UNIDAD
4

Paradigma funcional

Introducción a la unidad

Si hasta ahora pensabas que programar en ensamblador, C, Java, o incluso Ruby era todo lo que había en el mundo de los lenguajes de programación, pues no es así. Te imaginas programar el movimiento de los discos de la torre de Hanoi sin una estructura *if* o *case*, o sin el *while* o el *for*, o que no pudieras declarar una variable ni mucho menor le pudieras asignar un valor. Parece algo loco, pero hay un paradigma que en su definición pura no necesita de estos conceptos para realizar programas. Su trabajo lo logra utilizando funciones puras, trasladando el concepto de función matemática a la programación.

Objetivo particular de la unidad

Explicar los conceptos más importantes que dan soporte y fundamento al paradigma de programación funcional (funciones, recursividad y listas), y aplicarlos en la resolución de problemas algorítmicos propios de la automatización de la información y manipulación de datos.

LO QUE SÉ

Contesta las siguientes preguntas, escribiendo tu propia definición de los siguientes conceptos, no es necesario que busques las respuestas en diversas fuentes, contéstalas de acuerdo a lo que sepas.

- ¿Qué es una función?
- ¿Qué es la recursividad?
- ¿Cuál es el algoritmo para calcular de un número dado su factorial?
- ¿Qué es la modularidad?

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**

Temas de la unidad IV

1. Definición
2. Programas con funciones
3. Recursividad
4. Listas
5. Implementación de algoritmos
6. Lenguajes funcionales puros e híbridos
7. Introducción a los lenguajes funcionales
8. Campos de aplicación

Resumen de la unidad

La programación apareció a principio de los sesenta como un paradigma completamente distinto a los existentes. Su origen se debe a las necesidades que encontraron los investigadores en los campos de:

- La inteligencia artificial
- El cálculo simbólico.
- Pruebas de teoremas.
- Sistemas basados en reglas.
- Procesamiento del lenguaje natural.

El enfoque de este paradigma está en que los cálculos se aprecian como una función matemática con datos de entrada y de salida. Se pierde la de memoria, por lo que se carece de una instrucción de asignación. Los bucles se basan en el concepto de recursividad.

En general los lenguajes funcionales soportan los conceptos de: variable, asignación y bucle.

Tema 1. Definición

Objetivo del tema

Identificar el concepto de programación funcional.

Desarrollo

Es el paradigma de programación que difiere de otros paradigmas ya que define a los programas como función, y trata a las funciones como datos, logrando así minimizar los efectos colaterales de ejecución y la administración automática de la memoria. Con este paradigma se logra gran flexibilidad de los lenguajes, es conciso en la notación y la semántica es fácil de entender.

ACTIVIDAD 1

Realiza la lectura de los siguientes puntos y realiza la actividad.

1.1.1 Funciones y 1.1.2 Lenguajes, Capítulo 1 del libro *Programación Funcional* de Jeroen Fokker **(ANEXO 1)** Presentación Programación Declarativa, páginas 2, 3 y 4**. (ANEXO 2)**

Genera un documento de no más de dos cuartillas indicando los siguientes puntos:

- Definición propia de lenguajes funcionales
- Historia
- Características
- Lista de lenguajes funcionales

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 2. Programas con funciones

Objetivos del tema

Identificar un programa como una colección de funciones interrelacionadas.

Desarrollo

Un programa es una descripción de un cálculo. Un programa entonces es equivalente a una función matemática.

Una función es una regla que asocia a cada elemento *x* de algún conjunto *X* de valores un elemento *y* único de otro conjunto *Y* de valores. Matemáticamente se define una función así:

$$y = f(x) \text{ ó}$$
$$f: X \rightarrow Y$$

El conjunto X se llama dominio de *f*, y el conjunto Y se llama rango de *f*. La *x* en f(x) se llama variable independiente. Y la *y* del conjunto Y de la ecuación y=f(x) se llama variable dependiente. Cuando f no está definida para todas las *x* de X, se llama función parcial y cuando sí está definida para todas las *x* de X se llama función total.

Los programas, procedimientos y funciones se pueden representar por medio de una función. En el caso de un programa *x* representa la entrada y *y* la salida. En un procedimiento o función *x* representa los parámetros y *y* los valores devueltos. En el paradigma funcional no se hace distinción entre programa, función o procedimiento, sólo importa los valores de entrada y salida.

