Universidad Católica del norte

ESCUELA DE INGENIERÍA UCN COQUIMBO

LENGUAJES DE PROGRAMACIÓN

TALLER 1 – DISEÑADORES DE LENGUAJES

2° SEMESTRE 2020 – PROFESOR: BASTIÁN RUIZ

OBJETIVOS

- Construir un traductor para un nuevo lenguaje de programación, aplicando la teoría, técnicas y herramientas actualmente disponibles.
- Utilizar la herramienta Bison-Flex para analizar un texto y aplicar reglas gramaticales.

DESCRIPCIÓN DEL TRABAJO

Deberán crear un lenguaje de programación, diseñado por ustedes mismos, que implemente algunas características básicas de los lenguajes que se usan actualmente, además de funcionalidades que lo diferencien del resto.

Con el lenguaje diseñado, tendrán que crear un "compilador" (que en realidad es un traductor) que recibirá como archivo fuente un programa escrito en su lenguaje, que luego interpretará las instrucciones y finalmente entregará el resultado en un archivo de código fuente Python.

El resultado del taller deberá estar lo más completo posible, puesto que se utilizará el mismo código para el taller 2.

DETALLE

Para este taller se les solicita desarrollar un nuevo lenguaje de programación. A este lenguaje deberán asignarle un nombre tentativo (beta o nombre código) y diseñarle una sintaxis en BNF o EBNF, la que debe ser **ORIGINAL** y cumplir con las siguientes características básicas.

- Poseer mecanismos para obtener valores ingresados por el usuario, como también imprimir valores en la consola de comandos.
- Permitir el almacenamiento de datos durante la ejecución del programa, tanto de forma individual como una serie de valores guardados secuencialmente (estructuras).
- Aceptar tipos de dato numérico mixto (enteros y decimales) y booleanos.
- Contener las 4 operaciones aritméticas (suma, resta, multiplicación y división), junto con la operación de cálculo de resto.
- Poseer estructuras de control de flujo, tanto una sentencia de selección como una sentencia iterativa.
    - La sentencia de selección debe controlarse mediante una expresión booleana.
    - La sentencia iterativa debe controlarse mediante una expresión booleana, e incrementar un contador de iteración.

Posteriormente, deberán usar Bison-Flex para programar su gramática y semántica del lenguaje y construir el traductor. Este permitirá realizar la traducción del código fuente escrito en su lenguaje a código Python ejecutable. En otras palabras, el traductor generado deberá recibir el código fuente de su lenguaje con la extensión elegida y "compilar" dicho archivo que genere otro con las instrucciones en Python, el cual se ejecutará de forma aparte y deberá ejecutar el algoritmo definido en su lenguaje de origen.

RESTRICCIONES

Para el desarrollo educativo de este taller, se **PROHIBE TERMINANTENTE** utilizar funciones y librerías de Python que no sean de I/O desde teclado, consola o de archivos. En otras palabras, solo se pueden utilizar:

- Funciones de entrada y salida de datos por consola (input y print respectivamente).
- Funciones para transformación de datos (int, float y similares).

No se podrán utilizar otras funciones que no sean las indicadas arriba (P.E: len). Tampoco pueden declarar objetos en el código generado.

Cualquier cambio a estas restricciones será informado oportunamente en la plataforma virtual.

ENTREGABLES SOLICITADOS

Los grupos de trabajo deben estar constituidos de 5 integrantes, los cuales no podrán cambiar durante el desarrollo del proyecto. Pueden usar los foros de la plataforma virtual para buscar integrantes. Dicha conformación debe ser avisada mediante un foro o correo electrónico con los nombres de los integrantes hasta el 30 de octubre de 2020.

La entrega de este primer taller considera los siguientes elementos, los que deben ser subidos a la plataforma virtual antes del 13 de diciembre:

TRADUCTOR

El traductor de su lenguaje consiste en archivos fuente de Bison-Flex utilizados para construir el traductor en sí, junto con el código fuente extra necesario. Por obligación, el código fuente debe ser compilable (En el caso contrario, no se revisará). Los archivos del traductor son:

- Archivo con el código fuente del analizador léxico (.l).
- Archivo con el código fuente del analizador sintáctico (.y).
- Archivos con el código fuente extra para la traducción del lenguaje (.c y .h).

DOCUMENTACIÓN BÁSICA

Para entender el lenguaje desarrollado, deberán entregar un manual sencillo de su lenguaje. Este debe estar desarrollado en LaTeX y tiene que contener lo siguiente:

- Portada.
- Índice de contenidos, tablas y figuras.
- Información de compilación del proyecto (Generación del traductor).
- Descripción BNF o EBNF del lenguaje.
- Sintaxis detallada de cada funcionalidad con información de uso.
- 1 ejemplo de código fuente con su output correspondiente.

Para trabajar el código del proyecto (traductor) deberán como grupo crear un repositorio Git. Este repositorio debe tener acceso de edición para los integrantes del grupo y acceso de visualización para el profesor y ayudante. Dicho repositorio se utilizará también para control de avance.

EVALUACIÓN

El formato de evaluación del taller 1 consiste en una nota única, cuyos criterios de evaluación se distribuyen en la siguiente tabla, donde cada criterio corresponde a una nota del 1.0 al 7.0.

