LENGUAJES Y AUTOMATAS II

PROYECTO: COMPILADOR

Contenido

[1 Introducción 3](#_Toc128489134)

[2 Objetivo 3](#_Toc128489135)

[3 Investigación 3](#_Toc128489136)

[3.1 Expresiones regulares 3](#_Toc128489137)

[3.2 Compilador 4](#_Toc128489138)

[3.2.1 ¿Qué es un compilador? 4](#_Toc128489139)

[3.2.2 ¿Qué hace un compilador? 4](#_Toc128489140)

[3.2.3 Vista general de un compilador 4](#_Toc128489141)

[3.2.4 ¿Cómo se construyen los compiladores? 4](#_Toc128489142)

[3.3 Git y GitHub 4](#_Toc128489143)

[3.4 Conocimientos necesarios 5](#_Toc128489144)

[3.4.1 Diccionario 5](#_Toc128489145)

[3.4.2 Pila 6](#_Toc128489146)

[3.4.3 Sintaxis 6](#_Toc128489147)

[3.4.4 Semántica 6](#_Toc128489148)

[4 Elección de herramientas 7](#_Toc128489149)

[4.1 Python 7](#_Toc128489150)

[4.2 ANTLR 7](#_Toc128489151)

[4.3 Git 7](#_Toc128489152)

[5 Fases de desarrollo 8](#_Toc128489153)

[5.1 Primera fase (Analizador Léxico) 8](#_Toc128489154)

[5.1.1 Glosario de terminos respecto al análisis léxico 8](#_Toc128489155)

[5.2 Segunda fase 9](#_Toc128489156)

[5.3 Tercera fase 9](#_Toc128489157)

[6 Prototipo 9](#_Toc128489158)

[7 Resultados 9](#_Toc128489159)

[8 Conclusiones 9](#_Toc128489160)

# Introducción

Un compilador es un programa que traduce el código escrito en un lenguaje de programación a otro lenguaje. Esto se hace normalmente para crear un programa ejecutable que pueda ejecutarse en un ordenador. Los compiladores son una herramienta esencial para el desarrollo de software y se utilizan en muchos sectores. Suelen estar diseñados para optimizar el código que generan para que se ejecute de la forma más eficiente posible, y a menudo realizan otras tareas como la comprobación de errores y la optimización. Algunos compiladores populares son GCC, LLVM y Clang.

Sin embargo, los compiladores no están exentos de limitaciones. Uno de los mayores retos a la hora de diseñar compiladores es encontrar un equilibrio entre la optimización y la legibilidad del código. Un código muy optimizado puede ser difícil de leer y mantener, lo que puede dificultar su depuración y actualización en el futuro. En algunos casos, puede ser más importante dar prioridad a la legibilidad que a la optimización.

Otro problema de los compiladores es que a veces pueden introducir fallos o errores en el código. Esto puede ocurrir si el compilador hace suposiciones incorrectas sobre el código o si hay fallos en el propio compilador. Para solucionar este problema, muchos compiladores se someten a pruebas rigurosas y se actualizan con frecuencia para corregir los fallos que se descubren.

El desarrollo de éste proyecto tiene razón de dar al alumno la comprensión de la teoría de automatas y desarrollar habilidades necesarias para construir un compilador. En éste documento se encuentran datos relevantes de lo que he investigado, mis problemáticas a la hora de buscar información, los obstáculos o dudas que tuve y como las resolví, así como consejos para otros desarrolladores que tengan una asignación como el desarrollo de un compilador.

# Objetivo

* Implementar un compilador para un lenguaje específico considerando las etapas del mismo.
* Hacer representaciones de código
* Mostrar las fases de desarrollo del proyecto en sus ámbitos léxicos y sintácticos.

# Investigación

## Expresiones regulares

El uso de Regex es sólo en el análisis sintáctico de elementos muy simplistas, como es un número un nombre de variable, etc. Enfocarnos mucho en ellas no es muy útil, sin embargo, saberlas usar lo es demasiado. Identificar datos de entrada y hacer match usando las expresiones regulares nos permite adaptarnos a muchos problemas. De forma inmediata pensemos en algunos ejemplos “identifica todos los correos que pertenezcan a Gmail” “Identifica todos los correos que pertenezcan a Outlook”. Ahora supongamos que queremos invertir la palabra “Gmail” en todos los nombres de los correos coincidentes quedando así “liamG”. Saber usar las expresiones regulares nos ayuda a mejorar los programas en el ambito de leer los datos. Aun así, para el desarrollo de éste proyecto, saberlas usar es un requisito, sin embargo, el principal objetivo es tener un compilador.