ACTIVIDAD 1

Con base en la lectura de la página Tema 2: Características de la programación funcional http://www.dccia.ua.es/dccia/inf/asignaturas/LPP/2007-2008/tema-02.html, elabora un documento (no mayor a una cuartilla), mostrando la operación de sumatoria de un número como una función. En éste trabajo deberás señalar además las entradas de información y la salida de datos. Así mismo presentar 2 ejemplos de su aplicación.

La sumatoria de un numero entero n es la suma de 1, 2, 3, 4, 5, …, n

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Indicar cuáles de las siguientes expresiones son falsa y cuales verdaderas. Al final obtendrás tu calificación de manera automática.

		Verdadera	Falsa
1	Las funciones establecen la relación entre los parámetros (la 'entrada') y el resultado (la 'salida') de procesos definidos.	( )	( )
2	En la programación funcional un programa consiste en la definición de una sola función.	( )	( )
3	Un lenguaje funcional es LISP.	( )	( )
4	En los lenguajes funcionales, no existe la noción de posición de memoria y por tanto, la necesidad de una instrucción de asignación.	( )	( )
5	Con un lenguaje funcional no se requiere definir las funciones que emplearemos.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 3. Recursividad

Objetivos del tema

Identificar y aplicar la técnica de recursividad para los leguajes funcionales.

Desarrollo

La recursividad en términos de programas es cuando una función o método se llama a sí misma cuantas veces requiera para resolver un problema dado. Pero no quiere decir que sea la mejor forma de hacerlo ni la más eficiente, al contrario, la recursividad es cara en el uso de memoria y exige una mayor capacidad de procesamiento.

El ejemplo clásico de una función recursiva es el cálculo de la factorial de un número. Por ejemplo, 6! es igual a 6 * 5 * 4 * 3 * 2 * 1. Algebraicamente, podemos considerar el cálculo factorial como (n!):

n * (n-1) * (n-2) * (n-3)...(n - (n+1))

La definición recursiva del cálculo factorial en LISP es:

```
defun factorial (n)
 (cond ((zerop n) 1)
          (T
 (* n (factorial (1- n))))
 ) ;_ fin de cond
) ;_ fin de defun
```

ACTIVIDAD 1

Genera en un documento nuevamente el algoritmo para el cálculo de la sumatoria, presentado en el tema anterior, pero de tal forma que incluya la recursividad. Además, señala la(s) diferencia(s) con relación a tu primera versión.

El documento no deberá ser mayor a una cuartilla.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Busca en internet el código para Scheme que corresponda al cálculo del factorial de un número. Con la ayuda de la guía de Shceme, modifícalo para generar la sumatoria que desarrollaste en la Actividad 1 del Tema 2. Envía tu código.

Te proporciono algunos link que puedes consultar

- http://www.plt-scheme.org/
- http://docs.plt-scheme.org/guide/index.html

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Emplea la función factorial y calcula las siguientes operaciones.

Después une cada factorial con su resultado, arrastrando la cantidad al lugar correspondiente. Al final obtendrás tu calificación de manera automática.

1. Factorial(9) _____
2. Factorial(5) _____
3. Factorial(0) _____
4. Factorial(1) _____
5. Factorial(3) _____

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 4. Listas

Objetivo del tema

Identificar el concepto de Lista dentro del proceso de información.

Desarrollo

La lista es el elemento principal cuando se programa en un lenguaje funcional, ya que tradicionalmente una función estará implementada por listas de elementos. Tanto los datos como los programas son listas. De ahí viene el nombre del lenguaje LISP, que es un acrónimo de "LIST Processing". Por cierto, hay un chiste de esto: las listas en LISP están delimitadas por paréntesis, y entonces se dice que el significado de LISP es: "Lost In Stupid Parentheses".

En LISP hay dos tipos de elementos con los que se programa:

Átomos:
Son los datos elementales y pueden pertenecer a varios tipos: números, caracteres, cadenas de caracteres o símbolos.

Listas:
Son secuencias de átomos o de listas encerradas entre paréntesis. Además, existe una lista especial, "nil", que es la lista nula, que no tiene ningún elemento.

