**CICLO DE VIDA DE SOFTWARE**

APLICACIÓN PARA UNA CALCULADORA CIENTÍFICA INGENIERIL

**Planeación:**

Codificar un programa que permita resolver problemas a través de una aplicación de calculadora científica ingenieril, la cual constara de diversos apartados, el primero es una breve información y ejemplos de cada componente que integra la calculadora; el segundo es la calculadora científica, en las que encontremos desde funciones básicas como suma, resta hasta funciones trigonométricas, el tercer apartado es un área de conversión entre sistemas de numeración (decimal, binario, hexadecimal y Octal) y de escalas de temperatura (Celsius, Fahrenheit, Rankine Kelvin), el cuarto apartado es dedicado a las matrices, en el cual podemos encontrar la determinante de una matriz, la matriz transpuesta, suma, resta y multiplicación de matrices, entre otras funciones más; y el quinto y último apartado es el graficador, donde podremos encontrar los apartados correspondientes para cada tipo de gráfica que deseemos utilizar (polares, paramétricas y figuras de Lissajous).

**Análisis:**

Para poder realizar dicho programa de crear una calculadora científica ingenieril, para cada apartado nos apoyaremos de clases que nos permitan una solución más eficiente, en las cuales encontraremos las siguientes estructuras de clases.

Para el apartado de calculadora científica tenemos la clase Calculadora con la siguiente estructura:

|  |
| --- |
| Calculadora() |
| SumaXY(x,y,r)  RestaXY(x,y,r)  MultiXY(x,y,r)  DiviXY(x,y,r)  PotXY(x,y,r)  FactXoY(xoy, r)  SqrtXoY(xoy,r)  PotXoY2(xoy,r)  CosXoY(xoy,r)  SinXoY(xoy,r)  TanXoY(xoy,r)  AcosXoY(xoy,r)  AsinXoY(xoy,r)  AtanXoY(xoy,r)  CoshXoY(xoy,r)  SinhXoY(xoy,r)  TanhXoY(xoy,r)  CotXoY(xoy,r)  SecXoY(xoy,r)  CscXoY(xoy,r)  ExpoXoY(xoy,r)  LnXoY(xoy,r)  Log10XoY(xoy,r)  AbsXoY(xoy,r)  RoundXoY(xoy,r)  SingXoY(xoy,r)  Fx(x,fx,r)  Funx(x,fx)  Fy(y,fy,r)  Funy(y,fy)  Fxy(x,y,fxy,r)  Funxy(x,y,fxy) |
| Métodos de acceso y modificación a los miembros de la clase. |

Para el apartado de conversiones tenemos la clase Conversiones con la siguiente estructura.

|  |
| --- |
| Calculadora() |
| Dec2Bin(dec,bin)  Bin2Dec(bin, dec)  Dec2Hexa(dec, hexa)  Hexa2Dec(Hexa, dec)  Dec2Octal(dec,, octal)  Octal2Dec(octal, dec)  Cel2Far(cel, far)  Cel2Ran(far, ran)  Cel2Kel(cel, kel)  Far2Cel(far, cel)  Ran2Cel(ran, cel)  Kel2Cel(kel, cel) |
| Métodos de acceso y modificación a los miembros de la clase. |

Para el apartado de operaciones con matrices tenemos la clase CalcuMatrices con la siguiente estructura:

|  |
| --- |
| CalcuMatrices() |
| Columnas = int m  Filas = int n;  Elementos = double[,] elem  CalcuMatrices(m,n)  Int M()  Int N()  Double[,] Elem |
| Métodos de acceso y modificación a los miembros de la clase. |

**Diseño:**

Para el primer apartado de Teoría y Ejemplos tenemos el siguiente diseño:

Interfaz de usuario gráfica, Texto, Aplicación, Word

Descripción generada automáticamente

Para el segundo apartado de Calculadora Científica tenemos el siguiente diseño:

Tabla

Descripción generada automáticamente

Para el tercer apartado de Conversiones tenemos el siguiente diseño:

Imagen que contiene Aplicación

Descripción generada automáticamente

Para el apartado de Operaciones con matrices tenemos el siguiente diseño:

Interfaz de usuario gráfica, Word

Descripción generada automáticamente

Para el ultimo apartado del Graficador tenemos el siguiente diseño:

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**Implementación:**

Para el primer apartado de Teoría y Ejemplos tenemos la implementación del siguiente código para poder observar nuestros diversos documentos:

namespace ExamenCalcuNL15

{

/// <summary>

/// Lógica de interacción para Teoria.xaml

/// </summary>

public partial class Teoria : Page

{

public Teoria()

{

InitializeComponent();

}

private void BtCalcu\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/Calculadora.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtConversiones\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/Conversiones.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtConverEjem\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/ConversionesEjemplos.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtMatrices\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/Matrices.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtMatEjem\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/MatricesEjemplos.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtGraficador\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/Graficador.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

private void BtGrafiEjem\_Click(object sender, RoutedEventArgs e)

{

string file = System.IO.Path.GetDirectoryName(System.Reflection.Assembly.GetExecutingAssembly().Location) + "/GraficadorEjemplos.xps";

XpsDocument documento = new XpsDocument(file, System.IO.FileAccess.Read);

