

ACTIVIDAD 01: **TIPOS DE DATO PRIMITIVOS Y TIPOS DE DATO ESTRUCTURADOS**

ESTUDIANTE: **EFRAIN ROBLES PULIDO**

CODIGO: **221350095**

*NOMBRE DE LA MATERIA:* **ESTRUCTURAS DE DATOS I**

*SECCIÓN:* ***D12***

*CLAVE:* ***I5886***

FECHA: **DOMINGO 23 DE ENERO DE 2022**

**Tabla de autoevaluación:**

****

**Introducción:**

Para comenzar se tuvo que leer apropiadamente las instrucciones para crear primero una función para el menú donde nos indicara que letra se debe introducir para mostrar los tipos de datos primitivos o las matrices. Se utilizo el ciclo do-while para que si es diferente a “c” se repita el programa.

Si es “a” se deberá mostrar como en forma de tabla, el tamaño de bits y los valores máximos y mínimos de las variables primitivas mediante las funciones del programa como sizeof y los numeric\_limits.

Si es “b” se deberá de pedir el tamaño de las matrices con rango de 3 a 10 que estará en un ciclo do-while para confirmar que este en el rango acordado, si cumple, se crearáN matrices A,B,C,D como objetos, la matriz A y la matriz B tendran valores reales aleatorios de -100 a 100 con la librería random, para después hacer que la matriz C sea el calculo la multiplicación de las matrices A y B mediante una triple anidación del ciclo for y la sobrecarga de los operadores del \*, después hacer que la matriz D que será la suma de las matrices A y B que es directamente la suma de las matrices normales dentro de dos for anidados y la sobrecarga de operadores del +, también se utilizara hacer una sobrecarga del operador = para el correto funcionamiento del código, finalmente se mostrara las matrices apropiadamente con 2 decimales, utilizando también 2 for anidados para mostrarlos.

**Código fuente:**

#include <iostream>

#include <string>

#include <random>

#include <chrono>

#include <functional>

**using namespace std**;

**class** Matriz {

**private**:

**int** sizeM; //Atriutos

**float** matriz[10][10];

**public**:

Matriz() {} //Constructores

Matriz(**const** Matriz& mtrz) : sizeM(mtrz.sizeM) {

**for**(**int** i = 0; i < **this**->sizeM; i++) {

**for**(**int** j = 0; j < **this**->sizeM; j++) {

**this**->matriz[i][j] = mtrz.matriz[i][j];

}

}

}

Matriz(**const int**& N) : sizeM(N) {}

//Metodos

**void** fillMatriz() {//Llena matriz

**std**::**default\_random\_engine** generator(clock());//Llama generador de numero reales random

**std**::**uniform\_real\_distribution**<**double**> distribution(-100.00,100.00);

**for**(**int** i = 0; i < sizeM; i++) {

**for**(**int** j = 0; j < sizeM; j++) {

matriz[i][j] = distribution(generator);

}

}

}

**string** printMatriz() **const** { //Imprime matriz resultante

**string** \_space;

**char** myStr[11];

**for**(**int** i = 0; i < **this**->sizeM; i++) {

**for**(**int** j = 0; j < **this**->sizeM; j++) {

sprintf(myStr,"%8.2f", **this**->matriz[i][j]);

\_space+= myStr;

\_space+= "\t";

}

\_space+= "\n";

}

**return** \_space;

}

//Operadores

Matriz **operator** + (**const** Matriz& mtrz) {

Matriz resultante(sizeM);

**for**(**int** i = 0; i < **this**->sizeM; i++) {

**for**(**int** j = 0; j < **this**->sizeM; j++) {

resultante.matriz[i][j] = matriz[i][j] + mtrz.matriz[i][j];

}

}

**return** resultante;

}

Matriz **operator** \* (**const** Matriz& mtrz) {

Matriz resultante(sizeM);

**for**(**int** i = 0; i < sizeM; i++) {

**for**(**int** j = 0; j < sizeM; j++) {

**float** sum = 0;

**for**(**int** a = 0; a < sizeM; a++) {

sum += matriz[i][a] \* mtrz.matriz[a][j];

}

resultante.matriz[i][j] = sum;

}

}

**return** resultante;

}

Matriz& **operator** = (**const** Matriz& mtrz) {

**for**(**int** i = 0; i < sizeM; i++) {

**for**(**int** j = 0; j < sizeM; j++) {

matriz[i][j] = mtrz.matriz[i][j];

}

}

**return** \***this**;

}

};