Hay funciones en LISP cuyos nombres son símbolos (+ para la suma, * para el producto, etc.) por ejemplo: (+ 5 9).

ACTIVIDAD 1

Con base en la lectura de la página Listas, listas y más listas que se encuentra en http://docs.gimp.org/2.2/es/gimp-using-script-fu-tutorial-lists.html, elabora un documento de no más de dos cuartillas indicando:

- Definición de lista
- Elementos de una lista
- Construcción de una lista en Scheme
- Uso de las funciones list, cons y car, incluyendo dos ejemplo del uso de cada una de ellas.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Indica cuáles de las siguientes expresiones son verdaderas y cuáles falsas, para el lenguaje Scheme (dialecto de LISP): Al final obtendrás tu calificación de manera automática.

		Verdadera	Falsa
1	Una variable puede referirse a una lista de valores.	( )	( )
2	No puede declararse listas vacías.	( )	( )
3	La concatenación de listas se lleva a cabo con la función list.	( )	( )
4	Las listas se componen de una ‘cabeza’ y una ‘cola’.	( )	( )
5	En una lista, una ‘cola’ está conformada por los elementos de la lista sin incluir al primero.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 5. Implementación de algoritmos

Objetivo del tema

Identificar algunas técnicas básicas para la implementación de algoritmos.

Desarrollo

Este paradigma basa su programación en un conjunto de funciones (casi siempre recursivas) y alguna expresión cuya salida representa el resultado de algún algoritmo. Los lenguajes funcionales son declarativos, esto quiere decir que se describen relaciones entre las variables en términos de funciones y reglas de inferencia (procedimiento que infiere hechos a partir de otros hechos conocidos), dejando al traductor la responsabilidad de encontrar el mejor algoritmo para encontrar el resultado buscado. También se basa en el cálculo lambda λ con constantes; este cálculo ayuda a crear valores de funciones sin tener que darles un nombre. Las funciones de este tipo no modifican su salida con entradas iguales, y al no tener efectos colaterales cumplen con la regla de transparencia referencial.

Para implementar un algoritmo hay que considerar que un lenguaje de programación es completo en *Turing,* si tiene valores enteros, funciones aritméticas sobre dichos valores, así como un mecanismo para definir nuevas funciones utilizando las ya existentes, además de selección y recursión.

Al dejar que el mismo traductor implemente el mejor algoritmo para hacer una determinada tarea hace que los programas funcionales sean más independientes de la arquitectura de la computadora. Veamos un ejemplo de implementación funcional:

Vamos a implementar el algoritmo del MCD (máximo común divisor). Para resolverlo se utiliza el algoritmo de Euclides que consiste en varias divisiones euclidianas sucesivas.

En C nuestro algoritmo queda:

```
void mcd(int u, int v, int *x){
        int y, t, z;
        z = u;
        y = v;
        while (y!=0){
                t = y;
                y = z%y;
                z = t;
}
*x = z;
} // Fin de mcd

La versión funcional (sin asignación y con recursión) de este código es:

int mcd(int u, int v){
        if (v==0) return u;
        else return mcd(v, u%v);
}
```

Como se pudo observar las dos versiones son muy diferentes, y aunque hacen lo mismo, implementar el MCD de manera recursiva resulta en un código más compacto y elegante.

ACTIVIDAD 1

Investiga en Internet, mínimo tres fuentes, el algoritmo de Hanoi y posteriormente genera un documento no mayor a tres cuartillas, explicando cada una de las siguientes soluciones al problema.

1. Solución recursiva
2. Solución iterativa

Además, incluir en tus propias palabras las diferencias entre el uso de dichas soluciones. No olvides citar tus fuentes.

Busca en Internet el algoritmo de las Torres de Hanoi. Posteriormente genera un documento no mayor a tres cuartillas, explicando cada una las siguientes soluciones al problema:

3. Solución recursiva
4. Solución iterativa

Además, incluir en tus propias palabras las diferencias entre el uso de dichas soluciones.Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 6. Lenguajes funcionales puros e híbridos

Objetivo del tema

Distinguir las diferencias entre los lenguajes funcionales puros e híbridos.

