



Ministerio de Cultura y Educación
 Universidad Nacional de San Luis
 Facultad de Ciencias Físico Matemáticas y Naturales
 Departamento: Informatica
 Area: Area V: Automatas y Lenguajes

(Programa del año 2014)
 (Programa en trámite de aprobación)
 (Presentado el 23/06/2015 22:31:35)

I - Oferta Académica

| Materia | Carrera | Plan | Año | Período |
|----------------------------------|--------------|------------|------|-----------------|
| DISEÑO Y PARADIGMAS DE LENGUAJES | ING. INFORM. | 026/1 2 | 2014 | 2° cuatrimestre |

II - Equipo Docente

| Docente | Función | Cargo | Dedicación |
|------------------------------|----------------------|-----------|------------|
| ERRECALDE, MARCELO LUIS | Prof. Responsable | P.Adj Exc | 40 Hs |
| CAGNINA, LETICIA CECILIA | Prof. Co-Responsable | P.Adj Exc | 40 Hs |
| SOSA TORANZO, CECILIA LORENA | Auxiliar de Práctico | Beca Doc | 10 Hs |

III - Características del Curso

| Credito Horario Semanal | | | | |
|-------------------------|----------|-------------------|---------------------------------------|-------|
| Teórico/Práctico | Teóricas | Prácticas de Aula | Práct. de lab/ camp/ Resid/ PIP, etc. | Total |
| Hs | 2 Hs | 1 Hs | 2 Hs | 5 Hs |

| Tipificación | Periodo |
|--|-----------------|
| B - Teoria con prácticas de aula y laboratorio | 2° Cuatrimestre |

| Duración | | | |
|------------|------------|---------------------|-------------------|
| Desde | Hasta | Cantidad de Semanas | Cantidad de Horas |
| 19/08/2014 | 21/11/2014 | 15 | 75 |

IV - Fundamentación

Las prácticas de programación desarrolladas en cursos previos como Programación I, Programación II y Programación III, brindan al futuro Ingeniero una experiencia concreta en la resolución de problemas utilizando lenguajes de programación particulares. Esta experiencia brinda también al alumno una primera aproximación a distintos aspectos y construcciones de los lenguajes de programación utilizados, como así también a los paradigmas a los cuales estos pertenecen. Acotar el conocimiento de un Ingeniero a un paradigma o lenguaje particular, acota su visión de las herramientas existentes a las particularidades del lenguaje y paradigma utilizados. Esto no responde en general a las necesidades actuales, con un mercado laboral que plantea problemas de diversa índole, cuya resolución efectiva puede requerir de construcciones y paradigmas de lenguajes diversos. Todo esto además, en un contexto donde permanentemente se le ofrecen al profesional informático nuevos lenguajes (o adaptaciones de los ya existentes) y paradigmas que intentan dar respuesta a las necesidades que los nuevos tipos de aplicaciones demandan.

Es así que surge la necesidad de brindar al futuro Ingeniero una visión más global de los lenguajes, que profundice más allá de la consideración superficial de sus "características" y se exploren los principales conceptos de diseño subyacentes y su efecto sobre la implementación de los lenguajes. Esta visión, de acuerdo a la bibliografía especializada en el tema permite, entre otras cosas, mejorar la habilidad para desarrollar algoritmos eficaces, mejorar el uso del lenguaje de programación disponible, acrecentar el propio vocabulario con construcciones útiles de programación, hacer posible una mejor elección del lenguaje de programación y facilitar el aprendizaje de un nuevo lenguaje. Si a todo esto, se le suma la identificación de los

principios subyacentes a los principales paradigmas en lenguajes de programación, y una comparación crítica entre los mismos, se puede decir que se le brindan al Ingeniero, las herramientas necesarias para enfrentar sus necesidades presentes y futuras a la hora de elegir y usar de manera adecuada un lenguaje de programación.

V - Objetivos / Resultados de Aprendizaje

Al finalizar el curso se espera que el alumno sea capaz de:

- Tener una perspectiva general de los paradigmas claves que se usan en el desarrollo de lenguajes de programación modernos, sus bases teóricas, aplicativas y de implementación: lenguajes Imperativos, Funcionales, Lógicos y Orientados a Objetos.
- Desarrollar una visión clara del tipo de situaciones en que los distintos paradigmas son adecuados y hacer uso de lenguajes multi-paradigma que permitan una fácil integración de los mismos y su interacción con lenguajes de programación existentes.
- Desarrollar y ejecutar programas cortos usando un lenguaje multi-paradigma.
- Evaluar en forma crítica distintos lenguajes de programación existentes y futuros.
- Entender la implementación de distintos lenguajes con suficiente detalle como para reconocer la relación entre un programa fuente y su comportamiento en ejecución.
- Extender sus conocimientos sobre los temas anteriores con bibliografía adecuada y mínima supervisión.

