Compiladores Código: [**03307**]

Proyecto 1. Valor 1%

Temas de Estudio

- Tema 1. Introducción a los compiladores
- Tema 2. Un traductor simple orientado a la sintaxis

Objetivo

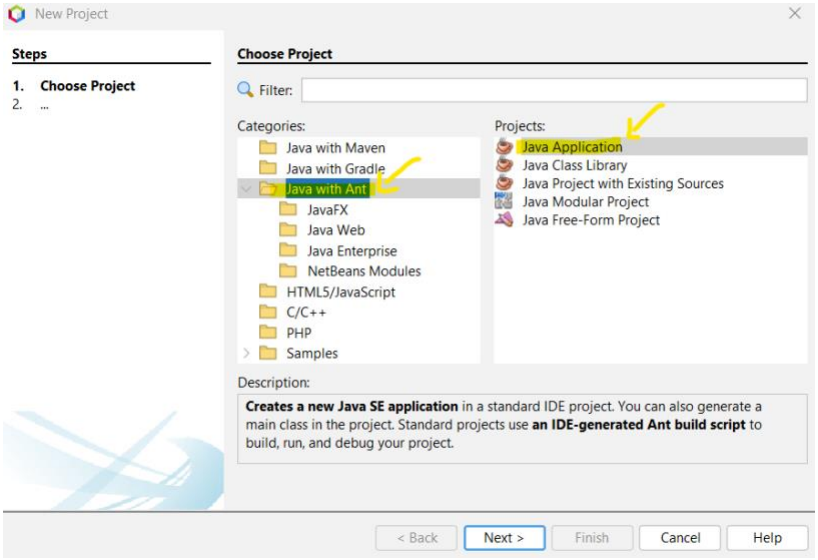
Aplicar los conocimientos adquiridos en el curso, centrándonos en la programación relacionada con los conceptos de compiladores, intérpretes, para validar a nivel de análisis léxico y la sintaxis, de un código fuente el cual es formado por entrada de cadenas de textos.

Software de Desarrollo

El proyecto debe desarrollarse utilizando el lenguaje de **programación Java** y el entorno de desarrollo **integrado (IDE) Apache NetBeans**, deberá instalar la versión más reciente o tres versiones antes de la última, no se aceptan entregas con proyectos desarrollados en versión obsoleta de Netbeans 8.2, en este enlace oficial podrá encontrar el instalador https://dlcdn.apache.org/netbeans/netbeans-installers/. Cuando ya lo tenga instalado debe considerar que hay tres maneras de crear proyectos en Apache Netbeans:

- Ant
- Gradle
- Maven

Se debe utilizar la primera manera que se refiere a proyectos tipo Ant, debido a que son proyectos que se moldean a nuestro aprendizaje y además tiene solución más liviana, en la siguiente imagen se muestra cómo crear un proyecto:



Se deberá crear específicamente un proyecto de tipo "***Java con Ant***" en modo carácter. Atención: No está permitido el uso de interfaces gráficas. Además, tenga en cuenta que **no** se permiten proyectos del tipo "Java con Maven" ni "Java con Gradle". Sino se cumple con lo indicado, el proyecto no será aceptado.

Desarrollo

PHP fue creado en 1994 por Rasmus Lerdorf, quien inicialmente desarrolló un conjunto de scripts CGI escritos en C conocidos como "Personal Home Page Tools" (Herramientas para Páginas Personales). PHP es un lenguaje interpretado, lo que significa que su código no necesita ser compilado antes de ejecutarse. En su lugar, un intérprete de PHP en el servidor lee y ejecuta el código directamente cada vez que se realiza una solicitud.

- Para este proyecto se requiere realizar un análisis que identifique los tokens o componentes léxicos en el código PHP, con el fin de verificar su uso adecuado y las estructuras sintácticas, de acuerdo con las reglas especificadas en la rúbrica del proyecto.

