Programación Funcional

LOOKING FOR AN OPTIMAL SOLUTION WITH FUNCTIONAL PROGRAMMING

Wilder Choque Cruz
gowilder@hotmail.com

Universidad Mayor De San Andrés, Facultad de Ciencias Puras, Carrera de Informática, Web Con Programación Funcional

Resumen
La programación funcional es un lenguaje de programación declarativo donde el programador especifica lo que quiere hacer, en lugar de lidiar con el estado de los objetos. Es decir, las funciones estarían en un primer lugar y nos centraremos en expresiones que pueden ser asignadas a cualquier variable.

Al escribir el código de forma declarativa, se busca que no juegue con los objetos y sea más legible. Lo normal sería que un desarrollador tuviera que hacer un bucle, iterando y crear una lógica, pero, con el lenguaje funcional, este te da funciones que hace que se parezca más a leer y escribir que a programar.

El origen del modelo de programación funcional, pese a ser algo de relativa reciente aceptación, tiene su origen en el cálculo lambda. El cálculo lambda es un sistema desarrollado en la década de los 30 del siglo XX, donde buscaban investigar la naturaleza de las funciones y la computabilidad.

Palabras Clave: Web, Programación Funcional, Sintaxis, lenguaje de programación

Abstract
Functional programming is a declarative programming language where the programmer specifies what he wants to do, rather than dealing with the state of objects. That is, the functions would be in the first place and we will focus on expressions that can be assigned to any variable.

By writing the code in a declarative way, it is intended that it does not play with the objects and is more readable. Normally, a developer would have to loop, iterate and create logic, but with functional language, it gives you functions that make it more like reading and writing than programming.

The origin of the functional programming model, despite being something of relative recent acceptance, has its origin in the lambda calculus. The lambda calculus is a system developed in the 30s of the 20th century, where they sought to investigate the nature of functions and computability.
Keywords: Web, Functional Programming, Syntax, programming language

1. INTRODUCCIÓN

programación funcional un lenguaje de programación declarativo donde el programador especifica lo que quiere hacer, en lugar de lidiar con el estado de los objetos. Es decir, las funciones estarían en un primer lugar y nos centraremos en expresiones que pueden ser asignadas a cualquier variable.

Se busca que no juegue con los objetos y sea más legible. Lo normal sería que un desarrollador tuviera que hacer un bucle, iterando y crear una lógica, pero, con el lenguaje funcional, este te da funciones que hace que se parezca más a leer y escribir que a programar.

El origen del modelo de programación funcional, pese a ser algo de relativa reciente aceptación, tiene su origen en el cálculo lambda. El cálculo lambda es un sistema desarrollado en la década de los 30 del siglo XX, donde buscaban investigar la naturaleza de las funciones y la computabilidad.

Entonces, aprendiendo a escribir con lambdas podremos hacer uso de la programación funcional. Antes de continuar, debemos dejar claro que las lambdas no es la única forma de lenguaje funcional pero sí la más extendida y que, por ejemplo, JavaScript es la que acepta.

Puedes pensar en una expresión lambda como un método anónimo. Tiene parámetros y un cuerpo como los métodos completos, pero no tiene un nombre como un método real. Las expresiones lambda se conocen como lambdas a secas.

1.1 Teoría

Como metodología se puede definir el camino o, también, el conjunto de procesos y procedimientos que, siendo altamente racionales, son usados para lograr un determinado objetivo y un conjunto de objetivos y que propende por fortalecer las leyes o conclusiones hasta las cuales llegue la investigación científica, los conocimientos que requieran determinadas habilidades y determinados cuidados. La palabra metodología hace referencia a un término demasiado amplio dado que acude a un concepto que, mayormente, puede ser reemplazado por la palabra método.

Etimológicamente la palabra metodología significa "el estudio del camino que lleva más allá" y es por esto que esta palabra acude a un concepto propio de la investigación científica que busca cumplir los objetivos que se han planteado en la ciencia. Todo esto lleva a pensar que la metodología, como concepto, puede aproximarse a un conjunto de procedimientos que establece el conocimiento que se deriva de una investigación científica o que determina un específico rumbo doctrinal.

Por su parte, la programación funcional se puede definir como un paradigma de la programación declarativa que se basa en la definición y uso de las funciones matemáticas en contraposición (no necesariamente antagónica) a la programación imperativa. En la programación funcional priman las funciones como núcleos básicos de trabajo que posibilitan obtener resultados a partir de determinados procesos, operaciones y estructuras. En la programación imperativa su esencia está fundamentada en el concepto de estados asociado a la utilización de variables que son las que almacenan la información y cuya interacción se logra a partir de la buena utilización de instrucciones de lenguajes de programación que buscan alterar los contenidos de las variables, interactuar con ellos y operarlos de forma que, en un momento dado, se obtenga el resultado que se espera .Es de anotarse que la utilización de variables (y de su concepto de "estado" asociado) implica efectos colaterales en el desarrollo y aplicación de un algoritmo en términos de lenguajes de programación . En el caso de la programación funcional, ésas se alimentan de argumentos que son los que nutren una función para que esta funcione valga esta redundancia. La eliminación de los efectos colaterales secundarios que involucra la programación imperativa permite entender y, en determinados casos, llegar a predecir el comportamiento de un programa específico.