Teoria1.Document = documento.GetFixedDocumentSequence();

}

}

}

Para el segundo apartado de Calculadora Científica tenemos la implementación de dos códigos, el de la clase Calculadora y el de la page xaml:

Implementación de la clase Calculadora

namespace ExamenCalcuNL15

{

internal class Calculadora

{

public Calculadora()

{

}

public string SumaXY(double x, double y, ref double r)

{

r = x + y;

return ("Suma");

}

public string RestaXY(double x, double y, ref double r)

{

r = x - y;

return ("Resta");

}

public string MultiXY(double x, double y, ref double r)

{

r = x \* y;

return ("Multiplicación");

}

public string DiviXY(double x, double y, ref double r)

{

r = x / y;

return ("División");

}

public string PotXY(double x, double y, ref double r)

{

r = Math.Pow(x, y);

return ("x elevado a la y");

}

public string PotYX(double x, double y, ref double r)

{

r = Math.Pow(y, x);

return ("y elevada a la x");

}

public string FactXoY(double xoy, ref double r)

{

double fact = 1;

if (xoy < 0)

{

fact = 0;

MessageBox.Show("No existe factorial de número negativo");

}

else if (xoy == 0) fact = 1;

else

{

for (int i = 1; i <= xoy; i++)

{

fact \*= i;

}

}

r = fact;

return ("Factorial de ");

}

public string SqrtXoY(double xoy, ref double r)

{

r = Math.Sqrt(xoy);

return ("Raíz cuadrada de ");

}

public string PotXoY2(double xoy, ref double r)

{

r = Math.Pow(xoy,2);

return (" elevada al cuadrado");

}

public string CosXoY(double xoy, ref double r)

{

r = Math.Cos(xoy);

return ("Coseno de ");

}

public string SinXoY(double xoy, ref double r)

{

r=Math.Sin(xoy);

return ("Seno de ");

}

public string TanXoY(double xoy, ref double r)

{

r = Math.Tan(xoy);

return ("Tangente de ");

}

public string AcosXoY(double xoy, ref double r)

{

r = Math.Acos(xoy);

return ("Arco coseno de ");

}

public string AsinXoY(double xoy, ref double r)

{

r = Math.Asin(xoy);

return ("Arco seno de ");

}

public string AtanXoY(double xoy, ref double r)

{

r = Math.Atan(xoy);

return ("Arco tangente de ");

}

public string CoshXoY(double xoy, ref double r)

{

r = Math.Cosh(xoy);

return ("Coseno hiperbólico de ");

}

public string SinhXoY(double xoy, ref double r)

{

r = Math.Sinh(xoy);

return ("Seno hiperbólico de ");

}

public string TanhXoY(double xoy, ref double r)

{

r = Math.Tanh(xoy);

return ("Tangente hiperbólica de ");

}

public string CotXoY(double xoy, ref double r)

{

r = 1 / Math.Tan(xoy);

return ("Cotangente de ");

}

public string SecXoY(double xoy, ref double r)

{

r = 1 / Math.Cos(xoy);

return ("Secante de ");

}

public string CscXoY(double xoy, ref double r)

{

r = 1 / Math.Sin(xoy);

return ("Cosecante de ");

}

public string ExpXoY(double xoy, ref double r)

{

r = Math.Exp(xoy);

return ("e a la ");

}

public string LnXoY(double xoy, ref double r)

{

r = Math.Log(xoy);

return ("Logaritmo natural de ");

}

public string Log10XoY(double xoy, ref double r)

{

r = Math.Log10(xoy);

return ("Logaritmo decimal de ");

}

public string AbsXoY(double xoy, ref double r)

{

r = Math.Abs(xoy);

return ("Valor absoluto de ");

}

public string RoundXoY(double xoy, ref double r)

{

r = Math.Round(xoy);

return ("Valor redondeado de ");

}

public string SingXoY(double xoy, ref double r)

{

r = Math.Sign(xoy);

return ("Sing de ");

}

public string fx(double x, string fx, ref double r)

{

r = funx(x, fx);

return ("Función f(x)");

}

public double funx(double x, string fx)

{

ExpressionParser parser = new ExpressionParser();

parser.Values.Add("x", x);

return (parser.Parse(fx));

}

public string fy(double y, string fy, ref double r)

{

r = funy(y, fy);

return ("Función f(y)");

}

public double funy(double y, string fy)

{

ExpressionParser parser = new ExpressionParser();

parser.Values.Add("y", y);

return (parser.Parse(fy));

}

public string fxy(double x, double y, string fxy, ref double r)

{

r = funxy(x, y, fxy);

return ("Función f(x,y)");

}

public double funxy(double x, double y, string fxy)

{

ExpressionParser parser = new ExpressionParser();

parser.Values.Add("x", x);

parser.Values.Add("y", y);

return (parser.Parse(fxy));

}

}

}

Implementación de la page xaml correspondiente:

namespace ExamenCalcuNL15

{

/// <summary>

/// Lógica de interacción para CalcuBasica.xaml

/// </summary>

public partial class CalcuBasica : Page

{

double x, y, r;

Calculadora cal;

string mensaje, fx, fy, fxy;

public CalcuBasica()

{

InitializeComponent();

cal = new Calculadora();

}

private void BtCosx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CosXoY(x, ref r) + "x";