- El intérprete que usted desarrollará en Apache NetBeans, utilizando el lenguaje Java en un proyecto tipo "Java con Ant", debe ser completamente independiente de una ruta física. Esto significa que el programa podrá ejecutarse en cualquier máquina y directorio sin depender de configuraciones específicas. Para su ejecución, se utilizará la consola de Windows (CMD), ejecutando el archivo compilado (.jar).

- El proyecto implicará la lectura de un archivo con extensión .php que contenga un ejemplo de código en PHP.

- Para que este archivo .php pueda ser leído, no deberá depender de un nombre específico ni de una ruta estática. En su lugar, deberá ubicarse en el mismo directorio que el archivo compilado .jar.
- Es necesario detectar y registrar los posibles errores en un archivo de salida con extensión .log, conforme a las especificaciones detalladas en la rúbrica.

- No se permite el uso de librerías de terceros, entendidas como fragmentos de código desarrollados por otras personas y encapsulados en componentes que podrían servir como referencia para la solución del problema.

Código fuente:

- El código que se muestra a continuación, corresponde al juego tradicional "**Ahorcado**" y está desarrollado en el lenguaje programación PHP.
- El código no tiene ningún error y lo puede ejecutar online en el siguiente sitio que interpreta código PHP https://www.programiz.com/php/online-compiler/
- Usted puede trabajar a partir de dicho código para realizar lo que se le solicita, pero tenga en cuenta que el código, será modificado por el tutor (a) para agregar errores según se mencionan en la rúbrica para poder comprobar si el estudiante realizó la validación que se le solicitó.

- A continuación, se especifica el código con el que se estará trabajando en este proyecto, por favor tome en consideración que dicho código está disponible en un archivo descargable en la plataforma “Moodle” con la extensión .php

```php
<?php
// Definimos un array con las palabras posibles para el juego
$palabras = ["compiladores", "sintaxis", "semantica"];

// Seleccionamos aleatoriamente una palabra secreta de la lista
$palabra_secreta = $palabras[array_rand($palabras)];

// Establecemos el número de intentos permitidos
$intentos = 6; // Número total de intentos que el jugador tiene para adivinar

// Inicializamos la variable que muestra las letras acertadas como guiones bajos
$letras_acertadas = str_repeat('_', strlen($palabra_secreta));
// Creamos un array para almacenar las letras que ya se han utilizado
$letras_usadas = [];

// Mostramos un mensaje de bienvenida y la palabra actual (oculta)
echo "¡Bienvenido al juego del ahorcado!\nPalabra: " . implode(' ', str_split($letras_acertadas)) . "\n";

// Iniciamos el bucle principal del juego
while ($intentos > 0 && $letras_acertadas != $palabra_secreta) {
    // Mostramos los intentos restantes y las letras que ya se han usado
    echo "Tienes $intentos intentos restantes.\nLetras usadas: " . implode(', ', $letras_usadas) . "\n";
    // Pedimos al usuario que ingrese una letra
    // Convertimos la letra a minúsculas y eliminamos espacios
    $letra = strtolower(trim(readline("Ingresa una letra: ")));

    // Verificamos si la entrada es válida
    if (strlen($letra) != 1 || !ctype_alpha($letra) || in_array($letra, $letras_usadas)) {
        echo "Entrada inválida o letra ya usada.\n";
        // Volvemos al inicio del bucle para pedir otra letra
        continue;
    }

    // Agregamos la letra ingresada a la lista de letras usadas
    // Añadimos la letra a la lista de letras utilizadas
    $letras_usadas[] = $letra;
    // Comprobamos si la letra está en la palabra secreta
    if (strpos($palabra_secreta, $letra) !== false) {
        // Si la letra es correcta, actualizamos las letras acertadas
        for ($i = 0; $i < strlen($palabra_secreta); $i++) {
            if ($palabra_secreta[$i] == $letra) {
                // Actualizamos la letra acertada en la posición correspondiente
                $letras_acertadas[$i] = $letra;
            }
        }
    } else {
        // Si la letra es incorrecta, disminuimos el número de intentos
        // Restamos un intento si la letra no está en la palabra secreta
        $intentos--;
    }
    // Mostramos la palabra actual con las letras acertadas
    echo "Palabra: " . implode(' ', str_split($letras_acertadas)) . "\n";
}
// Al finalizar, mostramos si el usuario ganó o perdió
echo $letras_acertadas == $palabra_secreta ? "¡Felicidades! Adivinaste: $palabra_secreta\n" : "Perdiste. Era: $palabra_secreta\n";
?>
```

