Universidad Tecnológica de Pereira

UNIVERSIDAD TECNOLÓGICA DE PEREIRA
FACULTAD DE INGENIERÍAS: ELÉCTRICA, ELECTRÓNICA, FÍSICA Y CIENCIAS DE LA COMPUTACIÓN
PROGRAMA INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

ASIGNATURA:	**PROGRAMACIÓN I**
CODIGO:	**IS105**
CREDITOS:	**5**
INTENSIDAD:	**6 horas semanales para 96 horas totales**
REQUISITOS:	**No tiene**

JUSTIFICACION

Los primeros computadores electrónicos se construyeron en los años cuarenta (40's). Los primerísimos modelos fueron 'programados' con grandes relés y pronto se almacenaron los programas en la memoria del computador, haciendo que los primeros lenguajes de programación hicieran su entrada. En aquel tiempo el uso de un computador era muy costoso y era lógico que el lenguaje de programación guardara mucha relación con la arquitectura del computador que nació bajo el modelo de máquina de Von Newman, donde el computador consta de una unidad de control y una memoria. Por eso un programa consistía en instrucciones para cambiar el contenido de la memoria ya que la unidad de control se encargaba de ejecutarlas. De esta manera se creó el estilo de programación imperativa, siendo entonces tradicional que el primer curso de programación de computadores se basara en lenguajes bajo el paradigma imperativo, pues existía la necesidad de hablar en términos de una memoria que cambie por instrucciones en un programa.

Desde antes de la existencia de los computadores se inventaron métodos para

resolver problemas. Los primeros algoritmos conocidos proceden de la antigua Mesopotamia, datan aproximadamente del 3.000 A.C., y estaban escritos en tablillas de arcilla. Su propósito era la realización de cálculos tan pragmáticos como el del capital resultante de un préstamo a interés compuesto y otros similares. Por tanto, no existía la necesidad de hablar en términos de una memoria que cambie por instrucciones en un programa. En la matemática de los últimos cuatrocientos años son muy importantes las funciones. Estas establecen la relación entre los parámetros (la 'entrada') y el resultado (la 'salida') de procesos definidos. Esto es que el resultado depende de una u otra forma de los parámetros. Por esa razón, una función es una buena manera de construir soluciones computacionales y es la base del paradigma de programación funcional. Con el tiempo, al bajar los precios de los computadores y al subir los precios de los programadores, llega a ser más importante describir las soluciones computacionales en un lenguaje que esté más cerca del 'mundo del hombre', que cerca del computador. Los lenguajes funcionales se unen a la tradición matemática y no están muy influidos por la arquitectura concreta del computador. Para este curso utilizaremos el lenguaje DrScheme que es un entorno grafico, interactivo e integrado de programación para el lenguaje Scheme.

Scheme es un dialecto pequeño y elegante de LISP que fue desarrollado en los 70's en el Massachusetts Institute Of Technology(MIT). La versatilidad de Scheme lo convierte en el lenguaje de elección para introducir a los estudiantes novatos de la programación.

El *"Institute* of Electrical and Electronics Engineers, Inc. Computer Society (IEEE-*CS)"* y la "Association *for Computing Machinery(ACM)"* en su documento de "CURRICULA-2004" propone dentro sus modelos para la enseñanza de la programación, el paradigma declarativo: Programación funcional.

En todo el mundo, Scheme rápidamente esta reemplazando los lenguajes del paradigma imperativo en los cursos de introducción de Ciencias de la Computación. (V.gr: Berkeley, Princeton, Cornell, MIT, Yale, UCLA, Universidad Javeriana Cali, Univalle, Politécnico de Barcelona, entre muchas otras.)

PARADIGMA DECLARATIVO: PROGRAMACIÓN FUNCIONAL

El paradigma declarativo, no se basa en el cómo se hace algo (cómo se logra un objetivo paso a paso), sino que describe (declara) cómo es algo. En otras palabras, se enfoca en describir las propiedades de la solución buscada, dejando indeterminado el algoritmo (conjunto de instrucciones) usado para encontrar esa

solución. Esto último se realizará mediante mecanismos internos de inferencia de información a partir de la descripción realizada. La programación funcional es uno de los fundamentales entre los llamados del paradigma declarativo. Como tal, permite aunar los componentes de especificación y programación en las tareas de solución automática de problemas.