Salida(r);

}

private void BtCosy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CosXoY(y, ref r) + "y";

Salida(r);

}

private void BtSinx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SinXoY(x, ref r) + "x";

Salida(r);

}

private void BtSiny\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SinXoY(y, ref r) + "y";

Salida(r);

}

private void BtTanx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.TanXoY(x, ref r) + "x";

Salida(r);

}

private void BtTany\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.TanXoY(y, ref r) + "y";

Salida(r);

}

private void BtAcosx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AcosXoY(x, ref r) + "x";

Salida(r);

}

private void BtAcosy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AcosXoY(y, ref r) + "y";

Salida(r);

}

private void BtAsinx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AsinXoY(x, ref r) + "x";

Salida(r);

}

private void BtAsiny\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AsinXoY(y, ref r) + "y";

Salida(r);

}

private void BtAtanx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AtanXoY(x, ref r) + "x";

Salida(r);

}

private void BtAtany\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AtanXoY(y, ref r) + "y";

Salida(r);

}

private void BtCoshx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CoshXoY(x, ref r) + "x";

Salida(r);

}

private void BtCoshy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CoshXoY(y, ref r) + "y";

Salida(r);

}

private void BtSinhx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SinhXoY(x, ref r) + "x";

Salida(r);

}

private void BtSinhy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SinhXoY(y, ref r) + "y";

Salida(r);

}

private void BtTanhx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.TanhXoY(x, ref r) + "x";

Salida(r);

}

private void BtTanhy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.TanhXoY(y, ref r) + "y";

Salida(r);

}

private void BtCotx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CotXoY(x, ref r) + "x";

Salida(r);

}

private void BtCoty\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CotXoY(y, ref r) + "y";

Salida(r);

}

private void BtSecx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SecXoY(x, ref r) + "x";

Salida(r);

}

private void BtSecy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SecXoY(y, ref r) + "y";

Salida(r);

}

private void BtCscx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CscXoY(x, ref r) + "x";

Salida(r);

}

private void BtCscy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.CscXoY(y, ref r) + "y";

Salida(r);

}

private void BtExpx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.ExpXoY(x, ref r) + "x";

Salida(r);

}

private void BtExpy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.ExpXoY(y, ref r) + "y";

Salida(r);

}

private void BtLnx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.LnXoY(x, ref r) + "x";

Salida(r);

}

private void BtLny\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.LnXoY(y, ref r) + "y";

Salida(r);

}

private void BtLog10x\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.Log10XoY(x, ref r) + "x";

Salida(r);

}

private void BtLog10y\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.Log10XoY(y, ref r) + "y";

Salida(r);

}

private void BtAbsx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AbsXoY(x, ref r) + "x";

Salida(r);

}

private void BtAbsy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.AbsXoY(y, ref r) + "y";

Salida(r);

}

private void BtRoundx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.RoundXoY(x, ref r) + "x";

Salida(r);

}

private void BtRoundy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.RoundXoY(y, ref r) + "y";

Salida(r);

}

private void BtSingx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SingXoY(x, ref r) + "x";

Salida(r);

}

private void BtSigny\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SingXoY(y, ref r) + "y";

Salida(r);

}

private void BtSuma\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SumaXY(x, y, ref r);

Salida(r);

}

private void BtResta\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.RestaXY(x, y, ref r);

Salida(r);

}

private void BtMulti\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.MultiXY(x, y, ref r);

Salida(r);

}

private void BtDivi\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.DiviXY(x, y, ref r);

Salida(r);

}

private void BtPotxy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.PotXY(x, y, ref r);

Salida(r);

}

private void BtPotyx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.PotYX(x, y, ref r);

Salida(r);

}

private void BtFactx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.FactXoY(x, ref r) + "x";

Salida(r);

}

private void BtFacty\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.FactXoY(y, ref r) + "y";

Salida(r);

}

private void BtSqrtx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SqrtXoY(x, ref r) + "x";

Salida(r);

}

private void BtSqrty\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.SqrtXoY(y, ref r) + "y";

Salida(r);

}

private void BtPotx2\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = "x" + cal.PotXoY2(x, ref r);

Salida(r);

}

private void BtPoty2\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = "y" + cal.PotXoY2(y, ref r);

Salida(r);

}

private void BtFx\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.fx(x, fx, ref r);

Salida(r);

}

private void BtFy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.fy(y, fy, ref r);

Salida(r);

}

private void BtFxy\_Click(object sender, RoutedEventArgs e)

{

Entrada();

mensaje = cal.fxy(x, y, fxy, ref r);

Salida(r);

}

private void Bt\_Borrar\_Click(object sender, RoutedEventArgs e)

{

LBsalida.Items.Clear();

}

public void Entrada()

{

try

{

x = double.Parse(Tbx.Text);

y = double.Parse(Tby.Text);

fx = Tbfx.Text;

fy = Tbfy.Text;

fxy = Tbfxy.Text;

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void Salida(double r)

{

LBsalida.Items.Add(r);

Tbmensaje.Text = mensaje;

}

}

}

Para el tercer apartado de conversiones tenemos la implementación de dos códigos, el de la clase Conversiones y el de la page xaml:

Implementación de la clase Conversiones

namespace ExamenCalcuNL15

{

internal class Conversiones

{

public Conversiones()

