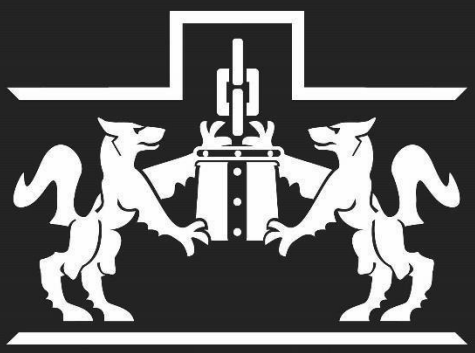


LA VERDAD
NOS HARÁ LIBRES

UNIVERSIDAD
IBEROAMERICANA
PUEBLA ®

Practica No 6.

Clases: Calculadora

18/11/2020

Otoño 2020

Ariel Hernández Barrón
Samara García González

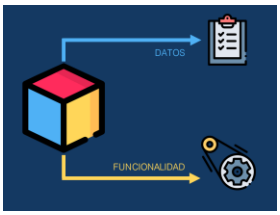
Laboratorio de Elementos Programables II

Profesor: Mario De Los Santos

Licenciatura

Ingeniería Mecatrónica

PRÁCTICA #6 “Práctica final del curso”



1. INTRODUCCIÓN

La presente práctica conjunta el conocimiento del 90% de los temas vistos a lo largo del semestre en la materia Elementos programables 2, siendo el principal propósito reafirmar el conocimiento e introducir el concepto de programación orientada a objetos y sus 5 pilares: Abstracción, encapsulamiento, polimorfismo, herencia y modularidad. La práctica consiste en la escritura de una clase multinivel que funciona en conjunto como una calculadora científica, que incluye operaciones aritméticas básicas(suma, resta, multiplicación, división y potencia), operaciones geométricas(área, volumen y perímetro de triángulos, rectángulos y círculos) Operaciones matriciales (suma, resta, multiplicación y transposición), Operaciones algebraicas(solución de sistemas de ecuaciones 2x2, distancia entre puntos, magnitud de un vector y suma de vectores), operaciones estadísticas(media , moda , promedio y mediana) , realizadas cada una en una clase, cuyos atributos hereda una superclase calculadora.

El usuario puede dimensionar el tamaño de los dígitos usados en cada operación, excepto en las operaciones con vectores, ya que estos son 2D y en las operaciones matriciales, con un tamaño por default de 3x3, los recursos usados en esta super clase incluyen, pero no se limitan a constructores, matadores, accesos, operaciones, funciones y punteros.

2. MARCO DE REFERENCIA

La programación Orientada a objetos (POO) es un paradigma de programación que usa objetos en sus interacciones, para diseñar aplicaciones y programas informáticos.
La POO es una forma especial de programar, este paradigma de programación es cercano a como expresamos las cosas en la vida real en nuestro día a día. La POO tiene como principio que todo en la vida es un objeto programable, entonces para empezar a programar con este paradigma se debe empezar con desarrollar el pensamiento basado en objetos. Existe un acuerdo acerca de qué características contempla la "orientación a objetos".

Abstracción

Denota las características esenciales de un objeto, donde se capturan sus comportamientos. El proceso de abstracción permite seleccionar las características relevantes dentro de un conjunto e identificar comportamientos comunes para definir nuevos tipos de entidades en el mundo real. La abstracción es clave en el proceso de análisis y diseño orientado a objetos, ya que mediante ella podemos llegar a armar un conjunto de clases que permitan modelar la realidad o el problema que se quiere atacar.

Encapsulamiento

Significa reunir todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción. Esto permite aumentar la de los componentes del sistema.

Polimorfismo

Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre; al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando

Herencia

Las clases no se encuentran aisladas, sino que se relacionan entre sí, formando una jerarquía de clasificación. Los objetos heredan las propiedades y el comportamiento de todas las clases a las que pertenecen. La herencia organiza y facilita el polimorfismo y el encapsulamiento, permitiendo a los objetos ser definidos y creados como tipos especializados de objetos preexistentes. Cuando un objeto hereda de más de una clase, se dice que hay herencia múltiple.

Modularidad

Se denomina "modularidad" a la propiedad que permite subdividir una aplicación en partes más pequeñas, cada una de las cuales debe ser tan independiente como sea posible de la aplicación en sí y de las restantes partes.

3. OPERACIONES

A continuación, se presenta una breve descripción de cada una de las operaciones realizadas separadas por la clase a la que pertenecen:

Operaciones aritméticas básicas

Suma: La suma o adición es una de las cuatro operaciones básicas de la aritmética que consiste en la adición de dos o más elementos para llegar a un resultado final.

$$Suma = a + b$$

Resta: La resta o la sustracción es una operación de aritmética que se representa con el signo; representa la operación de eliminación de objetos de una colección

$$Resta = a - b$$

Multiplicación: La multiplicación, en definitiva, consiste en tomar el multiplicando y sumarlo tantas veces como unidades contiene el multiplicador**.**

$$Multiplicación = a * b$$

División: La división es una de las operaciones básicas de la aritmética que consiste en separar en partes iguales un total.

$$División = a / b$$

Potencia: Las potencias son una manera abreviada de escribir una multiplicación formada por varios números iguales, La base es el número que se está multiplicando varias veces y el exponente es el número de veces que se multiplica la base.

$$Potencia = a \wedge b$$

Operaciones geométricas

Área: El área puede ser definida como la medida de la superficie, y se descubre partir de multiplicar la base por la altura en el caso del rectángulo, el resultado entre dos para el caso del triángulo y pi por radio al cuadrado en el caso del círculo.

$$Área\ (rectángulo) = b * h$$
$$Área\ (triángulo) = (b * h) / 2$$
$$Área\ (círculo) = \pi * r\wedge 2$$

Volumen: Volumen se considera el espacio ocupado por un cuerpo, es decir, su magnitud física comprendida en tres dimensiones: largo, ancho y alto.

$$Volumen = l * a * h$$

Perímetro: Se refiere al contorno de una superficie o de una figura y a la medida de ese contorno. En otras palabras, en una figura, el perímetro es la suma de todos sus lados.

$$Perímetro\ (rectángulo) = l+a+l+a$$
$$Perímetro\ (triángulo) = l+l+l$$
$$Perímetro\ (círculo) = \pi * d$$

Operaciones matriciales

Suma: La matriz suma de dos matrices con la misma dimensión es la matriz que tiene en la posición fila i y columna j la suma de los elementos de la misma posición en las matrices que sumamos. Es decir, la suma de matrices se calcula sumando los elementos que ocupan la misma posición**.**

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} =$$

$$= \begin{pmatrix} 1+5 & 2+6 \\ 3+7 & 4+8 \end{pmatrix} = \begin{pmatrix} 6 & 8 \\ 7 & 12 \end{pmatrix}$$

Resta: La matriz resta de dos matrices con la misma dimensión es la matriz que tiene en la posición fila i y columna j la resta de los elementos de la misma posición en las matrices que restamos. Es decir, la resta de matrices se calcula restando los elementos que ocupan la misma posición**.**

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} - \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} =$$

Multiplicación: Dos matrices A y B son multiplicables si el número de columnas de A coincide con el número de filas de B. El elemento cij de la matriz producto se obtiene multiplicando cada elemento de la fila i de la matriz A por cada elemento de la columna j de la matriz B y sumándolos.

$$\begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} -1 & -1 \\ 5 & 6 \end{pmatrix} =$$

$$= \begin{pmatrix} 2\cdot(-1)+0\cdot 5 & 2\cdot(-1)+0\cdot 6 \\ 1\cdot(-1)+3\cdot 5 & 1\cdot(-1)+3\cdot 6 \end{pmatrix} =$$

$$= \begin{pmatrix} -2 & -2 \\ 14 & 17 \end{pmatrix}$$

Transposición: Hallar matriz transpuesta: matr[m][n] su transpuesta es matr[n][m]. Se obtiene cambiando filas por columnas. Donde los elementos de la fila m ahora pertenecen a la columna m de la transpuesta.