Evaluación:

#	Requerimientos a evaluar
1	**Lectura archivo**: se debe leer un archivo de código PHP con extensión .php no debe leer otra extensión que no sea la indicada, de lo contrario no está cumpliendo con este punto. • La ejecución del ejecutable .jar debe poder realizarse desde la consola CMD o símbolo del sistema. • En la plataforma se provee un ejemplo en dicho formato de extensión indicado, debe utilizar ese. • Al leer el archivo su programa debe aceptar espacios en blanco, líneas en blanco, cualquier espacio en general que se encuentre y que se distinga entre mayúsculas y minúsculas. Es decir, el formato original del archivo debe ser respetado en su totalidad. • La lectura del archivo debe ser cargada en la memoria del programa para realizar los diferentes análisis y verificaciones propuestos en este proyecto. • Es importante, que el archivo .php, no dependa de un nombre en específico para poder ser leído, y que no dependa de una ruta estática o física ni configuraciones personalizadas por el desarrollador (estudiante), **de lo contrario no está cumpliendo con este punto.** • Donde se encuentre el compilado de su programa .jar en la misma carpeta esté el archivo .php a leer

2	**Manejo de errores**: A partir del archivo que se leyó se debe crear otro archivo con extensión .log
	• El nombre del nuevo archivo debe conservar el nombre original y agregarle un guion con la descripción "-errores.log" el archivo debe verse de esta manera: "nombreArchivo-errores.log" donde "nombreArchivo" depende del nombre que tenga dicho archivo de entrada .php
	• Se debe respetar el contenido y formato del archivo original .php, únicamente, deberá agregarle 4 dígitos al inicio de la línea seguidos del texto, por ejemplo:
	`0001 <?php`
	`0002 $palabras = ["compiladores", "sintaxis", "semantica"];`
	`0003 $palabra_secreta = $palabras[array_rand($palabras)];`
	• En los próximos puntos a evaluar de esta rúbrica, cuando se encuentre un incumplimiento, se deberá indicar como error utilizando el siguiente formato:
	○ Número del error: la tipificación es a criterio del desarrollador
	○ Descripción: corresponde a un error especifico del problema encontrado no se debe hacer errores generalizados sino debe considerar ser muy puntual con lo que sucede.
	Se debe ver de la siguiente manera:
	Error 200. No encuentra la etiqueta de apertura <?php
	• Otra manera de agregar los errores es al final del archivo, pero debe de indicar el número de línea donde encontró el error y respetar lo anterior, ejemplo:

	Error 200. Línea 0001. No encuentra la etiqueta de apertura <?php • Considere que en una misma línea puede haber más de un error que debe detectar.

3	**Identificadores**: En PHP, un identificador es un nombre utilizado para identificar variables, funciones, clases u otros elementos en el código.
	Los identificadores deben seguir ciertas reglas para ser válidos:
	• Comienzan con una letra o un guion bajo (_): Un identificador debe comenzar con una letra (a-z, A-Z) o con un guion bajo (_).
	• Pueden contener letras, números y guiones bajos: Después del primer carácter, los identificadores pueden contener letras, números (0-9) y guiones bajos.
	• Sensible a mayúsculas y minúsculas: PHP distingue entre mayúsculas y minúsculas en los identificadores. Por ejemplo, $Variable y $variable son diferentes. El signo dólar($) se usa para declarar y referencias variables.
	• La variable es precedida por un símbolo de igual y luego el contenido, para este caso se valida del igual hacia la izquierda
	• No puedes usar palabras reservadas de PHP para este proyecto solo validaremos las siguientes palabras reservadas.
	