Desarrollo

Matemáticamente las variables siempre representan valores reales, pero en los lenguajes de programación imperativos por ejemplo las variables se refieren a localizaciones de memoria, así como a valores. Ya que en matemáticas no existe este concepto de localización en la memoria o los valores de 1 de las variables, el enunciado $x = x + 1$ no tiene sentido. Por eso el paradigma funcional elimina el concepto de variable a excepción del uso como nombre para un valor. Por consecuencia, no hay operaciones de asignación. Sólo hay constantes, parámetros y valores.

Si un lenguaje de programación funcional (también llamado recursivo) trabaja de esta forma, sin variables ni operaciones de asignación, se dice que es un lenguaje funcional puro. La mayoría de los lenguajes funcionales conservan alguna idea de asignación, lo que los hace impuros o híbridos, pero permiten trabajar de forma pura si así lo requerimos. Además de la falta de asignaciones, tampoco hay ciclos. La forma de hacer que una operación se repita (a falta de ciclos *while*) es la recursión. Algunos de los principales lenguajes de programación funcional son: Hope, LML, Clean, Haskell, FP, Miranda, SML y LISP.

ACTIVIDAD 1

Los lenguajes funcionales híbridos son menos estrictos que los puros, ya que admiten conceptos tomados de los lenguajes imperativos, como las secuencias de instrucciones o la asignación de variables. En contraste, los lenguajes funcionales puros tienen una mayor potencia expresiva, conservando a la vez su transparencia referencial, algo que no se cumple siempre con un lenguaje funcional híbrido.

Con base en lo anterior, genera un documento de no más de dos cuartillas, indica cuales de los siguientes lenguajes funcionales son impuros y cuales son puros. Justifica tu respuesta.

- Hope
- LML
- Clean
- Haskell
- FP
- Miranda
- SML
- LISP
- SCHEME

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Indica cuáles de las siguientes expresiones son verdaderas y cuales falsas. Al final obtendrás tu calificación de manera automática.

		Verdadera	Falsa
1	Los lenguajes funcionales puros permiten la inclusión de técnicas de los lenguajes imperativos.	( )	( )
2	En los lenguajes funcionales híbridos se puede encontrar operaciones de asignación.	( )	( )
3	La transparencia referencial indica que el significado de una expresión depende únicamente del significado de sus subexpresiones.	( )	( )
4	Los lenguajes funcionales puros permiten las estructuras cíclicas como el while.	( )	( )
5	Los lenguajes funcionales implementan la recursión.	( )	( )

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 7. Introducción a los lenguajes funcionales

Objetivo del tema

Reconocer las características de programación con lenguajes funcionales.

Desarrollo

LISP: Abreviación de LISt Processor, fue desarrollado en el MIT en los 60. Está basado en el cálculo lambda λ desarrollado por Church. No hay un estándar para LISP, así que nos vamos a encontrar con muchas variantes de este lenguaje. Una de estas variantes es Scheme. Es así como surgen los dialectos. Scheme es un dialecto de LISP.

En Scheme los programas y datos son expresiones y son de dos tipos: átomos y listas. Los átomos son como las constantes e identificadores de un lenguaje imperativo como C: incluyen números, cadenas, nombres, funciones, etc. Una lista es simplemente una secuencia de expresiones separadas por espacios y rodeadas por paréntesis, por ejemplo: (+ 2 3).

Los programas se ejecutan evaluándolos como expresiones, y éstas a su vez se evalúan aplicando el primer elemento de una lista (que debe ser una función) al resto de los elementos que vienen siendo los argumentos. Ejemplo:

```
(mcd 8 18)
```

La función mcd se aplica a los argumentos 8 y 18.
Así, 2 + 3 se escribe (+ 2 3), en donde la función + se aplica a los argumentos 2 y 3.

El cálculo del MCD en Scheme-LISP quedaría así:

```
(define (mcd u v)
        (if (= v 0) u
                (mcd v(modulo u v))
        )
)
```

En la función mcd se utiliza la función if-then-else, pero a diferencia de los demás paradigmas, el *if* es una función, no una estructura de control selectiva. (if a b c) significa:

```
if a
        then b
        else c
```

Esta función representa tanto el control como el cómputo de un valor. Primero se evalúa *a*, y dependiendo del resultado, se evalúa ya sea *b* o *c*, convirtiéndose el valor resultante en el valor devuelto por la función. El *if* en otros lenguajes carece de valor.