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix}$$

$$A^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

Operaciones algebraicas

Solución de sistemas de ecuaciones 2x2: Un sistema lineal de dos ecuaciones con dos incógnitas es un sistema lineal de ecuaciones formado por sólo dos ecuaciones que admite un tratamiento particularmente simple, con infinito número de soluciones.

$$\begin{cases} x+5y=5 \\ 3x-5y=3 \end{cases} \qquad \begin{cases} y=-2x+1 \\ 4x+2y=3 \end{cases}$$

Distancia entre puntos: Cualquier punto del Plano se puede localizar con respecto a un par de ejes Perpendiculares dando las distancias del punto a cada uno de los ejes, para esto se realizan las operaciones de distancia entre puntos como la raíz de la suma de las diferencias al cuadrado entre coordenadas, x, y

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Magnitud de un vector: . La magnitud de un vector nos permite conocer el tamaño del vector sin importar dirección o sentido, se obtiene con la raíz cuadrada de la suma de sus componentes al cuadrado.

$$|\vec{a}| = \sqrt{x^2 + y^2 + z^2}$$

Suma de vectores: La suma y resta de vectores se realiza sumando o restando cada una de las componentes de cada uno y da como resultado otro vector. Para sumar dos vectores, los mismos deben tener la misma cantidad de componentes.

$$\vec{R} = \vec{A} + \vec{B}$$
$$\vec{R} = (4,2) + (-1,5)$$
$$\vec{R} = (3,7)$$

Operaciones estadísticas

Media : Una media o promedio es una medida de tendencia central. Resulta al efectuar una serie determinada de operaciones con un conjunto de números y que, en determinadas condiciones, puede representar por sí solo a todo el conjunto.

$$\bar{X} = \frac{x_1 + x_2 + x_3 + x_4 + \cdots + x_n}{N}$$

Moda: La moda es el dato de mayor frecuencia, así que si quieres calcularlo solo debes fijarte del dato que más se repite entre todos. Si los datos te lo presentan en una tabla o una gráfica, la moda será la categoría con mayor frecuencia absoluta. La formula para datos agrupados es la siguiente:

$$\text{Mo} = Li + \left(\frac{\Delta 1}{\Delta 1 + \Delta 2}\right) * c$$

Mediana: En el ámbito de la estadística, la mediana representa el valor de la variable de posición central en un conjunto de datos ordenados.

4.DIAGRAMA DE FLUJO

A continuación, se muestra el diagrama de flujo que sigue la lógica de la herencia de clases, comenzando con la clase más básica y esencial hasta llegar a la calculadora que hereda todas las clases anteriores.