{

}

public void Dec2Bin(int dec, ref string bin)

{

bin = Convert.ToString(dec, 2);

}

public void Bin2Dec(string bin, ref int dec)

{

int sum = 0;

char[] array = bin.ToCharArray();

Array.Reverse(array);

for (int i = 0; i < array.Length; i++)

{

if (array[i] == '1')

{

sum += (int)Math.Pow(2, i);

}

}

dec = sum;

}

public void Dec2Hexa(int dec, ref string hexa)

{

char[] letras = { 'A', 'B', 'C', 'D', 'E', 'F' };

int divisor = 16;

int resto = 0;

hexa = "";

for (int i = dec % divisor, j = 0; dec > 0; dec /= divisor, i = dec % divisor, j++)

{

resto = i % divisor;

if (resto >= 10)

{

hexa = letras[resto - 10] + hexa;

}

else

{

hexa = resto + hexa;

}

}

}

public void Hexa2Dec(string hexa, ref int dec)

{

dec = 0;

int divisor = 16;

for (int i = 0, j = hexa.Length - 1; i < hexa.Length; i++, j--)

{

if (hexa[i] >= '0' && hexa[i] <= '9')

{

dec += (int)Math.Pow(divisor, j) \* Convert.ToInt32(hexa[i] + "");

}

else if (hexa[i] >= 'A' && hexa[i] <= 'F')

{

dec += (int)Math.Pow(divisor, j) \* Convert.ToInt32((hexa[i] - 'A' + 10) + "");

}

}

}

public void Dec2Octal(int dec, ref string octal)

{

int divisor2 = 8;

int digito = 0;

int oct = 0;

for (int i = dec % divisor2, j = 0; dec > 0; dec /= divisor2, i = dec % divisor2, j++)

{

digito = i % divisor2;

oct += digito \* (int)Math.Pow(10, j);

}

octal = oct.ToString();

}

public void Octal2Dec(string octal, ref int dec)

{

dec = 0;

int digito = 0;

int divisor = 10;

for (int i = int.Parse(octal), j = 0; i > 0; i /= divisor, j++)

{

digito = (int)i % divisor;

dec += digito \* (int)Math.Pow(8, j);

}

}

public void Cel2Far(double cel, ref double far)

{

far = (1.8 \* cel) + 32;

}

public void Cel2Ran(double far, ref double ran)

{

ran = far + 460;

}

public void Cel2Kel(double cel, ref double kel)

{

kel = cel + 273;

}

public void Far2Cel(double far, ref double cel)

{

cel = (far - 32)/1.8;

}

public void Ran2Cel(double ran, ref double cel)

{

cel = (ran - 0.6)/1.8;

cel = cel - 273;

}

public void Kel2Cel(double kel, ref double cel)

{

cel = kel - 273;

}

}

}

Implementación de la page xaml correspondiente

namespace ExamenCalcuNL15

{

/// <summary>

/// Lógica de interacción para Conversion.xaml

/// </summary>

public partial class Conversion : Page

{

Conversiones Conver;

string bin, hexa, octal;

double cel, far, ran, kel;

int dec;

public Conversion()

{

InitializeComponent();

Conver = new Conversiones();

}

private void RbtDec\_Checked(object sender, RoutedEventArgs e)

{

Lb1.Content = "Decimal";

Lb2.Content = "Binario";

Lb3.Content = "Hexadecimal";

Lb4.Content = "Octal";

}

private void RbtBin\_Checked(object sender, RoutedEventArgs e)

{

Lb1.Content = "Binario";

Lb2.Content = "Decimal";

Lb3.Content = "Hexadecimal";

Lb4.Content = "Octal";

}

private void RbtHexa\_Checked(object sender, RoutedEventArgs e)

{

Lb1.Content = "Hexadecimal";

Lb2.Content = "Decimal";

Lb3.Content = "Binario";

Lb4.Content = "Octal";

}

private void RbtOctal\_Checked(object sender, RoutedEventArgs e)

{

Lb1.Content = "Octal";

Lb2.Content = "Decimal";

Lb3.Content = "Binario";

Lb4.Content = "Hexadecimal";

}

private void BtConvertir\_Click(object sender, RoutedEventArgs e)

{

if (RbtDec.IsChecked == true)

{

EntradaDec();

Conver.Dec2Bin(dec, ref bin);

Conver.Dec2Hexa(dec, ref hexa);

Conver.Dec2Octal(dec, ref octal);

SalidaDec2(bin, hexa, octal);

}

if(RbtBin.IsChecked == true)

{

EntradaBin();

Conver.Bin2Dec(bin, ref dec);

Conver.Dec2Hexa(dec, ref hexa);

Conver.Dec2Octal(dec, ref octal);

SalidaBin2(dec, hexa, octal);

}

if (RbtHexa.IsChecked == true)

{

EntradaHexa();

Conver.Hexa2Dec(hexa, ref dec);

Conver.Dec2Bin(dec, ref bin);

Conver.Dec2Octal(dec, ref octal);

SalidaHexa2(dec, bin, octal);

}

if (RbtOctal.IsChecked == true)

{

EntradaOctal();

Conver.Octal2Dec(octal, ref dec);