**class** TiposDatosPrimitivos {

**public**:

**void** InfoData() {

**cout**<<"\n Tipo de dato Bits Valor minimo Valor maximo"<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Caracter Con Signo "<<**sizeof**(**signed char**)\*8<<" "<<+**numeric\_limits**<**signed char**>::**min**()<<" "<<+**numeric\_limits**<**signed char**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Caracter Sin Signo "<<**sizeof**(**unsigned char**)\*8<<" "<<+**numeric\_limits**<**unsigned char**>::**min**()<<" "<<+**numeric\_limits**<**unsigned char**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Entero Corto Con Signo "<<**sizeof**(**signed short int**)\*8<<" "<<+**numeric\_limits**<**signed short int**>::**min**()<<" "<<+**numeric\_limits**<**signed short int**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Entero Corto Sin Signo "<<**sizeof**(**unsigned short int**)\*8<<" "<<+**numeric\_limits**<**unsigned short int**>::**min**()<<" "<<+**numeric\_limits**<**unsigned short int**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Entero Largo Con Signo "<<**sizeof**(**signed long int**)\*8<<" "<<+**numeric\_limits**<**signed long int**>::**min**()<<" "<<+**numeric\_limits**<**signed long int**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Entero Largo Sin Signo "<<**sizeof**(**unsigned long int**)\*8<<" "<<+**numeric\_limits**<**unsigned long int**>::**min**()<<" "<<+**numeric\_limits**<**unsigned long int**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Real De Precision Simple "<<**sizeof**(**float**)\*8<<" "<<+**numeric\_limits**<**float**>::**min**()<<" "<<+**numeric\_limits**<**float**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------"<<**endl**;

**std**::**cout**<<"Real De Doble Precision "<<**sizeof**(**double**)\*8<<" "<<+**numeric\_limits**<**double**>::**min**()<<" "<<+**numeric\_limits**<**double**>::**max**()<<**endl**;

**cout**<<"---------------------------------------------------------------------------------\n"<<**endl**;

}

};

**void** menu() {

**cout**<<"\tSeleccione una opcion:"<<**endl**;

**cout**<<"a) Tamaño y rangos de los Tipos de Dato Primitivos"<<**endl**;

**cout**<<"b) Ejemplo de uso de Tipo de dato Estructurado"<<**endl**;

**cout**<<"c) Salir\n"<<**endl**;

}

**int** main() {

**char** opcMenu;

**do** {

menu();

**cin**>>opcMenu;

**if**(opcMenu=='a') {

TiposDatosPrimitivos ejemplo;

ejemplo.InfoData();

}

**else if**(opcMenu=='b') {

**int** N = 0;

**do** {

**cout**<<"Tamaño de matriz (3 a 10): "<<**endl**;

**cin**>>N;

}

**while**((N<3)||(N>10));

Matriz A(N), B(N), C(N), D(N);

A.fillMatriz();

**cout** << "Matriz A:" << **endl**;

**cout** << A.printMatriz() << **endl** << **endl**;

B.fillMatriz();

**cout** << "Matriz B:" << **endl**;

**cout** << B.printMatriz() << **endl** << **endl**;

C = A \* B;

**cout** << "Matriz C - Multiplicacion de las matrices:" << **endl**;

**cout** << C.printMatriz() << **endl** << **endl**;

D = A + B;

**cout** << "Matriz D - Suma de las matrices:" << **endl**;

**cout** << D.printMatriz() << **endl** << **endl**;

}

}

**while**(opcMenu!='c');

**cout**<<"Fin del programa\n"<<**endl**;

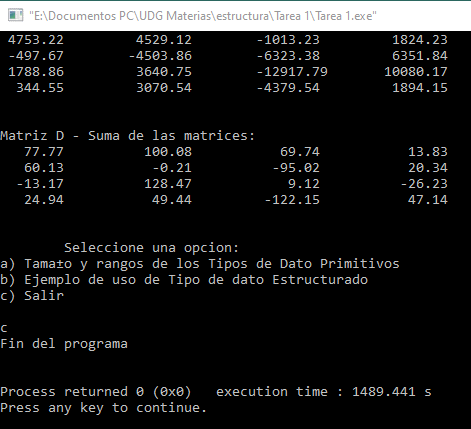
**return** 0;

Texto

Descripción generada automáticamente }

Texto

Descripción generada automáticamente



**Conclusión:**

En esta práctica fue difícil de hacer debido que tuve que investigar cómo se debe de programar en POO ya que en las siguientes practicas tendremos que usarlo muy seguido, primero hice el programa como me enseñaron que es de la forma estructurada, una vez me salió el programa, lo pase a POO en donde tuve que usar las clases para poner sus atributos(las variables a utilizar), construcciones(como se inicializarían los atributos) y métodos(las acciones que harán nuestros objetos) para poder crear los objetos que serían mis matrices, también me fue difícil entender como era la sobrecarga de operadores debido a que se sobrescribe los operadores del mismo programa para poder ajustarlas a nuestras necesidades pero con mucha investigación y pruebas pude realizar el código. Además, también tuve que investigar cómo se sacar los números aleatorios para las matrices que se utilizó funciones de la librería random y chrono para que si fueran aleatorios, también se utilizó la librería limits para poder usar sus funciones para encontrar el valor máximo y mínimo de las variables primitivas y el sizeof() \* 8 para encontrar la cantidad de bits que usan. Tengo mucho de que aprender sobre POO debido que es una forma muy diferente a como me enseñaron programar.