Los lenguajes funcionales han tenido un gran uso en lo académico ya que permiten fortalecer el concepto de función y sus características, fomentar la notación formal y establecer un nexo íntimo entre las matemáticas y la programación de computadores. Algunos de los lenguajes funcionales (o que soportan programas puramente funcionales) son Scheme, Caml,F Sharp,Scala y Haskell. La programación funcional también ha tenido grandes aplicaciones en campos como la estadística, las matemáticas y el análisis financiero. Es de anotar que la filosofía de la programación funcional (estructuración de una solución algorítmica a través de funciones, construcción de funciones sin incluir el concepto de "estados", aprovechamiento de las características de las funciones llamados, recursividad, operatividad) puede

implementarse también en lenguajes de programación imperativos e, incluso, a partir del concepto de métodos en la Programación Orientada a Objetos llegan a ser de gran utilidad en este otro paradigma.

Este articulo se aplica una comparativa de lenguajes de programación que usa estas características de programación funcional dando un enfoque como está estructurado y que tan factible es de dar soluciones a un problema. Aplicando otro punto sería un Benchmarking (comparación) entre c#, java, Python, f#, scala. Detallando que el benchmarking consiste en evaluar y analizar los procesos, productos, servicios y/o demás aspectos de otras compañías o áreas para compararlos y tomarlos como punto de referencia para tus futuras estrategias.

2. MATERIALES Y MÉTODOS/METODOLOGÍA

La metodología utilizada en el desarrollo de esta investigación se explica en la Tabla 1 en donde se explican las fases planteadas con su correspondiente descripción.

	Lenguajes de Programación
Algoritmo 1	1.-Descripcion de función.
Algoritmo 3	
Algoritmo 4	
Algoritmo 5	

Tabla 1

	Algoritmos	Pseudocodigo
1	Factorial Recursivo	FORMA RECURSIVA: función Factorial(número): si número == 0 ó 1: retornar 1; si no: retornar número*Factorial(número-1);
2	Número al cuadrado	Pseudocódigo 1. Algoritmo el cuadrado de numero 2. Var 3. A: entero; 4. B 5. Inicio 6. Escriba("ingrese numero"); 7. Lea(a); 8. Cuadrado=a*a; 9. Escriba("el resultado es:", cuadrado); 10. Fin

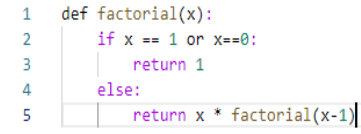
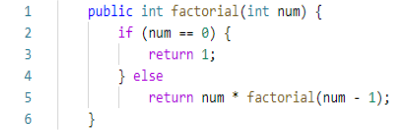
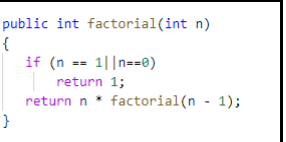
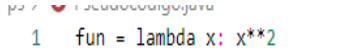
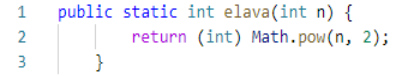
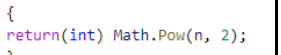
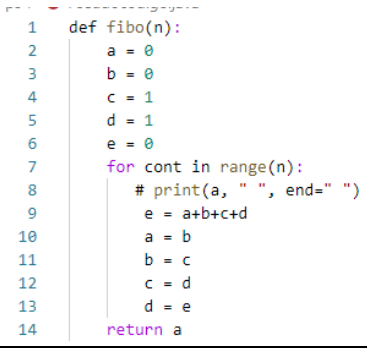
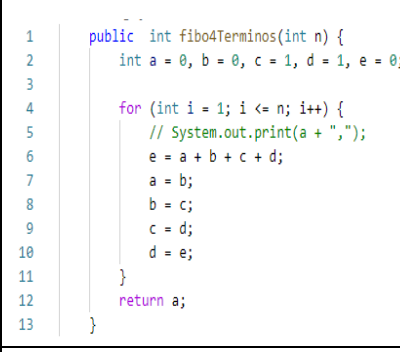
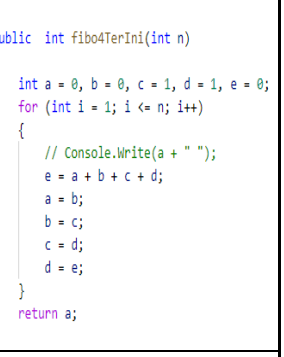
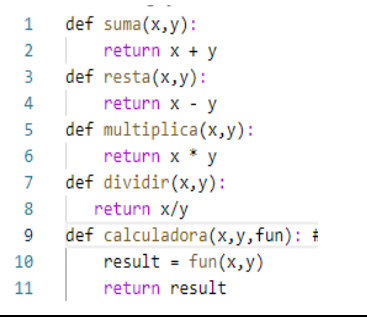
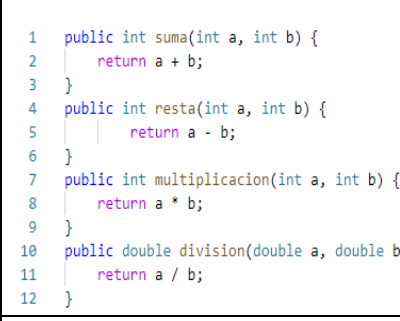
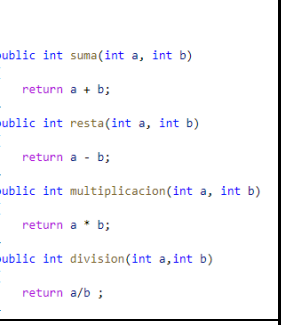
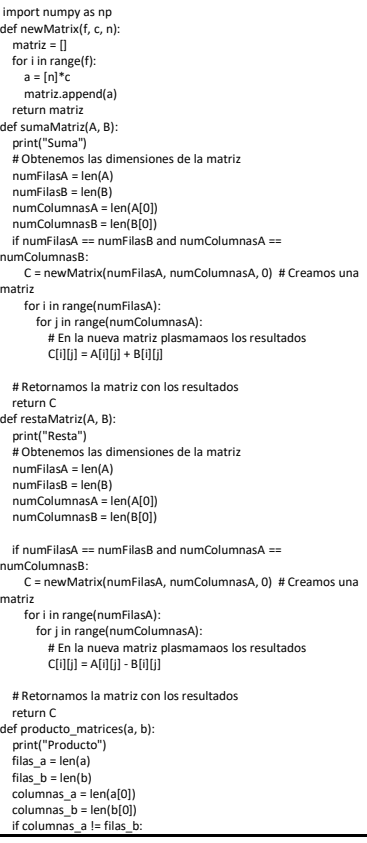
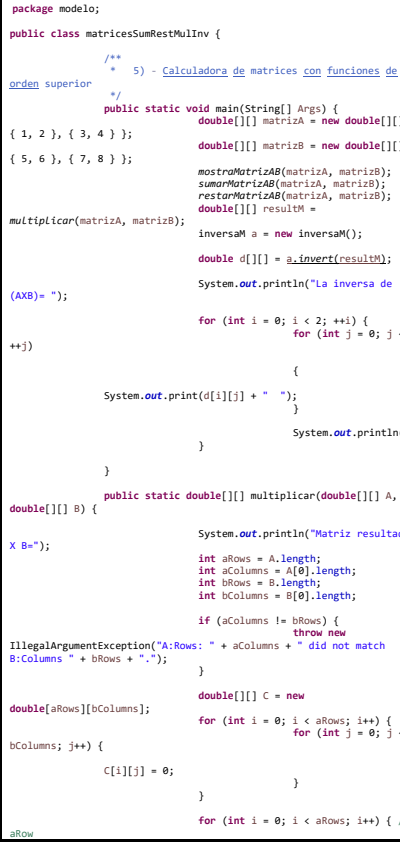
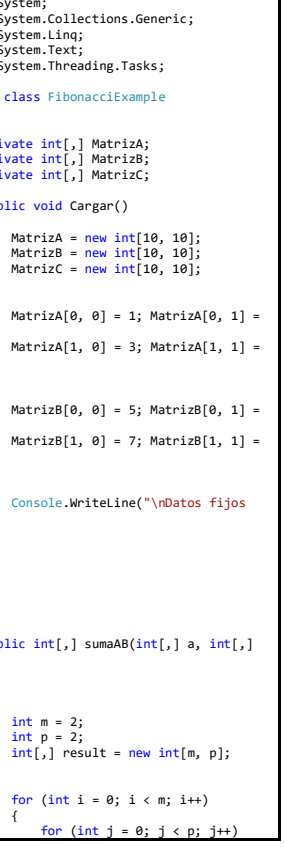
3	Fibonacci de 4 términos iniciales	1 Proceso fibo4Terminos (int n)
		2 {
		3 a = 0 Es Entero;
		4 b = 0 Es Entero;
		5 c = 1 Es Entero;
		6 d = 1 Es Entero;
		7 Mientras (int i = 1; i <= n; i++) Hacer {
		8
		9 e <-a + b + c + d;
		10 a = b;
		11 b = c;
		12 c = d;
		13 d = e;
		14
		15 }
		16 returna a;
		17 }
		18