Conver.Dec2Bin(dec, ref bin);

Conver.Dec2Hexa(dec, ref hexa);

SalidaOctal2(dec, bin, hexa);

}

}

public void EntradaDec()

{

try

{

dec = int.Parse(Tb1.Text);

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void EntradaBin()

{

try

{

bin = Tb1.Text;

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void EntradaHexa()

{

try

{

hexa = Tb1.Text;

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void EntradaOctal()

{

try

{

octal = Tb1.Text;

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void SalidaDec2(string bin, string hexa, string octal)

{

Tb2.Text = bin;

Tb3.Text = hexa;

Tb4.Text = octal;

}

public void SalidaBin2(int dec, string hexa, string octal)

{

Tb2.Text = dec.ToString();

Tb3.Text = hexa;

Tb4.Text = octal;

}

public void SalidaHexa2(int dec, string bin, string octal)

{

Tb2.Text = dec.ToString();

Tb3.Text= bin;

Tb4.Text = octal;

}

public void SalidaOctal2(int dec, string bin, string hexa)

{

Tb2.Text = dec.ToString();

Tb3.Text = bin;

Tb4.Text = hexa;

}

private void RbtCelsius\_Checked(object sender, RoutedEventArgs e)

{

Lb5.Content = "Celsius";

Lb6.Content = "Fahrenheit";

Lb7.Content = "Rankine";

Lb8.Content = "Kelvin";

}

private void RbtFahrenheit\_Checked(object sender, RoutedEventArgs e)

{

Lb5.Content = "Fahrenheit";

Lb6.Content = "Celsius";

Lb7.Content = "Rankine";

Lb8.Content = "Kelvin";

}

private void RbtRankine\_Checked(object sender, RoutedEventArgs e)

{

Lb5.Content = "Rankine";

Lb6.Content = "Celsius";

Lb7.Content = "Fahrenheit";

Lb8.Content = "Kelvin";

}

private void RbtKelvin\_Checked(object sender, RoutedEventArgs e)

{

Lb5.Content = "Kelvin";

Lb6.Content = "Celsius";

Lb7.Content = "Fahrenheit";

Lb8.Content = "Rankine";

}

private void BtConvertirTempe\_Click(object sender, RoutedEventArgs e)

{

if(RbtCelsius.IsChecked == true)

{

EntradaCel();

Conver.Cel2Far(cel, ref far);

Conver.Cel2Ran(far, ref ran);

Conver.Cel2Kel(cel, ref kel);

SalidaCel(far, ran, kel);

}

if(RbtFahrenheit.IsChecked == true)

{

EntradaFar();

Conver.Far2Cel(far, ref cel);

Conver.Cel2Ran(far, ref ran);

Conver.Cel2Kel(cel, ref kel);

SalidaFar(cel, ran, kel);

}

if(RbtRankine.IsChecked == true)

{

EntradaRan();

Conver.Ran2Cel(ran, ref cel);

Conver.Cel2Far(cel, ref far);

Conver.Cel2Kel(cel, ref kel);

SalidaRan(cel, far, kel);

}

if(RbtKelvin.IsChecked == true)

{

EntradaKel();

Conver.Kel2Cel(kel, ref cel);

Conver.Cel2Far(cel, ref far);

Conver.Cel2Ran(far, ref ran);

SalidaKel(cel, far, ran);

}

}

public void EntradaCel()

{

try

{

cel = double.Parse(Tb5.Text);

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void SalidaCel(double far, double ran, double kel)

{

Tb6.Text = far.ToString() + " °F";

Tb7.Text = ran.ToString() + " °R";

Tb8.Text = kel.ToString() + " °K";

}

public void EntradaFar()

{

try

{

far = double.Parse(Tb5.Text);

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void SalidaFar(double cel, double ran, double kel)

{

Tb6.Text = cel.ToString() + " °C";

Tb7.Text = ran.ToString() + " °R";

Tb8.Text = kel.ToString() + " °K";

}

public void EntradaRan()

{

try

{

ran = double.Parse(Tb5.Text);

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void SalidaRan(double cel, double far, double kel)

{

Tb6.Text = cel.ToString() + " °C";

Tb7.Text = far.ToString() + " °F";

Tb8.Text = kel.ToString() + " °K";

}

public void EntradaKel()

{

try

{

kel = double.Parse(Tb5.Text);

}

catch

{

MessageBox.Show("Error de entrada de datos...");

}

}

public void SalidaKel(double cel, double far, double ran)

{

Tb6.Text = cel.ToString() + " °C";

Tb7.Text = far.ToString() + " °F";

Tb8.Text = ran.ToString() + " °R";

}

}

}

Para el cuarto apartado de Operaciones con matrices tenemos la implementación de dos códigos, el de la clase CalcuMatrices y el de la page xaml:

Implementación de la clase CalcuMatrices

namespace ExamenCalcuNL15

{

internal class CalcuMatrices

{

private int m;

private int n;

double[,] elem;

public CalcuMatrices(int m, int n)

{

this.m = m;

this.n = n;

elem = new double[m, n];

}

public int M

{

get { return m; }

set { m = value; }

}

public int N

{

get { return n; }

set { n = value; }

}

public double[,] Elem

{

get { return elem; }

set { elem = value; }

}

}

}

Implementación de la page xaml correspondiente

namespace ExamenCalcuNL15

{

/// <summary>

/// Lógica de interacción para Matrices.xaml

/// </summary>

public partial class Matrices : Page

{

int m, n, m2 ,n2;

