UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS

Organización de Lenguajes y Compiladores 2
Primer semestre 2024

CÓDIGO	781	**CRÉDITOS**	5
ESCUELA	Ciencias y sistemas	**ÁREA**	Ciencias de la computación
PRE-REQUISITO	777 – Organización de lenguajes y compiladores 1 772 – Estructuras de datos	**POST REQUISITO**	281 – Sistemas operativos 1
CATEGORÍA	Obligatorio	**SEMESTRE**	Primero 2023
HORAS POR SEMANA DEL CURSO	4	**HORAS POR SEMANA DE LABORATORIO**	2
COORDINADOR DE ÁREA	Ing. Luis Espino		

Distribución de secciones

Clase magistral

Sección	Sala meet	De	A	Lu	Ma	Mi	Ju	Vi	Sa	Catedrático
A		07:10	8:50	X						Bayron Wosvely López López
A		12:10	13:50						X	Bayron Wosvely López López
B		07:10	10:30		X					Luis Fernando Espino Barrios
N		19:00	20:40		X		X			Edgar Rubén Sabán Raxon

Laboratorio

Sección	Sala meet	De	A	Lu	Ma	Mi	Ju	Vi	Sa	Tutor Académico
		17:20	19:00	X						Henry Ronely Mendoza Aguilar
B	https://meet.google.com/yoi-vmgv-gcf	19:00	20:40			X				José Andres Rodas Arrecis
N		16:30	17:10					X		Daniel Enrique Santos Godoy

Descripción

El laboratorio de organización de lenguaje y compiladores 2 perteneciente al área de ciencias de la computación, parte de una base teórica obtenida en la clase magistral y continúa con la implementación de un intérprete o compilador aplicando distintas fases para su correcto desarrollo.

Durante el desarrollo de laboratorio se explica el proceso para la construcción el cual consta de la teoría, análisis, diseño, eficiencia e implementación, se repasan conceptos de la clase aplicados de forma práctica como la traducción dirigida por la sintaxis, generación y optimización de código intermedio; que son útiles en muchos contextos más allá de compiladores los cuales pueden ser ingeniería de software y diseño de sistemas.

Se utilizarán herramientas que permitan la generación de analizadores léxicos y sintácticos para el desarrollo de las actividades de laboratorio.

Competencia general

Que el estudiante con base al contenido aprendido en clase y laboratorio conozca el proceso de desarrollo y construcción intérpretes y compiladores con la ayuda de generadores de análisis léxicos y sintácticos, el estudiante con la base práctica y teórica deberá ser capaz de realizar un análisis e implementar los problemas planteados en las actividades de laboratorio.

Metodología

- Se desarrollarán ejercicios prácticos, que serán la implementación de los ejercicios resueltos en la clase magistral.
- Se realizarán evaluaciones lo cual permitirá verificar los avances de los estudiantes en las actividades de laboratorio.
- Además de la bibliografía recomendada, se proporcionará al alumno material complementario que le ayude a conocer distintas técnicas en la elaboración de intérpretes y compiladores.
- Se realizarán dos proyectos para poder evaluar los conceptos adquiridos en clase, tomando en cuenta que pueden incluirse temas de cursos prerrequisitos.

Evaluación de rendimiento académico

El laboratorio tiene una ponderación de 34 puntos distribuidos de la siguiente manera:

Actividad	Ponderación	Publicación	Fecha entrega	Días	Porcentaje
Primer proyecto	15.3	14 de febrero	19 de marzo	38	40%
Segundo proyecto	18.7	20 de marzo	30 de abril	47	60%
Total	**34**	**-**	**-**	**-**	**100%**

Observaciones:

- Para aprobar el laboratorio se debe tener una nota final igual o mayor al 61% de los puntos, es decir 20.74 puntos de 34.
- Los proyectos son INDIVIDUALES, no está permitido compartir código con ningún estudiante. Las copias parciales o totales de los proyectos tendrán una nota de 0 puntos y los responsables serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas.
    - El código fuente de los proyectos deberá estar publicado en github haciendo uso de un repositorio privado donde el estudiante deberá proporcionar acceso a los auxiliares.
- Se tendrá una fecha y hora preestablecida para la entrega de los proyectos, al utilizar github se tomará como entrega el código fuente existente hasta el último commit realizado, por lo tanto, **si se hacen <u>más commits luego de la fecha y hora indicados no se tendrá</u> <u>derecho a calificación</u>.**

Sesiones

Las clases de laboratorio estarán divididas en sesiones, siendo una por semana y la misma para los laboratorios de todas las secciones.

