Proyecto final INF-319 Programación funcional

INF-319 Functional Programming final Project

Alfredo Adolfo Alvarez Acuña

Alfredoaldaralvarez@gmail.com

Universidad mayor de san andres facultad de ciencias puras y naturales carrera de informática

Resumen

En el siguiente articulo compararemos el código de cinco programas, y cada uno estará escrito en cinco lenguajes diferentes, y estos lenguajes se dividirán en lenguajes no funciones, lenguajes intermedios y lenguajes funcionales. Primero veremos el pseudocodigo y luego haremos una pequeña comparación con los lenguajes de programación.

Los lenguajes funcionales tienen su complejidad ya que se necesita tener recursividad para poder entender este tema en los lenguajes no funcionales solo se llama al método o función que se desea llamar por eso es importante conocer que es recursividad y como se utiliza

Palabra Clave: programación funcional, lenguajes de programación

Abstract

In the next article we will compare the code of five programs, and each one will be written in five different languages, and these languages will be divided into non-functional languages, intermediate languages and functional languages. First we will see the pseudocode and then we will do a little comparison with the programming languages.

Functional languages have their complexity since it is necessary to have recursion to be able to understand this issue in non-functional languages, only the method or function that you want to call is called, so it is important to know what recursion is and how it is used

Keyword: functional programming, programming languages

1. INTRODUCCION

1.1. Programacion Funcional

Para entender mejor la programación funcional, debemos tener claro que con ella resolvemos a la pregunta ¿Qué? mientras que la programación imperativa responde a la pregunta ¿Cómo?

Al responder la pregunta del "¿Qué?", nos enfocamos en el resultado y no en el procedimiento. Esto implica un nivel mayor de abstracción, pero también que la programación es independiente del contexto.

Uno de los principios del paradigma es hacer que las funciones sean lo más específicas posible. De esta manera se cumple otro de los principios de este paradigma: la reutilizacion de código -pues, como veremos, las funciones retornarán lo mismo siempre a lo largo de toda la ejecución del programa-

1.2. Paradigma declarativo

La intención es el que se hace, más no el cómo se hace.

Una de las preguntas que todo programador de calidad debe hacerse antes de empezar a trabajar es ¿Que tecnología es la mejor para solucionar este problema? Una pregunta complicada pero que con la respuesta indicada puede facilitar mucho el trabajo, es por ello que usualmente se opta por usar una tecnología con la cual se encuentre familiarizado, es por ello que la programación declarativa no se nos suele pasar por la cabeza, ya que usualmente usamos la programación imperativa.

¿De que se trata la programación declarativa?

En el paradigma de programación declarativa a diferencia del paradigma de programación imperativa el programa se describe en términos de su respuesta, más no en el conjunto de secuencias e instrucciones

para llegar a ella, esto puede llegar a ser complicado debido a que usualmente los algoritmos programados se realizan en torno a los pasos para la solución de un problema específico, pero para el paradigma declarativo es más importante tener bien definido un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones las cuales modelan la solución al problema en cuestión.

1.2.1. DECLARATIVO

- Control del programador es nulo en la secuencia del programa
- Se enfoca en el que
- Lista como estructura fundamental
- No hay asignación destructiva
- Solo existen valores y expresiones matemáticas que generan nuevos valores desde los ya declarados
- El control no es responsabilidad del programador

1.2.2. IMPERATIVO

- El programador tiene control sobre la secuencia del programa
- Se enfoca en el cómo
- Se fija en como realizar operaciones ayudándose de patrones de control de flujo
- Asignación destructiva
- Presenta efectos colaterales
- El control por parte del programador es excesivo

2. MATERIALES Y METODOS/METODOLOGIA

Se tuvo que resolver 5 problemas y se programa en 5 lenguajes distintos donde veremos a los lenguajes no funcionales y funcionales.