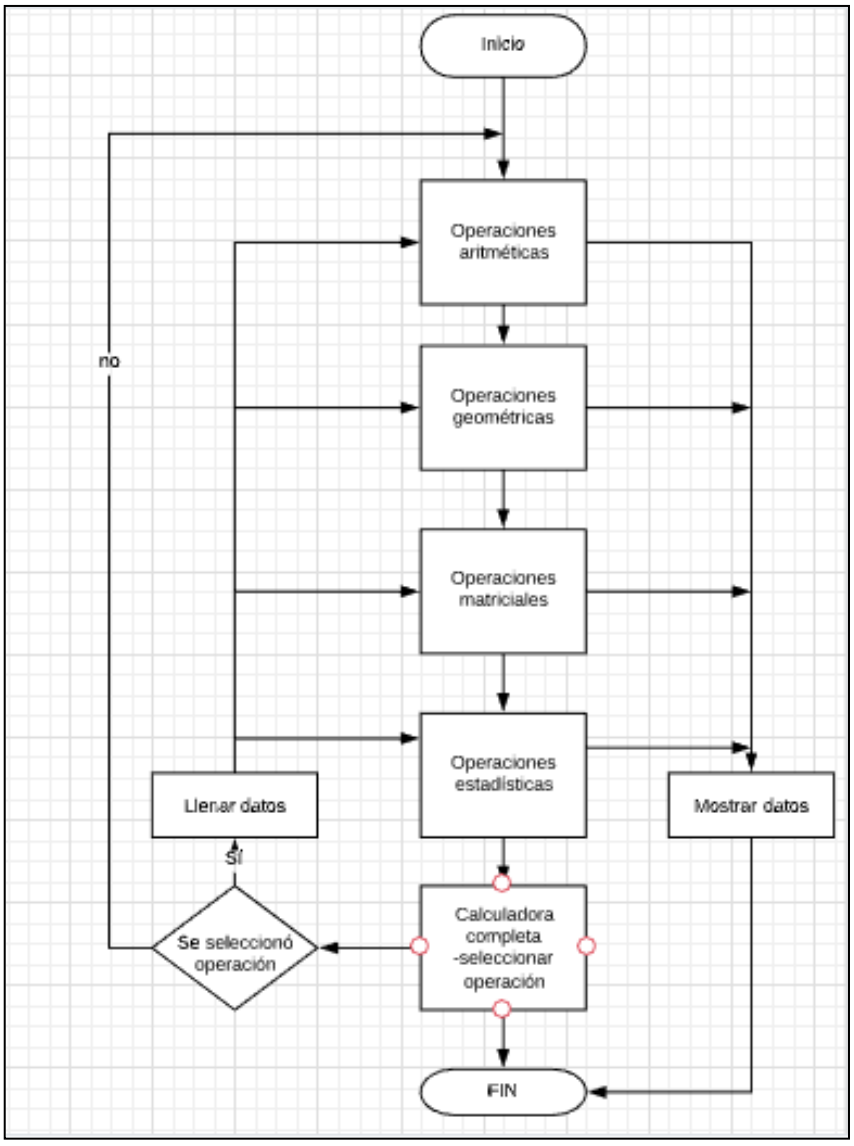


Fig.1 Diagrama de flujo del ejercicio

CÓDIGO

A continuación, se muestran las capturas de pantalla del código, en donde se puede ver implementado el uso de clases, funciones, constructores y destructores, entre otros, la selección se realiza en primer instancia por un Switch case en la clase final “calculadora” que permite seleccionar una clase, es decir un tipo de operación, posteriormente el programa se redirige a un submenú en el que se selecciona la operación especifica, para posteriormente realizar lo contenido dentro de la función.

```
#include <iostream>
#include <bits/stdc++.h> //libreria sort
#include <math.h> //Libreria operaciones matematicas
using namespace std;
/*CLASES*/
class OpBasicas //Clase de operaciones fundamentales
{
public:
  double add(int i);
  double sub(int i);
  double mul(int i);
  double pot();
  double div(int i);
};
class Geometrico: public OpBasicas //Clase de operaciones geometricas
{
  double ancho;
  double largo;
  double profundidad;
  double base;
  double altura;
  float radio;

public:
  void setAncho(); //Mutadores
  void setLargo();
  void setProfundidad();
  void setBase();
  void setAltura();
  void setRadio();

  double areaRec();
  double perimetroRec();
  double volumenRec();
  double areaTria();
  double perimetroTria();
  double volumenTria();
  double areaCir();
  double perimetroCir();
  double volumenCir();
};

class Matrices: public Geometrico //Clase de operaciones matriciales
{
  double A[50][50] = {0}; //matriz de usuario 1
  double B[50][50] = {0}; //matriz de usuario 2
  double C[50][50] = {0}; //transpuesta de 1
  double D[50][50] = {0}; //transpuesta de 2
  double R[50][50] = {0}; //matriz resultado
```

```
public:
  void setData(int sizex, int sizey);
  void sumaM(int sizex, int sizey);
  void restaM(int sizex, int sizey);
  void multM(int sizex, int sizey);
  void transM(int sizex, int sizey);
  void printR(int sizex, int sizey);
};

class Vectoriales: public Matrices //Clase de operaciones vectoriales
{
  int x3,x4;
  int y3,y4;
  int z3,z4;

  public: //publico
  void setx(int x1,int x2);//MUTADORES
  void sety(int y1, int y2);
  void setz (int z1, int z2);

  int getx();//ACCESOS
  int gety();
  int getz();

  void ecuaciones();//OPERACIONES
  void distancia();
  void mvector();
  void sumavectores();
};

class Estadistica: public Vectoriales //Clase de operaciones estadisticas
{
  double A[50] = {0}; // matriz de usuario

public:
  void setData(int i);
  double media(int i);
  double moda(int i);
  double mediana(int i);

};

class Fullcalc: public Estadistica //Clase de calculadora completa
{
public:
  void OpBasic();
  void Geometry();
  void Matrix();
  void Vectorial();
  void Estadisticas();
};
```

```
int main() //interfaz de usuario
{
  int menu = 0;
  Fullcalc calculadora; //CREACION DE OBJETO DE CLASE GENERAL

  cout << "1.Basica"
  << "\n2.Geometrica"
  << "\n3.Matricial"
    << "\n4.Vectorial"
    << "\n5.Estadistica"
  << "\n0.SALIR"
  << "\n" <<endl;

  do {

    cout << "\nINGRESE TIPO DE OPERACION ";
    cin >> menu;

    switch (menu) {
      case 1: // OPERACIONES BASICAS
        calculadora.OpBasic();
      break;
      case 2: // OPERACIONES GEOMETRICAS
        calculadora.Geometry();
      break;
      case 3: // OPERACIONES MATRICIALES
        calculadora.Matrix();
      break;
      case 4: // OPERACIONES VECTORIALES
        calculadora.Vectorial();
      break;
      case 5: // OPERACIONES ESTADISTICAS
        calculadora.Estadisticas();
      break;
    }

  } while (menu>= 1 && menu <= 5);
}

void Fullcalc::OpBasic() //funciÃ³n menÃº de la calculadora bÃ¡sica
{
  int s = 0, cant = 0;
  OpBasicas calculadora; //creacion de objeto en clase basica

  cout << "1.SUMA"
  << "\n2.RESTA"
  << "\n3.MULTIPLICA"
  << "\n4.DIVIDE"
  << "\n5.POTENCIA"
  << "\n0.SALIR"
  << "\n";
```

```
  do {

  cout << "\nINGRESE SU OPERACION: ";
  cin >> s;

  switch (s) {
  case 1: // SUMA
     cout << "\nSUMA, INGRESE CANTIDAD DE NUMEROS: ";
    cin >> cant;
   cout << calculadora.add(cant) << endl;
   break;
  case 2: // RESTA
     cout << "\nRESTA, INGRESE CANTIDAD DE NUMEROS: ";
     cin >> cant;
   cout << calculadora.sub(cant) << endl;
   break;
  case 3: // MULTIPLICACION
     cout << "\nMULT, INGRESE CANTIDAD DE NUMEROS: ";
     cin >> cant;
   cout << calculadora.mul(cant) << endl;
   break;
  case 4: // DIVISION
     cout << "\nDIV, INGRESE CANTIDAD DE NUMEROS: ";
     cin >> cant;
   cout << calculadora.div(cant) << endl;
   break;
  case 5: // POTENCIA
   cout << calculadora.pot() << endl;
   break;
  }

 } while (s>= 1 && s <= 5);
}

double OpBasicas::add(int i) //OPERADOR SUMA, cantidad de datos
{
  int num = 0, result = 0;
  for (int x = 0; x < i; x++) //ciclo para el total de digitos
  {
     cout << "ingresa numero " << x << ": " << endl;
     cin >> num;
     result += num; //suma a variable resultado
  }
  return result; //regresar suma total
}

double OpBasicas::sub(int i) //OPERADOR RESTA, cantidad de datos
{
  int num = 0, result = 0;
  cout << "ingresa numero 0: "<< endl;
  cin >> num;
```

```
  result = num; //valor inical de la resta
  for (int x = 1; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
      result -= num; //resta a variable resultado
  }
  return result; //regresar resta total
}

double OpBasicas::mul(int i) //OPERADOR MUlTIPLICACION, cantidad de datos
{
  int num = 1, result = 1;
  for (int x = 0; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
      result *= num; //multiplicacion a variable resultado
  }
  return result; //regresar mult total
}

double OpBasicas::pot() //OPERADOR POTENCIA, cantidad de datos
{
  int num = 0, power = 0, result = 0;

    cout << "ingresa numero: " << endl;
    cin >> num;
    cout << "ingresa potencia: " << endl;
    cin >> power;

    return pow(num, power); //regresa potencia de los numeros dados por usuario
}

double OpBasicas::div(int i) //OPERADOR DIVISION, cantidad de datos
{
  int num = 1, result = 1;
  cout << "ingresa numero 0: " << endl;
  cin >> num;
  result = num; //primer valor de resultado para la division
  for (int x = 1; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
    if (num == 0){ //si el numero es 0 se tiene un error
      cout << "division entre 0" <<endl;
      return 0;
    } else {
       result /= num; // division al resultado
    }
  }
```

```
  return result; // regresar division
}

void Fullcalc::Geometry() //funciÃ³n menÃº de la calculadora geometrica
{
  int menu1 = 0, menu2 = 0;
  Geometrico calculadora; //creacion de objeto en clase geometrica

  cout << "1.RECTANGULO"
  << "\n2.TRIANGULO"
  << "\n3.CIRCULO"
  << "\n0.SALIR"
  << "\n" <<endl;

  do {

    cout << "\nINGRESE SU FIGURA ";
    cin >> menu1;

    switch (menu1) { //PRIMER SWITCH CASE PARA DETERMINAR LA FIGURA
      case 1: // RECTANGULO
        cout << "1.PERIMETRO"
        << "\n2.AREA"
        << "\n3.VOLUMEN"
        << "\n0.SALIR"
        << "\n" <<endl;
        cout << "\nINGRESE SU OPERACION ";
        cin >> menu2;

        switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
          case 1: // PERIMETRO
            calculadora.setAncho();
            calculadora.setLargo();
            cout << "RESULTADO: " << calculadora.perimetroRec() << endl;
          break;
          case 2: // AREA
            calculadora.setAncho();
            calculadora.setLargo();
            cout << "RESULTADO: " << calculadora.areaRec() << endl;
          break;
          case 3: // VOLUMEN
            calculadora.setAncho();
            calculadora.setLargo();
            calculadora.setProfundidad();
            cout << "RESULTADO: " << calculadora.volumenRec() << endl;
          break;
      }
```

```
  break;

     case 2: // TRIANGULO
       cout << "1.PERIMETRO"
         << "\n2.AREA"
         << "\n3.VOLUMEN"
         << "\n0.SALIR"
         << "\n" <<endl;
       cout << "\nINGRESE SU OPERACION ";
       cin >> menu2;

       switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
       case 1: // PERIMETRO
         calculadora.setBase();