## Compilador

### ¿Qué es un compilador?

Un compilador es un programa especial que traduce el código fuente de un lenguaje de programación a código máquina, bytecode u otro lenguaje de programación. El código fuente suele estar escrito en un lenguaje de alto nivel legible por humanos, como Java o C++. Un programador escribe el código fuente en un editor de código o en un entorno de desarrollo integrado (IDE) que incluya un editor, y guarda el código fuente en uno o varios archivos de texto. Un compilador compatible con el lenguaje de programación fuente lee los archivos, analiza el código y lo traduce a un formato adecuado para la plataforma de destino.

Los compiladores que traducen el código fuente a código máquina se dirigen a sistemas operativos y arquitecturas de ordenador específicos. Este tipo de salida se denomina a veces código objeto (que no está relacionado con la programación orientada a objetos). El código máquina resultante está formado íntegramente por bits binarios (1 y 0), por lo que puede ser leído y ejecutado por los procesadores de los ordenadores de destino.

### ¿Qué hace un compilador?

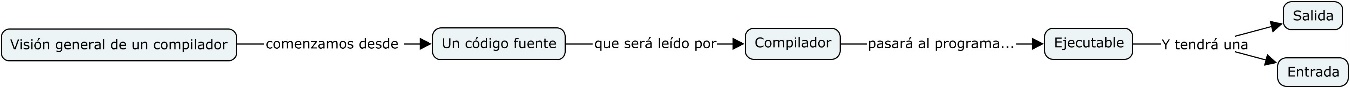
Los compiladores generan código maquina y trabajan tomando el código fuente como entrada y realizando una serie de transformaciones sobre él para generar el código de salida. La primera etapa del proceso suele ser el análisis léxico, que consiste en dividir el código en una serie de tokens. A continuación, estos tokens se analizan sintácticamente para determinar la estructura del código.

Una vez determinada la estructura del código, el compilador puede empezar a generar el código de salida. Esto se hace normalmente en una serie de etapas, cada una de las cuales realiza una transformación diferente en el código. Estas transformaciones incluyen aspectos como la optimización, la generación de código y la comprobación de errores.

Una de las ventajas de utilizar un compilador es que puede ayudar a detectar errores en el código antes de ejecutarlo. Esto puede ahorrar mucho tiempo y esfuerzo en el proceso de depuración. Además, los compiladores a menudo pueden generar código más eficiente que el que un programador humano podría escribir a mano.

En general, los compiladores son una herramienta esencial para el desarrollo de software, y han sido fundamentales en el desarrollo de muchas de las tecnologías de las que dependemos hoy en día.

### Vista general de un compilador



### ¿Cómo se construyen los compiladores?

## Conocimientos necesarios

En la parte práctica de éste proyecto se manejarán mucho los diccionarios y las pilas en Python, por ende, es necesario tener nociones o práctica en el uso de éstos.

### Git y GitHub

Git es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia, la confiabilidad y compatibilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente.



### Sintaxis

Conjunto de reglas que definen las combinaciones de símbolos que se consideran construcciones correctamente estructuradas.

Ejemplos:

“Por supuesto hija, puedes traer a tu novio a casa”.

Ésta frase es sintácticamente correcta en español, pero no es un programa de Java.

class Foo {

public int j;

public int foo (int k) {

return j+k;

}

}

Ésta frase está correcta en Java, pero no es un programa en C.

A menudo la sintaxis es expresada usando gramáticas libres de contexto, los elementos básicos como palabras se especifican a través de expresiones regulares.

Ejemplo para expresiones algebraicas.

expr ⇀ num

| ‘(‘ expr ‘)’

| expr ‘+’ expr

| expr ‘-‘ expr

| expr ‘\*’ expr

| expr ‘/’ expr

NUM ⇀ [0-9] + ( ‘.’ [0-9]+)

### Semántica

La semántica de un lenguaje de programación describe lo que significa un programa bien construido.

### Diccionario

En Python, un diccionario es una colección no-ordenada de valores que son accedidos a traves de una clave. Es decir, en lugar de acceder a la información mediante el índice numérico, como es el caso de las listas y tuplas, es posible acceder a los valores a través de sus claves, que pueden ser de diversos tipo.

### Pila

Una pila es un TAD que tiene las siguientes operaciones (se describe también la acción que lleva adelante cada operación):

\_\_init\_\_: inicializa una pila nueva, vacía.

apilar: agrega un nuevo elemento a la pila.

desapilar: elimina el tope de la pila y lo devuelve. El elemento que se devuelve es siempre el último que se agreg6.

es\_vacia: devuelve True o False según si la pila está vacía o no.