Como primera herramienta se utilizó Eclipse para java, Visual studio para c#, PowerShell para scala y Visual Studio Code para Python también se utilizo aplicaciones de consola también se utilizo un navegador para poder correr las aplicaciones en web y ver su funcionamiento

2.1. Serie Fibonacci

2.1.1. Pseudocodigo

```
Función Fibonacci(n):
        a = 0
        b = 1
        for i < 0 to n:
                res + = ”,”
                aux = a
                a=b
                b = aux+a

                print(a)

        return res
```

2.1.2. Algoritmo Fibonacci 2 terminos

Tabla Comparativa de código escrito entre los distintos lenguajes:

Pseudocodigo

```
Función Fibonacci(n):
        a = 0
        b = 0
        c = 1
        d = 1
        for i < 0 to n:
                f = a+b+c+d
                a=b
                b = c
```

```
c = d
d = f

print(f)

return res
```

	Python	Java	C#	Scala

Fibonacci	def func_fibo(n):
	  a = 0
	  b = 1
	  res = "0"
	  for i in range(n):
	  res += ", "
	  aux = a
	  a = b
	  b = aux+a
	  res += str(a)
	return res
		lim = n
		a = 0;
		b = 1;
		out.print("0, ");
		for (int i = 2; i <= lim; i++)
		{
		  aux = a;
		  a = b;
		  b = aux + a;
		  out.print(a + ", ");
		  if (i % 10 == 0 && i != 0)
		out.print("
		");
		  }
			int n ;
			int a = -1;
			int b = 1;
			int c;
			String aux = "";
			for (int i = 1; i <= n; i++)
			{
			  c = a + b;
			  a = b;
			  b = c;
			  aux = aux + c + ", ";
			  }
				def fib1(n : Int) : Int =
				n match {
				case 0 | 1 => n
				case _ => fib1(n-1) +
				fib1(n-2)
				}

2.1.3. Algoritmo Primos

	Python	Java	C#	Scala

Primos	def func_primo(n):
	  a = 2
	  res =""
	  while n!=0:
	  b=0
	  for i in range(1,a):
	  if(a%i==0):
	  b=b+1
	  if(b==1):
	  n=n-1
	  res += ", "
	  res += str(a)
	  a=a+1
	  return res
		int lim,p1=1;
		while(lim!=0){
		  int p2 = 0;
		  for(inti=1;i<=p1;i++){
		  if(p1%i==0){
		  p2++;
		  }
		}
		if(p2==2){
		  lim--;
		out.print(p1+", ");
		  }
		p1++;
		}
			int p1 = 1;
			String p3 = "";
			while (n1!=0) {
			int p2 = 0;
			for (int i = 1; i <= p1; i++) {
			  if (p1 % i == 0) {
			  p2++;
			  }
			  }
			  if (p2 == 2) {
			  n1--;
			  p3 = p3 + p1+", ";
			  }
			  p1++;}
				def fibo4(n: Int): Int = {
				  n match {
				  case 0 | 1 | 2 => 0
				  case 3 => 1
				  case _ => fibo4(n - 1) +
				fibo4(n - 2) + fibo4(n - 3) +
				fibo4(n - 4)
				  }
				}