CalcuMatrices matA;

CalcuMatrices matB;

CalcuMatrices matC;

CalcuMatrices matIn;

CalcuMatrices mat2A;

CalcuMatrices mat2B;

CalcuMatrices mat2C;

CalcuMatrices mat2In;

CalcuMatrices matSuRe;

CalcuMatrices matMul;

TextBox[,] TbMat;

TextBox[,] TbMat2;

public Matrices()

{

InitializeComponent();

}

public void LeerMat()

{

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matA.Elem[i, j] = double.Parse(TbMat[i, j].Text);

}

}

}

public void MostrarMat()

{

string aux;

LbMat.Items.Add("Matriz A");

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n; j++)

{

aux += matA.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi.Text = "Mostrando matriz";

}

private void BtTamaño\_Click(object sender, RoutedEventArgs e)

{

m = int.Parse(Tbm.Text);

n = int.Parse(Tbn.Text);

int x = 0;

int y = -20;

matA = new CalcuMatrices(m, n);

TbMat = new TextBox[m, n];

StackPanel1.Children.Clear();

TbIndi.Text = "Tamaño de la matriz " + m + "x" + n + " establecido";

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

TbMat[i, j] = new TextBox();

TbMat[i, j].Width = 30;

TbMat[i, j].Height = 20;

TbMat[i, j].HorizontalAlignment = HorizontalAlignment.Left;

TbMat[i, j].VerticalAlignment = VerticalAlignment.Top;

if (j == 0)

{

TbMat[i, j].Margin = new Thickness(30 \* j, x, 0, 0);

}

else

{

TbMat[i, j].Margin = new Thickness(30 \* j, y, 0, 0);

}

StackPanel1.Children.Add(TbMat[i, j]);

}

}

}

private void BtMostrar\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat();

MostrarMat();

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtTranspuesta\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat();

matB = new CalcuMatrices(n, m);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

matB.Elem[i, j] = matA.Elem[j, i];

}

}

string aux;

LbMat.Items.Add("Transpuesta de la matriz A");

for (int i = 0; i < n; i++)

{

aux = "";

for (int j = 0; j < m; j++)

{

aux += matB.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi.Text = "Mostrando matriz transpuesta";

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtInversa\_Click(object sender, RoutedEventArgs e)

{

if(m==n)

{

LeerMat();

matIn = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matIn.Elem[i, j] = 0;

if (i == j)

{

matIn.Elem[i, j] = 1;

}

}

}

double pivote;

double aux;

for (int i = 0; i < m; i++)

{

pivote = matA.Elem[i, i];

for (int k = 0; k < n; k++)

{

matA.Elem[i, k] = matA.Elem[i, k] / pivote;

matIn.Elem[i, k] /= pivote;

}

for (int j = 0; j < n; j++)

{

if (i != j)

{

aux = matA.Elem[j, i];

for (int k = 0; k < n; k++)

{

matA.Elem[j, k] = matA.Elem[j, k] - aux \* matA.Elem[i, k];

matIn.Elem[j, k] = matIn.Elem[j, k] - aux \* matIn.Elem[i, k];

}

}

}

}

string aux2;

LbMat.Items.Add("Inversa de la matriz A");

for (int i = 0; i < n; i++)

{

aux2 = "";

for (int j = 0; j < m; j++)

{

aux2 += matIn.Elem[i, j] + "\t";

}

aux2 += "\t";

LbMat.Items.Add(aux2);

}

TbIndi.Text = "Mostrando matriz inversa";

}

else

{

MessageBox.Show("La matriz no es cuadrada");

}

}

private void BtDeterminante\_Click(object sender, RoutedEventArgs e)

{

if(m == n)

{

LeerMat();

double pivote, pivote1, aux;

double det = 1;

for (int i = 0; i < m; i++)

{

pivote = matA.Elem[i, i];

for (int j = i + 1; j < n; j++)

{

pivote1 = matA.Elem[j, i];

aux = pivote1 / pivote;

for (int k = 0; k < m; k++)

{

matA.Elem[j, k] = matA.Elem[j, k] - aux \* matA.Elem[i, k];

}

}

}

LbMat.Items.Add("Determinante de la matriz A");

for (int i = 0; i < n; i++)

{

det \*= matA.Elem[i, i];

}

LbMat.Items.Add(det);

TbIndi.Text = "Mostrando el determinante";

}

else

{

MessageBox.Show("La matriz no es cuadrada");

}

}

private void BtMultiplicar\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat();

matB = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matB.Elem[i, j] = double.Parse(TbMulti.Text) \* matA.Elem[i, j];

}

}

string aux;

LbMat.Items.Add("Matriz A multiplicada por " + TbMulti.Text);

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n; j++)

{

aux += matB.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi.Text = "Mostrando matriz multiplicada por " + TbMulti.Text;

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtElevar\_Click(object sender, RoutedEventArgs e)

{

if (m == n)

{

LeerMat();

matB = new CalcuMatrices(m, n);

matC = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matB.Elem[i, j] = matA.Elem[i, j];

