Tipos de Dato Primitivo y Tipos de Datos Estructurados

Actividad de Aprendizaje 01

***Alumno***: Mariscal Rodríguez Omar Jesús

***Código***: 220858478

***Profesor:*** Gutiérrez Hernández Alfredo

***Fecha***: 17 de Agosto de 2025

***Centro Universitario de Ciencias Exactas e Ingenierías***

***Materia***: Estructuras de Datos

***Clave:*** V0731 ***Sección***: D02



Contenido

[Test de Autoevaluación 3](#_Toc206178495)

[Introducción y Abordaje del Problema 4](#_Toc206178496)

[1.- Fase de Investigación 4](#_Toc206178497)

[2.- Planteamiento General 5](#_Toc206178498)

[3.- Programación 5](#_Toc206178499)

[Código Fuente 7](#_Toc206178500)

[Carpeta include 7](#_Toc206178501)

[*Carpeta src* 10](#_Toc206178502)

[Ejecución del Programa 21](#_Toc206178503)

[Conclusiones. 24](#_Toc206178504)

# Test de Autoevaluación

|  |  |  |  |
| --- | --- | --- | --- |
| ***Autoevaluación*** | | | |
| ***Concepto*** | ***Sí*** | ***No*** | ***Acumulado*** |
| Bajé el trabajo de internet o alguien me lo pasó (aunque sea de forma parcial) | ***-100 pts*** | ***0 pts*** | ***0*** |
| Incluí el código fuente ***en formato de texto (sólo si funciona cumpliendo todos los requerimientos)*** | ***+25 pts*** | ***0 pts*** | ***25*** |
| Incluí las ***impresiones de pantalla (sólo si funciona cumpliendo todos los requerimientos)*** | ***+25 pts*** | ***0 pts*** | ***25*** |
| Incluí una ***portada*** que identifica mi trabajo (nombre, código, materia, fecha, título) | ***+25 pts*** | ***0 pts*** | ***25*** |
| Incluí una ***descripción y conclusiones*** de mi trabajo | ***+25 pts*** | ***0 pts*** | ***25*** |
| ***Suma:*** | | | ***100*** |

# Introducción y Abordaje del Problema

Esta actividad tuvo su principal propósito en ser un repaso de todo lo que hemos cursado de la carrera hasta ahora, desde los fundamentos de programación hasta la Programación Orientada a Objetos, por lo fue aplicar conocimiento que ya se tenía en práctica.

Dividida en dos partes; la impresión de la tabla de datos obtenida del mismo lenguaje y el uso del dato compuesto en forma de matrices, el proceso de realización de esta actividad siguió este flujo:

***1.- Fase de Investigación***

***2.- Planteamiento General***

***3.- Programación***

## 1.- Fase de Investigación

**Aleatoriedad de Números Reales**

La actividad requería apartados de los cuales tuve que investigar, como por ejemplo, ***números reales generados de manera aleatoria y que no se repitan en cada ejecución,*** presentar ***información en tablas.***

Para la ***aleatoriedad*** investigué en el link de la documentación de la librería random que se nos dio: <https://cplusplus.com/reference/random/> , de ahí, aprendí cómo funciona su generación de números pseudoaleatorios, y conocí que necesitan tanto un ***generador*** como una ***distribución.***

Al necesitar sólo números aleatorios, utilicé el generador de números aleatorios por defecto: <https://cplusplus.com/reference/random/default_random_engine/> y una distribución real y uniforme <https://cplusplus.com/reference/random/uniform_real_distribution/> para hacerlos lo más random posible.

La librería trabaja con una ***seed,*** que es un número sobre el cuál se basan los números aleatorios, una misma seed produce los mismos números aleatorios en ese orden; investigando por la documentación, existe una clase llamada “random\_device” que genera números no secuenciales, aleatorios con una distribución uniforme, pero no se puede limitar, por lo que la utilicé como una seed aleatoria: <https://cplusplus.com/reference/random/random_device/>

**Estética del Output**

Solucionado el tema de la aleatoriedad, para presentar la información de una manera más visual (sobre todo para la tabla de datos primitivos), podría haberlo hecho manual (como se hizo con el menú principal por ser una tabla más chica), pero investigando conocí sobre la librería iomanip: <https://cplusplus.com/reference/iomanip/?kw=iomanip> que permite manipular de manera más interesante los string’s.