         cout << "RESULTADO: " << calculadora.perimetroTria() << endl;
         break;
       case 2: // AREA
         calculadora.setBase();
         calculadora.setAltura();
         cout << "RESULTADO: " << calculadora.areaTria() << endl;
         break;
       case 3: // VOLUMEN
         calculadora.setBase();
         calculadora.setAltura();
         cout << "RESULTADO: " << calculadora.volumenTria() << endl;
         break;
       }
       break;

       case 3: //CIRCULO
         cout << "1.PERIMETRO"
           << "\n2.AREA"
           << "\n3.VOLUMEN"
           << "\n0.SALIR"
           << "\n" <<endl;
         cout << "\nINGRESE SU OPERACION ";
         cin >> menu2;

         switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
         case 1:
           calculadora.setRadio(); // PERIMETRO
           cout << "RESULTADO: " << calculadora.perimetroCir() << endl;
           break;
         case 2:
           calculadora.setRadio(); // AREA
           cout << "RESULTADO: " << calculadora.areaCir() << endl;
           break;
         case 3: // VOLUMEN
           calculadora.setRadio();
```

```
cout << "RESULTADO: " << calculadora.volumenCir() << endl;
            break;
          }
          break;
      }

  } while (menu1>= 1 && menu1 <= 3);
}

/*FUNCIONES PARA OBTENER DATOS*/

void Geometrico::setAncho()
{
  int A;
  cout << "ingrese ancho: "<<endl;
  cin >> A;
  if(A<0) // si el valor ingresado es negativo, convertirlo
    A=A*-1;
  ancho = A;
}

void Geometrico::setLargo()
{
  int L;
  cout << "ingrese largo: "<<endl;
  cin >> L;
  if(L<0) // si el valor ingresado es negativo, convertirlo
    L=L*-1;
  largo = L;
}

void Geometrico::setProfundidad()
{
  int P;
  cout << "ingrese profundidad: "<<endl;
  cin >> P;
  if(P<0) // si el valor ingresado es negativo, convertirlo
    P=P*-1;
  profundidad = P;
}

void Geometrico::setBase()
{
  int B;
  cout << "ingrese base: "<<endl;
  cin >> B;
  if(B<0) // si el valor ingresado es negativo, convertirlo
    B=B*-1;
  base = B;
}
```

```
void Geometrico::setAltura()
{
  int H;
  cout << "ingrese altura: "<<endl;
  cin >> H;
  if(H<0) // si el valor ingresado es negativo, convertirlo
    H=H*-1;
  altura = H;
}

void Geometrico::setRadio()
{
  int R;
  cout << "ingrese radio: "<<endl;
  cin >> R;
  if(R<0) // si el valor ingresado es negativo, convertirlo
    R=R*-1;
  radio = R;
}

/* OPERADORES */
double Geometrico::areaRec() //AREA RECTANGULO
{
  return ancho*largo;
}

double Geometrico::perimetroRec() //PERIMETRO RECTANGULO
{
  return (ancho+largo)*2;
}

double Geometrico::volumenRec() // VOLUMEN RECTANGULO
{
  return(ancho*largo*profundidad);
}

double Geometrico::areaTria() //AREA TRIANGULO
{
  return (base*altura)/2;
}

double Geometrico::perimetroTria() // PERIMETRO TRIANGULO
{
  return base*3;
}

double Geometrico::volumenTria() //VOLUMEN TRIANGULO
{
  return((sqrt(3)/4)*(base*base)*altura);
}
```

```
double Geometrico::areaCir() //AREA CIRCULO
{

  return (3.1416*(radio*radio));
}

double Geometrico::perimetroCir() //PERIMETRO CIRCULO
{
  return ((2 * 3.1416) * radio);
}

double Geometrico::volumenCir() //VOLUMEN CIRCULO
{
  return ((4/3)*3.1416)*(pow(radio,3));
}

void Fullcalc::Matrix() //funciÃ³n menÃº de la calculadora matricial
{
  int menu = 0, sizex = 0, sizey = 0;
  Matrices calculadora; //creacion de objeto en clase matricial

  cout << "1.SUMA"
  << "\n2.RESTA"
  << "\n3.MULTIPLICACION"
    << "\n4.TRANSPOSICION"
  << "\n0.SALIR"
  << "\n" <<endl;

    do {
      cout << "\nINGRESE SU OPERACION ";
      cin >> menu;

      switch (menu) {
        case 1: // SUMA
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.sumaM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
        case 2: // RESTA
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.restaM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
        case 3: // MULTIPLICACION
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
```

```
            sizey = sizex;
            calculadora.setData(sizex, sizey); //llenado de matriz
          calculadora.multM(sizex, sizey); // Suma de matriz
            calculadora.printR(sizex, sizey); // impresion de matriz
          break;
          case 4: // TRANSPOSICION
            cout << "tamaÃ±o de matrices: "<<endl;
            cin >> sizex;
            sizey = sizex;
            calculadora.setData(sizex, sizey); //llenado de matriz
          calculadora.transM(sizex, sizey); // Suma de matriz
            calculadora.printR(sizex, sizey); // impresion de matriz
          break;
      }

    } while (menu>= 1 && menu <= 4);
}

/*FUNCIONES PARA OBTENER DATOS*/

void Matrices::setData(int sizex, int sizey) // tamaÃ±o en x, tamaÃ±o en y
{
  cout << "MATRIZ A: " <<endl;
  for (int x=0; x<sizex; x++){ //llenado de las matrices
    for (int i=0; i<sizey; i++){
          cout << "Fila: " << x << " columna: " << i <<endl;
          cin >> A[i][x];
      }
  }

  cout << "MATRIZ B: " <<endl;
  for (int x=0; x<sizex; x++){ //llenado de las matrices
    for (int i=0; i<sizey; i++){
          cout << "Fila: " << x << " columna: " << i <<endl;
          cin >> B[i][x];
      }
  }
}

/* OPERADORES */

void Matrices::sumaM(int sizex, int sizey) //SUMA, tamaÃ±o en x, tamaÃ±o en y
{
  for (int x=0; x<sizex; x++){ //SUMA MATRICIAL
    for (int i=0; i<sizey; i++){
          R[i][x] = A[i][x] + B [i][x];
      }
  }
}

void Matrices::restaM(int sizex, int sizey) //RESTA, tamaÃ±o en x, tamaÃ±o en y
```

```
{
  for (int x=0; x<sizex; x++){ //RESTA MATRICIAL
    for (int i=0; i<sizey; i++){
         R[i][x] = A[i][x] - B [i][x];
      }
  }
}

void Matrices::multM(int sizex, int sizey) //MULTIPLCACION, tamaÃ±o en x, tamaÃ±o en y
{
  for (int x=0; x<sizex; x++){ //MUlTIPLICACION MATRICIAL
    for (int i=0; i<sizey; i++){
         R[i][x] = A[i][x] * B [i][x];
      }
  }
}

void Matrices::transM(int sizex, int sizey) //TRANSPUESTA, tamaÃ±o en x, tamaÃ±o en y
{
  int i,j,aux1,aux2;
  for (int x=0; x<sizex; x++) {
     for(int i=0; i<sizey; i++) {
       //matriz1 cambio de posiciÃ³n
       aux1 = A[x][i]; //variable temporal para guardado
       C[x][i] = A[i][x];
       A[i][x] = aux1;
       //matriz2 cambio de posiciÃ³n
       aux2 = B[x][i]; //variable temporal para guardado
       D[x][i] = B[i][x];
       B[i][x] = aux2;
      }
  }
}

/*FUNCIONES PARA IMPRESION*/

void Matrices::printR(int sizex, int sizey)
{
  for (int x=0; x<sizex; x++){ //impresion de matriz resultado
    for (int i=0; i<sizey; i++){
         cout << R[i][x] <<" ";
      }
    cout <<endl;
  }
  for (int x=0; x<sizex; x++){ //impresion de matriz transpuesta 1
    for (int i=0; i<sizey; i++){
         cout << C[i][x] <<" ";
      }
    cout <<endl;
  }
  for (int x=0; x<sizex; x++){ //impresion de matriz transpuesta 2
    for (int i=0; i<sizey; i++){
```

```
        cout << D[i][x] <<" ";
      }
     cout <<endl;
   }
}

void Fullcalc::Vectorial() //funciÃ³n menÃº de la calculadora vectorial
{
  int x1=0,x2=0,y1=0,y2=0,s,z1=0,z2=0;
  Vectoriales calculadora; //creacion de objeto en clase vectorial

  cout<< "1.SISTEMA 2X2"
  << "\n2.DISTANCIA 2 PUNTOS"
  << "\n3.MAGNITUD VECTOR"
  << "\n4.SUMA DE VECTORES"
  << "\n0.SALIR"
  << "\n";

  do {
  cout << "\nINGRESE SU OPERACION ";
  cin >> s;

  switch (s) {
  case 1: // SISTEMA 2*2
   calculadora.setx(x1,x2); //captura de informacion
     calculadora.sety(y1,y2);//captura de informacion
     calculadora.setz(z1,z2);//captura de informacion
     cout<<"LA SOLUCION AL SISTEMA 2X2 ES:"<<endl;
     calculadora.ecuaciones();
   break;
  case 2: // DISTANCIA ENTRE PUNTOS
   calculadora.setx(x1,x2);//captura de informacion
     calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA DISTANCIA ENTRE DOS PUNTOS ES:"<<endl;
   calculadora.distancia();
   break;
  case 3: // MAGNITUD DE VECTOR
   calculadora.setx(x1,x2);//captura de informacion
     calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA MAGNITUD DEL VECTOR ES:"<<endl;
   calculadora.mvector();
   break;
  case 4: // SUMA DE VECTORES
   calculadora.setx(x1,x2);//captura de informacion
     calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA SUMA DE VECTORES ES:"<<endl;
   calculadora.sumavectores();
   break;
  case 0:
   cout << "SALIR" << endl;
   break;
  }
```

```
 } while (s>= 1 && s <= 4);

}

/*FUNCIONES PARA OBTENER DATOS*/

void Vectoriales::setx(int x1, int x2){
 cout<<"INGRESE VALOR DE X1"<<endl;//pide valor al usuario y lo guarda
 cin>>x1;
 x3 = x1;//asigna el valor a una nueva variable
 cout<<"INGRESE VALOR DE X2"<<endl;
 cin>>x2;
 x4 = x2;
}

void Vectoriales::sety (int y1, int y2){
 cout<<"INGRESE VALOR DE y1"<<endl;//pide valor al usuario y lo guarda
 cin>>y1;
 y3= y1;//asigna valor a una nueva variable
 cout<<"INGRESE VALOR DE y2"<<endl;
 cin>>y2;
 y4= y2;
}

void Vectoriales::setz (int z1, int z2){
 cout<<"INGRESE VALOR DE RESULTADO EC1"<<endl;//pide valor al usuario y lo guarda
 cin>>z1;
 z3  = z3;//asigna valor a una nueva variable
 cout<<"INGRESE VALOR DE RESULTADO EC2"<<endl;
 cin>>z2;
 z4  = z2;//resultado
}

int Vectoriales::getx(){//regresa el valor de las nuevas variables
 return x3;
 return x4;
}

int Vectoriales::gety(){//regresa el valor de las nuevas variables
 return y3;
 return y4;
}

int Vectoriales::getz(){//Regresa el valor de las nuevas variables
 return z3;
 return z4;
}

/* OPERADORES */

void Vectoriales::ecuaciones(){
float y,x,z,xr,yr,zr;//eSolucion de ecuaciones 2x2 por métoddo de eliminación
```

```
x=-x3*x4;
y=-x4*x3;
z=-x4*x3;

xr=x3*x4;
yr=x3*y4;
zr=x3*z4;

x=x+xr;
y=y+yr;
z=z+zr;
y=z/y;

z=z3-y*y3;
x=z/x3;

cout<<"VALOR DE Y ES:"<<y<<"VALOR DE X ES:"<<x;
}

void Vectoriales::distancia(){//distancia entre puntos por formula
  float a,b,c;
  a = (x4 - x3)*(x4 - x3);
  b = (y4 - y3)*(y4- y3);
  c= sqrt(a+b);
  cout<<"LA DISTANCIA ENTRE PUNTOS ES:"<<c<<endl;

}

void Vectoriales::mvector(){//magnitud vector por formula
  cout<<"COLOCAR VALORES DE X2,Y2=0"<<endl;
  float a,b,c;
  a = x3;
  b = y3;
  c= sqrt((a*a)+(b*b));
  cout<<"LA MAGNITUD DEL VECTOR ES:"<<c<<endl;
}

void Vectoriales::sumavectores(){//suma de vectores por formula
  float a,b;
  a = x3 + x4;
  b = y3 + y4;
  cout<<"VALOR DE X ES:"<<a<<"EL VALOR DE Y ES:"<<b;
}

void Fullcalc::Estadisticas() //funciÃ³n menÃº de la calculadora estadistica
{
 int menu = 0, n = 0;
 Estadistica calculadora; //creacion de objeto en clase estadistica

 cout << "1.MEDIA"
 << "\n2.MODA"
 << "\n3.MEDIANA"
```

```
  << "\n0.SALIR"
  << "\n" <<endl;

    do {
      cout << "\nINGRESE SU OPERACION ";
      cin >> menu;

      switch (menu) {
        case 1: // MEDIA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.media(n) << endl;
         break;
        case 2: // MODA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.moda(n) << endl;
         break;
        case 3: // MEDIANA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.mediana(n) << endl;
         break;
      }

    } while (menu>= 1 && menu <= 3);
}

/*FUNCIONES PARA OBTENER DATOS*/

void Estadistica::setData(int i)
{
  for (int x = 0; x<i; x++){ //LLENADO DE MATRIZ
    cout << "digito " << x << ":"<<endl;
    cin >> A[x];
  }
}

/* OPERADORES */

double Estadistica::media(int i) // tamaÃ±o
{
  int result = 0;
  for (int x = 0; x<i; x++){ // SUMA DE TODOS LOS DATOS
    result += A[x];
  }
  return result / i; // DIVISION PARA OBTENER PROMEDIO
}
```

```
double Estadistica::moda(int i) // tamaÃ±o
{

  sort(A, A+i); // funcion para ordenar la matriz
  int temp = A[0]; // primer valor de la moda
  int moda = temp, contador1 = 1, contador2 = 1;

  for (int x = 1; x<i; x++){

    if (A[x] == temp){ // si el valor del ciclo es igual a la moda original
      contador1++; //sumar uno a contador
    } else{ // si no es igual

      if (contador1 > contador2){ // contador es mas grande que contador 2
        contador2 = contador1; // cambio de valor
        moda = temp; // moda es igual al ultimo valor en temp
       }
      contador1 = 1; // reseteo de contador 1
      temp = A[x]; // temporal es igual a nuevo valor del arreglo
    }
  }

    return moda;
}

double Estadistica::mediana(int i) // tamaÃ±o
{

  double mediana = 0;
  sort(A, A + i); // funcion para ordenar matriz

  if (i%2 == 0){ // si es par
    mediana = (A[(i/2)-1] + A[i/2])/2; // se toman los dos valores de en medio y se promedia
  } else { // si es impar
    mediana = A[(i/2)]; // se toma el valor de la mitad
  }

  ret#include <iostream>
#include <bits/stdc++.h> //libreria sort
#include <math.h> //Libreria operaciones matematicas
using namespace std;
/*CLASES*/
class OpBasicas //Clase de operaciones fundamentales
{
public:
  double add(int i);
  double sub(int i);
  double mul(int i);
  double pot();
  double div(int i);
};
```

```
class Geometrico: public OpBasicas //Clase de operaciones geometricas
{
  double ancho;
  double largo;
  double profundidad;
  double base;
  double altura;
  float radio;

public:
  void setAncho(); //Mutadores
  void setLargo();
  void setProfundidad();
  void setBase();
  void setAltura();
  void setRadio();

  double areaRec();
  double perimetroRec();
  double volumenRec();
  double areaTria();
  double perimetroTria();
  double volumenTria();
  double areaCir();
  double perimetroCir();
  double volumenCir();
};

class Matrices: public Geometrico //Clase de operaciones matriciales
{
  double A[50][50] = {0}; //matriz de usuario 1
  double B[50][50] = {0}; //matriz de usuario 2
  double C[50][50] = {0}; //transpuesta de 1
  double D[50][50] = {0}; //transpuesta de 2
  double R[50][50] = {0}; //matriz resultado

public:
  void setData(int sizex, int sizey);
  void sumaM(int sizex, int sizey);
  void restaM(int sizex, int sizey);
  void multM(int sizex, int sizey);
  void transM(int sizex, int sizey);
  void printR(int sizex, int sizey);
};

class Vectoriales: public Matrices //Clase de operaciones vectoriales
{
  int x3,x4;
  int y3,y4;
  int z3,z4;

  public: //publico
```

```
  void setx(int x1,int x2);//MUTADORES
  void sety(int y1, int y2);
  void setz (int z1, int z2);

  int getx();//ACCESOS
  int gety();
  int getz();

  void ecuaciones();//OPERACIONES
  void distancia();
  void mvector();
  void sumavectores();
};

class Estadistica: public Vectoriales //Clase de operaciones estadisticas
{
  double A[50] = {0}; // matriz de usuario

public:
  void setData(int i);
  double media(int i);
  double moda(int i);
  double mediana(int i);

};

class Fullcalc: public Estadistica //Clase de calculadora completa
{
public:
  void OpBasic();
  void Geometry();
  void Matrix();
  void Vectorial();
  void Estadisticas();
};

int main() //interfaz de usuario
{
  int menu = 0;
  Fullcalc calculadora; //CREACION DE OBJETO DE CLASE GENERAL

  cout << "1.Basica"
  << "\n2.Geometrica"
  << "\n3.Matricial"
    << "\n4.Vectorial"
    << "\n5.Estadistica"
  << "\n0.SALIR"
  << "\n" <<endl;

  do {

    cout << "\nINGRESE TIPO DE OPERACION ";
```

```
    cin >> menu;

    switch (menu) {
      case 1: // OPERACIONES BASICAS
        calculadora.OpBasic();
      break;
      case 2: // OPERACIONES GEOMETRICAS
        calculadora.Geometry();
      break;
      case 3: // OPERACIONES MATRICIALES
        calculadora.Matrix();
      break;
      case 4: // OPERACIONES VECTORIALES
        calculadora.Vectorial();
      break;
      case 5: // OPERACIONES ESTADISTICAS
        calculadora.Estadisticas();
      break;
    }

  } while (menu>= 1 && menu <= 5);
}

void Fullcalc::OpBasic() //funciÃ³n menÃº de la calculadora bÃ¡sica
{
  int s = 0, cant = 0;
  OpBasicas calculadora; //creacion de objeto en clase basica

  cout << "1.SUMA"
  << "\n2.RESTA"
  << "\n3.MULTIPLICA"
  << "\n4.DIVIDE"
  << "\n5.POTENCIA"
  << "\n0.SALIR"
  << "\n";

  do {

  cout << "\nINGRESE SU OPERACION: ";
  cin >> s;

  switch (s) {
  case 1: // SUMA
      cout << "\nSUMA, INGRESE CANTIDAD DE NUMEROS: ";
    cin >> cant;
   cout << calculadora.add(cant) << endl;
   break;
  case 2: // RESTA
      cout << "\nRESTA, INGRESE CANTIDAD DE NUMEROS: ";
      cin >> cant;
   cout << calculadora.sub(cant) << endl;
   break;
```

```
  case 3: // MULTIPLICACION
     cout << "\nMULT, INGRESE CANTIDAD DE NUMEROS: ";
     cin >> cant;
   cout << calculadora.mul(cant) << endl;
   break;