}

}

for (int a = 0; a < (int.Parse(TbElevar.Text)) - 1; a++)

{

for (int i = 0; i < m; i++)

{

for (int j = 0; j < m; j++)

{

for (int k = 0; k < n; k++)

{

matC.Elem[i, j] += matB.Elem[i, k] \* matA.Elem[k, j];

}

}

}

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matB.Elem[i, j] = matC.Elem[i, j];

matC.Elem[i, j] = 0;

}

}

}

string aux;

LbMat.Items.Add("Matriz A elevada a la " + TbElevar.Text);

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n; j++)

{

aux += matB.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi.Text = "Mostrando matriz elevada a la " + TbElevar.Text;

}

else

{

MessageBox.Show("El número de columnas de la primera matriz debe coincidir con el número de filas de la segunda matriz");

}

}

public void LeerMat2()

{

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

mat2A.Elem[i, j] = double.Parse(TbMat2[i, j].Text);

}

}

}

public void MostrarMat2()

{

string aux;

LbMat.Items.Add("Matriz B");

for (int i = 0; i < m2; i++)

{

aux = "";

for (int j = 0; j < n2; j++)

{

aux += mat2A.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi2.Text = "Mostrando matriz";

}

private void BtTamaño2\_Click(object sender, RoutedEventArgs e)

{

m2 = int.Parse(Tbm2.Text);

n2 = int.Parse(Tbn2.Text);

int x = 0;

int y = -20;

mat2A = new CalcuMatrices(m2, n2);

TbMat2 = new TextBox[m2, n2];

StackPanel2.Children.Clear();

TbIndi2.Text = "Tamaño de la matriz " + m2 + "x" + n2 + " establecido";

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

TbMat2[i, j] = new TextBox();

TbMat2[i, j].Width = 30;

TbMat2[i, j].Height = 20;

TbMat2[i, j].HorizontalAlignment = HorizontalAlignment.Left;

TbMat2[i, j].VerticalAlignment = VerticalAlignment.Top;

if (j == 0)

{

TbMat2[i, j].Margin = new Thickness(30 \* j, x, 0, 0);

}

else

{

TbMat2[i, j].Margin = new Thickness(30 \* j, y, 0, 0);

}

StackPanel2.Children.Add(TbMat2[i, j]);

}

}

}

private void BtMostrar2\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat2();

MostrarMat2();

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtTranspuesta2\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat2();

mat2B = new CalcuMatrices(n2, m2);

for (int i = 0; i < n2; i++)

{

for (int j = 0; j < m2; j++)

{

mat2B.Elem[i, j] = mat2A.Elem[j, i];

}

}

string aux;

LbMat.Items.Add("Transpuesta de la matriz B");

for (int i = 0; i < n2; i++)

{

aux = "";

for (int j = 0; j < m2; j++)

{

aux += mat2B.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi2.Text = "Mostrando matriz transpuesta";

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtInversa2\_Click(object sender, RoutedEventArgs e)

{

if (m2 == n2)

{

LeerMat2();

mat2In = new CalcuMatrices(m2, n2);

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

mat2In.Elem[i, j] = 0;

if (i == j)

{

mat2In.Elem[i, j] = 1;

}

}

}

double pivote;

double aux;

for (int i = 0; i < m2; i++)

{

pivote = mat2A.Elem[i, i];

for (int k = 0; k < n2; k++)

{

mat2A.Elem[i, k] = mat2A.Elem[i, k] / pivote;

mat2In.Elem[i, k] /= pivote;

}

for (int j = 0; j < n2; j++)

{

if (i != j)

{

aux = mat2A.Elem[j, i];

for (int k = 0; k < n2; k++)

{

mat2A.Elem[j, k] = mat2A.Elem[j, k] - aux \* mat2A.Elem[i, k];

mat2In.Elem[j, k] = mat2In.Elem[j, k] - aux \* mat2In.Elem[i, k];

}

}

}

}

string aux2;

LbMat.Items.Add("Inversa de la matriz B");

for (int i = 0; i < n2; i++)

{

aux2 = "";

for (int j = 0; j < m2; j++)

{

aux2 += mat2In.Elem[i, j] + "\t";

}

aux2 += "\t";

LbMat.Items.Add(aux2);

}

TbIndi2.Text = "Mostrando matriz inversa";

}

else

{

MessageBox.Show("La matriz no es cuadrada");

}

}

private void BtDeterminante2\_Click(object sender, RoutedEventArgs e)

{

if (m2 == n2)

{

LeerMat2();

double pivote, pivote1, aux;

double det = 1;

for (int i = 0; i < m2; i++)

{

pivote = mat2A.Elem[i, i];

for (int j = i + 1; j < n2; j++)

{

pivote1 = mat2A.Elem[j, i];

aux = pivote1 / pivote;

for (int k = 0; k < m2; k++)

{

mat2A.Elem[j, k] = mat2A.Elem[j, k] - aux \* mat2A.Elem[i, k];

}

}

}

LbMat.Items.Add("Determinante de la matriz B");

for (int i = 0; i < n2; i++)

{

det \*= mat2A.Elem[i, i];

}

LbMat.Items.Add(det);

TbIndi2.Text = "Mostrando el determinante";

}

else

{

MessageBox.Show("La matriz no es cuadrada");