Usando mucho la función setw: <https://cplusplus.com/reference/iomanip/setw/> para establecer el ancho de las columnas en la tabla; además de ello, utilicé sstrings: <https://cplusplus.com/reference/sstream/?kw=sstream> para su objeto osstringstream <https://cplusplus.com/reference/sstream/ostringstream/> para poder operar y transformarlo string’s de manera más libre, esto lo pensé más que nada para el método to\_string() que tendría que tener los objetos de tipo matriz.

Una vez recopilada esta información, tenía el conocimiento necesario para llevar a cabo la actividad.

## 2.- Planteamiento General

El paradigma de programación era orientado a objetos, por lo que primero se aplicó un proceso mental de ***abstracción*** para determinar los objetos que intervienen, y de estos identifiqué los siguientes:

1.- Tabla de Tipos de Datos (DataTypeTable)

2.- Matrices (Matrix)

3- Menu (Menu)

DataTypeTable será una clase con un único método estático (ya que no necesitamos instanciar directamente la clase) que imprima la tabla de tipos de datos.

Matrix es el tipo de dato compuesto que debe cumplir varios objetivos: primero inicializarse en 0’s, después rellenarse con valores reales aleatorios en el rango establecido, sobrecargar sus operadores de suma y multiplicación, finalmente, un método to\_string() para su impresión en formato de tabla.

Menu imprimiría las opciones para acceder a las dos funciones que pide el programa, y cada uno de estas sería un método privado.

Al planear estos objetos, se imagina un caso donde el usuario ingrese una cantidad errónea, por ejemplo en el menú, o en la matriz ingresar un tamaño mayor a 10; para ello y para mantener el principio de Principio de Responsabilidad Única lo mejor posible, surgió una clase más:

4.- Utilidades (Utilities)

Esta recopilará funciones útiles varias, como en la entrada de datos, surgió la idea de un método estático llamado inputPositiveIntegerLimit que haría que el usuario ingrese sí o sí, un número entre dos cantidades.

Una vez con estas clases planteadas, se procedió con la programación.

## 3.- Programación

Con el planteamiento y la investigación ya hechas, la programación se agilizó bastante, se dividió en modularización con una estructura donde en una carpeta include van las definiciones de las clases en .h y en el src va el main y las implementaciones.

De ahí, lo interesante sería abordar el algoritmo de multiplicación de matrices, donde recurrí a un ciclo triple anidado, dado que se dan multiplicaciones, donde n es el tamaño de la matriz, y de ahí es el recorrer las matrices en orden.

# Código Fuente

## Carpeta include

**DataTypeTable.h**

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: DataTypeTable.h\*/

#pragma once

namespace Tables {

// Clase con un método estático, no necesitamos instanciarlo y que ocupe

// memoria.

class DataTypeTable {

 public:

  /// @brief Impresión de la tabla de valores

  static void showDataTypeTable();

};

}  // namespace Tables

**Matrix.h**

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: Matrix.h\*/

#pragma once

#include <iostream>

namespace CompositeData {

/// @brief Clase de matrices y operaciones definidas entre ellas.

class Matrix {

 private:

  int size;

  float values[10][10];

 public:

  /// @brief Constructor para incializar Size

  /// @param  Size tamaño que se tomará en cuenta para las operaciones y

  /// rellenado

  Matrix(int);

  /// @brief Limpiar la matriz y dejarla llena de ceros

  void zeros();

  /// @brief Llenar el size con valores reales aleatorios en el rango

  /// establecido

  void randomizer();

  /// @brief Modificar el Size de lo que se tomará en cuenta para las

  /// operaciones

  /// @param  Size Nuevo tamaño para tomarlo en cuenta

  void setSize(int);