Contenido

Unidad 1: Repaso

1. Procesadores de lenguaje
2. La estructura de un compilador
    2.1. Fases de un compilador
        2.1.1. Fase de análisis
        2.1.2. Fase de síntesis
3. Administración de la tabla de símbolos
4. Mecanismos para el paso de parámetros
5. Llamada por valor
6. Llamada por referencia
7. Llamada por nombre
8. Definición de sintaxis

- **Sesiones estimadas: 2 sesiones** (Del 16 de enero al 29 de enero)

Unidad 2: Traducción dirigida por la sintaxis

1. Esquemas de traducción orientados por la sintaxis
    1.1. Con acciones dentro de las producciones
    1.2. Eliminación de la recursividad por la izquierda de esquemas de traducción
    1.3. Con atributos heredados por la izquierda
2. Aplicaciones de la traducción dirigida por la sintaxis
    2.1. Expresiones aritméticas con un esquema de traducción sencillo
    2.2. Construcción de árboles de análisis sintáctico
        2.2.1. Expresiones
    2.3. Recorrido de un árbol de análisis sintáctico (intérprete)
        2.3.1. Obtener el valor implícito de una expresión
        2.3.2. Validación de errores
        2.3.3. Comprobación de tipos
        2.3.4. Conversión de tipos
3. Construcción de árboles de análisis sintáctico
    3.1. Sentencias de selección
    3.2. Sentencias cíclicas
    3.3. Sentencias de transferencia
4. Recorrido de un árbol de análisis sintáctico

4.1. Validación de errores
4.2. Comprobación de tipos
4.3. Estructuras y técnicas auxiliares para manejo de sentencias de transferencia 5. Construcción de árboles de análisis sintáctico
5.1. Métodos y funciones
6. Recorrido de un árbol de análisis sintáctico
6.1. Validaciones semánticas para métodos y funciones
6.2. Acceso a atributos
7. Construcción de árboles de análisis sintáctico
7.1. Arreglos
7.2. Structs
8. Recorrido de un árbol de análisis sintáctico
8.1. Acceso a arreglos
8.2. Validación de errores en arreglos
- **Sesiones estimadas: 4 sesiones** (Del 30 de enero al 19 de febrero)

Unidad 3: Introducción a la generación de código intermedio

1. Código de tres direcciones
1.1. Direcciones e instrucciones
1.2. Cuádruplos
2. Modelo básico orientado a objetos para la generación de código intermedio
2.1. ¿Qué es lo mínimo que se necesita para generar código intermedio?
2.1.1. Árbol de análisis sintáctico
2.1.2. Tabla de símbolos
2.2. Administración de la tabla de símbolos para un generador de código intermedio
2.2.1. Elementos esenciales de un símbolo
2.2.2. Tablas de símbolos encadenadas por alcance
2.2.2.1. Definición de ámbito
2.2.2.2. Implementación de un ámbito
3. Tipos y declaraciones
3.1. Expresiones de tipos y equivalencias
3.2. Declaraciones y distribución de almacenamiento
3.3. Secuencias de las declaraciones
3.4. Campos en registros
4. Construcción y recorrido de árbol para generación de código de tres direcciones
4.1. Traducción de expresiones
4.1.1. Operaciones dentro de expresiones
4.1.2. Traducción incremental
4.1.3. Direccionamiento de los elementos de un arreglo
4.1.4. Traducción de referencias a arreglos
4.1.5. Manejo de cadenas
4.2. Comprobación de tipos
4.2.1. Reglas para la comprobación de tipos
4.2.2. Conversiones de tipos
4.2.3. Sobrecarga de funciones y operadores
4.2.4. Inferencia de tipos
4.3. Flujo de control
4.3.1. Expresiones booleanas
4.3.2. Código de corto circuito
4.3.3. Instrucciones de flujo de control
4.3.3.1. If
4.3.3.2. If – else
4.3.3.3. While
4.3.3.4. For
4.3.3.5. Repeat
4.3.3.6. DoWhile
4.3.3.7. Switch
4.3.3.8. Traducción de sentencias de transferencia

4.3.3.8.1. Sentencia break
4.3.3.8.2. Sentencia continue
4.3.3.8.3. Salto incondicional goto
4.3.4. Traducción del flujo de control de las expresiones booleanas
4.3.5. Valores booleanos y código de salto
4.4. Parcheo de retroceso
4.4.1. Instrucciones de flujo de control
4.5. Código intermedio para procedimientos
4.5.1.Métodos y funciones
4.5.2.Variables globales
4.5.3.Variables locales
4.5.4.Parámetros
4.5.5.Llamadas
4.6. Código intermedio para structs
4.6.1. Manejo de structs
4.6.2. Auto referenciación y acceso a atributos de struct
4.7. Generación de grafo ast
• **Sesiones estimadas: 6 sesiones** (Del 20 de febrero al 2 de abril)