4	Calculadora con funciones de orden superior	**Pseudolenguaje** **Inicio** **Escribir** 'Proporcione A y B' **Leer** (A y B) **Escribir** 'Opciones:' **Escribir** ' 1.- Suma' **Escribir** ' 2.- Resta' **Escribir** ' 3.- Multiplicación' **Escribir** ' 4.- División' **Escribir** ' Opción ?' **Leer** Opción **Según** Opción **sea** 1: C A+B **Escribir** 'Suma=',C 2: C A-B **Escribir** 'Resta=',C 3: **Si** B ≠ 0 **entonces** C A ÷B **Escribir** 'División=',C **sino** **Escribir** 'Error !!' **Fin_si** 4: C A x B **Escribir** 'Multiplicación =' **Fin_según** **Fin**
5	Calculadora de matrices con funciones de orden superior	**Pseudolenguaje** **Inicio** **Escribir** 'Proporcione A y B' **Leer** (A y B) **Escribir** 'Opciones:' **Escribir** ' 1.- Suma' **Escribir** ' 2.- Resta' **Escribir** ' 3.- Multiplicación' **Escribir** ' 4.- División' **Escribir** ' Opción ?' **Leer** Opción **Según** Opción **sea** 1: C A+B **Escribir** 'Suma=',C 2: C A-B **Escribir** 'Resta=',C 3: **Si** B ≠ 0 **entonces** C A ÷B **Escribir** 'División=',C **sino** **Escribir** 'Error !!' **Fin_si** 4: C A x B **Escribir** 'Multiplicación =' **Fin_según** **Fin**

Tabla 2.
Pseudocodigo

En la Tabla 3 se desarrolla estos puntos

- Lenguajes no funcionales: c#, java
- Lenguajes intermedios: Python.