}

}

private void BtMultiplicar2\_Click(object sender, RoutedEventArgs e)

{

try

{

LeerMat2();

mat2B = new CalcuMatrices(m2, n2);

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

mat2B.Elem[i, j] = double.Parse(TbMulti2.Text) \* mat2A.Elem[i, j];

}

}

string aux;

LbMat.Items.Add("Matriz B multiplicada por " + TbMulti2.Text);

for (int i = 0; i < m2; i++)

{

aux = "";

for (int j = 0; j < n2; j++)

{

aux += mat2B.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi2.Text = "Mostrando matriz multiplicada por " + TbMulti2.Text;

}

catch

{

MessageBox.Show("Error de datos");

}

}

private void BtElevar2\_Click(object sender, RoutedEventArgs e)

{

if (m2 == n2)

{

LeerMat2();

mat2B = new CalcuMatrices(m2, n2);

mat2C = new CalcuMatrices(m2, n2);

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

mat2B.Elem[i, j] = mat2A.Elem[i, j];

}

}

for (int a = 0; a < (int.Parse(TbElevar2.Text)) - 1; a++)

{

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < m2; j++)

{

for (int k = 0; k < n2; k++)

{

mat2C.Elem[i, j] += mat2B.Elem[i, k] \* mat2A.Elem[k, j];

}

}

}

for (int i = 0; i < m2; i++)

{

for (int j = 0; j < n2; j++)

{

mat2B.Elem[i, j] = mat2C.Elem[i, j];

mat2C.Elem[i, j] = 0;

}

}

}

string aux;

LbMat.Items.Add("Matriz B elevada a la " + TbElevar2.Text);

for (int i = 0; i < m2; i++)

{

aux = "";

for (int j = 0; j < n2; j++)

{

aux += mat2B.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

TbIndi2.Text = "Mostrando matriz elevada a la " + TbElevar2.Text;

}

else

{

MessageBox.Show("El número de columnas de la primera matriz debe coincidir con el número de filas de la segunda matriz");

}

}

private void BtSumaMats\_Click(object sender, RoutedEventArgs e)

{

if (m == n)

{

LeerMat();

LeerMat2();

matSuRe = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matSuRe.Elem[i, j] = matA.Elem[i, j] + mat2A.Elem[i, j];

}

}

string aux;

LbMat.Items.Add("Suma de las matrices A y B");

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n; j++)

{

aux += matSuRe.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

}

else

{

MessageBox.Show("Las dimensiones de ambas matrices deben ser iguales para poder sumarlas");

}

}

private void BtRestaMats\_Click(object sender, RoutedEventArgs e)

{

if (m == n)

{

LeerMat();

LeerMat2();

matSuRe = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

matSuRe.Elem[i, j] = matA.Elem[i, j] - mat2A.Elem[i, j];

}

}

string aux;

LbMat.Items.Add("Resta de las matrices A y B");

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n; j++)

{

aux += matSuRe.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

}

else

{

MessageBox.Show("Las dimensiones de ambas matrices deben ser iguales para poder Restarlas");

}

}

private void BtMultiMats\_Click(object sender, RoutedEventArgs e)

{

if (m == n2)

{

LeerMat();

LeerMat2();

matMul = new CalcuMatrices(m, n);

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n2; j++)

{

for (int k = 0; k < n; k++)

{

matMul.Elem[i, j] += matA.Elem[i, k] \* mat2A.Elem[k, j];

}

}

}

string aux;

LbMat.Items.Add("Multiplicación de las matrices A y B");

for (int i = 0; i < m; i++)

{

aux = "";

for (int j = 0; j < n2; j++)

{

aux += matMul.Elem[i, j] + "\t";

}

aux += "\t";

LbMat.Items.Add(aux);

}

}

else

{

MessageBox.Show("El número de columnas de la matriz A debe coincidir con el número de filas de la matriz B");

}

}

private void BtBorrar\_Click(object sender, RoutedEventArgs e)

{

LbMat.Items.Clear();

}

}

}

Para el quinto apartado del Graficador tenemos la implementación del código de la page xaml que nos lanza el tipo de graficas que deseamos observar:

namespace ExamenCalcuNL15

{

/// <summary>

/// Lógica de interacción para Graficador.xaml

/// </summary>

public partial class Graficador : Page

{

bool ban=false;

public Graficador()

{

InitializeComponent();

}

private void BtGrafiDeter\_Click(object sender, RoutedEventArgs e)

{

GraficaEstablecida f1 = new GraficaEstablecida();

f1.Show();

}

private void BtGrafica\_Click(object sender, RoutedEventArgs e)

{

GraficaIngresada f1 = new GraficaIngresada();

f1.Show();

}

private void BtGraficaPolar\_Click(object sender, RoutedEventArgs e)

{

GraficaPolar f1 = new GraficaPolar();

f1.Show();

}

private void BtGraficaParame\_Click(object sender, RoutedEventArgs e)

{

GraficaParametrica f1 = new GraficaParametrica();

f1.Show();

}

private void BtGraficaLissajous\_Click(object sender, RoutedEventArgs e)

{

GraficaLissajous f1 = new GraficaLissajous();

f1.Show();

}

}

}

**Procesamiento:**

IDE Visual Studio