Los lenguajes funcionales ofrecen al programador un buen número de recursos expresivos que permiten resolver problemas complejos mediante programas pequeños y robustos. Entre ellos cabe destacar:

1. Unsistemadetipospolimórficosquepermitedefinirunaampliavariedadde estructuras de datos de uso genérico.

2. La posibilidad de definir funciones que aceptan otras funciones como argumentos y devuelven funciones como resultado.

3. Facilidad es para definir y manipular estructuras de datos infinitas.

4. Un modelo computacional simple, claro y bien fundamentado, etc.

De no menor importancia es la posibilidad de razonar, de forma sencilla, acerca de las propiedades de los programas: su corrección, su eficacia, su comportamiento en ejecución, ... Esto permite optimizar las tareas de implementación de los lenguajes funcionales.

Podemos encontrar, en casi todos los lenguajes de programación funcional, un núcleo común de conceptos y técnicas asentado sobre bases matemáticas firmemente establecidas. En esta asignatura estudiamos dichos y su utilización en la definición de implementaciones correctas y eficientes de los lenguajes de programación que se enmarcan en este paradigma.

OBJETIVOS

La Programación declarativa es un paradigma de programación basado en la lógica en el que se estudian de forma simple muchos aspectos avanzados de los lenguajes de programación modernos.

OBJETIVO GENERAL

Profundizar en los aspectos formales y aplicados de la programación declarativa y, en particular, en uno de los dos (2) paradigmas más representativos del estilo de programación declarativa: la programación funcional, siguiendo un esquema uniforme basado en explotar la lógica ecuacional para el estilo funcional.

OBJETIVOS ESPECIFICOS

- Describir los conceptos principales de la programación funcional y de la Lógica de Programación - Explicar las características básicas y fundamentales de un lenguaje de programación funcional (Scheme).

- Tener una primera toma de contacto con los lenguajes declarativos-funcionales, resolviendo problemas típicos con el uso de tales lenguajes. - Alcanzar la capacidad de plasmar los conceptos de la programación funcional en más de una herramienta de programación.

CONTENIDO

1. Introducción a la informática. (objetivo: tener un punto de partida en común)

- Hardware
- Código binario
- Almacenamiento
- Software: paradigmas, lenguajes de programación, clasificación de software: sistema, desarrollo, aplicativo.
- Estructura de un programa.
- Tipos de datos: Numéricos (Enteros y de Punto Flotante), Alfanuméricos y Lógicos
- Expresiones aritméticas: diferentes notaciones: prefija, infija, sufija, árboles de sintaxis.
- Representación de la información: Ascii, Unicode y BitCode.

2. Introducción al lenguaje Dr. Scheme

- Características fundamentales de la programación funcional
- Reseña histórica de Scheme
- Elementos básicos de Scheme:
- Caracteres
- Cadenas de caracteres
- Números
- Booleanos
- Identificadores: variables y palabras reservadas
- Expresiones
- Literales
- Variables
- Operadoresaritméticos
- Variables y programas simples

3. Funciones

- Definición de función
- Reglas de ámbito léxico
- Definiciones internas
- Composición de funciones

4. Predicadosysentenciascondicionales

- Operadores relacionales
- Operadores lógicos

- Predicados primitivos
- Predicados simbólicos
- Predicados numéricos
- Predicados de equivalencia
- Formas especiales condicionales:
- Forma especial “cond”
- Forma especial “cond – else”
- Forma especial “if”
- Estructuras de bloques

5. Recursión e Iteración

- Recursión simple
- Recursión múltiple
- Manejo de ciclos simples y anidados
- Funciones utilizadas como parámetros

6. Gráficos

- Introducción a:
  - Gráficos
- Estructuras y gráficas simples
- Operaciones para el manejo del mouse

7. Tipos compuestos de datos, ordenamiento y búsqueda

- Estructuras de datos
- Pares

- Listas

- Operaciones con listas

METODOLOGIA

Los estudiantes deberán preparar, antes de la clase, los temas asignados por el profesor. Como apoyo el profesor podrá publicar material en una página Web y/o entregarlo en conferencias. Bajo el esquema de trabajo de esta materia, preparar un tema significa ESTUDIARLO. Estos capítulos pueden ser complementados con la bibliografía que se presenta al final de este documento.

El trabajo en clase se centrará en presentar los temas en forma magistral, resolver las dudas encontradas por los estudiantes durante la preparación del material, la solución de ejercicios que se hayan asignado, pero sobre todo en discutir nuevos ejercicios que permitan alcanzar mayor claridad en cada tema.

También se harán trabajos tendientes a desarrollar en el estudiante la capacidad de traducir a un lenguaje de programación la solución dada a diferentes problemas.

Cada grupo podrá contar con un auxiliar de cátedra o monitor que será apoyo, principalmente, en el lenguaje de programación, y quien atenderá a los estudiantes durante la semana en horarios convenidos de común acuerdo con la mayoría de los estudiantes. La asistencia a consultas tanto al profesor como al monitor serán valoradas. Adicionalmente, el profesor atenderá previa cita a los estudiantes, en forma personalizada dos (2) horas adicionales a la semana, en horario que será convenido con los estudiantes.