2.1.4. Algoritmo Fibonacci 4 terminos

	Python	Java	C#	Scala

Fibonacci 4 terminos	def func_fibo2(n):
	  a = 0
	  b = 0
	  c = 1
	  d = 1
	  res ="0, 0, 1, 1"
	  if(n==1):
	  return "0"
	  else:
	  if(n==2):
	  return "0, 0"
	  else:
	  if(n==3):
	  return "0, 0,
	1"
	  else:
	  if(n==4):
	  return "0, 0,
	1, 1"
	  else:
		int a = 0;
		  int b = 0;
		  int c = 1;
		  int d = 1;
		  int f;
		  if(n==1){
		  }
		  else{
		  if(n==2){
		  out.print("0, 0"+
		", ");}
		else{if(n==3){out.print("0,
		0, 1" + ", ");
		}
		else{
		  if(n==4){
		out.print("0, 0, 1, 1" + ",
		");
		}
			int a = 0;
			  int b = 0;
			  int c = 1;
			  int d = 1;
			  int f;
			  String aux = "0, 0,
			1, 1, ";
			  if (n == 1){
			Label9.Text = "0".ToString();
			}
			else {if (n == 2){
			Label9.Text = "0,
			0".ToString();
			}else {
			if (n == 3)
			{Label9.Text = "0, 0,
			1".ToString();}
			else {
				def fibo4(n: Int): Int = {
				  n match {
				  case 0 | 1 | 2 => 0
				  case 3 => 1
				  case _ => fibo4(n - 1) +
				fibo4(n - 2) + fibo4(n - 3) +
				fibo4(n - 4)
				  }
				}

	n=n-3
	for i in range(1,n):
	
	f=a+b+c+d
	a=b
	b=c
	c=d
	d=f
	res =res+ ", "
	res =res+ str(f)
	return res
		else{out.print("0, 0, 1, 1" + ", ");
		n = n-4;
		for(int i=1;i<=n;i++){
		f = a+b+c+d;
		a=b;
		b=c;
		c=d;
		d=f;
		out.print(f + ", ");
		}
		}
		}
		}
		}
			if (n == 4){
			Label9.Text = "0, 0, 1, 1".ToString();}
			else {n = n - 4;
			for (int i = 1; i <= n; i++)
			{f = a + b + c + d;
			a = b;
			b = c;
			c = d;
			d = f;
			aux = aux + f + ", ";
			}
			Label9.Text = aux.ToString();
			}}}}	

2.1.5. Algoritmos Calculadora(suma,resta,multiplicacion,division)

	Python	Java	C#	Scala

Calculadora	def func_calcu(a,b,func):
	res = func(a,b)
	return str(res)
	
	def inter(a,b,op):
	if (op == 'suma'):
	return func_calcu(a,b,suma)
	elif (op == 'resta'):
	return func_calcu(a,b,resta)
	elif (op == 'mult'):
	return func_calcu(a,b,mult)
	elif (op == 'div'):
	return func_calcu(a,b,div)
	else:
	return "Operacion no reconocida"
		else if(request.getParameter("a_calcu") != null && request.getParameter("b_calcu") != null && request.getParameter("op_calcu") != null){out.print(" Calculadora ");
		String op = request.getParameter("op_calcu");
		int a =
		int b =
		if(op.equals("suma") || op.equals("resta") || op.equals("multi") || op.equals("divi")){
		out.print(op+"
		");
		Callable cmd = new Calcu();
		int res = invoke(cmd, a, b, op);
		out.print(res);
		}
		else out.print("Operacion no reconocida");
		}
			public int operación(Func<int, int, int> op, int num1, int num2)
			{
			return op(num1, num2);
			}
			public int suma(int number1, int number2)
			{
			return number1 + number2;
			}
			public int resta(int number1, int number2)
			{
			return number1 - number2;
			}
			public int mult(int number1, int number2)
			{
			return number1 * number2;
			}
			public int div(int number1, int number2)
			{
			return number1 / number2;
			}
				def calculadora(operacion: String): (Int, Int) => Int = {
				operacion match {
				case "suma" => (x: Int, y: Int) => x + y
				case "resta" => (x: Int, y: Int) => x - y
				case "multiplicacion" => (x: Int, y: Int) => x * y
				case "division" => (x: Int, y: Int) => x / y
				}
				}

2.1.6. Calculadora Matriz

	Python	Java	C#	Scala

Calculadora Matriz	def sumat(m1, m2):
	res = [[0,0],[0,0]]
	cad = ""
	for i in range(2):
	for j in range(2):
	res[i][j] = m1[i][j] + m2[i][j]
		String op = request.getParameter("op_matrices");
		out.print(op+"
		");
		int a = Integer.parseInt(request.getParameter("mat100"));
			{
			return op(mat1, mat2);
			}
			