La principal desventaja de los lenguajes funcionales es la ineficiencia en su ejecución. Por ser dinámicos, deben ser interpretados más que compilados, con la consecuente pérdida de velocidad de ejecución.

ACTIVIDAD 1

Con base en el algoritmo para la solución de las Torres de Hanoi, genera la el código fuente para SCHEME y envíalo para su revisión.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

Dada las siguientes funciones en Scheme:

```
(define (a x) (+ x x))
(define (b x) 5)
(define (c x y)  (* x  y))
(define d
  (lambda (n)
    (if (= n 0) 1
        (* n (d (- n 1))))))
```

Indica el resultado de cada una de las siguientes expresiones.

1. (a 8) _____
2. (b 66) _____
3. (c 5 8) _____
4. (d 7) _____
5. (c 7 (d 4)) _____

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	Descripción

Tema 8. Campos de aplicación

Objetivo del tema

Identificar los campos de aplicación para los lenguajes funcionales.

Desarrollo

Los lenguajes funcionales son más cercanos a la manera en que funciona la mente humana, pues permiten a los programadores describir sus algoritmos como expresiones que serán evaluadas. Este paradigma encuentra diversas aplicaciones en áreas como las bases de datos, ingeniería del software, procesadores de lenguajes, inteligencia artificial, redes neuronales y sistemas expertos. Los lenguajes funcionales permiten la creación de procedimientos en tiempo de ejecución, lo que lleva a un gran nivel de modularidad que difícilmente puede alcanzarse en otros paradigmas de programación.

ACTIVIDAD 1

Busca en la Web la historia y evolución de LISP. Haz un árbol genealógico de los dialectos que han surgido a partir de este lenguaje; sus creadores, años de aparición y equipos en los que pueden correr.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

ACTIVIDAD 2

Busca en la Web un programa en LISP y otro en Java o C que implemente el algoritmo del movimiento de la Reina en el juego de ajedrez. Compara el rendimiento de cada uno en la ejecución y analiza por qué uno es más lento que los otros, aunque sean menos líneas de código.

Escribe tu análisis en no más de una cuartilla.

Para enviar tu respuesta, pulsa el botón **Editar mi envío;** se mostrará un editor de texto en el cual puedes redactar tu información; una vez que hayas concluido, salva tu actividad pulsando el botón **Guardar cambios**.

ACTIVIDAD 3

Investiga y realiza un mapa conceptual sobre las características de Haskell. Debe incluir las diferencias y similitudes con Scheme.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

Autoevaluación

De las siguientes aplicaciones, subraya de color azul, aquellas que son adecuadas para el uso de los lenguajes funcionales:

a) Resolución de rompecabezas lógicos.
b) Generación de gráficas 3D.
c) Exploración de grafos.
d) Permutaciones y combinaciones empleando listas.
e) Administración de bases de datos.

Cuestionario de Autoevaluación

Contesta las siguientes preguntas.

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar.** Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

1. ¿Cómo maneja los programas el paradigma funcional?
2. ¿Qué es una función?
3. ¿Qué es una variable independiente?
4. ¿Qué es una variable dependiente?
5. ¿Para qué sirve el cálculo lambda (λ)?

Examen de Autoevaluación

Elije el inciso que conteste correctamente cada pregunta. Al final obtendrás tu calificación de manera automática.

1. La *y* del conjunto Y de la ecuación y=f(x) se llama:

- a) Cálculo lambda (λ)
- b) Variable dependiente
- c) Variable independiente
- d) Procedimiento

2. A los lenguajes funcionales también se les llama:

- a) Lenguajes lógicos
- b) Lenguajes procedurales
- c) Lenguajes imperativos
- d) Lenguajes recursivos

3. En los lenguajes funcionales, las operaciones repetitivas no se expresan mediante ciclos o lazos, sino mediante:

- a) Funciones de control
- b) Funciones secuenciales
- c) Funciones recursivas
- d) Funciones imperativas

4. En el paradigma funcional se eliminan los conceptos de:

- a) variable como una localización de memoria y operaciones de asignación
- b) variable como nombre para un valor y operaciones de asignación
- c) operaciones de repetición y de recursividad
- d) operaciones de repetición y asignación

5. El valor devuelto por *(* (+ 50 30) (- 80 50))* es:

- a) 1400
- b) 2400
- c) 3400
- d) 4000

6. En LISP, los programas son:

- a) Procedimientos anidados
- b) Funciones recursivas
- c) Expresiones de listas
- d) Listas recursivas

Observa el contenido de cada columna, después lee con cuidado cada uno de los incisos y colócalos donde corresponda arrastrando la fase al lugar indicado. Al final obtendrás tu calificación de manera automática.