	Algoritmos	Python	Java	C#
1	Factorial Recursivo			
2	Número al cuadrado			
3	Fibonacci de 4 términos iniciales			
4	Calculadora con funciones de orden superior			
5	Calculadora de matrices con funciones de orden superior			

```
    return None
  # Asignar espacio al producto. Es decir, rellenar con "espacios vacíos"
  producto = []
  for i in range(filas_b):
    producto.append([])
    for j in range(columnas_b):
      producto[i].append(None)
  # Rellenar el producto
  for c in range(columnas_b):
    for i in range(filas_a):
      suma = 0
      for j in range(columnas_a):
        suma += a[i][j]*b[j][c]
      producto[i][c] = suma
  return producto
def imprimirAB(a, b):
  print("------------A")
  show_matrix(a)
  print("------------B ")
  show_matrix(b)
  print("--------------")
def show_matrix(M):
  """ Imprime los valores almacenados en la matriz """
  filas = len(M)
  columnas = len(M[0])
  for i in range(filas):
    for j in range(columnas):
      # Imprime de una forma elegante la matriz
      print("| {0} ".format(M[i][j]), sep=',', end='')
    print('|')
matriz_a = [
  [1, 2],
  [3, 4],
]
matriz_b = [
  [5, 6],
  [7, 8],

]
imprimirAB(matriz_a, matriz_b)
def invMatriz(A):
  Minv = np.linalg.inv(A)
  return Minv
def calculadora(x, y, fun):  # function as an argument
  result = fun(x, y)
  show_matrix(result)
def calculadoral(x, fun):  # function as an argument
  result = fun(x)
  show_matrix(result)
calculadora(matriz_a, matriz_b, sumaMatriz)
calculadora(matriz_a, matriz_b, restaMatriz)
calculadora(matriz_a, matriz_b, producto_matrices)
print("-------Inversa--------")
calculadoral(producto_matrices(matriz_a, matriz_b), invMatriz)
NOTA: 101 LINEAS
```

```
                                        for (int j = 0; j < bColumns; j++) { // bColumn
                                            for (int k = 0; k < aColumns; k++) { // aColumn
                C[i][j] += A[i][k] * B[k][j];
                                            }
                                        }
                                    }
                                    for (int i = 0; i < 2; i++) {
                                        for (int j = 0; j < 2; j++)
                System.out.print(C[i][j] + " ");
                                        System.out.println();
                                    }
                                    return C;
                }

                public static void mostraMatrizAB(double[][] matrizA, double[][] matrizB) {
                                    double[][] matrizResultado;
                                    int filasA = matrizA.length;
                                    int columnasA = matrizA[0].length;

                                    double filasB = matrizB.length;
                                    double columnasB = matrizB[0].length;

                                    System.out.println("Primera matriz:A");
                                    for (int i = 0; i < filasA; i++) {
                                        for (int j = 0; j < columnasA; j++) {
                System.out.print(matrizA[i][j] + "   ");
                                        }

                System.out.println("");
                                    }

                                    System.out.println("Segunda matriz:B");
                                    for (int i = 0; i < filasB; i++) {
                                        for (int j = 0; j < columnasB; j++) {
                System.out.print(matrizB[i][j] + "   ");
                                        }

                System.out.println("");
                                    }

                }

                public static double[][] sumarMatrizAB(double[][] matrizA, double[][] matrizB) {
                                    double[][] matrizResultado;
                                    int filasA = matrizA.length;
                                    int columnasA = matrizA[0].length;

                                    double filasB = matrizB.length;
                                    double columnasB = matrizB[0].length;

                                    if (filasA == filasB && columnasB == columnasA) {

                                        matrizResultado = new double[filasA][columnasA];
                                        for (int i = 0; i < filasA; i++) {
                                            for (int j = 0; j < columnasA; j++) {
                matrizResultado[i][j] = matrizA[i][j] + matrizB[i][j];
                                            }
                                        }

                                    } else {
                                        throw new Error("Las matrices deben tener la misma cantidad de filas que columnas");
                                    }
                                    System.out.println("Matriz resultado A+B=");
                                    for (int i = 0; i < filasA; i++) {
                                        for (int j = 0; j < columnasA; j++) {
                System.out.print(matrizResultado[i][j] + "   ");
                                        }

                System.out.println("");
                                    }
                                    return matrizResultado;
                }

                public static double[][] restarMatrizAB(double[][] matrizA, double[][] matrizB) {
                                    double[][] matrizResultado;
                                    int filasA = matrizA.length;
                                    int columnasA = matrizA[0].length;

                                    double filasB = matrizB.length;
                                    double columnasB = matrizB[0].length;

                                    if (filasA == filasB && columnasB == columnasA) {

                                        matrizResultado = new double[filasA][columnasA];
                                        for (int i = 0; i < filasA; i++) {
                                            for (int j = 0; j < columnasA; j++) {