  case 4: // DIVISION
     cout << "\nDIV, INGRESE CANTIDAD DE NUMEROS: ";
     cin >> cant;
   cout << calculadora.div(cant) << endl;
   break;
  case 5: // POTENCIA
   cout << calculadora.pot() << endl;
   break;
  }

 } while (s>= 1 && s <= 5);
}

double OpBasicas::add(int i) //OPERADOR SUMA, cantidad de datos
{
  int num = 0, result = 0;
  for (int x = 0; x < i; x++) //ciclo para el total de digitos
  {
     cout << "ingresa numero " << x << ": " << endl;
     cin >> num;
     result += num; //suma a variable resultado
  }
  return result; //regresar suma total
}

double OpBasicas::sub(int i) //OPERADOR RESTA, cantidad de datos
{
  int num = 0, result = 0;
  cout << "ingresa numero 0: "<< endl;
  cin >> num;
  result = num; //valor inical de la resta
  for (int x = 1; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
     result -= num; //resta a variable resultado
  }
  return result; //regresar resta total
}

double OpBasicas::mul(int i) //OPERADOR MUlTIPLICACION, cantidad de datos
{
  int num = 1, result = 1;
  for (int x = 0; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
```

```
    result *= num; //multiplicacion a variable resultado
  }
  return result; //regresar mult total
}

double OpBasicas::pot() //OPERADOR POTENCIA, cantidad de datos
{
  int num = 0, power = 0, result = 0;

    cout << "ingresa numero: " << endl;
    cin >> num;
    cout << "ingresa potencia: " << endl;
    cin >> power;

    return pow(num, power); //regresa potencia de los numeros dados por usuario
}

double OpBasicas::div(int i) //OPERADOR DIVISION, cantidad de datos
{
  int num = 1, result = 1;
  cout << "ingresa numero 0: " << endl;
  cin >> num;
  result = num; //primer valor de resultado para la division
  for (int x = 1; x < i; x++) //ciclo para el total de digitos -1
  {
    cout << "ingresa numero " << x << ": " << endl;
    cin >> num;
    if (num == 0){ //si el numero es 0 se tiene un error
      cout << "division entre 0" <<endl;
      return 0;
    } else {
       result /= num; // division al resultado
    }
  }
  return result; // regresar division
}

void Fullcalc::Geometry() //funciÃ³n menÃº de la calculadora geometrica
{
  int menu1 = 0, menu2 = 0;
  Geometrico calculadora; //creacion de objeto en clase geometrica

  cout << "1.RECTANGULO"
  << "\n2.TRIANGULO"
  << "\n3.CIRCULO"
  << "\n0.SALIR"
  << "\n" <<endl;

  do {

    cout << "\nINGRESE SU FIGURA ";
    cin >> menu1;
```

```
    switch (menu1) { //PRIMER SWITCH CASE PARA DETERMINAR LA FIGURA
      case 1: // RECTANGULO
        cout << "1.PERIMETRO"
         << "\n2.AREA"
         << "\n3.VOLUMEN"
         << "\n0.SALIR"
         << "\n" <<endl;
        cout << "\nINGRESE SU OPERACION ";
        cin >> menu2;

        switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
          case 1: // PERIMETRO
            calculadora.setAncho();
            calculadora.setLargo();
            cout << "RESULTADO: " << calculadora.perimetroRec() << endl;
          break;
          case 2: // AREA
            calculadora.setAncho();
            calculadora.setLargo();
            cout << "RESULTADO: " << calculadora.areaRec() << endl;
          break;
          case 3: // VOLUMEN
            calculadora.setAncho();
            calculadora.setLargo();
            calculadora.setProfundidad();
            cout << "RESULTADO: " << calculadora.volumenRec() << endl;
          break;
        }
        break;

        case 2: // TRIANGULO
          cout << "1.PERIMETRO"
            << "\n2.AREA"
            << "\n3.VOLUMEN"
            << "\n0.SALIR"
            << "\n" <<endl;
          cout << "\nINGRESE SU OPERACION ";
          cin >> menu2;

          switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
          case 1: // PERIMETRO
            calculadora.setBase();
            cout << "RESULTADO: " << calculadora.perimetroTria() << endl;
            break;
          case 2: // AREA
            calculadora.setBase();
            calculadora.setAltura();
            cout << "RESULTADO: " << calculadora.areaTria() << endl;
            break;
          case 3: // VOLUMEN
            calculadora.setBase();
            calculadora.setAltura();
            cout << "RESULTADO: " << calculadora.volumenTria() << endl;
```

```
        break;
       }
       break;

       case 3: //CIRCULO
        cout << "1.PERIMETRO"
          << "\n2.AREA"
          << "\n3.VOLUMEN"
          << "\n0.SALIR"
          << "\n" <<endl;
        cout << "\nINGRESE SU OPERACION ";
        cin >> menu2;

        switch (menu2) { //SEGUNDO SWITCH CASE PARA DETERMINAR LA
OPERACION
        case 1:
          calculadora.setRadio(); // PERIMETRO
          cout << "RESULTADO: " << calculadora.perimetroCir() << endl;
          break;
        case 2:
          calculadora.setRadio(); // AREA
          cout << "RESULTADO: " << calculadora.areaCir() << endl;
          break;
        case 3: // VOLUMEN
          calculadora.setRadio();
          cout << "RESULTADO: " << calculadora.volumenCir() << endl;
          break;
        }
        break;
     }

  } while (menu1>= 1 && menu1 <= 3);
}

/*FUNCIONES PARA OBTENER DATOS*/

void Geometrico::setAncho()
{
  int A;
  cout << "ingrese ancho: "<<endl;
  cin >> A;
  if(A<0) // si el valor ingresado es negativo, convertirlo
    A=A*-1;
  ancho = A;
}

void Geometrico::setLargo()
{
  int L;
  cout << "ingrese largo: "<<endl;
  cin >> L;
  if(L<0) // si el valor ingresado es negativo, convertirlo
```

```
   L=L*-1;
  largo = L;
}

void Geometrico::setProfundidad()
{
  int P;
  cout << "ingrese profundidad: "<<endl;
  cin >> P;
  if(P<0) // si el valor ingresado es negativo, convertirlo
    P=P*-1;
  profundidad = P;
}

void Geometrico::setBase()
{
  int B;
  cout << "ingrese base: "<<endl;
  cin >> B;
  if(B<0) // si el valor ingresado es negativo, convertirlo
    B=B*-1;
  base = B;
}

void Geometrico::setAltura()
{
  int H;
  cout << "ingrese altura: "<<endl;
  cin >> H;
  if(H<0) // si el valor ingresado es negativo, convertirlo
    H=H*-1;
  altura = H;
}

void Geometrico::setRadio()
{
  int R;
  cout << "ingrese radio: "<<endl;
  cin >> R;
  if(R<0) // si el valor ingresado es negativo, convertirlo
    R=R*-1;
  radio = R;
}
/* OPERADORES */
double Geometrico::areaRec() //AREA RECTANGULO
{
  return ancho*largo;
}
double Geometrico::perimetroRec() //PERIMETRO RECTANGULO
{
  return (ancho+largo)*2;
}
```

```
double Geometrico::volumenRec() // VOLUMEN RECTANGULO
{
 return(ancho*largo*profundidad);
}

double Geometrico::areaTria() //AREA TRIANGULO
{
 return (base*altura)/2;
}

double Geometrico::perimetroTria() // PERIMETRO TRIANGULO
{
 return base*3;
}

double Geometrico::volumenTria() //VOLUMEN TRIANGULO
{
 return((sqrt(3)/4)*(base*base)*altura);
}

double Geometrico::areaCir() //AREA CIRCULO
{

 return (3.1416*(radio*radio));
}

double Geometrico::perimetroCir() //PERIMETRO CIRCULO
{
 return ((2 * 3.1416) * radio);
}

double Geometrico::volumenCir() //VOLUMEN CIRCULO
{
 return ((4/3)*3.1416)*(pow(radio,3));
}

void Fullcalc::Matrix() //funciÃ³n menÃº de la calculadora matricial
{
 int menu = 0, sizex = 0, sizey = 0;
 Matrices calculadora; //creacion de objeto en clase matricial

 cout << "1.SUMA"
 << "\n2.RESTA"
 << "\n3.MULTIPLICACION"
  << "\n4.TRANSPOSICION"
 << "\n0.SALIR"
 << "\n" <<endl;

  do {
    cout << "\nINGRESE SU OPERACION ";
    cin >> menu;
```

```
      switch (menu) {
        case 1: // SUMA
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.sumaM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
        case 2: // RESTA
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.restaM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
        case 3: // MULTIPLICACION
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.multM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
         case 4: // TRANSPOSICION
           cout << "tamaÃ±o de matrices: "<<endl;
           cin >> sizex;
           sizey = sizex;
           calculadora.setData(sizex, sizey); //llenado de matriz
         calculadora.transM(sizex, sizey); // Suma de matriz
