LENGUAJES Y AUTOMATAS II

PROYECTO: COMPILADOR

# 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?

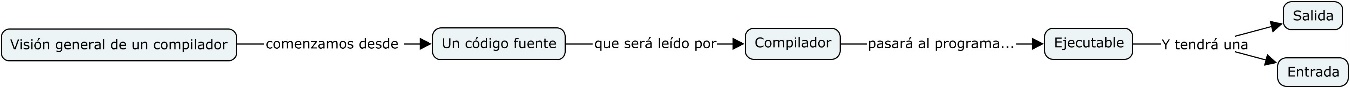
Los compiladores generan código maquina y trabajan tomando el código fuente como entrada y realizando una serie de transformaciones sobre él para generar el código de salida. La primera etapa del proceso suele ser el análisis léxico, que consiste en dividir el código en una serie de tokens. A continuación,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

## Consejos de desarrollo

### Documentación y control de versiones

Para un optimo desarrollo de un proyecto, documentar y llevar un control de versiones nos ayudará a agilizar nuestro desarrollo sobre todo si se trabaja en equipo, así como solucionar errores rápidamente e identificar errores fácilmente.

## 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.



## 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.

### 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).

### 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.

# Elección de herramientas

## Python

## ANTLR

## Git

# Fases de desarrollo

## Primera fase

## Segunda fase

## Tercera fase

# Prototipo

# Resultados

# 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.