			//operaciones
				def sumaMatriz(ma: Array[Array[Int]], mb: Array[Array[Int]]): Array[Array[Int]] = {
				var mc = Array.ofDim[Int](2, 2)

```
            cad +=
str(res[i][j]) + " "
        cad+="<br>"
    return cad

def resmatri(m1, m2):
    res = [[0,0],[0,0]]
    cad = ""
    for i in range(2):
        for j in range(2):
            res[i][j] =
m1[i][j] - m2[i][j]
            cad +=
str(res[i][j]) + " "
        cad+="<br>"
    return cad

def mulmat(m1, m2):
    res = [[0,0],[0,0]]
    cad = ""
    for i in range(2):
        for j in range(2):
            for k in
range(2):
                res[i][j] +=
m1[i][k] * m2[k][j]
            cad +=
str(res[i][j]) + " "
        cad+="<br>"
    return cad
```

```
            int b =
Integer.parseInt(request.getParameter("
mat101"));
            int c =
Integer.parseInt(request.getParameter("
mat110"));
            int d =
Integer.parseInt(request.getParameter("
mat111"));
            int a1 =
Integer.parseInt(request.getParameter("
mat200"));
            int b1 =
Integer.parseInt(request.getParameter("
mat201"));
            int c1 =
Integer.parseInt(request.getParameter("
mat210"));
            int d1 =
Integer.parseInt(request.getParameter("
mat211"));

            int[][] m1 = {{a,b},{c,d}};
            int[][] m2 =
{{a1,b1},{c1,d1}};
    if(op.equals("suma") ||
op.equals("resta") ||
op.equals("multi")){
            Callable cmd = new Calcu();
        int[][] res = mat(cmd, m1, m2, op);
        for(int i=0; i<2; i++){
            for(int j=0; j<2; j++){
                    out.print(res[i][j]+"
");
            }
            out.print("<br>");
        }
    }
    else out.print("Operacion no
reconocida");
```

```
    public int[,]
sumamat(int[,] mat1,
int[,] mat2)
    {
        int[,] res = new
int[2, 2];
        for (int i = 0; i <
2; i++)
        {
            for (int j = 0; j
< 2; j++)
            {
                res[i, j] =
mat1[i, j] + mat2[i, j];
            }
        }
        return res;
    }

    //operaciones
    public int[,]
resmat(int[,] mat1, int[,]
mat2)
    {
        int[,] res = new
int[2, 2];
        for (int i = 0; i <
2; i++)
        {
            for (int j = 0; j
< 2; j++)
            {
                res[i, j] =
mat1[i, j] - mat2[i, j];
            }
        }
        return res;
    }

    public int[,]
mulmat(int[,] mat1, int[,]
mat2)
    {
        int[,] res = new
int[2, 2];
        int n = 2;
        for (int i = 0; i <
n; i++)
        {
            for (int j = 0; j
< n; j++)
            {
                res[i, j] = 0;
                for (int k =
0; k < n; k++)
                {
                    res[i, j]
+= mat1[i, k] * mat2[k,
j];
                }
            }
        }
```

```
  for (i <- 0 until
ma.length) {
    for (j <- 0 until
ma(0).length) {
      mc(i)(j) = ma(i)(j) +
mb(i)(j)
    }
  }
  mc
}


def restaMatriz(ma:
Array[Array[Int]], mb:
Array[Array[Int]]):
Array[Array[Int]] = {
  var mc =
Array.ofDim[Int](2, 2)
  for (i <- 0 until
ma.length) {
    for (j <- 0 until
ma(0).length) {
      mc(i)(j) = ma(i)(j) -
mb(i)(j)
    }
  }
  mc
}

def
calculadoraMatriz(operaci
on: String):
(Array[Array[Int]],
Array[Array[Int]]) =>
Array[Array[Int]] = {
  operacion match {
    case "suma" =>
sumaMatriz
    case "resta" =>
restaMatriz
  }
}
```

			return res; }	

3. RESULTADO Y DISCUSIÓN

En el código escrito en cada problema se puede apreciar, sobre todo en las funciones de orden superior, se nota la diferencia entre lenguajes funcionales y no funcionales. En los lenguajes funcionales es posible declarar las funciones de orden superior con muy pocas líneas y de forma muy sencilla. En Python y C# también es relativamente sencillo declarar y utilizar funciones de orden superior.