           calculadora.printR(sizex, sizey); // impresion de matriz
         break;
      }

    } while (menu>= 1 && menu <= 4);
}

/*FUNCIONES PARA OBTENER DATOS*/

void Matrices::setData(int sizex, int sizey) // tamaÃ±o en x, tamaÃ±o en y
{
  cout << "MATRIZ A: " <<endl;
  for (int x=0; x<sizex; x++){ //llenado de las matrices
    for (int i=0; i<sizey; i++){
         cout << "Fila: " << x << " columna: " << i <<endl;
         cin >> A[i][x];
      }
  }
```

```
  cout << "MATRIZ B: " <<endl;
  for (int x=0; x<sizex; x++){ //llenado de las matrices
    for (int i=0; i<sizey; i++){
         cout << "Fila: " << x << " columna: " << i <<endl;
         cin >> B[i][x];
      }
  }
}

/* OPERADORES */

void Matrices::sumaM(int sizex, int sizey) //SUMA, tamaÃ±o en x, tamaÃ±o en y
{
  for (int x=0; x<sizex; x++){ //SUMA MATRICIAL
    for (int i=0; i<sizey; i++){
         R[i][x] = A[i][x] + B [i][x];
      }
  }
}

void Matrices::restaM(int sizex, int sizey) //RESTA, tamaÃ±o en x, tamaÃ±o en y
{
  for (int x=0; x<sizex; x++){ //RESTA MATRICIAL
    for (int i=0; i<sizey; i++){
         R[i][x] = A[i][x] - B [i][x];
      }
  }
}
void Matrices::multM(int sizex, int sizey) //MULTIPLCACION, tamaÃ±o en x, tamaÃ±o en y
{
  for (int x=0; x<sizex; x++){ //MUlTIPLICACION MATRICIAL
    for (int i=0; i<sizey; i++){
         R[i][x] = A[i][x] * B [i][x];
      }
  }
}
void Matrices::transM(int sizex, int sizey) //TRANSPUESTA, tamaÃ±o en x, tamaÃ±o en y
{
  int i,j,aux1,aux2;
  for (int x=0; x<sizex; x++) {
      for(int i=0; i<sizey; i++) {
        //matriz1 cambio de posiciÃ³n
        aux1 = A[x][i]; //variable temporal para guardado
        C[x][i] = A[i][x];
        A[i][x] = aux1;
        //matriz2 cambio de posiciÃ³n
        aux2 = B[x][i]; //variable temporal para guardado
        D[x][i] = B[i][x];
        B[i][x] = aux2;
      }
  }
}
```

```
/*FUNCIONES PARA IMPRESION*/

void Matrices::printR(int sizex, int sizey)
{
  for (int x=0; x<sizex; x++){ //impresion de matriz resultado
    for (int i=0; i<sizey; i++){
         cout << R[i][x] <<" ";
      }
    cout <<endl;
  }
  for (int x=0; x<sizex; x++){ //impresion de matriz transpuesta 1
    for (int i=0; i<sizey; i++){
         cout << C[i][x] <<" ";
      }
    cout <<endl;
  }
  for (int x=0; x<sizex; x++){ //impresion de matriz transpuesta 2
    for (int i=0; i<sizey; i++){
         cout << D[i][x] <<" ";
      }
    cout <<endl;
  }
}

void Fullcalc::Vectorial() //funciÃ³n menÃº de la calculadora vectorial
{
  int x1=0,x2=0,y1=0,y2=0,s,z1=0,z2=0;
  Vectoriales calculadora; //creacion de objeto en clase vectorial

  cout<< "1.SISTEMA 2X2"
  << "\n2.DISTANCIA 2 PUNTOS"
  << "\n3.MAGNITUD VECTOR"
  << "\n4.SUMA DE VECTORES"
  << "\n0.SALIR"
  << "\n";

  do {
  cout << "\nINGRESE SU OPERACION ";
  cin >> s;

  switch (s) {
  case 1: // SISTEMA 2*2
   calculadora.setx(x1,x2); //captura de informacion
      calculadora.sety(y1,y2);//captura de informacion
      calculadora.setz(z1,z2);//captura de informacion
      cout<<"LA SOLUCION AL SISTEMA 2X2 ES:"<<endl;
      calculadora.ecuaciones();
   break;
  case 2: // DISTANCIA ENTRE PUNTOS
   calculadora.setx(x1,x2);//captura de informacion
      calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA DISTANCIA ENTRE DOS PUNTOS ES:"<<endl;
```

```
 calculadora.distancia();
   break;
  case 3: // MAGNITUD DE VECTOR
   calculadora.setx(x1,x2);//captura de informacion
   calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA MAGNITUD DEL VECTOR ES:"<<endl;
   calculadora.mvector();
   break;
  case 4: // SUMA DE VECTORES
   calculadora.setx(x1,x2);//captura de informacion
      calculadora.sety(y1,y2);//captura de informacion
   cout<<"LA SUMA DE VECTORES ES:"<<endl;
   calculadora.sumavectores();
   break;
  case 0:
   cout << "SALIR" << endl;
   break;
  }

 } while (s>= 1 && s <= 4);

}

/*FUNCIONES PARA OBTENER DATOS*/

void Vectoriales::setx(int x1, int x2){
 cout<<"INGRESE VALOR DE X1"<<endl;//pide valor al usuario y lo guarda
 cin>>x1;
 x3 = x1;//asigna el valor a una nueva variable
 cout<<"INGRESE VALOR DE X2"<<endl;
 cin>>x2;
 x4 = x2;
}

void Vectoriales::sety (int y1, int y2){
 cout<<"INGRESE VALOR DE y1"<<endl;//pide valor al usuario y lo guarda
 cin>>y1;
 y3= y1;//asigna valor a una nueva variable
 cout<<"INGRESE VALOR DE y2"<<endl;
 cin>>y2;
 y4= y2;
}

void Vectoriales::setz (int z1, int z2){
 cout<<"INGRESE VALOR DE RESULTADO EC1"<<endl;//pide valor al usuario y lo guarda
 cin>>z1;
 z3  = z3;//asigna valor a una nueva variable
 cout<<"INGRESE VALOR DE RESULTADO EC2"<<endl;
 cin>>z2;
 z4  = z2;//resultado
}
```

```
int Vectoriales::getx(){//regresa el valor de las nuevas variables
 return x3;
 return x4;
}

int Vectoriales::gety(){//regresa el valor de las nuevas variables
 return y3;
 return y4;
}

int Vectoriales::getz(){//Regresa el valor de las nuevas variables
 return z3;
 return z4;
}

/* OPERADORES */

void Vectoriales::ecuaciones(){
float y,x,z,xr,yr,zr;//eSolucion de ecuaciones 2x2 por métoddo de eliminacion
x=-x3*x4;
y=-x4*x3;
z=-x4*x3;

xr=x3*x4;
yr=x3*y4;
zr=x3*z4;

x=x+xr;
y=y+yr;
z=z+zr;
y=z/y;

z=z3-y*y3;
x=z/x3;

cout<<"VALOR DE Y ES:"<<y<<"VALOR DE X ES:"<<x;
}

void Vectoriales::distancia(){//distancia entre puntos por formula
   float a,b,c;
   a = (x4 - x3)*(x4 - x3);
   b = (y4 - y3)*(y4- y3);
   c= sqrt(a+b);
   cout<<"LA DISTANCIA ENTRE PUNTOS ES:"<<c<<endl;

}

void Vectoriales::mvector(){//magnitud vector por formula
   cout<<"COLOCAR VALORES DE X2,Y2=0"<<endl;
   float a,b,c;
   a = x3;
   b = y3;
```

```
   c= sqrt((a*a)+(b*b));
   cout<<"LA MAGNITUD DEL VECTOR ES:"<<c<<endl;
}

void Vectoriales::sumavectores(){//suma de vectores por formula
   float a,b;
   a = x3 + x4;
   b = y3 + y4;
   cout<<"VALOR DE X ES:"<<a<<"EL VALOR DE Y ES:"<<b;
}

void Fullcalc::Estadisticas() //funciÃ³n menÃº de la calculadora estadistica
{
  int menu = 0, n = 0;
  Estadistica calculadora; //creacion de objeto en clase estadistica

  cout << "1.MEDIA"
  << "\n2.MODA"
  << "\n3.MEDIANA"
  << "\n0.SALIR"
  << "\n" <<endl;

    do {
      cout << "\nINGRESE SU OPERACION ";
      cin >> menu;

      switch (menu) {
        case 1: // MEDIA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.media(n) << endl;
         break;
        case 2: // MODA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.moda(n) << endl;
         break;
        case 3: // MEDIANA
           cout << "numero de digitos: "<<endl;
           cin >> n;
           calculadora.setData(n); //captura de datos
         cout << "RESULTADO: " << calculadora.mediana(n) << endl;
         break;
      }

    } while (menu>= 1 && menu <= 3);
}
```

```
/*FUNCIONES PARA OBTENER DATOS*/

void Estadistica::setData(int i)
{
  for (int x = 0; x<i; x++){ //LLENADO DE MATRIZ
    cout << "digito " << x << ":"<<endl;
    cin >> A[x];
  }
}

/* OPERADORES */

double Estadistica::media(int i) // tamaÃ±o
{
  int result = 0;
  for (int x = 0; x<i; x++){ // SUMA DE TODOS LOS DATOS
    result += A[x];
  }
  return result / i; // DIVISION PARA OBTENER PROMEDIO
}

double Estadistica::moda(int i) // tamaÃ±o
{

  sort(A, A+i); // funcion para ordenar la matriz
  int temp = A[0]; // primer valor de la moda
  int moda = temp, contador1 = 1, contador2 = 1;

  for (int x = 1; x<i; x++){

    if (A[x] == temp){ // si el valor del ciclo es igual a la moda original
      contador1++; //sumar uno a contador
    } else{ // si no es igual

      if (contador1 > contador2){ // contador es mas grande que contador 2
        contador2 = contador1; // cambio de valor
        moda = temp; // moda es igual al ultimo valor en temp
      }
      contador1 = 1; // reseteo de contador 1
      temp = A[x]; // temporal es igual a nuevo valor del arreglo
    }
  }

    return moda;
}

double Estadistica::mediana(int i) // tamaÃ±o
{

  double mediana = 0;
  sort(A, A + i); // funcion para ordenar matriz
```

```
  if (i%2 == 0){ // si es par
    mediana = (A[(i/2)-1] + A[i/2])/2; // se toman los dos valores de en medio y se promedia
  } else { // si es impar
    mediana = A[(i/2)]; // se toma el valor de la mitad
  }

  return mediana;

}return mediana;

}
```

