Ministerio de Cultura y Educación **(Programa del año 2011)**
Universidad Nacional de San Luis **(Programa en trámite de aprobación)**
Facultad de Ciencias Físico Matemáticas y Naturales **(Presentado el 11/08/2011 20:49:11)**
Departamento: Informatica
Area: Area V: Automatas y Lenguajes

I - Oferta Académica

Materia	Carrera	Plan	Año	Período
DISEÑO Y CONSTRUCCION DE COMPILADORES	LIC.EN CS.DE LA COMPUTACION	18/11	2011	2° cuatrimestre

II - Equipo Docente

Docente	Función	Cargo	Dedicación
ESQUIVEL, SUSANA CECILIA	Prof. Responsable	P.Tit. Exc	40 Hs
ARAGON, VICTORIA SOLEDAD	Responsable de Práctico	JTP Exc	40 Hs
FERRETTI, EDGARDO	Auxiliar de Práctico	JTP Exc	40 Hs

III - Características del Curso

Credito Horario Semanal				
Teórico/Práctico	**Teóricas**	**Prácticas de Aula**	**Práct. de lab/ camp/ Resid/ PIP, etc.**	**Total**
Hs	3 Hs	2 Hs	5 Hs	10 Hs

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

Duración			
Desde	**Hasta**	**Cantidad de Semanas**	**Cantidad de Horas**
09/08/2011	18/11/2011	15	150

IV - Fundamentación

El diseño y construcción de compiladores deben formar parte del conocimiento general de un Licenciado en Ciencias de la Computación, con el perfil requerido por el correspondiente plan de estudios.

Esta materia profundiza los conocimientos introductorios dados en la materia Análisis Comparativo de Lenguajes y aplica conceptos ya adquiridos en Autómatas y Lenguajes. Además las técnicas involucradas en la construcción de un compilador pueden aplicarse tanto en el ámbito del desarrollo de software de base como en el desarrollo de software de aplicación.

Los contenidos del curso se corresponden con las propuestas curriculares recomendadas por la ACM/IEEE Computer Society Joint Curriculum Task Force en 2001, para el área de compiladores.

V - Objetivos / Resultados de Aprendizaje

Desarrollar en el alumno la capacidad de diseñar e implementar un compilador para un subconjunto del Lenguaje de Programación C++.

VI - Contenidos

BOLILLA 1: INTRODUCCION

Concepto y tipos de traductores. Compiladores de una o más pasadas. Estructura Clásica de un Compilador: Modelo

de Análisis y Síntesis de un compilador. Sintaxis y Semántica.

BOLILLA 2: ANALIZADOR LEXICOGRAFICO O SCANNER

Breve recapitulación de los conceptos fundamentales.

BOLILLA 3: TABLA DE SIMBOLOS

Finalidad. Entradas de la tabla de símbolos. Descripción de objetos en la tabla de símbolos. Tratamiento de nombres de longitud fija y variable. Posibles implementaciones: lista, hash, árbol, otras. Representación de la información sobre tipos. Manipulaciones básicas. Control de bloques y de amplitud.

BOLILLA 4: PARSING TOP DOWN DETERMINISTICO

Definiciones de gramáticas LL(k) y LL(k) fuerte. Condición de LL(k) fuerte. Conjuntos FIRSTk, FOLLOWk, algoritmos de construcción de dichos conjuntos. Teoremas Fundamentales. Análisis top down por medio de tablas para k = 1. Análisis sintáctico top-down por el método recursivo descendente. Generalidades. Algoritmos de implementación de un parser descendente recursivo.

BOLILLA 5: DETECCION Y RECUPERACION DE ERRORES

Errores en las etapas lexicográfica, sintáctica, semántica y de ejecución. Esquema de recuperación de errores en entornos de parsing top-down.

BOLILLA 6: AMBIENTES DE TIEMPO DE EJECUCION

Revisión de las características básicas de los lenguajes de programación. Representación de tiempo de ejecución de objetos computacionales estáticos, semidinámicos y dinámicos. Organización de la memoria en tiempo de ejecución. Administración dinámica de la memoria como stack: activación de procedimientos, referenciación local, registros de activación, referenciación global, vector display. Administración dinámica de la memoria como heap: organización, administración del heap con elementos de tamaño fijo y variable, asignación inicial y reuso.

BOLILLA 7: REPRESENTACIONES INTERMEDIAS DE LOS PROGRAMAS FUENTES

Notación polaca, de n-uplas, de árboles de sintaxis abstracta. Código de máquina abstracta. Portabilidad.

BOLILLA 8: SISTEMA DE EJECUCION BASICO Y EXTENDIDO

Representación interna de los objetos en tiempo de ejecución: lógicos, caracteres, enteros, reales, arreglos. Análisis y traducción de expresiones aritméticas y lógicas, asignaciones, estructuras de control a nivel de sentencias, entrada-salida, estructuras de control a nivel de procedimientos, pasaje de parámetros por dirección y por valor. Referenciación global y local. Variables subindicadas: acceso, pasaje de arreglos y elementos de arreglos como parámetros.