VI - Contenidos

UNIDAD I.

Razones del estudio de lenguajes de programación. Historia de los lenguajes de programación. Características de un buen lenguaje. La estructura y operación de una computadora. Computadora de hardware, de firmware y simulada por software. Traductores. Computadoras virtuales y ligaduras. Tiempos de ligadura. Paradigmas de lenguajes: imperativo, funcional, orientado a objetos y lógico.

UNIDAD II.

Especificación de Tipos de datos elementales y estructurados. Declaraciones. Chequeo de tipos. Conversión de tipos. Equivalencia de tipos.

UNIDAD III.

Administración de la memoria. Fases de la administración de memoria. Administración de memoria estática. Administración de memoria basada en Pila. Heap con elementos de tamaño fijo y de tamaño variable. Recolección de Basura. Ejemplos en Java.

UNIDAD IV.

Control de subprogramas. Llamada-retorno simple. Definición y activación de subprogramas. Subprogramas recursivos. Control de datos. Ambientes de referenciación. Alcance estático y dinámico. Estructura de bloques. Datos compartidos en subprogramas. Parámetros. Pasaje de parámetros. Ambientes comunes explícitos. Alcance dinámico. Alcance estático.

UNIDAD V.

Tipos de datos abstractos. Evolución del concepto de tipo de datos. Ocultamiento de la información. Encapsulamiento mediante subprogramas. Tipos de datos abstractos genéricos. Programación Orientada a Objetos: Herencia, Polimorfismo y ligadura dinámica. Subclases y subtipos. Herencia simple y múltiple. Aspectos de diseño en Smalltalk, C++ y Java.

UNIDAD VI.

Variantes en control de subprogramas. Excepciones. Corutinas. Subprogramas planificados. Comandos en guardia. Tareas. Ejemplos en Lenguajes representativos de distintos paradigmas.

UNIDAD VII.

Sintaxis de los lenguajes de programación. Criterios sintácticos generales. Etapas de la Traducción. Análisis sintáctico: Técnicas de Análisis Top-Down y Bottom-Up. Implementación de Analizadores Sintácticos.

UNIDAD VIII.

Lenguajes de programación multi-paradigma. Definición. Utilidad. Integrando características del enfoque imperativo, orientado a objetos, lógico y funcional. El caso de LEDA. Otros paradigmas de programación: lenguajes que soportan distribución y concurrencia. Lenguajes con restricciones. Programación Visual. Meta-programación. Características multi-paradigma en Python, Perl, MATLAB, Ruby y Oz.

VII - Plan de Trabajos Prácticos

Los prácticos de la asignatura son del tipo:

- a) Prácticos de aula, con un total de 30 horas.
- b) Prácticos de formación experimental, con un total de 15 horas.

a) Prácticos de aula:

Práctico 1: Tipos de Datos Elementales.

Objetivo: ejercitar sobre ligadura y tiempos de ligadura, comparando al menos dos lenguajes de distintos paradigmas. Además investigar sobre el funcionamiento de la máquina virtual Java.

Metodología: entrega de informe escrito de la experiencia realizada. Discusión en clase.

Práctico 2: Tipos de Datos Estructurados.

Objetivo: análisis de la especificación e implementación de tipos de datos en diferentes lenguajes. Comparación de las ventajas y desventajas del chequeo de tipos estático y dinámico.

Metodología: resolución de ejercicios en lápiz y papel.

Práctico 3: Administración de Memoria.

Objetivo: implementar en el lenguaje Java un heap de elementos de tamaño variable y el algoritmo de recolección de basura.

Metodología: resolución de ejercicios en lápiz y papel.

Práctico 4: Control de Secuencia y Datos en Subprogramas.

Objetivo: realizar ejercicios referidos a ambientes de referenciación, implementación de reglas de alcance estático y dinámico. Desarrollo de ejercicios con variantes en control de subprogramas: excepciones y concurrencia en Java.

Metodología: resolución de ejercicios en lápiz y papel. Investigación y discusión sobre las variantes en control de subprogramas, analizando estas construcciones en los lenguajes estudiados que las provean.

Práctico 5: Tipos de Datos Abstractos y POO.

Objetivo: análisis de la forma en que conceptos como herencia, encapsulamiento y polimorfismo basado en enlace dinámico, son especificados e implementados en distintos lenguajes de programación orientada a objetos.

Metodología: resolución de ejercicios en lápiz y papel. Implementar algunos de los ejercicios para corroborar los conceptos en máquina.

Práctico 6: Lenguajes Multi-paradigma.

Objetivo: este práctico deberá estar compuesto por ejercicios que cubran, al menos, los temas programación imperativa, programación orientada a objetos, programación funcional y programación lógica.

Metodología: resolución de ejercicios de tipo teórico, enunciados de problemas cuya resolución consista del uso de cada uno de los paradigmas. Indicar para cada uno encontradas en la resolución usando distintos paradigmas.

b) Prácticos de formación experimental:

Práctico 1: Implementación del Algoritmo de Recolección de Basura.