El comportamiento de una pila se puede describir mediante la frase "Lo último que se apiló es lo primero que se usa", que es exactamente lo que uno hace con una pila (de platos por ejemplo): en una pila de platos uno sólo puede ver la apariencia completa del plato de arriba, y sólo puede tomar el plato de arriba (si se intenta tomar un plato del medio de la pila lo más probable es que alguno de sus vecinos, o él mismo, se arruine).

### Teoría de lenguajes

Para especificar tokens en nuestro compilador usaremos conceptos de la teoría de lenguajes, los principales conceptos son los siguientes:

* Alfabeto: Un conjunto finito de símbolos.
* Palabra: Una secuencia finita de símbolos del alfabeto.
* Lenguaje: Un conjunto de palabras sobre un alfabeto.

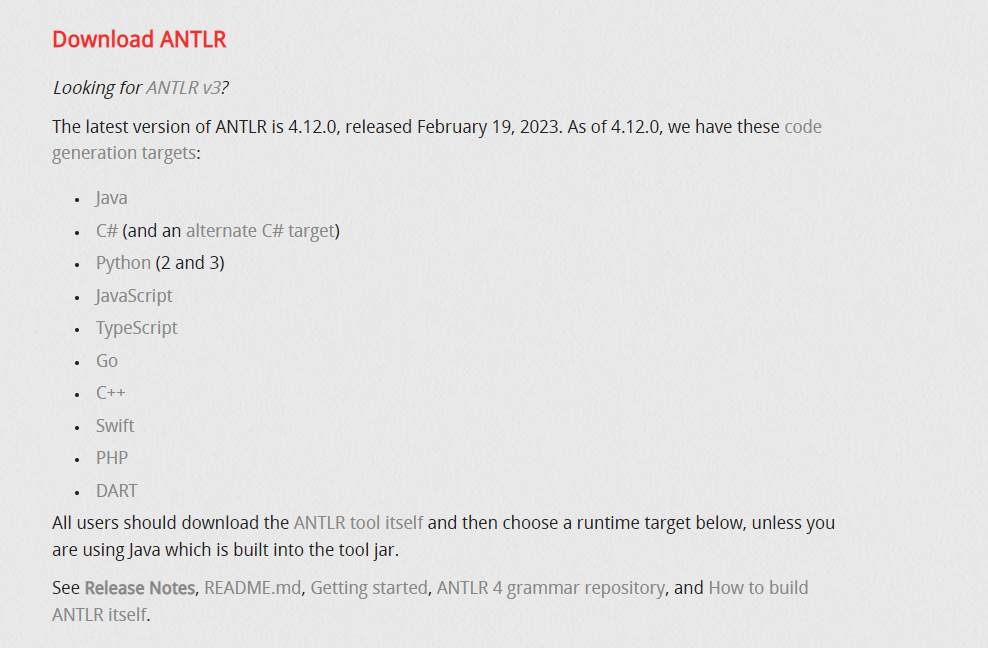
# Elección de herramientas

## Python

Python es un lenguaje de alto nivel de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código, se utiliza para desarrollar aplicaciones de todo tipo, ejemplos: Instagram, Netflix, Spotify, Panda3D, entre otros. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. Se selecciona Python como lenguaje principal de desarrollo como ya se menciona, por su facilidad de lectura, por ser un lenguaje con bastas herramientas, una gran comunidad y bastante información en internet; lo cual enriquece el proceso de aprendizaje y desarrollo de éste proyecto de compilador.

## ANTLR

Dadas las especificaciones léxicas y sintácticas del lenguaje de programación nos ayudará a construir automáticamente el escáner el analizador y el AST. (Nosotros daremos ésta especificaciones).



ANTLR cuenta con varios targets, ese es el listado de targets con los que podemos trabajar, como se puede ver incluye Java, así que **es una excelente alternativa a JavaCC.** Para aquellos que quieran manejar java.

## Git

Esa es una herramienta de control de versiones y es bastante útil para el desarrollo, así como para agilizar y dividir las tareas entre los miembros del equipo por ello será una herramienta base en éste proyecto.

# Fases de desarrollo

## Primera fase (Analizador Léxico)

El analizador léxico (o escáner) convierte una secuencia de caracteres en una secuencia de tokens. Antes de comenzar a desarrollar debemos refrescar la memoria respecto a conceptos básicos de programación.