COMPETENCIAS

COMPETENCIAS TRANSVERSALES / GENÉRICAS:

- Aprendizaje autónomo
- Capacidad de análisis y síntesis
- Capacidad de aplicar los conocimientos a la práctica
- Resolución de problemas
- Trabajo individual y por parejas
- Comunicación oral y escrita

COMPETENCIAS ESPECÍFICAS:

- Cognitivas (Saber):
    - Idioma
    - Matemáticas
    - Nuevas tecnologías TIC
    - Conocimientos de informática
    - Procedimentales / Instrumentales (Saber hacer):
    - Redacción en interpretación de documentación técnica
    - Estimación y programación del trabajo
    - Planificación, organización y estrategia.
- Actitudinales (Ser):
    - Calidad
    - Toma de decisión
    - Capacidad de iniciativa y participación

TÉCNICAS DOCENTES

Las técnicas docentes que se van a utilizar son:

- Clases de teoría

- Exposiciones sobre trabajos de casos prácticos.
- Desarrollo en clase, de proyectos dirigidos por el docente
- Tutorías colectivas de teoría
- Clases de prácticas
- Corrección de las prácticas
- Tutorías colectivas de prácticas
- Tutorías individualizadas

DESARROLLO Y JUSTIFICACIÓN:

Clases de teoría:

- Se hará una reseña inicial del contenido de cada tema y se indicará su relación con los otros temas.
- Al comenzar la explicación de una sección de un tema, se indicarán las relaciones que posee con otras secciones del mismo tema o de temas diferentes.

- Se explicará detenidamente cada sección de cada tema teórico.

Exposiciones:

- El profesor propondrá los trabajos sobre trabajos de casos prácticos, que los estudiantes deberán preparar y exponer a lo largo del curso.
- Los trabajos podrán hacerse individualmente o en parejas.

Acerca de las prácticas:

Las prácticas y tutorías colectivas de prácticas, están sujetas a la disponibilidad de salas con computadores.
- Las prácticas persiguen consolidar el conocimiento adquirido sobre los lenguajes funcionales y sus peculiaridades expresivas. Consistirán en la realización de pequeños programas que permitan bosquejar las posibilidades de aplicación de estos lenguajes a problemas.
- Se presentarán los recursos informáticos necesarios para el desarrollo de las prácticas. Vgr: intérpretes (drscheme), editores de texto, entorno gráfico integrado, etc.
- Se describirán los objetivos que se pretenden conseguir con la elaboración de cada una de las prácticas.
- Se utilizará el tablero para el desarrollo de los fundamentos prácticos y el proyector de transparencias y el material informático (hardware: computadores; software: drscheme, editores de texto, etc.) para desarrollar los ejemplos.
- Se entregará a los estudiantes fotocopias o información digital de manuales del lenguaje DrScheme, ejemplos prácticos y los enunciados de las prácticas.
- Los estudiantes desarrollarán las prácticas codificando y documentando los programas.
- Se podrá corregir y evaluar en presencia del estudiante los trabajos de prácticas que haya realizado.
- Se propenderá por indicarle al estudiante los posibles fallos y proponerle posibles soluciones alternativas.

Tutorías colectivas de teoría o prácticas

Es una actividad desarrollada dentro de las horas de clase
- El profesor responderá a las preguntas que les planteen los estudiantes procurando que ellos intenten deducir las repuestas correctas.
- Se procurará que las preguntas que se planteen no sean dudas particulares de

un estudiante, sino dudas generales que puedan tener la mayoría de los estudiantes. Las dudas particulares se deben plantear en las tutorías individuales. - El profesor también podrá plantear preguntas a los estudiantes para comprobar si han aprendido correctamente los conceptos fundamentales de la asignatura.

Tutorías individualizadas:

- Según es reglamento estudiantil vigente, en su articulo 60.("ARTÍCULO *60o.: El estudiante de la Universidad tiene derecho a:.....Ser* asistido, *asesorado y oído por quienes tienen la responsabilidad administrativa y docente.". Subrayado nuestro),* estas tutorías están enmarcadas dentro de la actividad docente y los horarios deberán ser concertados con todos los estudiantes o con la mayoría cuando con todos no sea posible.
- Los estudiantes con el fin de poder organizar y garantizar que la atención sea individual, deberá solicitar con anticipación cita con el profesor.
- Los estudiantes deben utilizar estas tutorías a lo largo de todo el curso y no sólo antes de la fecha del examen.
- El profesor intentará resolver las dudas particulares que pueda tener cada estudiante en relación con los temas de teoría, los trabajos de las exposiciones, las prácticas, etc. - Aunque las dudas más simples puedan plantearse mediante correo electrónico, es preferible que haya una reunión del profesor y el estudiante para resolver las dudas más complejas.
- La Universidad podrá disponer como recurso adicional un *"asistente de cátedra o monitor",* que podrá ser un estudiante de semestres superiores, según el reglamento que sobre este particular maneje la Universidad.