  /// @brief Representar la matriz en un string

  /// @return String con una representación de los valores

  std::string to\_string();                     // Llevarlo a un formato legible

  Matrix operator+(const Matrix& otro) const;  // Sobrecarga del operador suma

  Matrix operator\*(

      const Matrix& otro) const;  // Sobrecarga del operador multiplicación

};

}  // namespace CompositeData

**Interfaces.h**

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: Interfaces.h\*/

#pragma once

#include <iostream>

// Namespace de Utilidades varias relacionadas al input

namespace Utilities {

/// @brief Utilidades Varias

class InputUtilities {

 public:

  /// @brief Pedir al usuario un entero positivo en un rango determinado. Se

  /// ejecuta un bucle hasta que se ingrese una cantidad válida

  /// @param  label Texto que se imprimira para pedir la información

  /// @param  lowerLimit Número mínimo que el usuario puede ingresar

  /// @param  upperLimit Numero máxmo que el usuario puede ingresar

  /// @return Seleccion de la selección del usuario.

  static int inputPositiveIntegerLimit(std::string, int, int);

};

}  // namespace Utilities

namespace Menus {

/// @brief Menu Principal y sus distintas bifurcaciones a otros

/// Submenús/Funciones

class Menu {

 public:

  /// @brief Menu Principal desde el que se acceden los demás submenús/funciones

  void mainMenu();

 private:

  /// @brief Mostrar la tabla de Datos

  void dataTypes();

  /// @brief Mostrar el ejemplo de Datos Compuestos (Matrices)

  void compositeData();

  /// @brief Mensaje de despedida

  void exit();

};

}  // namespace Menus

## Carpeta src

DataTypeTable.cpp

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: DataTypeTable.cpp\*/

#include "DataTypeTable.h"  //Incluir la definición de la clase

#include <iomanip>  //Librería para el control de entrada y salida de datos. Usada para darle el formato a la tabla

#include <iostream>  //Librería de Entrada y Salida de Datos Estándar

#include <limits>  //Librería para extreaer del lenguaje el tamaño de los tipos de datos.

// using std's para agilizar funciones recurrentes en el archivo

using std::cout;

using std::left;

using std::numeric\_limits;

using std::setw;

using std::string;

// Función  para imprimir la tabla de valores con funciones del lenguaje

void Tables::DataTypeTable::showDataTypeTable() {

  // Medidas para el ancho de cada apartado

  const int widthType = 30;

  const int widthBits = 10;

  const int widthMin = 15;

  const int widthMax = 15;

  // Función para imprimir línea horizontal

  auto linea = [&]() {

    cout << "+" << string(widthType, '-') << "+" << string(widthBits, '-')

         << "+" << string(widthMin, '-') << "+" << string(widthMax, '-')

         << "+\n";

  };

  // Cabeceraas

  linea();

  cout << "|" << setw(widthType) << left << "Tipo de dato"

       << "|" << setw(widthBits) << left << "Bits"

       << "|" << setw(widthMin) << left << "Valor minimo"

       << "|" << setw(widthMax) << left << "Valor maximo" << "|\n";

  linea();

  // Filas de los distintos tipos de datos

  // Usamos setw para establecer los anchos de cada apartado

  cout << "|" << setw(widthType) << left << "Caracter con signo"

       << "|" << setw(widthBits) << left << sizeof(char) \* 8 << "|"

       << setw(widthMin) << left << int(numeric\_limits<char>::min()) << "|"

       << setw(widthMax) << left << int(numeric\_limits<char>::max()) << "|\n";

  cout << "|" << setw(widthType) << left << "Caracter sin signo"

       << "|" << setw(widthBits) << left << sizeof(unsigned char) \* 8 << "|"

       << setw(widthMin) << left << int(numeric\_limits<unsigned char>::min())

       << "|" << setw(widthMax) << left

       << int(numeric\_limits<unsigned char>::max()) << "|\n";

  cout << "|" << setw(widthType) << left << "Entero corto con signo"

       << "|" << setw(widthBits) << left << sizeof(int) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<int>::min() << "|"

