenidos, la matriz resultado tendrá ‘n’ filas, donde n es el número de matrices en la configuración del candado, y la matriz tendrá 2 columnas. En la primera columna se ubica el orden de la matriz (matriz que compone el candado, no la matriz resultado), y en la segunda irá el estado de rotación de la misma.

Y, por último, se le despliega al usuario los resultados que se obtuvieron de manera que pueda visualizar cada matriz.

**Esquema.**

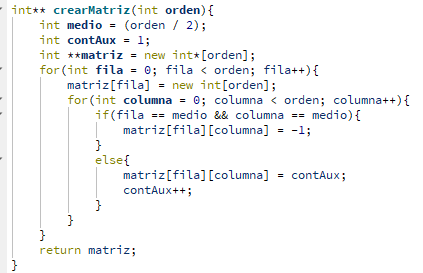
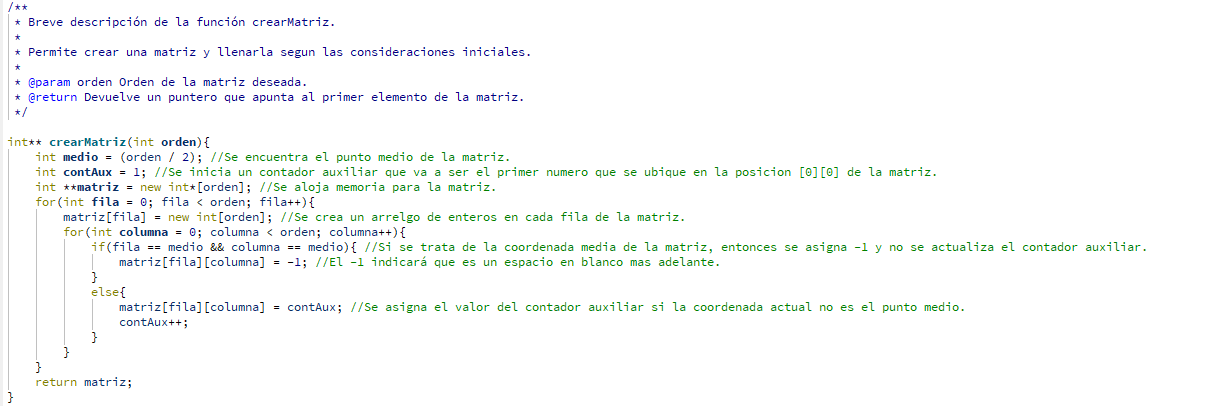
A continuación, se presentará un esquema de los dos módulos principales de la solución en donde se resuelve el problema, estos son: **Estructura** y **Candado**.

**Algoritmos implementados.**

En el desarrollo del codigo implementamos los dos módulos principales de los cuales fueron explicadas las tareas más representativas como pudo ver en el esquema anterior, en esta sección ahondaremos un poco más en la implementación de todos los módulos, repasando cada una de las funciones que lo componen:

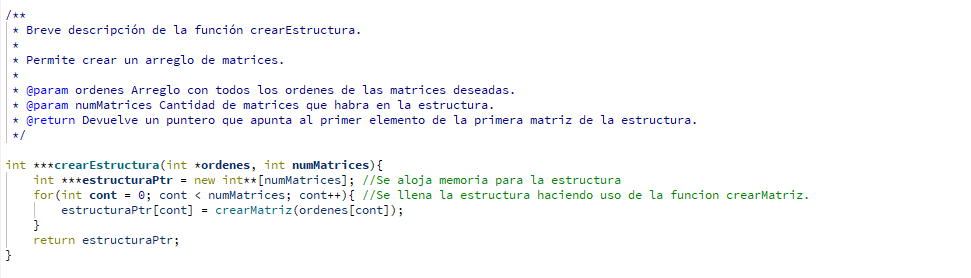
**Módulo Estructuras:**

* **Función crearMatriz:**



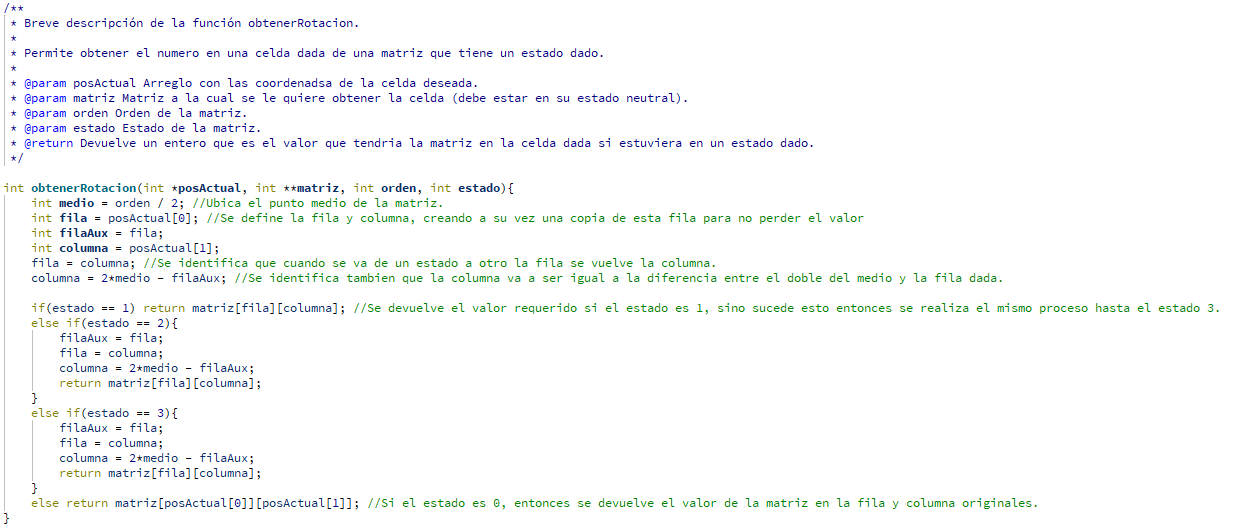
Esta función permite crear la estructura cuadrada que se menciona en las consideraciones iniciales del problema, recibe solamente el orden de la estructura que se desee crear, incluye dos ciclos ideados para recorrer completamente la estructura y llenarla de números, si la posicion corresponde a la del medio se le asigna el numero -1, que después será interpretado como un espacio en blanco.

* **Función crearEstructura:**



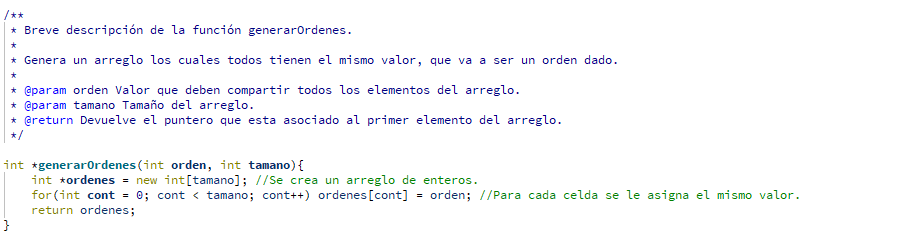
Esta función permite crear un arreglo de matrices al cual llamaremos estructura, en el codigo se utiliza esta estructura para poder tener acceso a cualquier celda de las matrices que compongan vayan a componer mi configuración de candado, se entrega mediante un puntero la posicion de memoria de esta estructura (el primer elemento de la primera matriz).

* **Función obtenerRotacion:**



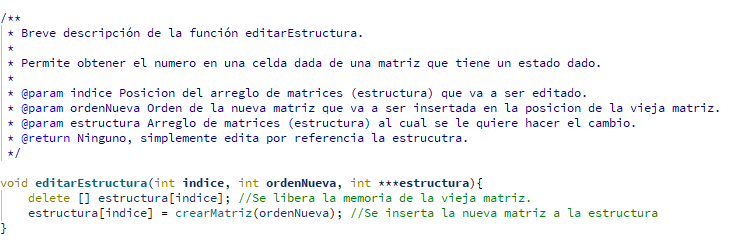
Esta funcion permite simular la rotación de una matriz y obtener el número que estaría en la celda especificada si el estado de la matriz fuera el que se ingresó en la funcion, nos dimos cuenta de que rotar la matriz completamente para cada comparación era un esfuerzo computacional innecesario, por lo tanto a base de tanteo dedujimos una relacion para hallar aquel valor de la matriz en la celda especificada con el estado especificado.

* **Función generarOrdenes:**



Esta funcion simplemente genera un arreglo de números que tiene un número fijo en cada elemento de él, en este caso el orden especificado, será de mucha utilidad en otras funciones ya que permitirá obtener los órdenes de las matrices que intervienen en la generación del candado, y además de eso se podrá editar este orden a conveniencia para encontrar una combinación.

* **Función editarEstructura:**



La funcion editarEstructura tiene como utilidad principal borrar completamente una matriz de una estructura de matrices, y seguido a eso se crea una nueva de un orden especificado, es útil cuando se estan comparando dos matrices y una de ellas impide que el criterio de comparación se cumpla, entonces se aumenta el orden de la matriz de una de ellas (se debe borrar la matriz anterior e insertar una nueva) y se compara otra vez hasta encontrar la combinación valida.