                matrizResultado[i][j] = matrizA[i][j] - matrizB[i][j];
                                            }
                                        }

                                    } else {
                                        throw new Error("Las matrices deben tener la misma cantidad de filas que columnas");
                                    }
                                    System.out.println("Matriz resultado A-B=");
                                    for (int i = 0; i < filasA; i++) {
                                        for (int j = 0; j < columnasA; j++) {
                System.out.print(matrizResultado[i][j] + "   ");
                                        }

                System.out.println("");
                                    }
                                    return matrizResultado;
                }

}
///----
package modelo;

    //Programa de ejemplo para calcular la inversa de una matriz

    import java.util.Scanner;
```

```
            {
                result[i, j] = a[i, j] + b[i, j];
            }
        }
        return result;
    }

    public int[,] restaAB(int[,] a, int[,] b)
    {

        int m = 2;
        int p = 2;
        int[,] result = new int[m, p];


        for (int i = 0; i < m; i++)
        {
            for (int j = 0; j < p; j++)
            {
                result[i, j] = a[i, j] - b[i, j];
            }
        }
        return result;
    }



    public  int[,] multiplicaAB(int[,] a, int[,] b)
    {


        int m = 2;
        int n = 2;
        int p = 2;
        int[,] result = new int[m, p];
        for (int i = 0; i < m; i++)
            for (int j = 0; j < p; j++)
            {
                int d = 0;
                for (int k = 0; k < n; k++)
                    d += a[i, k] * b[k, j];
                result[i, j] = d;
            }
        return result;
    }

    public void mostrarMatrizX(int[,] MatrizC)
    {

        for (int i = 0; i < 2; i++)
        {
            Console.Write("\n");
            for (int j = 0; j <2; j++)
            {
                Console.Write(MatrizC[i, j] + "   ");
            }
        }

        Console.ReadKey();

    }


    public static int fibo4TerIni(int n)
    {
        int a = 0, b = 0, c = 1, d = 1, e = 0;
        for (int i = 1; i <= n; i++)
        {
            // Console.Write(a + " ");
            e = a + b + c + d;
            a = b;
            b = c;
            c = d;
            d = e;
        }
        return a;

    }

    public static int factorial(int n)
    {
        if (n == 1)               // Aseguramos que termine (caso base)
            return 1;
        return n * factorial(n - 1);  // Si no es 1, sigue la recursión
    }
    public static int elevado2(int n)
    {
        return(int) Math.Pow(n, 2); ;  // Si no es 1, sigue la recursión
    }
    public static void Main(string[] args)
    {

        FibonacciExample pv = new FibonacciExample();

        pv.Cargar();
        Console.Write("\n A");
        pv.mostrarMatrizX(pv.MatrizA);
        Console.Write("\n B");
        pv.mostrarMatrizX(pv.MatrizB);
        Console.Write("\nA + B");

pv.mostrarMatrizX(pv.sumaAB(pv.MatrizA, pv.MatrizB));
        Console.Write("\nA - B");
```

```
    public class inversaM

    {
        public static void main(String argv[])
        {
            Scanner input = new Scanner(System.in);
            System.out.println("Enter the dimension of
square matrix: ");
            int n = 2;
            double a[][]= new double[n][n];

            System.out.println("Enter the elements of
matrix: ");

            a[0][0]=19; a[0][1]=22;
            a[1][0]=43; a[1][1]=50;
            double d[][] = invert(a);

            System.out.println("La inversa es: ");

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

                {System.out.print(d[i][j]+"  ");}

                System.out.println();
            }
            input.close();
        }


        public static double[][] invert(double a[][])
{

            int n = a.length;
            double x[][] = new double[n][n];
            double b[][] = new double[n][n];
            int index[] = new int[n];
            for (int i=0; i<n; ++i)
                b[i][i] = 1;

     // Transform the matrix into an upper triangle

            gaussian(a, index);

     // Update the matrix b[i][j] with the ratios stored
            for (int i=0; i<n-1; ++i)
                for (int j=i+1; j<n; ++j)
                    for (int k=0; k<n; ++k)
                        b[index[j]][k]
                                -=
a[index[j]][i]*b[index[i]][k];


     // Perform backward substitutions
            for (int i=0; i<n; ++i)             {
                x[n-1][i] = b[index[n-1]][i]/a[index[n-
1]][n-1];
                for (int j=n-2; j>=0; --j)
{
                    x[j][i] = b[index[j]][i];
                    for (int k=j+1; k<n; ++k)
                    {
                        x[j][i] -=
a[index[j]][k]*x[k][i];
                    }

                    x[j][i] /= a[index[j]][j];
                }
            }
            return x;
        }

    // Method to carry out the partial-pivoting Gaussian
    // elimination.  Here index[] stores pivoting order.

        public static void gaussian(double a[][], int
index[])  {

            int n = index.length;
            double c[] = new double[n];

     // Initialize the index
            for (int i=0; i<n; ++i)
                index[i] = i;

     // Find the rescaling factors, one from each row
            for (int i=0; i<n; ++i) {
                double c1 = 0;
                for (int j=0; j<n; ++j) {
                    double c0 = Math.abs(a[i][j]);
                    if (c0 > c1) c1 = c0;
                }
                c[i] = c1;
            }

     // Search the pivoting element from each column
            int k = 0;
            for (int j=0; j<n-1; ++j) {
                double pi1 = 0;
                for (int i=j; i<n; ++i)  {

                    double pi0 =
Math.abs(a[index[i]][j]);
                    pi0 /= c[index[i]];
                    if (pi0 > pi1) {