	class | function | if | while
	echo | else | continue | for
	return | class | public | null
	implode | false | true | array_rand
	str_repeat | strlen | str_split | strtolower
	trim | readline | ctype_alpha | in_array
	strpos | | |
	
	• **Sensible a mayúsculas y minúsculas**: PHP distingue entre mayúsculas y minúsculas en los identificadores. Por ejemplo, $Variable y $variable son diferentes.
	
	• Ejemplos de identificadores válidos:
	$variable; // Correcto, comienza con una letra
	$_variable123; // Correcto, comienza con un guion bajo y puede contener números
	$VariableConMayus; // Correcto, PHP distingue entre mayúsculas y minúsculas
	
	• Ejemplos de identificadores no válidos:
	$123variable; // Incorrecto, no puede comenzar con un número
	$#variable; // Incorrecto, el símbolo `#` no está permitido
	$if; // Incorrecto, "if" es una palabra reservada

4	**Etiquetas apertura y cierre**

	Las etiquetas de apertura y cierre en PHP (<?php y ?>) tienen algunas reglas sintácticas importantes que debes seguir para que el código funcione correctamente. Aquí están las principales reglas: 1. Etiqueta de apertura: <?php • Debe usarse al comienzo de un bloque de código PHP. • Es la forma estándar para abrir un bloque PHP y es compatible en todas las configuraciones de PHP. • No puede haber ningún espacio en blanco ni texto antes de esta etiqueta 2. Etiqueta de cierre: ?> • Se usa para cerrar el bloque de código PHP. • Es opcional al final de archivos que contienen solo PHP. ¿En muchos proyectos, es común omitir ?> para evitar problemas con espacios en blanco adicionales. Pero para este proyecto será obligatorio que esté en el código a evaluar, en caso que no esté se debe evidenciar el error. O bien si se encuentra en una posición diferente a la última línea debe dar error.

5	**Comentarios**
	Los **comentarios** en PHP son una forma de agregar notas o explicaciones dentro del código que no afectan su ejecución. Son útiles para documentar el código, explicar funciones o dejar recordatorios para otros desarrolladores (o para ti mismo). PHP admite dos estilos de comentarios:
	1. Comentarios de una sola línea
	• **Con //**
	php
	// Esto es un comentario de una línea
	echo "Hola, mundo!"; // Imprime un saludo
	• **Con #**:
	php
	# Esto también es un comentario de una línea
	echo "Hola, mundo!";
	2. Comentarios de varias líneas:
	Estos comentarios se utilizan para escribir anotaciones más largas o para comentar bloques de código. Se encierran entre /* y */
	Para efectos de este proyecto no serán evaluados los comentarios de varias líneas y se entenderán que están correctos en caso de encontrarse.

Las reglas a cumplir son las siguientes para este proyecto (podrá diferir con las instrucciones reales de PHP, por eso debe basarse en estas reglas)

Evitar en estructuras de control y expresiones: No coloques comentarios en medio de estructuras de control (if, for, while, etc.) ni dentro de expresiones complejas, ya que pueden interrumpir el flujo del código y causar errores.