Unidad 4: Optimización de código intermedio
1. Fundamentos de optimización de código intermedio
1.1. Definición, objetivos e importancia de la optimización de código intermedio
2. Optimización por bloques
2.1. Generación de diagrama de bloques
2.2. Reglas de optimización por bloques
2.2.1. Restricciones en los bloques
2.3. Implementación de optimización por bloques
3. Optimización por mirilla

• **Sesiones estimadas: 3 sesiones** (Del 1 de abril de al 1 de mayo)
• **Tomar en cuenta que se tomará una sesión para la conferencia de cada laboratorio.**

Calificación de proyectos

- La calificación de los proyectos se realizará de forma virtual y desde los archivos ejecutables.
- Los archivos de entrada serán publicados 3 días antes de la calificación. Es responsabilidad del estudiante probar los archivos de entrada e indicar durante la calificación funcionalidades no realizadas.
- La calificación del proyecto será personal y existirá un tiempo límite.
- Anomalías o copias detectadas de proyectos tendrán de manera automática una nota de 0 puntos y los involucrados serán reportados a la Escuela de Ingeniería en Ciencias y Sistemas, para que se apliquen las sanciones correspondientes.
- Existirán horarios para la calificación de cada proyecto, por el cual el estudiante deberá de elegir el horario que mejor le convenga.
- Anomalías detectadas en los archivos entregables tendrá de manera automática una nota de 0 puntos, por ejemplo: no se envió el código correcto, se envió parte del código y no el código completo, archivos ajenos a los entregables del proyecto, no se hizo uso de las herramientas descritas en el enunciado de cada proyecto, entre otras.

Bibliografías y recursos

- **Libros de texto**
  - *Compiladores, Principios, Técnicas y Herramientas Aho, Sethi y Ullmam. PEARSON ADDISON- WESLEY, 2008, segunda edición.*

- **Documentación de herramientas para generadores de análisis léxicos y sintácticos**
  - Carter, Z. (2009). *Jison / Documentación*. Disponible en: https://zaa.ch/jison/docs/
  - Klein, G., Rowe, S. and Décamps, R. (1998). *JFlex - manual*. Disponible en: https://jflex.de/manual.html
  - Viswanadha, S. and Sankar, S. (n.d.). *JavaCC / Documentación*. Disponible en: https://javacc.github.io/javacc.
  - *CUP / Documentación*. Disponible en: http://www2.cs.tum.edu/projects/cup/docs.php.
  - *Ply / Documentación*. Disponible en: https://www.dabeaz.com/ply/

- **Tutoriales**
  - Python – Ply
    - https://ericknavarro.io/2020/02/10/24-Mi-primer-proyecto-utilizando-PLY/
    - https://ericknavarro.io/2020/03/15/26-Interprete-sencillo-utilizando-PLY/
    - http://kunusoft.com/slides/compi2/manual_ply_python/
    - http://kunusoft.com/slides/compi2/guia_heredados_pila/index.php?pic=0
  - Java – JavaCC
    - https://ericknavarro.io/2020/02/10/23-Mi-primer-proyecto-utilizando-JavaCC/
    - https://ericknavarro.io/2020/03/07/25-Interprete-sencillo-utilizando-Javacc/
  - Java – Flex y Cup
    - https://ericknavarro.io/2019/04/26/02-Mi-primer-proyecto-utilizando-Jlex-y Cup-Windows/
    - https://ericknavarro.io/2019/04/26/05-Interprete-sencillo-utilizando-Java-Jlex y-Cup/
  - Javascript/NodeJS – Jison
    - https://ericknavarro.io/2019/07/21/17-Mi-primer-proyecto-utilizando-Jison Linux/
    - https://ericknavarro.io/2019/08/01/20-Interprete-sencillo-utilizando-Jison-con Nodejs-Ubuntu/
  - C# – Irony
    - https://ericknavarro.io/2019/07/24/19-Mi-primer-proyecto-utilizando-Irony Windows/
    - https://ericknavarro.io/2019/08/07/22-Interprete-sencillo-utilizando-Irony-con CS/
  - Visual Basic – Gold Parser
    - https://ericknavarro.io/2019/07/22/18-Mi-Primer-Proyecto-Utilizando-GOLD Parser-Windows/
    - https://ericknavarro.io/2019/08/07/21-Interprete-sencillo-utilizando-GOLD Parser-y-Visual-Basic/