LINK AL REPOSITORIO:

https://github.com/Samara332/Calculadora/settings

CONCLUSIONES

Samara: Al terminar la práctica se reafirmaron los conceptos aprendidos a lo largo del semestre, además de agregar conceptos nuevos como los pilares de la programación orientada a objetos, que inconscientemente ya habíamos utilizado en prácticas anteriores a la materia como en proyectos de Arduino o elementos 1. Existieron problemas solo al interpretar lo requerido, puesto que en un inicio se había heredado únicamente de las clases hacia la calculadora, pero no se heredaban entre ellas, por lo que se desaprovechaba el potencial de la herencia, que es el tema central de esta práctica. La materia en general me resultó muy fructífera, realmente sentí que aprendía aún estando en línea, la programación es un tema que me gusta bastante, sin embargo, no tenía sentido de muchas cosas importantes para su aplicación. Ahora me siento más segura en cuanto al manejo de herramientas que se dan por sentadas en otras clases, además creo que a pesar de parecer un programa muy ambicioso al inicio del curso , se cubrieron todos los temas con explicaciones claras y concisas.

Ariel: El tema de programación orientada a objetos me resulta particularmente interesante puesto que es precisamente éste, el que permite diseñar aplicaciones, programas informáticos etc. que nos serán muy útiles tanto en la vida universitaria como profesional, esta práctica me pareció muy adecuada pues nos hace sirvió para poner en práctica todos los conocimientos previos y aprendidos en la clase, desde tipos de datos hasta los últimos conceptos aprendidos como el uso de clases, herencia, polimorfismo etc. Al momento de desarrollar el ejercicio no tuvimos problemas graves, únicamente una confusión, puesto que al inicio solo existía herencia de las clases a la calculadora pero no interactuaban entre ellas, se tuvo que reacomodar los programas para que no fueran independientes, pero fuera de eso no existieron mayores trabas. La materia de elementos2 me pareció muy dinámica y variada, pues no solo aprendimos c++ si no un poco de diseño de PCBs, lo cual también es una herramienta necesaria pero que no se enseña como tal en la carrera.

Fuentes:

Codingame. (2020). Punteros en C++ Manejo dinámico. octubre 5,2020, de Codingame Sitio web: https://www.codingame.com/playgrounds/51214/manejo-dinamico-de-memoria-y-polimorfismo-practica-4/punteros-en-c

R.Steven (2001). Función strlen ANSI C. octubre 5,2020, de C++ con clase Sitio web: http://c.conclase.net/librerias/?ansifun=strlen

Aprendoencasa (2019) Suma de matrices dinámicas. Octubre 5,2020 de Aprendo en casa Sitio web: https://www.aprendoencasa.com/matrices-suma-resta-y-traspuesta/

Nube programación (2020) Multiplicación de matrices en C. Octubre 5,2020 Sitio Web: http://eenube.com/index.php/ldp/c/130-multiplicacion-de-matrices-en-c