### Glosario de terminos respecto al análisis léxico

|  |  |  |
| --- | --- | --- |
| Termino | Significado | Ejemplo |
| Delimitadores/puntuación | Un delimitador es un carácter único o una serie de caracteres que indica el principio o el final de una declaración específica, cadena o conjunto de cuerpo de función. Los delimitadores se utilizan para especificar caracteres de conjuntos de códigos o cadenas de datos, sirven como límites de datos y códigos y facilitan la interpretación del código. La puntuación se refiere al carácter en sí, mientras que la delimitación se refiere a la combinación de los caracteres de puntuación para delimitar algo. En ocasiones pueden funcionar también como operadores. | Delimitación:  ( ) [ ] { } “ ”  Puntuación:  , : . ` = ; += -= \*= /= //= %= &= |= ^= >>= <<= \*\*= |
| Identificador | Un identificador es el nombre empleado para identificar una variable, una función, una clase, un módulo u otro objeto. | public class **animales** { }  public void **metodoEjemplo** ();  String **perro** = “Guau guau”;  En donde las palabras en negritas serían nuestros identificadores. |
| Literales |  |  |
| Operadores | Son caracteres especiales designados (como su nombre indica) a realizar operaciones. | Integer suma = 1**+**1;  String maullido **=** “maw miau nya”;  String ladrido **=** “wof, guau, wuff”;  String sonidosAnimal **=** maullido**+**ladrido;  En donde los símbolos de rojo serían operadores. |
| Palabra clave o palabra reservada | En programación, una palabra clave es una palabra que está reservada por un programa, porque la palabra tiene un significado especial. Las palabras clave pueden ser comandos o parámetros. | **String** perro = “Guagu guau”; en donde la palabra de color verde sería nuestra palabra reservada. |

Objetivos de ésta fase de desarrollo

* Simplificar el trabajo del analizador sintáctico
* Descartar detalles irrelevantes como espacios en blanco y comentarios

Notas adicionales

Respecto a las diferencias entre un Scanner y un Parser tenemos que **un escáner simplemente convierte una cadena de entrada (digamos un archivo) en una lista de tokens.** Estos tokens representan cosas como identificadores, paréntesis, operadores, etc, mientras que **un analizador sintáctico convierte esta lista de tokens en un objeto** similar a un árbol para representar cómo encajan los tokens para formar un todo cohesivo (a veces denominado frase).

En el caso de los analizadores sintácticos de lenguajes de programación, el resultado suele denominarse árbol sintáctico abstracto (AST). Cada nodo del AST representa una construcción diferente del lenguaje, por ejemplo, una sentencia IF sería un nodo con 2 o 3 subnodos, un nodo CONDITION, un nodo THEN y potencialmente un nodo ELSE.

A partir de la definición léxica el escáner es un autómata determinista que produce como salida los tokens reconocidos. Su construcción va así: reglas con expresiones regulares > automatas finitos no deterministas > automatas finitos deterministas > tablas.

Nosotros definimos la léxica del lenguaje, las definimos con expresiones regulares.

### Reglas importantes para hacer un analizador Léxico

En éste apartado me enfoqué en algunas reglas para cuando vayamos a escribir nuestros analizador léxico. Para simplificar y facilitar su estudio he listado elementos de documentación básica en una lista y creado una tabla.

* \*Indica cero o más
* | indica unión
* ‘..’ indica rangos
* ‘?’ indica uno o ninguno
* + indica uno o más
* Juntar indica concatenación
* La palabra reservada skip no reporta el token
* Los nombres de todos los tokens deben ser en mayúsculas
* Los textos entre comillas representan ese texto
* Se pueden agrupar los elementos por paréntesis

|  |  |  |
| --- | --- | --- |
| **TOKEN:** | **Símbolo** | **Cierre** |
| RET: | ‘return’ | ; |
| LPAREN: | ‘(‘ | ; |
| RPAREN: | ‘)’ | ; |
| ADD: | ‘+’ | ; |
| SUB: | ‘-‘ | ; |
| MUL: | ‘\*’ | ; |
| DIV: | ‘;’ | ; |
| DIGIT: | ‘0’..’9’ | ; |
| LETTER: | [a-zA-Z] | ; |
| NUMBER: | (DIGIT)+ | ; |
| IDENT: | LETTER (LETTER | DIGIT)\* | ; |
| WS: | [ \t\n]+ -> skip | ; |

## Segunda fase

## Tercera fase

# Prototipo

# Resultados

Anexos

Nombre de archivo, Archivo

# Conclusiones

En conclusión, los compiladores son una herramienta fundamental en el desarrollo de software y desempeñan un papel vital en la creación del software y las tecnologías que utilizamos a diario. A medida que avance la tecnología, los compiladores seguirán evolucionando y mejorando, contribuyendo a que el desarrollo de software sea más rápido, sencillo y eficaz que nunca. A pesar de estos problemas, los compiladores son una parte esencial del proceso de desarrollo de software. Permiten escribir código en un lenguaje de alto nivel más expresivo y fácil de leer que el código máquina. También ayudan a detectar errores en una fase temprana del proceso de desarrollo, lo que puede ahorrar mucho tiempo y esfuerzo a largo plazo.