En cambio, en Java la declaración de funciones de orden superior es más complicada, necesitando incluir una interface, una clase que herede esa interface y un tercer método para invocar a esa clase, se nota la diferencia y que Java no está construido ni pensado para programación funcional.

Por otro lado, se intentó escribir un código parecido en cada lenguaje, exceptuando algunas funciones incluidas de cada lenguaje, como por ejemplo, la función yield en F# y Scala que para el problema separar listas, permite resolver el problema con mucho menos código que en el resto de lenguajes.

4. CONLUSIONES

La programación se puede ver y escuchar como algo muy complicado de hacer pero realmente cualquiera lo puede hacer si así se le propone ya que solo se aplica un conocimiento general de las cosas; una vez que se comienza a hacer es muy entretenido y cautivador el ser capas de crear algo con el solo hecho de pensarlo para luego llevarlo a cabo. Al mismo tiempo vamos aprendiendo a mejorar ese pensamiento, también a poder transmitirlo a una computadora y hacia las demás personas que aún desconocen el gran mundo de la programación.

Al adentrarnos en este tema es fácil darse cuenta de las grandes maravillas que se pueden lograr, no solo conociendo y superando nuestras capacidades si no también conociendo la manera que conlleva el pensar de las computadoras, nos permiten superar cualquier tipo de limites y va marcando un paso hacía el futuro en el cual la tecnología estará inmersa en todos los lugares por lo cual es necesario aprender y no hay manera mejor que comenzar aprendiendo a entender los códigos y la forma en que todos estos programas que vemos funcionan; para luego nosotros mismos innovar la manera en que funciona el mundo por medio de la creación de nuevos programas o incluso de nuevos códigos.

La programación nos permite desarrollarnos en todas las áreas ya que nos permite mejorar y ampliar la manera en que pensamos y no solo se limita a una forma individual si no que nos permite influenciar a los demás, demostrándoles de todo lo que se puede llegar a lograr con el simple hecho de escribir un código, si bien la programación es algo muy sencillo y el hecho de que esto sea así es que sin darnos cuenta forma parte de nuestras vidas ya que esta basada completamente a cosas que pueden ser aplicadas en cualquier ámbito; puede ir desde hacer una simple pagina web o programas tan complejos capaces de controlar incluso de manera más eficiente negocios u empresas, pero además de fundamente en cosas más accesibles al comprendimiento de este mismo como el hecho de aprender un sistema numérico aún más sencillo del que ocupamos usualmente(decimal) y es aprendiendo cosas tan sencillas como el binario del cual cabe mencionar que es fundamental para el funcionamiento de las computadoras.

5. REFERENCIAS

Carter, M. N. (2021, 12 diciembre). POO VS PF: Una falsa dicotomía - Matías Navarro Carter. Medium.
Recuperado 13 de diciembre de 2021, de https://medium.com/@mnavarrocarter/poo-vs-pf-una-falsa dicotom%C3%ADa-c3a0f52e68a
Programación funcional VS Programación orientada a objetos (OOP) ¿Cuál es mejor. . .? (2020, 3 diciembre).
ICHI.PRO. Recuperado 13 de diciembre de 2021, de https://ichi.pro/es/programacion-funcional-vs programacion-orientada-a-objetos-oop-cual-es-mejor-143036073092390
Programación funcional: ideal para algoritmos. (2021, 3 diciembre). IONOS Digitalguide. Recuperado 13 de

diciembre de 2021, de https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/programacion-funcional/