**Práctica 1: Uso de Arreglos Bidimensionales**

# Introducción

El objetivo de esta práctica es implementar un programa en C++ que maneje un arreglo bidimensional n x n (con n ≤ 10), lo llene con valores aleatorios del 1 al 100, muestre la matriz, calcule la suma total de todos los elementos, cuente la cantidad de números pares, impares y divisibles entre 7, y finalmente obtenga y muestre la matriz transpuesta.

# Desarrollo

**Análisis del problema.**

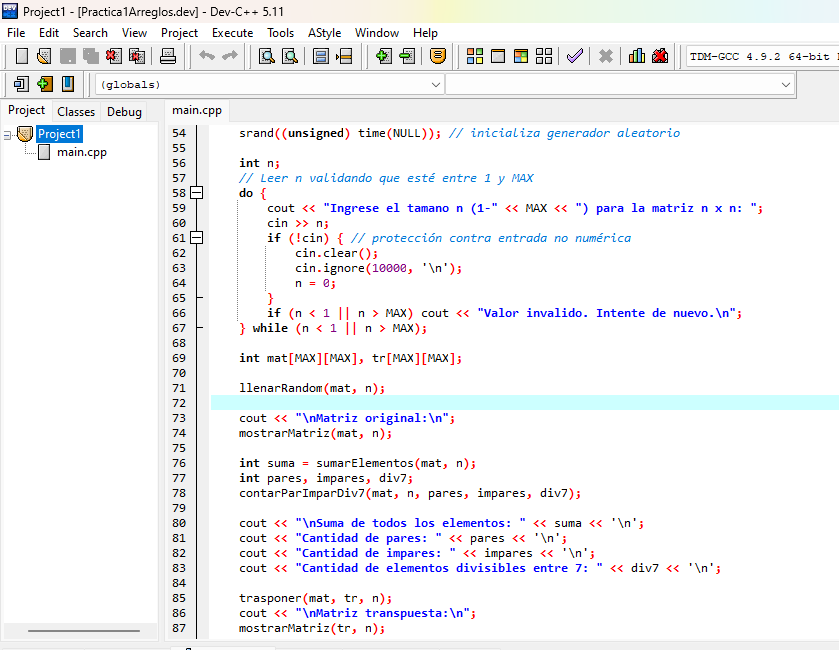
El problema consiste en manipular una matriz cuadrada de tamaño n x n donde n es definido por el usuario (1 ≤ n ≤ 10). Se deben cumplir los siguientes requisitos:  
- Llenar la matriz con valores enteros aleatorios entre 1 y 100.  
- Mostrar la matriz en pantalla.  
- Calcular la suma de todos los elementos.  
- Contar la cantidad de elementos pares, impares y divisibles entre 7.  
- Generar y mostrar la matriz transpuesta.

**Solución propuesta**.

Para cumplir con los requerimientos, el programa se modularizó en funciones:  
- llenarRandom(): llena la matriz con números aleatorios.  
- mostrarMatriz(): despliega la matriz alineada en pantalla.  
- sumarElementos(): suma todos los valores.  
- contarParImparDiv7(): cuenta pares, impares y divisibles entre 7.  
- trasponer(): genera la transpuesta.  
El programa principal solicita el valor de n, valida su rango y ejecuta las funciones.

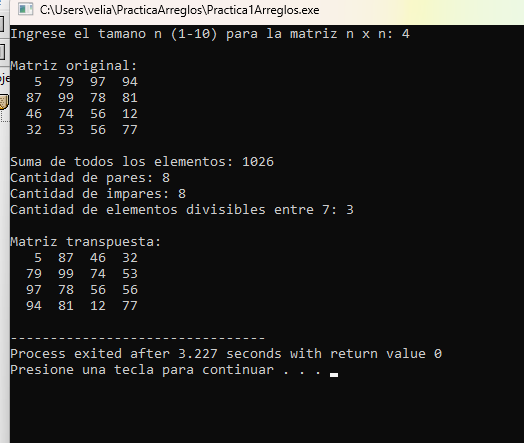
# Código Fuente

#include <iostream>  
#include <cstdlib>  
#include <ctime>  
#include <iomanip>  
using namespace std;  
  