MECANISMOS DE CONTROL Y SEGUIMIENTO

El profesor podrá comprobar el grado de seguimiento de la asignatura mediante:

- La asistencia a las clases de teoría y prácticas
- Las exposiciones de temas de teoría.
- La corrección de las prácticas.
- Las tutorías personales
- Los parciales
- Los exámenes de corta duración (Quiz).

ORGANIZACIÓN SEMANAI

Nro.	Semana	Temas	Clases de teoría (Horas)	Tutorías Profesor (Horas)	Práctica (Horas)	Examen (Horas)
1		Introducción al lenguaje - Características fundamentales de la programación funcional - Reseña histórica de Scheme - Elementos básicos de Scheme: - Caracteres, Cadenas	4	2	0	
2		- Números - Booleanos - Identificadores: variables y palabras reservadas - Expresiones - Literales - Variables	4	2	0	
3		- Operadores aritméticos - Variables y programas simples	4	2	2	
4		Funciones - Definición de función - Reglas de ámbito léxico - Definiciones internas - Composición de funciones	4	2	0	
5		Predicados y sentencias condicionales - Operadores relacionales - Operadores lógicos - Predicados primitivos - Predicados simbólicos - Predicados numéricos	4	2	2	
6		- Predicados de equivalencia - Formas esp. condicionales Forma especial cond"	6	2	0	
7		Forma especial "cond – else" Forma especial "if"	4	2	0	2

8		- Estructuras de bloques Recursión e Iteración - Recursión simple - Recursión múltiple	6	2	2	
9		- Manejo de ciclos simples y anidados - Funciones utilizadas como parámetros	4	2	2	
10		Gráficos y tipos compuestos de datos simples Introducción a: - Gráficos	4	2	0	2
11		Introducción a: - Estructuras de datos - Gráficas simples	4	2	2	
12		Introducción a: - Gráficas simples	4	2	0	
13		Introducción a: - Pares - Listas	4	2	2	
14		- Operaciones con listas	4	2	2	
15		Tipos compuestos de datos -Vector - Operaciones con vectores	4	2	2	
16		Tipos compuestos de datos - Pares - Listas - Operaciones con listas	4	2	0	

EVALUACIÓN

Según el reglamento estudiantil vigente, en sus artículos 72 y 73.“...ARTÍCULO ***72o.:*** *Se entiende por Prueba Parcial aquella que se realiza individualmente para verificar el logro de los objetivos de las diferentes unidades o temas en que se divide cada asignatura. Estas* ***no podrán ser menos de dos*** *para cada asignatura...* ***ARTÍCULO 73o.:*** *Se entiende por Prueba Final aquella que se realiza individualmente para verificar el logro de los objetivos generales de cada asignatura. Esta prueba se realizará con estricta observancia de las fechas establecidas en el calendario académico...”,subrayado* y resaltado nuestro.

Se recomienda al profesor el siguiente sistema de evaluación:

- I Previa: 20%
- II Previa: 20%
- Talleres, Tareas, etc.: 20%
- Exàmen Final: 20%

- Proyectos: 20%

BIBLIOGRAFIA

MATTHIAS, Felleisen, FINDLER Robert Bruce, FLATT Matthew, KRISHNAMURTHI Shriram, “How to Design Programs An Introduction to Computing and Programming”, The MIT Press, Cambridge, Massachusetts. London, England Last modified: Wednesday, September 24th, 2003 US/Eastern, tomado de internet en: http://www.htdp.org/2003-09-26/Book, el 28 de julio de 2006.

ABELSON, H., Sussman, G. J. y SUSSMAN, J. “Structure and Interpretations of Computers Programs”. Second edition. The MIT Electrical Engineering and Computers Science Series, 1996. ISBN: 0-262-01153-0.

ABELSON, H., Sussman, G. J. y Sussman, J. “Structure and Interpretations of Computers Programs”. The MIT Electrical Engineering and Computers Science Series, 1993. ISBN: 0-262-01077-1.

KELSEY, R., CLINGER, W, Rees, J. y otros: “Revised5 Report on the Algorithmic Language Scheme”, 1998. http://www.uco.es/~ma1fegan/manuales/lia/r5rs.pdf

HARVEY, B. y WRIGHT, M. “Simply Scheme: Introducing Computer Science”. The MIT Press, 1994. ISBN: 0-262-08226-8.