1.

procedure mcd(u, v: in integer; x out integer)is
y, t, z: integer;
begin
z:=u;
y:=v;
loop
exit when y=0
t:=y;
y:=z mod y;
z:=t;
end loop;
x:=z;
end mcd;
	(define (mcd u v)
	(if (= v 0) u
	(mcd v (modulo u v))))
		function mcd(u, v: in integer) return integer is
		begin
		if v=0 the
		return u;
		else
		return mcd(v, u mod v);
		end if;
		end mcd;
			Int mcd(int u, int v){
			if (v==0) return u;
			else
			return mcd(v, u % v);
			}

a) Versión funcional recursiva en C del MCD
b) Versión no funcional en ADA del MCD
c) Versión del tipo y = f(x) del MCD en ADA
d) Versión funcional en LISP del MCD

2.

En *y = f(x)* ...	**En un programa ...**	**En una función ...**

a) ***x*** representa los parámetros y ***y*** los valores devueltos
b) ***x*** representa cualquier valor proveniente de ***X*** y ***y*** la variable dependiente
c) ***x*** representa las entradas y ***y*** las salidas

Bibliografía básica

Autor	Capítulo	Páginas

Sitios electrónicos

Sitio	**Descripción**

LO QUE APRENDÍ

Responde las siguientes preguntas:

Realiza tu actividad en un documento en Word, guárdala en tu computadora y una vez concluida, presiona el botón **Examinar**. Localiza tu archivo donde lo guardaste, selecciónalo y presiona **Subir este archivo** para guardarlo en la plataforma.

1. ¿Cuáles son las características de la programación funcional?
2. ¿Qué es una función?
3. ¿Qué es la recursividad?
4. ¿Cómo es una función recursiva?
5. ¿SCHEME es un lenguaje funcional puro? ¿Por qué?
6. ¿Qué es una lista para los lenguajes funcionales?
7. ¿Cómo se utiliza la instrucción cdr con las listas en SCHEME?
8. ¿La memoria se puede manipular directamente empleando lenguajes funcionales? ¿Por qué?
9. ¿La recursividad es aplicable de forma exclusiva para los lenguajes funcionales? ¿Por qué?

Glosario de la unidad

Paradigmas Declarativos

Modelos de desarrollo: Funcional, Lógico y de Flujo de Datos. Se construye señalando hechos, reglas, restricciones, ecuaciones, transformaciones y otras propiedades derivadas del conjunto de valores que configuran la solución. [Programación III - http://exa.unne.edu.ar/depar/areas/informatica/informat.htm]

Paradigma Funcional

Modelo matemático de composición funcional donde el resultado de un cálculo es la entrada del siguiente, y así sucesivamente hasta que una composición produce el valor deseado. [Programación III - http://exa.unne.edu.ar/depar/areas/informatica/informat.htm]

Agente

Cualquier elemento capaz de percibir su entorno (recibir entradas), procesar dichas percepciones e interactuar con su entorno (proporcionar salidas).

Inteligencia artificial

Rama de la ciencia informática dedicada al desarrollo de agentes racionales no vivos.

Recursión o recursividad

Definición de un elemento (problema, estructura de datos, objeto) en términos de sí mismo.

Función recursiva

Una función que se llama a sí misma.

Lista

Grupo de elementos atómicos similares.

Lenguaje funcional híbrido

Lenguaje funcional que incluye variables, operaciones de asignación o ciclos.

Lenguaje funcional puro

Lenguaje funcional que carece de mecanismos para la declaración de variables, operaciones de asignación o ciclos.

MESOGRAFÍA

Bibliografía básica

Bibliografía complementaria

Sitios electrónicos

(ANEXO 1) Capítulo 1 del libro *Programación Funcional* de Jeroen Fokker

Descargar archivo PDF Jeroen Fokker que se encuentra en la plataforma

(ANEXO 2) Descarga *Programación declarativa* que se encuentra en la plataforma