       << setw(widthMax) << left << numeric\_limits<int>::max() << "|\n";

  cout << "|" << setw(widthType) << left << "Entero corto sin signo"

       << "|" << setw(widthBits) << left << sizeof(unsigned int) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<unsigned int>::min() << "|"

       << setw(widthMax) << left << numeric\_limits<unsigned int>::max()

       << "|\n";

  cout << "|" << setw(widthType) << left << "Entero largo con signo"

       << "|" << setw(widthBits) << left << sizeof(long int) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<long int>::min() << "|"

       << setw(widthMax) << left << numeric\_limits<long int>::max() << "|\n";

  cout << "|" << setw(widthType) << left << "Entero largo sin signo"

       << "|" << setw(widthBits) << left << sizeof(unsigned long int) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<unsigned long int>::min()

       << "|" << setw(widthMax) << left

       << numeric\_limits<unsigned long int>::max() << "|\n";

  cout << "|" << setw(widthType) << left << "Real de precision simple"

       << "|" << setw(widthBits) << left << sizeof(float) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<float>::min() << "|"

       << setw(widthMax) << left << numeric\_limits<float>::max() << "|\n";

  cout << "|" << setw(widthType) << left << "Real de doble precision"

       << "|" << setw(widthBits) << left << sizeof(double) \* 8 << "|"

       << setw(widthMin) << left << numeric\_limits<double>::min() << "|"

       << setw(widthMax) << left << numeric\_limits<double>::max() << "|\n";

  // linea final

  linea();

}

Matrix.cpp

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: Matrix.cpp\*/

#include "Matrix.h"  //Definición de la clase

#include <iomanip>   //Crear tabla para imprimir la Matrix

#include <iostream>  //Entrada y Salida Estándar

#include <random>    //Biblioteca para dar reales aleatorios

#include <sstream>   //Para facilitar el método to\_string

using CompositeData::Matrix;

// Método Constructor

Matrix::Matrix(int size) {

  this->size = size;

}

// Setter del Size de la matriz

void Matrix::setSize(int size) {

  this->size = size;

}

void Matrix::zeros() {  // Inicializar las cantidades en 0 como buena practica

  for (int i = 0; i < 10; i++)

    for (int j = 0; j < 10; j++)

      this->values[i][j] = 0;

}

void Matrix::randomizer() {

  // Generar una serie de números pseudoaleatorios para llenar la matriz

  std::random\_device

      seed;  // Un generador de semilla aleatoria en cualquier ejecución

  std::default\_random\_engine engine(

      seed());  // Usamos el motor aleatorio por defecto, el más simple, con la

                // semilla generada

  std::uniform\_real\_distribution<float> distribution(

      -100.00, 100.00);  // Crear el objeto de una distribución normal de reales

                         // en el rango de -100.00 a 100.00

  // Ciclo anidado para llenar la matriz con los números pesudoaleatorios

  for (int i = 0; i < size; i++)

    for (int j = 0; j < size; j++)

      values[i][j] = (round(distribution(engine) \* 100) / 100);

}

std::string Matrix::to\_string() {

  std::ostringstream oss;

  const int width = 10;  // Anchura de cada celda

  // Creación de una función lambda para imprimir una línea

  auto linea = [&]() {

    oss << "+";

    for (int i = 0; i < size; i++)

      oss << std::string(width, '-') << "+";

    oss << "\n";

  };

  linea();

  // Imprimir los valores valores de manera ordenada

  for (int i = 0; i < size; i++) {

    for (int j = 0; j < size; j++) {

      oss << "|" << std::setw(width) << std::left << values[i][j];

    }

    oss << "|" << "\n";

    linea();

  }

  return oss.str();  // Retornar el string

}

// Operador de Suma

Matrix Matrix::operator+(const Matrix& otro) const {

  Matrix sum(size);

  for (int i = 0; i < size; i++)

    for (int j = 0; j < size; j++)

      sum.values[i][j] = this->values[i][j] + otro.values[i][j];

  return sum;