COMPONENTE	CRITERIO	PORCENTAJE	DESCRIPCIÓN
MANUAL (40%)	Forma	10%	Construcción del informe en LaTeX cumpliendo con las normativas entregadas.
	Generación del compilador	5%	Indica la forma en como se puede crear el traductor con solo tener los archivos fuente.
	Sintaxis del lenguaje	45%	Detalle sintáctico de cada funcionalidad del lenguaje.
	Descripción (E)BNF	30%	Diseño del lenguaje en BNF o EBNF
	Ejemplos	10%	Ejemplos funcionales escritos en su lenguaje.
	TOTAL	**100%**	
TRADUCTOR (60%)	Input/Output	10%	Obtención de datos e impresión por consola.
	Operaciones Math	30%	Posibilidad de Sumar, restar, multiplicar y dividir valores y/o variables.
	Variables y Estructuras	10%	Instanciación de variables y estructuras de datos sencillas.
	Tipos de datos	10%	Declaración de tipos de datos numeral y booleano.
	Control de Flujo	25%	Incluir condicionales y ciclos
	Originalidad	5%	Creatividad de la estructura del lenguaje
	Historial	10%	Reportes semanales de actividad en Git.
	TOTAL	**100%**	

REGLAS DE ENTREGA

Respecto a la entrega de trabajos:

- Asegúrese que el archivo subido se encuentre en buen estado (no corrupto). Si no, **no podrá ser revisado e implicará la nota mínima en el taller (1.0)**.
- No se recibirán trabajos después de la fecha límite (13 de diciembre de 2020 23:59). Por lo tanto, si no registra una entrega antes del cierre del trabajo, se puntuará con la nota mínima (1.0). Por consiguiente, no se recibirán entregas por correo electrónico (no insista).

Respecto al manual:

- El manual debe desarrollarse en LaTeX y subir una copia en PDF. Si se envía en otro formato que no sea .pdf (.doc, .rtf, .txt, etc.) o el manual está hecho en otro procesador (Ej. Word), el manual de usuario **no será revisado y se puntuará con la nota mínima (1.0)**.

Respecto al traductor:

- Si no se respetan las restricciones indicadas en el apartado “RESTRICCIONES”, **el taller no será revisado, siendo evaluado con nota mínima 1.0**.
- Si el código fuente generado por el traductor no es código Python, **el taller se reprueba automáticamente, siendo evaluado con nota mínima 1.0**.
- Si el código fuente del taller no compila con las instrucciones entregadas en el manual, **el trabajo no será revisado, siendo evaluado el apartado de “Traductor” con nota mínima 1.0**.

EXTENSIÓN DEL TALLER 1

Como se anunció previamente, se extenderá este taller en plazo y condiciones. Por lo mismo, se realizará un solo taller (este) manteniendo las mismas condiciones de evaluación y aprobación (menor a 4.0, reprobado).

La fecha de entrega de este taller está definida para el 17 de enero de 2021, donde deberán enviar el código fuente del lenguaje y su manual de usuario. Para este taller, se les solicitará agregar ciertas funcionalidades. Estas están definidas a continuación en base al nombre tentativo de cada lenguaje.

Pytagoras

Pytagoras está ganando tanta popularidad por su forma de programar que ha sido bien vista para desarrollar aplicaciones de arquitectura. Por lo mismo, para no quedar obsoleto, es que se decidió implementar 2 funciones nuevas de forma nativa. La primera es una función para el cálculo de áreas. Estas áreas pueden ser de cuadrados o rectángulos. Y la segunda es una función de rendimiento de terreno, de tal forma que se genera una estructura que contenga el ancho y el largo de la superficie posible, y en esta estructura sea posible agregar distintos cuadrados o rectángulos, con el fin de saber si es posible construir los cuartos en ese terreno.

Cui#

Un grupo de diseñadores gráficos requiere una aplicación que permita procesar imágenes, y como necesitan desarrollar una aplicación específica, encontraron que la sintaxis y funcionalidades de Cui# son lo suficientemente robustas para poder construirla. Por lo mismo y para cumplir con su cometido, Cui# debe tener las siguientes funcionalidades: Debe tener una función nativa para leer imágenes PNG, y una función para guardar o imprimir la imagen (cualquiera de las 2). La función de lectura recibe la ruta donde se encuentra la imagen, y la función de guardado/impresión debe recibir dos parámetros, uno para la ubicación de la imagen (omitir si se imprimirá por pantalla) y el segundo parámetro para configurar la imagen, estas opciones son: Escala de Grises, Sepia o Reducción al 50%. Para que la aplicación logre su cometido, pueden utilizar las librerías de manejo de imágenes de Python.

Gollum

Gollum está diseñado para ser un lenguaje orientado a la administración económica, por lo que deben crear tres funciones para la prueba hogareña. La primera es importar un archivo de texto (la función recibe la ruta del archivo), que por cada línea tiene tres atributos separados por coma: Fecha, Operación (“Guardar” o “Recuperar”) y saldo. Este deberá generar una estructura donde por medio de una función y la fecha se pueda sumar o restar valores. Finalmente, debe haber una función de guardar el informe, utilizando la misma estructura y ordenando por fecha de la más antigua a la más nueva.

LPScript-4

En algunas ocasiones, a los programadores les piden realizar páginas web, pero muchas veces no tienen los conocimientos necesarios para generar un archivo. Por lo mismo, vieron el potencial de LPScript-4, y sus desarrolladores decidieron que sería bueno sumar 3 funcionalidades. La primera es la creación de una estructura que guarde todas las propiedades del documento web. En segundo lugar, la generación lineal de elementos web, donde se recibe el contenido del texto, la etiqueta HTML que lo engloba y sus estilos respectivos, como el color del texto o del fondo. Se debe tener en cuenta que un elemento puede estar dentro de otro. Por último, se necesita guardar este elemento en un archivo html, que debe ser generado por el lenguaje por medio de una ruta donde se guardará el archivo.