const int MAX = 10;  
  
void llenarRandom(int a[][MAX], int n) {  
 for (int i = 0; i < n; ++i)  
 for (int j = 0; j < n; ++j)  
 a[i][j] = rand() % 100 + 1;  
}  
  
void mostrarMatriz(int a[][MAX], int n) {  
 for (int i = 0; i < n; ++i) {  
 for (int j = 0; j < n; ++j)  
 cout << setw(4) << a[i][j];  
 cout << '\n';  
 }  
}  
  
int sumarElementos(int a[][MAX], int n) {  
 int suma = 0;  
 for (int i = 0; i < n; ++i)  
 for (int j = 0; j < n; ++j)  
 suma += a[i][j];  
 return suma;  
}  
  
void contarParImparDiv7(int a[][MAX], int n, int &pares, int &impares, int &div7) {  
 pares = impares = div7 = 0;  
 for (int i = 0; i < n; ++i)  
 for (int j = 0; j < n; ++j) {  
 if (a[i][j] % 2 == 0) ++pares;  
 else ++impares;  
 if (a[i][j] % 7 == 0) ++div7;  
 }  
}  
  
void trasponer(int a[][MAX], int t[][MAX], int n) {  
 for (int i = 0; i < n; ++i)  
 for (int j = 0; j < n; ++j)  
 t[j][i] = a[i][j];  
}  
  
int main() {  
 srand((unsigned) time(NULL));  
 int n;  
 do {  
 cout << "Ingrese el tamano n (1-10): ";  
 cin >> n;  
 } while (n < 1 || n > 10);  
  
 int mat[MAX][MAX], tr[MAX][MAX];  
 llenarRandom(mat, n);  
 cout << "\nMatriz original:\n";  
 mostrarMatriz(mat, n);  
  
 int suma = sumarElementos(mat, n);  
 int pares, impares, div7;  
 contarParImparDiv7(mat, n, pares, impares, div7);  
  
 cout << "\nSuma: " << suma << "\n";  
 cout << "Pares: " << pares << "\n";  
 cout << "Impares: " << impares << "\n";  
 cout << "Divisibles entre 7: " << div7 << "\n";  
  
 trasponer(mat, tr, n);  
 cout << "\nMatriz transpuesta:\n";  
 mostrarMatriz(tr, n);  
 return 0;  
}



# Evidencias de Ejecución

Ejemplo con n=4:



## Tiempos Estimados vs Reales

|  |  |  |
| --- | --- | --- |
| Actividad | Tiempo Estimado | Tiempo Real |
| Análisis del problema | 30 min | 30 min |
| Implementación del código | 60 min | 40 min |
| Pruebas y depuración | 30 min | 30 min |
| Documentación y reporte | 60 min | 30 min |

## Defectos Encontrados y Correcciones

- Error: ingreso de valores fuera de rango para n.  
 Corrección: se añadió un ciclo do-while que valida que 1 ≤ n ≤ 10.  
  
- Error: posible entrada no numérica que detenía el programa.  
 Corrección: se agregó validación con cin.clear() y cin.ignore() en una versión avanzada.  
  
- Error: alineación desigual al mostrar la matriz.  
 Corrección: se usó setw(4) de la librería iomanip para mejorar el formato.

## Evidencia del Repositorio GitHub

El código y la documentación fueron subidos al repositorio GitHub en la carpeta:  
/practicas/practica1/  
  
Enlace al repositorio: [https://github.com/ZavalaVelia/practica1.git]  
Se incluyen los archivos main.cpp, README.md e Informe\_Practica1.docx.

# Conclusiones

La práctica permitió aplicar el manejo de arreglos bidimensionales en C++ y el uso de funciones para modularizar el código. Se comprobó que es posible realizar operaciones como suma, conteo de elementos bajo ciertas condiciones y la obtención de la transpuesta de manera eficiente. El ejercicio ayudó a reforzar conceptos básicos de programación estructurada y el uso de la librería estándar de C++.