}

// Operador de Multiplicación

Matrix Matrix::operator\*(const Matrix& otro) const {

  Matrix times(size);

  // Recorrer ambas matrices en un determinado orden siguiendo el algoritmo de

  // multiplicación.

  for (int i = 0; i < size; i++)

    for (int j = 0; j < size; j++)

      for (int k = 0; k < size; k++)

        times.values[i][j] += (this->values[i][k] \* otro.values[k][j]);

  return times;

}

Interfaces.cpp

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: Interfaces.cpp\*/

#include "Interfaces.h"     //Importación del Menú y las Utilidades

#include "DataTypeTable.h"  //Imporatación de la clase de tabla de valores

#include "Matrix.h"         //Importación de la clase de matrices

#include <iostream>  //Entrada y Salida Estándar

#include <sstream>   //Para manipular strings con mayor facilidad

// std's frecuentes

using std::cin;

using std::cout;

using std::endl;

namespace Utilities {

int InputUtilities::inputPositiveIntegerLimit(std::string label,

                                              int lowerLimit,

                                              int upperLimit) {

  int answer;

  // Ciclo PostPrueba para obtener un entero en un rango determinado

  do {

    system("CLS");

    cout << label;  // Impresión de un mensaje

    cin >> answer;

    // Evaluación de la respuesta

    if (answer < lowerLimit || answer > upperLimit) {

      system("CLS");

      cout << "Se ha ingresado un numero fuera de lo establecido." << endl;

      cout << "Por favor, intentelo nuevamente." << endl << endl;

      system("PAUSE");

    }

  } while (answer < lowerLimit || answer > upperLimit);

  return answer;

}

}  // namespace Utilities

namespace Menus {

// Menú Principal del Programa

void Menu::mainMenu() {

  system("CLS");

  // El menú principal es un string acompañado de ostringstream para poder

  // llevarlo facilmente a las funciones de utilidades.

  std::ostringstream oss;

  oss << "+------------------------------------------------+\n";

  oss << "|                 Menu Principal                 |\n";

  oss << "+------------------------------------------------+\n";

  oss << "| 1. Mostrar Tabla de Tipos de Datos             |\n";

  oss << "| 2. Ejemplo del uso de Tipo de Dato Compuesto   |\n";

  oss << "| 3. Salir                                       |\n";

  oss << "+------------------------------------------------+\n";

  oss << "Ingrese una opcion: ";

  int selection = Utilities::InputUtilities::inputPositiveIntegerLimit(

      oss.str(), 1, 3);  // Elección del Usuario

  // Redirigir a la opción correspondiente

  switch (selection) {

    case 1:

      dataTypes();

      break;

    case 2:

      compositeData();

      break;

    case 3:

      exit();

      break;

      // No hay necesidad de un case default; esto por las utilidades que no

      // permiten avanzar si no se ingresa un número correcto.

  }

}

// Mostrar la tabla de tipos de datos

void Menu::dataTypes() {

  system("CLS");

  cout << "La siguiente tabla muestra la informacion de distintos tipos de "

          "datos obtenidos directamente del lenguaje: "

       << endl;

  Tables::DataTypeTable::showDataTypeTable();  // Uso del método estático (sin

                                               // instancia requerida)

  system("PAUSE");

  mainMenu();  // Regresamos al menú principal

}

// Menú de Datos Compuestos

void Menu::compositeData() {

  system("CLS");

  // Pedir el Size de la matriz

  int size = Utilities::InputUtilities::inputPositiveIntegerLimit(

      "Ingrese un valor entre 3 y 10 para determinar la capacidad de la "

      "matriz: ",

      3, 10);

  // Instanciar las matrices A y B

  CompositeData::Matrix matrixA(size);

  CompositeData::Matrix matrixB(size);

  // Inicializarlas en 0's

  matrixA.zeros();

  matrixB.zeros();

  // Llenarlas con números aleatorios reales en el rango determinado

  matrixA.randomizer();

  matrixB.randomizer();