BOLILLA 9: TRADUCCION DIRIGIDA POR LA SINTAXIS

Definiciones dirigidas por la sintaxis: atributos, atributos sintetizados y heredados, forma general de una definición dirigida por la sintaxis, grafo de dependencia de atributos. Definiciones S-atribuidas: traductores bottom up. Definiciones L-atribuidas. Esquemas de traducción dirigidos por la sintaxis. Restricciones para el cálculo de los valores de los atributos. Implementación de definiciones L-atribuidas en un entorno de parser top down. Eliminación de recursividad a izquierda en los esquemas de traducción. Parsing descendente recursivo que implementa definiciones L-atribuidas.

BOLILLA 10: CONTROL DE TIPOS

Introducción. Expresiones de tipo. Sistema de tipos. Salvaguarda de información del tipo asociado a los identificadores. Chequeo de tipo de expresiones y sentencias. Equivalencia de expresiones de tipo estructural.

BOLILLA 11: GENERACION DE CODIGO INTERMEDIO

Ubicación del generador de código dentro del proceso de compilación. Esquemas de traducción dirigidos por la sintaxis para la generación de código intermedio para expresiones, asignaciones, estructuras de control a nivel de sentencias y unidades. Generación de código intermedio embebido en un parser descendente recursivo.

VII - Plan de Trabajos Prácticos

1. Plan de trabajos prácticos de aula

PRACTICO 1: Tabla de Símbolos.
PRACTICO 2: Gramáticas LL(k).
PRACTICO 3: Parsing Descendente Recursivo.
PRACTICO 4: Recuperación de errores.
PRACTICO 5: Traducción dirigida por la sintaxis en entorno top down.
PRACTICO 6: Sistema de Ejecución Básico.
PRACTICO 7: Chequeo de tipos.
PRACTICO 8: Generación de Código.

2. Plan de trabajos prácticos de laboratorio: Actividad de diseño y proyecto que tiene por finalidad la implementación de módulos de un compilador (75 hs.)

2.1. Completar el módulo del Administrador de Tabla de Símbolos.
2.2. Completar el módulo del parser descendente recursivo.
2.3. Diseño e implementación de un módulo de recuperación de errores.
2.4. Diseño e implementación de un módulo de control de tipos.
2.5. Generación de código en una representación intermedia.

VIII - Regimen de Aprobación

F.1. Régimen para alumnos regulares

Para regularizar la materia los alumnos deberán aprobar:

1. Los prácticos de programación propuestos (fuente y ejecutable) y/o su correspondiente recuperación en las fechas estipuladas a tal efecto. La no aprobación de un trabajo práctico de máquina y/o su recuperación implicará la pérdida automática de la regularidad de la materia.

2. Un examen parcial práctico o su respectiva recuperación, con un porcentaje de ejercicos correctos del 70%.

3. Aquellos alumnos que han presentado, en tiempo y forma, el comprobante de trabajo, contarán con una recuperación adicional.

F.2. Los alumnos que han regularizado la materia para aprobarla deberán rendir un examen final que podrá ser escrito u oral.

F.3. Régimen de alumnos libres

En estos casos, el alumno tendrá una evaluación dividida en partes. En una se pedirá un Trabajo Especial, el cual es un compilador desarrollado bajo las pautas que se dan en el curso de la asignatura. En otra parte se tomará un examen escrito de carácter práctico. Finalmente, una parte oral y/o escrita
de teoría. Para su aprobación, se requiere la aprobación de las tres partes.

IX - Bibliografía Básica

[1] Aho, A. y Ullman, J. : "The Theory of Parsing, Translation and Compiling", Vol. I y II, Prentice Hall.
[2] Aho, A. y Ullman, J. : "Principles of Compiler Design", Addison Wesley.
[3] Aho, Setti y Ullman : "Compilers. Principles, Techniques and Tools", Addison Wesley.

X - Bibliografia Complementaria

[1] Gries, Davis: "Compilers Construction", John Wiley.
[2] Backhouse, R.C.: "Syntax of Programming Language", Prentice Hall.
[3] Davie, A. et al.: "Recursive Descendent Compiling",John Wiley.
[4] Barret y Coch: "Compiler Construction: Theory and Practice", Chicago SRA.
[5] Bauer et. al.: "An Advanced Course on Compilers", Springer Verlag.
[6] Waite y Goos: "Compiler Construction", Springer Verlag.
[7] Holub Allen: "Compiler Design in C", Prentice Hall
[8] Tremblay y Sorenson : “The Theory and Practice of Compiler Writing”, Mc Graw Hill, Computer Science Series.

XI - Resumen de Objetivos

Desarrollar en el alumno la capacidad de diseñar e implementar un compilador para un subconjunto del Lenguaje de Programación C++.

XII - Resumen del Programa

Descripción de los módulos de un compilador. Análisis Lexicográfico. Tabla de Símbolos. Análisis Sintáctico. Recuperación de errores. Análisis Semántico. Traducción dirigida por la sintáxis. Chequeo de tipos. Generación de código.

XIII - Imprevistos

XIV - Otros

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