                        pi1 = pi0;
                        k = i;
                    }
                }


       // Interchange rows according to the pivoting
order
                int itmp = index[j];
                index[j] = index[k];
                index[k] = itmp;
```

```
pv.mostrarMatrizX(pv.restaAB(pv.MatrizA,
pv.MatrizB));
        Console.Write("\nA X B");

pv.mostrarMatrizX(pv.multiplicaAB(pv.Matriz
A, pv.MatrizB));




        int n = 6; string cad = "";
        for (int i = 0; i < n; i++)
        {
            cad = cad + fibo4TerIni(i) +
",";
        }
        Console.Write("fac="+factorial(5) +
" ");
        Console.Write("n^2=" +elevado2(4) +
" ");
        Console.Write(cad + " ");

        Console.ReadKey();
    }
}
//----
using System;

namespace Rosetta
{
    internal class Vector
    {
        private double[] b;
        internal readonly int rows;

        internal Vector(int rows)
        {
            this.rows = rows;
            b = new double[rows];
        }

        internal Vector(double[] initArray)
        {
            b =
(double[])initArray.Clone();
            rows = b.Length;
        }

        internal Vector Clone()
        {
            Vector v = new Vector(b);
            return v;
        }

        internal double this[int row]
        {
            get { return b[row]; }
            set { b[row] = value; }
        }

        internal void SwapRows(int r1, int
r2)
        {
            if (r1 == r2) return;
            double tmp = b[r1];
            b[r1] = b[r2];
            b[r2] = tmp;
        }

        internal double norm(double[]
weights)
        {
            double sum = 0;
            for (int i = 0; i < rows; i++)
            {
                double d = b[i] *
weights[i];
                sum +=  d*d;
            }
            return Math.Sqrt(sum);
        }

        internal void print()
        {
            for (int i = 0; i < rows; i++)
                Console.WriteLine(b[i]);
            Console.WriteLine();
        }

        public static Vector operator-
(Vector lhs, Vector rhs)
        {
            Vector v = new
Vector(lhs.rows);
            for (int i = 0; i < lhs.rows;
i++)
                v[i] = lhs[i] - rhs[i];
            return v;
        }
    }

    class Matrix
    {
        private double[] b;
        internal readonly int rows, cols;

        internal Matrix(int rows, int cols)
        {
            this.rows = rows;
            this.cols = cols;
            b = new double[rows * cols];
        }

        internal Matrix(int size)
        {
            this.rows = size;
            this.cols = size;
```

```
                for (int i=j+1; i<n; ++i) {
                    double pj =
a[index[i]][j]/a[index[j]][j];

     // Record pivoting ratios below the diagonal
                    a[index[i]][j] = pj;

     // Modify other elements accordingly
                    for (int l=j+1; l<n; ++l)
                        a[index[i]][l] -=
pj*a[index[j]][l];
                }
            }
        }
    }
```

```
            b = new double[rows * cols];
            for (int i = 0; i < size; i++)
                this[i, i] = 1;
        }

        internal Matrix(int rows, int cols,
double[] initArray)
        {
            this.rows = rows;
            this.cols = cols;
            b =
(double[])initArray.Clone();
            if (b.Length != rows * cols)
throw new Exception("bad init array");
        }

        internal double this[int row, int
col]
        {
            get { return b[row * cols +
col]; }
            set { b[row * cols + col] =
value; }
        }

        public static Vector
operator*(Matrix lhs, Vector rhs)
        {
            if (lhs.cols != rhs.rows) throw
new Exception("I can't multiply matrix by
vector");
            Vector v = new
Vector(lhs.rows);
            for (int i = 0; i < lhs.rows;
i++)
            {
                double sum = 0;
                for (int j = 0; j <
rhs.rows; j++)
                    sum += lhs[i,j]*rhs[j];
                v[i] = sum;
            }
            return v;
        }

        internal void SwapRows(int r1, int
r2)
        {
            if (r1 == r2) return;
            int firstR1 = r1 * cols;
            int firstR2 = r2 * cols;
            for (int i = 0; i < cols; i++)
            {
                double tmp = b[firstR1 +
i];
                b[firstR1 + i] = b[firstR2
+ i];
                b[firstR2 + i] = tmp;
            }
        }

        //with partial pivot
        internal bool InvPartial()
        {
            const double Eps = 1e-12;
            if (rows != cols) throw new
Exception("rows != cols for Inv");
            Matrix M = new Matrix(rows);
//unitary
            for (int diag = 0; diag < rows;
diag++)
            {
                int max_row = diag;
                double max_val =
Math.Abs(this[diag, diag]);
                double d;
                for (int row = diag + 1;
row < rows; row++)
                    if ((d =
Math.Abs(this[row, diag])) > max_val)
                    {
                        max_row = row;
                        max_val = d;
                    }
                if (max_val <= Eps) return
false;
                SwapRows(diag, max_row);
                M.SwapRows(diag, max_row);
                double invd = 1 /
this[diag, diag];
                for (int col = diag; col <
cols; col++)
                {
                    this[diag, col] *=
invd;
                }
                for (int col = 0; col <
cols; col++)
                {
                    M[diag, col] *= invd;
                }
                for (int row = 0; row <
rows; row++)
                {
                    d = this[row, diag];
                    if (row != diag)
                    {
                        for (int col =
diag; col < this.cols; col++)
                        {
                            this[row, col]
-= d * this[diag, col];
                        }
                        for (int col = 0;
col < this.cols; col++)
                        {
                            M[row, col] -=
```