  // MatrizC y MatrizD obtenidas de las operaciones sobrecargadas

  CompositeData::Matrix matrixC = matrixA \* matrixB;

  CompositeData::Matrix matrixD = matrixA + matrixB;

  // Impresión de cada matriz con el método to\_string()

  cout << "Matriz A: " << endl;

  cout << matrixA.to\_string() << endl;

  cout << "Matriz B: " << endl;

  cout << matrixB.to\_string() << endl;

  cout << "Matriz C, Multiplicacion AxB" << endl;

  cout << matrixC.to\_string() << endl;

  cout << "Matriz D, Suma de A+B" << endl;

  cout << matrixD.to\_string() << endl;

  system("PAUSE");

  // Regresamos al menú principal

  mainMenu();

}

// Mensaje de Exit

void Menu::exit() {

  system("CLS");

  // Despedida

  cout << "Saliendo del Programa...\n" << endl;

  cout << "Programa Hecho por: Mariscal Rodriguez Omar Jesus" << endl;

  cout << "Materia: Estructuras de Datos" << endl;

  cout << "Profesor: Gutierrez Hernandez Alfredo.\n" << endl;

  cout << "Tenga un Lindo Dia :)" << endl;

}

}  // namespace Menus

main.cpp

/\*Nombre: Mariscal Rodríguez Omar Jesús

Fecha de Entrega: 17 de Agosto de 2025

Problema: Actividad de Aprendizaje 01: Tipos de Dato Primitivo y Tipos de Datos

Compuestos Archivo: main.cpp\*/

#include "Interfaces.h"

int main() {

  Menus::Menu menu;  // Instancia del Menú Principal

  menu.mainMenu();  // El main sólo necesita correr el menú principal. La lógica

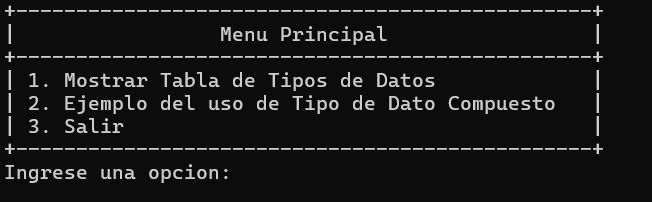
                    // se ejecuta detrás.

  return 0;

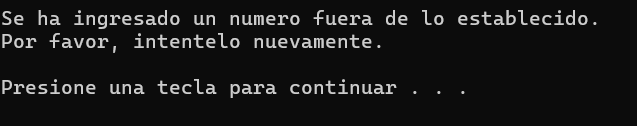
}

# Ejecución del Programa

Empezamos ejecutando el programa, lo que libera el menú principal:

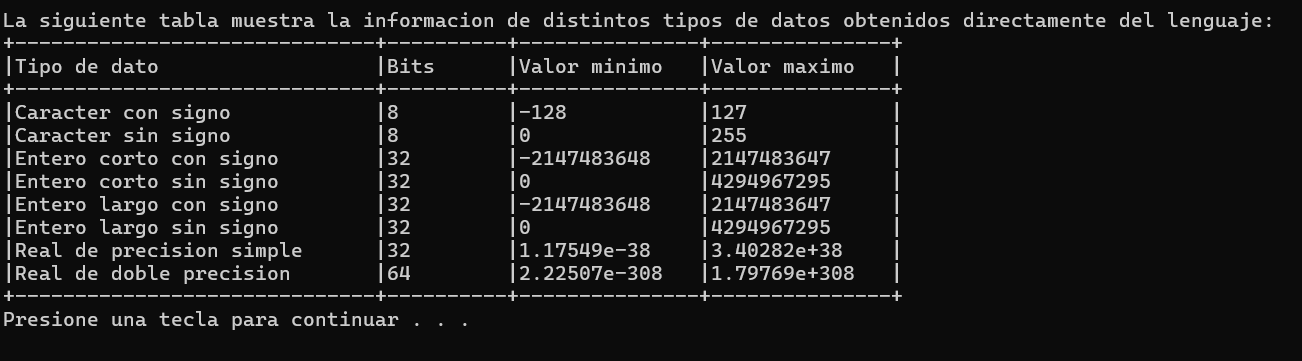


Gracias al objeto Utilities, si se ingresa una opción menor a 1 o mayor a 3 nos muestra el siguiente mensaje:

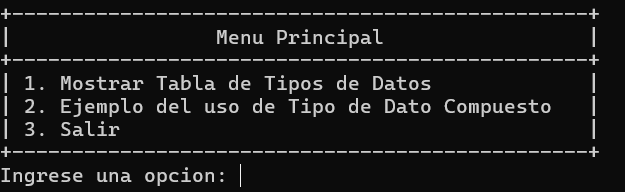


Evitando bug’s en el proceso y limpiando la consola para que no se sobresature en el proceso.

Si le damos a la primera opción en el menú principal, el programa avanza así:

Nota: los tipo Char muestran el valor en ASCII, no el carácter que representa.

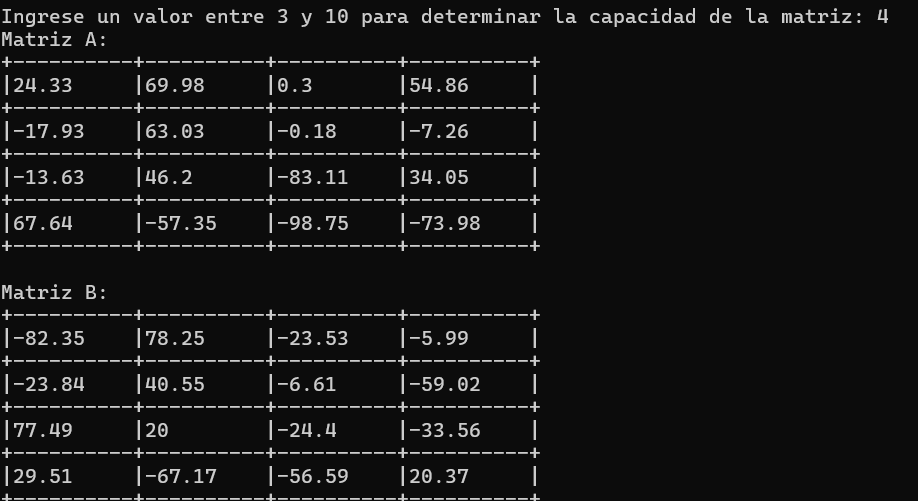
Dándole a cualquier tecla regresamos al menú principal:



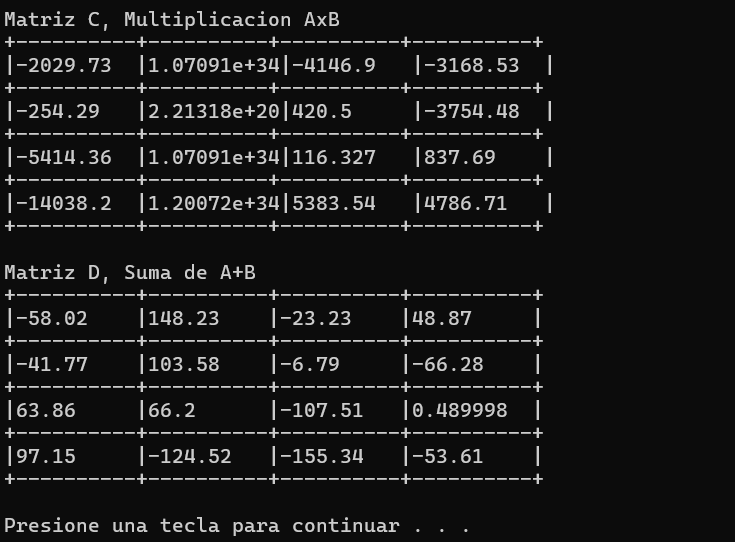
Y si ahora vamos a la opción 2:



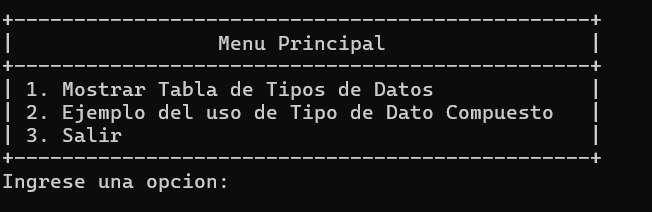
El input sigue restringido por Utilities, pero esta vez de 3 a 10; ingresando, por ejemplo, tamaño de matriz 4:



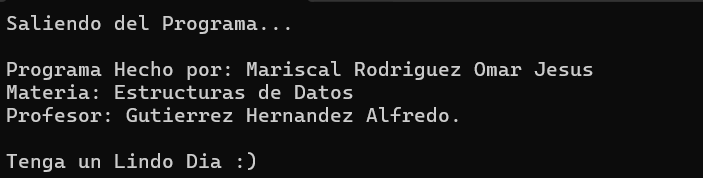
Muestra la matriz A y B con números reales aleatorios entre -100.00 y 100.00 con a lo mucho, dos decimales.

Inmediatamente después aparece lo siguiente:

Las matrices calculadas de su suma y su multiplicación mediante el algoritmo de multiplicación de matrices; si presionamos cualquier tecla volvemos al menú principal:



Podemos volver a cualquiera de las opciones, 1 nos seguirá mostrando la tabla de los tipos de datos y 2 nos seguirá generando matrices con otros números aleatorios que cumplen con las mismas características; todo esto cíclicamente hasta que, finalmente, presionemos la tecla 3, lo que nos muestra lo siguiente:



Después de esto, el programa finaliza y salimos.

# Conclusiones.

El trabajo me sirvió bastante para repasar todos los conceptos que he estudiado hasta ahora en la carrera, en la introducción describí como investigué en diversas fuentes de documentación, esto no es algo que haya tenido que hacer semestres anteriores, y a decir verdad, al principio estaba algo perdido, pero leyendo y siendo curioso con las librerías puedo aprender cómo se compone y tener más herramientas para diversos tipos de programas, y veo como son casi puramente objetos lo que nos proporcionan y como tienen relaciones de dependencias o composiciones detrás de ellas. Creo que son recursos valiosos y saber manejarse entre ellos me da mucha versatilidad como programador; sin ir más lejos, en este trabajo revisé documentación de las librerías random, iomanip y sstream para darle un estilo diferente a mi programa o para cumplir ciertos requisitos.

Otro punto a mencionar es Visual Studio Code, yo lo utilizaba desde antes, pero nunca había trabajado con proyectos dentro de él, y con la extensión de C/C++ Project Maker fue útil para crear la estructura, no obstante, experimenté problemas a la hora de intentar correr los programas al inicio, mis inclusiones tenían referencias inválidas, lo que me decía que Visual Studio no estaba compilando todo el proyecto, solo el main (o eso creía). Investigando sobre ello, creo que el problema estaba en que la estructura que me daba C/C++ Project Maker tenía rutas diferentes donde yo tenía mi compilador MSYS2, por lo que tuve que modificar el tasks.json y el launch.json para adaptarlos a mi equipo, de paso, incluí el path a las carpetas para que las inclusiones en mi código fuente no se vieran como “../include/ejemplo.h” y poder solo hacer “ejemplo.h”, tengo mucho que aprender de Visual Studio Code y los .json, me intriga mucho e investigaré mucho al respecto.

Al manejar la lógica del programa, lo más interesante fue la multiplicación de matrices, que descomponiéndola y simplificando el problema (divide y vencerás) pude idear un algoritmo lo suficientemente bueno como para adaptarlo al programa; el punto de partida que seguí fue descubrir cuantas operaciones hay en una multiplicación de matrices cuadradas (), ahí decidiendo usar una triple iteración para simular más fielmente el algoritmo.

En síntesis, fue un trabajo sumamente interesante, enriquecedor y hasta divertido de realizar por tantos elementos que involucraba y una nueva manera de trabajar con Visual Studio Code que no había probado antes.