**Desafio#1 Informática 2. Álvaro Gómez, Juan David Jiménez.**

**Modulo 1.**

#### **A).** Problema:

El problema consiste en desarrollar un módulo que permita crear estructuras de datos de tamaño variable, con espacio en blanco en el centro, como se indica en el código y en las especificaciones del main proporcionado.

#### Consideraciones Iniciales:

**Creación de la Estructura M:** Se debe implementar una función **crearEstructuraM(int tamano)** que reciba como parámetro el tamaño deseado para la estructura de datos y devuelva una instancia de la estructura **EstructuraM** con las características especificadas.

**Tamaño Impar:** Se debe asegurar que el tamaño de la estructura **EstructuraM** sea impar, aumentándolo en 1 si es par, para cumplir con la condición de tener espacio en blanco en el centro.

**Memoria Dinámica:** La matriz de enteros dentro de la estructura **EstructuraM** se debe crear utilizando asignación de memoria dinámica.

**Rellenar la Matriz:** Se debe llenar la matriz con valores -1 (espacios en blanco) en todas las posiciones excepto en el centro, donde se llenará con números del 1 al tamaño^2 excluyendo el centro.

#### Alternativa de Solución Propuesta:

Para abordar este problema, se propone la siguiente alternativa de solución:

**Función crearEstructuraM(int tamano):**

Verificar si el tamaño es par y aumentarlo en 1 si es necesario para asegurar que sea impar.

Crear la estructura **EstructuraM** y asignar memoria dinámica para la matriz de enteros.

Llenar la matriz con valores -1 en todas las posiciones.

Rellenar el centro de la matriz con números del 1 al tamaño^2 excluyendo el centro.

**Consideraciones de Eficiencia:**

Utilizar la asignación de memoria dinámica de manera eficiente para evitar problemas de fragmentación de memoria.

Implementar un algoritmo eficiente para rellenar la matriz y asegurar que cumpla con las condiciones requeridas.

**Gestión de Memoria:**

Implementar la función **liberarEstructuraM(EstructuraM &estructura)** para liberar la memoria asignada a la estructura **EstructuraM** cuando ya no sea necesaria.

**Modularidad y Reutilización:**

Mantener la separación de archivos (estructuraM.h y estructuraM.cpp) para una estructura modular y facilitar la reutilización del código.

Proporcionar una función **imprimirEstructuraM(EstructuraM &estructura)** para mostrar por pantalla la estructura creada.

Con esta alternativa de solución, se busca crear un módulo eficiente y modular que permita generar estructuras de datos de tamaño variable, con espacio en blanco en el centro, cumpliendo con las especificaciones indicadas en el problema. Además, se presta atención a la gestión de memoria y se garantiza la modularidad y reutilización del código para un desarrollo más organizado y mantenible.

**B).Aqui me falta desarrolar el esquema**

**C).** **Algoritmos implementados:**

**1.** crearEstructuraM

**2.** liberarEstructuraM

**3.** imprimirEstructuraM

**C.)**

**E).**