				d * M[diag, col];
				}
				}
				}
				}
				b = M.b;
				return true;
				}
				
				internal void print()
				{
				for (int i = 0; i < rows; i++)
				{
				for (int j = 0; j < cols; j++)
				Console.Write(this[i,j].ToString()+" ");
				Console.WriteLine();
				}
				}
				}
				
				}
				
				namespace Rosetta
				{
				class Program
				{
				static void Main(string[] args)
				{ //
				Matrix M = new Matrix(2, 2, new double[] { 19,22,43,50});
				M.InvPartial();
				M.print();
				Console.ReadKey();
				}
				}
				}
				//12,5 -5,49999999999999
				//-10,75 4,74999999999999

Tabla 3.

En la Tabla 4 se desarrolla este punto.

- Lenguajes funcionales: f#, scala

	Algoritmos	Scala	F#

1	Factorial Recursivo	`def factorial(n: Int): Int = {`
		`  if (n == 0)`
		`  return 1`
		`  else`
		`  return n * factorial(n-1)`
		`}`
			`let rec factorial n =`
			`  match n with`
			`  | 0 | 1 -> 1`
			`  | _ -> n * factorial(n-1)`

2	Número al cuadrado	`val eleva = (x:Int) => scala.math.pow(x,2)`	`let cuadrado = fun x -> pown x 2`

3	Fibonacci de 4 términos iniciales	`def fib2( n : Int ) : Int = {`
		`  var a = 0`
		`  var b = 0`
		`  var c = 1`
		`  var d = 1`
		`  var e = 0`
		
		`  for(i <- 1 to n){`
		`  e=a+b+c+d;`
		`  a = b;`
		`  b = c;`
		`  c = d;`
		`  d = e;`
		`  }`
		`  return a`
		`}`
			`let rec fib n =`
			`  match n with`
			`  | 0 -> 0`
			`  | 1 -> 0`
			`  | 2 -> 1`
			`  | 3 -> 1`
			`  | _ -> fib (n - 1) + fib (n - 2)+ fib (n - 3)+ fib (n - 4)`

4	Calculadora con funciones de orden superior		

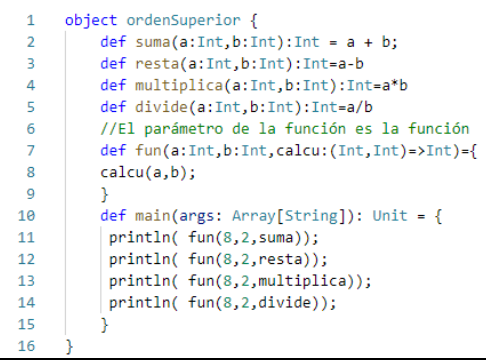
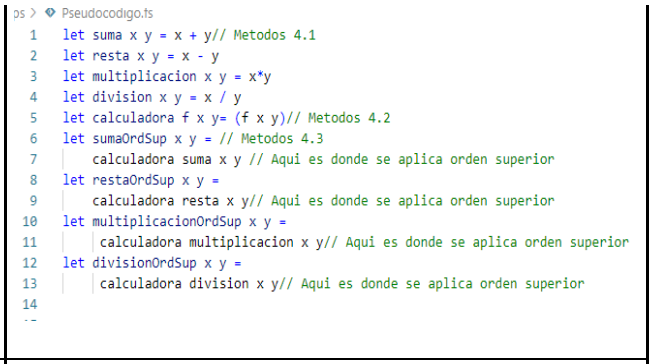
			

Tabla 4.

3. RESULTADOS Y DISCUSIÓN

En referencia con la metodología descrita en la Tabla 1, se explican en la Tabla 3 y Tabla 4 los objetivos o propósitos que se persiguieron con cada una de las fases implementadas para el desarrollo de la investigación.

En referencia con los temas abordados, que se presentan en la Tabla 3 vale la pena anotar que la selección de dichos temas (resolución de problemas, notación computacional y operadores) se pude decir que los códigos son más robustos y no dan un detalle de la estructura a solución es muy abstracto y no hay tanta facilidad de tener una análisis concreto que hace dicho código de manera específica Python es más accesible intuitivo más fácil de codificar y analizar hay mayor documentación el único detalle son las librerías que a un no hay documentación certera pues Django hay poca documentación respecto a sus librerías cabe decir que Django es más intuitivo y fácil de manejar rápido y su estructura de modelo vista controlado es concreto ,pasando a java es código extenso hay mucha documentación pero en la parte se programación funcional recién se está incursionando pues el java 8 se está implementando, referente al código es más procesos extensos mucho detalle otro aporte seria para la parte de web el uso serveles pues hace una conexión con el servidor es muy tedioso pues es el intermediario ,jsp es código extenso nada fácil de manejar pide bastantes importaciones de librerías código nada fácil de manejar. Por último, C# es un leguaje similar a java, pero en las llamadas a una función son muy tediosas no hay una estructura accesible, pasando a la parte Web es accesible amigable de manejar hay herramientas fáciles de manejar solo es importante tener una estructura definida para no tener problemas a la hora de hacer compilar las librerías.

En referencia con los temas abordados, que se presentan en la Tabla 4 se pude comentar que de manera comparativa que si pone a evaluar entre Scala y F# son parecidos en estructura, pero el que es más fácil de aprender es Scala por su sintaxis fácil de reconocer y más fácil de resolver un problema pues en la tabla 4 se evidencia sus soluciones elegantes concretas bien estructuradas conjuntos de temas bien planteadas al problema, Menos código y mejor solución.

Dando un breve aporte de otros leguajes quiero mostrar una Tabla 5 el cual refleja el uso mundial de otros lenguajes de Programación Funcional.