Ejemplo correcto

```
// Este es un comentario de una sola línea usando //
$variable = 10; // Inicialización de variable

# Este es otro comentario de una sola línea usando #
if ($variable > 5) { // Comprobamos si la variable es mayor que 5
    echo "La variable es mayor que 5"; # Mensaje de salida
}
```

Ejemplo incorrecto, nótese que interrumpe antes del ;

```
$variable = 10 # Error: falta el punto y coma aquí antes del comentario
; // Esto generará un error de sintaxis
```

Otro incorrecto, nótese que no ha terminado la sentencia del if

```
if ($variable > // Comentario incorrecto aquí
10) {
    echo "La variable es mayor que 10";
}
```

Otro incorrecto

El comentario interrumpe la condición `while` antes de que termine, lo que provoca un error de sintaxis, ya que la condición queda incompleta.

```
while ($intentos > // Comentario incorrecto en medio de la condición
0) {
    echo "Intentos restantes: $intentos";
    $intentos--;
}
```

Otro incorrecto porque interrumpe la instrucción del while con su paréntesis redondo de cierre de la condición.

```php
while ($intentos > 0 # Comentario incorrecto en medio de la condición
) {
    echo "Intentos restantes: $intentos";
    $intentos--;
}
```

Es un comentario incorrecto debido a que no tiene un cierre y interrumpe ese comentario principal

```php
<?php // comentario incorrecto
$variable = 10;
```

6 **Etiquete apertura y cierre:** En PHP, las reglas para validar la etiqueta de apertura <?php y la de cierre ?> son las siguientes:

1. **Etiqueta de apertura <?php**
   - Siempre debes usar <?php para iniciar el código PHP
   - Debe iniciar al principio de un archivo o en cualquier parte del mismo, pero debe estar fuera de comentarios y no puede estar dentro de una cadena de texto.

```php
<?php
// Definimos un array con las palabras posibles para el juego
$palabras = ["compiladores", "sintaxis", "semantica"]; // Lista de palabras del juego

// Seleccionamos aleatoriamente una palabra secreta de la lista
$palabra_secreta = $palabras[array_rand($palabras)]; // La palabra que el usuario debe adivinar
```

2. **Etiqueta de cierre ?>**
   - La etiqueta cierre debe estar al final del archivo
   - Para la verificación de este proyecto la etiqueta del cierre será obligatoria
   - No debe haber espacios o saltos de línea después de ?>

```php
// Al finalizar, mostramos si el usuario ganó o pe
echo $letras_acertadas == $palabra_secreta ? "¡Fe
?>
```

En caso que no se cumpla ninguna regla de las etiquetas de apertura o cierre deberá generar error.

7 **Documentar:** Documente en un archivo de texto con extensión .txt llamado “README.txt”, los pasos necesarios para ejecutar su programa, junto con cualquier información adicional que desee clarificar al profesor sobre su proyecto. Asegúrese de verificar y confirmar que estos pasos funcionen correctamente antes de la entrega.

Honestidad Académica

https://audiovisuales.uned.ac.cr/play/player/23048

Nota Importante

Cada estudiante es responsable del contenido que entrega, si no es el archivo correcto, no podrá entregarlo posterior a la fecha establecida.

Si el contenido del archivo coincide con algún otro estudiante, o se comprueba que no es de su autoría, se aplicaría lo indicado en la plataforma en el documento Lineamientos ante casos de plagio

Indicaciones Importantes

- Es obligatorio que incluya todo el directorio donde se encuentra < nombre del instrumento>.
- La < nombre del instrumento > debe estar desarrollado en [IDE de desarrollo] que es la herramienta oficial del curso.
- El programa debe ser modular, utilizando de la mejor manera funciones definidas por usted.
- Los trabajos deben realizarse en forma individual. Dentro del código del programa debe de indicar la documentación que explique cómo fue realizado el programa.
- Si utiliza código de algún ejemplo del libro, o de otra fuente que no sea de su autoría, debe de indicarlo.
- Comprima todos los archivos en un solo archivo .zip o .rar.
- Nombre del archivo que envía: debe ser nombre y primer apellido del estudiante, y nombre de la tarea. Ejemplo: JuanRojas-tarea1.
- La entrega de la <Nombre del instrumento>en las fechas establecidas en la plataforma de aprendizaje en línea Moodle en el apartado que se indique.
- Si no concluyó a tiempo la tarea, debe entregar lo que pudo hacer e incluir una carta explicando las razones por las cuales no finalizó.