Objetivos: desarrollo de ejercicios donde se utilicen e implementen facilidades de administración de memoria utilizando el lenguaje Java. También se incluirán en este laboratorio el uso de las facilidades que Java provee para manejo de excepciones threads, etc.

Metodología: resolución de ejercicios en computadoras de escritorio que permitan visualizar los conceptos mencionados.

Práctico 2: Análisis Sintáctico LR utilizando la herramienta Cup.

Objetivos: comprensión básica del funcionamiento de un parser a través de la utilización de herramientas gratuitas disponibles para su generación.

Metodología: resolución de ejercicios en computadoras de escritorio utilizando la herramienta Cup.

VIII - Regimen de Aprobación

A. Régimen para alumnos Regulares:

1) Aprobar los prácticos de laboratorio.

2) Entregar, resueltos, al menos el 80% de los ejercicios de prácticos de aula, solicitados por la cátedra.

3) Aprobar un examen parcial, que incluye todos los prácticos, con al menos el 70% correcto del total.

Al menos el 50% de cada uno de los ejercicios involucrados en el parcial deberán ser completados correctamente para considerar su aprobación.

4) El parcial tendrá 1 (una) recuperación, más la recuperación adicional para aquellos alumnos que trabajan, según conste en la lista de regulares emitida por Sección Alumnos.

5) Se requiere el 70% de asistencia a clase, condición que será flexibilizada para los alumnos que hayan presentado el correspondiente certificado de trabajo en Sección Alumnos.

B. Régimen para alumnos Promocionales:

1) Ídem a lo requerido para alumnos Regulares, salvo que el alumno deberá asistir al 80% de las clases tanto teóricas como prácticas.

2) Aprobar con un mínimo de 7 (siete) un examen integrador oral y/o escrito al final del cuatrimestre.

3) La nota final se computará promediando las notas obtenidas en cada uno de los puntos mencionados previamente.

C. Para aprobar la materia, los alumnos regulares deberán rendir un examen final, el cual podrá ser oral y/o escrito.

D. No se admite rendir la materia en condición de libre.

IX - Bibliografía Básica

[1] "Programming Languages - Design and Implementation". Pratt, Terrence y Zelkowitz, Marvin. Cuarta edición. Prentice Hall, 2001.

[2] "Lenguajes de Programación - Diseño e Implementación". Pratt, Terrence y Zelkowitz, Marvin. Tercera edición. Prentice Hall, 1999.

[3] "Concepts of Programming Languages". Sebesta, Robert. Addison-Wesley. Cuarta Edición, 1999 y Sexta Edición, 2004.

[4] "El lenguaje de Programación C". Kernighan, Brian y Ritchie, Dennis. Prentice Hall, 1991.

[5] Apunte de la cátedra: "Aspectos de la Traducción de Lenguajes de Programación".

[6] Apunte de la cátedra: "Evolución del Concepto de Tipo de Datos-Tipo de Datos Abstractos-Programación Orientada a Objetos".

X - Bibliografía Complementaria

- [1] "Smalltalk-80. The Language and its implementation". Goldberg, Adele y Robson, David. Addison-Wesley, 1985.
- [2] The Java class libraries. Chan, Patrick - Lee, Rosanna y Kramer, Douglas. Addison Wesley. Segunda Edición, 1998.
- [3] "Piensa en Java". Eckel Bruce. Pearson Alhambra. Cuarta Edición, 2007.
- [4] Aho A. - Ullman J. The theory of parsing, translation and compiling. Vol. I. Prentice Hall (1973).

XI - Resumen de Objetivos

El curso tiene como objetivo introducir al alumno a la problemática del diseño e implementación de lenguajes de programación, incluyendo fundamentos teóricos y prácticos. El estudio se realiza teniendo en cuenta todos los paradigmas actuales de programación, realizando un estudio comparativo de las técnicas de implementación de cada uno de ellos.

XII - Resumen del Programa

Historia de los lenguajes de programación. Evolución de los paradigmas de programación. Computadoras virtuales. Características esenciales de los lenguajes de programación y su implementación: tipos de datos y su representación, control de secuencia y datos en subprogramas. Administración de Memoria. Abstracción de Datos. Análisis sintáctico. Técnicas Top-Down y Bottom-Up. Programación Multi-paradigma. Variantes en el control de subprogramas.

XIII - Imprevistos

| |
|--|
| |
|--|

XIV - Otros

| |
|--|
| |
|--|

ELEVACIÓN y APROBACIÓN DE ESTE PROGRAMA

| ELEVACIÓN y APROBACIÓN DE ESTE PROGRAMA | |
|---|----------------------|
| | Profesor Responsable |
| Firma: | |
| Aclaración: | |
| Fecha: | |