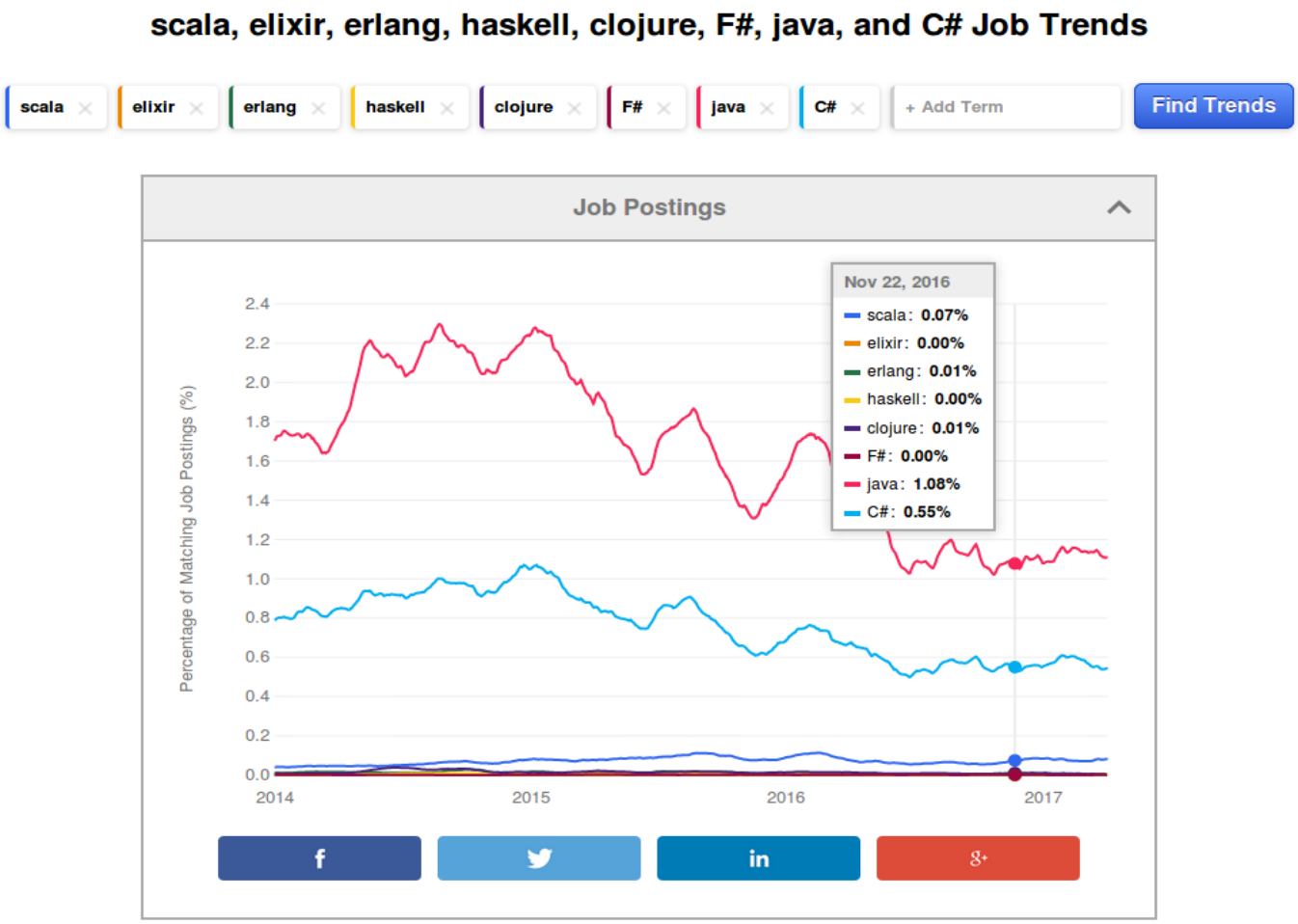


Tabla 5.

De manera general se realiza comparativa.

Ventajas

- Altos niveles de abstracción: El código muestra un mayor énfasis en el "¿qué se hace?" en lugar del "¿cómo se hace?".
- Código declarativo y comprensible: Debido a los altos niveles de abstracción, los programas que aplican este paradigma suelen ser más cortos y fáciles de entender que sus versiones en programación imperativa.
- La evaluación perezosa: Esta estrategia de evaluación permite realizar cálculos por demanda, evitando gasto computacional innecesario. El ejemplo más claro está en la utilización de listas infinitas.
- Las características del paradigma, en especial la utilización de funciones puras, permiten realizar ciertas optimizaciones particulares.

Desventajas

- Dificultad inicial para producir buen código: Esto debido a que un programador suele estar acostumbrado al pensamiento de la programación imperativa, tomando un poco de tiempo que la persona logre adaptarse y generar código útil.
- Generación de grandes cantidades de short-lived garbage: Esto se debe principalmente a la característica de inmutabilidad. Los garbage collectors tienden a optimizar este aspecto.

- Menor eficiencia en el uso de CPU comparados con su contraparte imperativa: Debido principalmente a que muchas estructuras de variables mutables (como los arreglos) tienen una sencilla implementación en un paradigma imperativo, mientras que en la programación funcional no es fácil crear componentes homólogos inmutables con la misma eficiencia.

4. CONCLUSIONES

En definitiva, el lenguaje de programación funcional se trata de una forma de ocuparse muy útil en casos determinados, como, por ejemplo, si estamos desarrollando un proyecto en el que trabajan varias personas o que heredará otro equipo, porque es una forma muy intuitiva y sencilla de programar.

REFERENCIAS

1] Vega, A.M. y Espinel, A., Aspectos fundamentales para la enseñanza de la programación básica en ingeniería. Revista Avances en Sistemas e
Informática, 7, pp. 7-13. 2010.
[2] Fincher. S., ¿What are we doing when we teach programming? 29th ASEE/IEEE Frontiers in Education Conference. San Juan, Puerto Rico.
10-13 de Noviembre de 1999, Sesión 12ª4.

Enlaces de Investigación:
http://ferestrepoca.github.io/paradigmas-de-programacion/progfun/funcional_teoria/index.html
https://es.wikipedia.org/wiki/Programaci%C3%B3n_funcional
https://www.genbeta.com/desarrollo/el-resurgir-de-la-programacion-funcional
https://www.djangoproject.com/
https://visualstudio.microsoft.com/es/vs/community/