Rúbrica de Evaluación

Criterio	Cumple a satisfacción lo indicado en la evaluación	Cumple medianamente en lo indicado en la evaluación	Cumple en contenido y formato, pero los aportes no son significantes	No cumple o no presenta lo solicitado
1. **Lectura del archivo:** Al leer el archivo su programa debe aceptar espacios en blanco, líneas en blanco, cualquier espacio en general que se encuentre y que se distinga entre mayúsculas y minúsculas. Es decir, el formato original del archivo debe ser respetado en su totalidad. La ejecución del ejecutable .jar debe poder realizarse desde la consola CMD o símbolo del sistema.	5	2	1	0
2. **Lectura del archivo**: Es importante, que el archivo .php, no dependa de un nombre en específico para poder ser leído, y que no dependa de una ruta estática o física ni configuraciones personalizadas por el desarrollador	10	5	1	0
3. **Lectura del archivo**: Donde se encuentre el compilado de su programa .jar en la misma carpeta esté el archivo .php a leer	5	2	1	0
4. P**ara el manejo de errores**: A partir del archivo que se leyó se debe crear otro archivo con extensión .log	5	2	1	0
5. **Para el manejo de errores**: El nombre del nuevo archivo debe conservar el nombre original y agregarle un guion con la descripción “-errores.log” el archivo debe verse de esta manera: “nombreArchivo-errores.log” donde “nombreArchivo” depende del nombre que tenga dicho archivo de entrada .php	5	2	1	0
6. **Para el manejo de errores**: Se debe respetar el contenido y formato del archivo original .php, únicamente, deberá agregarle 4 dígitos al inicio de la línea seguidos del texto	5	3	1	0
7. **Para el manejo de errores:** Se deberá indicar como error utilizando el siguiente formato: - Número del error: la tipificación es a criterio del desarrollador - Descripción: corresponde a un error especifico del problema encontrado no se debe hacer errores generalizados sino debe considerar ser muy puntual con lo que sucede	10	5	2	0

8.**Identificadores**: Comienzan con una letra o un guion bajo (_): Un identificador debe comenzar con una letra (a-z, A-Z) o con un guion bajo (_).	5	3	1	
9.**Identificadores**: Sensible a mayúsculas y minúsculas: PHP distingue entre mayúsculas y minúsculas en los identificadores. Por ejemplo, $Variable y $variable son diferentes.	5	3	1	0
10.**Identificadores**: No puedes usar palabras reservadas de PHP	5	3	1	
11. **Etiqueta Apertura <?php** - Siempre debes usar <?php para iniciar el código PHP - Debe iniciar al principio de un archivo o en cualquier parte del mismo, pero debe estar fuera de comentarios y no puede estar dentro de una cadena de texto	10	5	2	0
12. **Etiqueta de cierre ?>** - La etiqueta cierre debe estar al final del archivo - Para la verificación de este proyecto la etiqueta del cierre será obligatoria - No debe haber espacios o saltos de línea después de ?>	10	5	2,5	0
13. **Comentarios**: No puedes poner comentarios en lugares donde el intérprete espera que haya código válido. Si colocas un comentario en un lugar como este, el intérprete generará un error porque no podrá encontrar la instrucción o el valor que espera. No puedes poner comentarios en la misma línea donde hay una instrucción si el comentario rompe la sintaxis, como en una declaración de variable antes del punto y coma, una condición, o una estructura de control.	15	7,5	3,5	0
14. **Documentar**: Documente en un archivo de texto con extensión .txt llamado “README.txt”, los pasos necesarios para ejecutar su programa, junto con cualquier información adicional que desee clarificar al profesor sobre su proyecto. Asegúrese de verificar y confirmar que estos pasos funcionen correctamente antes de la entrega.	5	2,5	1	0
TOTAL	**100**	**50**	**20**	**0**