

Tecnológico Nacional de México

Instituto Tecnológico de Tepic

Ingeniería en Sistemas Computacionales

Lenguajes y Autómatas I

Tema 6. Máquinas de Turing

Dr. Francisco Ibarra Carlos

Equipo:

20400783. Moreno Duran Juan Diego

21400654. Betancourt Espericueta Jesús Ismael

21400655. Betancourt Espericueta Jorge Ismael

21400673. Enciso Ramirez Daniel Alexis

21400748. Portugal De La Paz Yves Manuel

Tepic; Nayarit a 31 de mayo de 2024

Repositorio GITHUB

https://github.com/DiegoMoreno7/LYA_LAUTISC

Video proyecto final

 **VIDEO LAUSTIC.mp4**

Contenido

1. Preliminares.....	3
1.1 Agradecimientos.....	3
1.2 Resumen.....	3
1.3. Colash de imágenes.....	4
2. Generalidades del proyecto.....	4
2.1. Introducción.....	4
2.2. Descripción de la empresa y área de trabajo.....	5
2.3. Problemas que resolver.....	5
2.4. Objetivos.....	6
2.4.1 General.....	6
2.4.2 Específicos.....	6
2.5 Justificación.....	6
3. Marco teórico.....	7
4. Desarrollo.....	8
4.1. Manual de Usuario.....	8
4.2. Manual técnico.....	18
4.2.1. Requerimientos técnicos.....	18
4.2.2. Requerimientos mínimos de software.....	18
4.2.3. Herramientas utilizadas.....	18
4.2.4. Instalación.....	19
5. Resumen.....	20
Resumen del video.....	20
6. Conclusiones.....	21
7. Competencias desarrolladas.....	22
8. Fuentes de información.....	24
9. Glosario.....	27

1. Preliminares

1.1 Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que han contribuido a la realización de este proyecto de desarrollo de nuestro compilador. Este proyecto ha sido un desafío significativo y ha requerido la colaboración y el apoyo de muchas personas, sin las cuales no habríamos podido lograr nuestros objetivos.

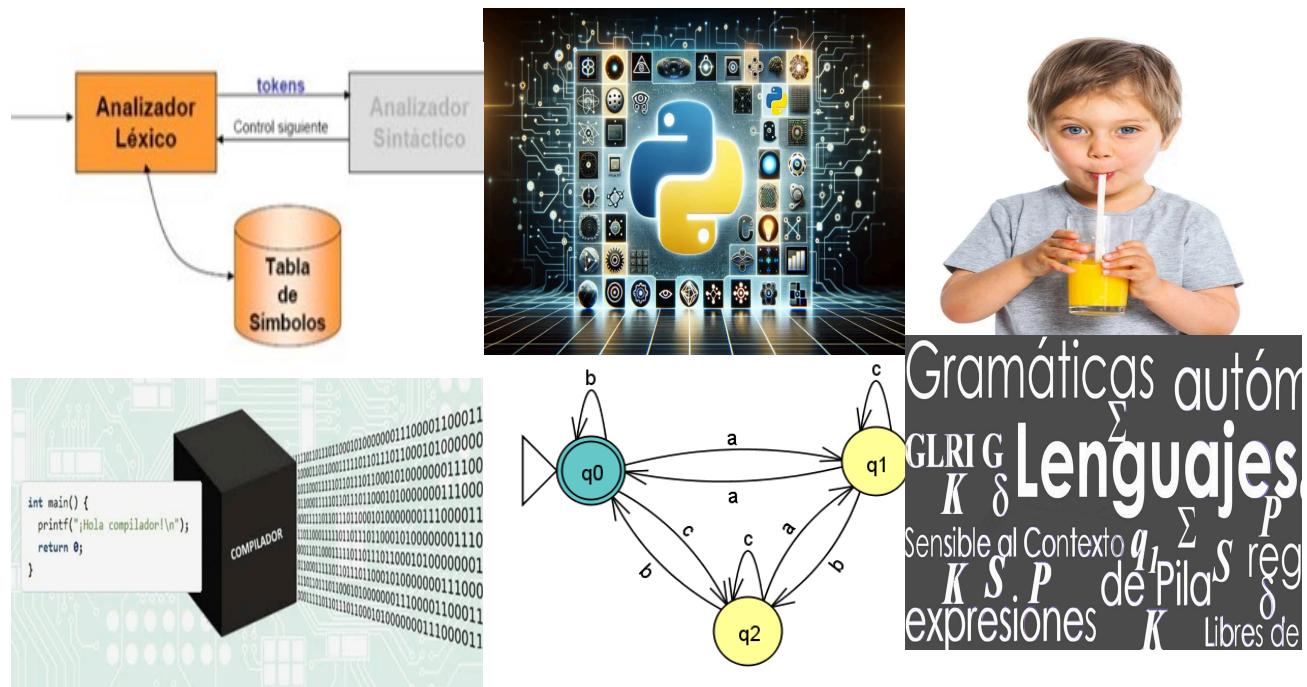
En primer lugar, agradecemos profundamente al profesor Francisco Ibarra Carlos por su inestimable guía y apoyo a lo largo de todo el proceso. Sus conocimientos, paciencia y disposición para ayudar nos han sido de gran ayuda y han sido fundamentales para el éxito de este proyecto. Su entusiasmo por la materia y su capacidad para transmitir conceptos complejos de manera clara y comprensible han sido inspiradores y motivadores para nosotros.

Finalmente, a nuestras familias y amigos, les agradecemos por su paciencia y apoyo incondicional. Su comprensión y aliento nos han permitido dedicar el tiempo y el esfuerzo necesarios para completar este proyecto.

1.2 Resumen

El proyecto se centra en el diseño y desarrollo de un lenguaje de programación llamado “FlavorCode”, específicamente dirigido a mejorar la operación de las máquinas dispensadoras de bebidas. Este lenguaje integra principios de compiladores para ofrecer opciones de personalización al consumidor y mejorar la eficiencia en la distribución de bebidas.

1.3. Colash de imágenes



2. Generalidades del proyecto

2.1. Introducción

El proyecto que presentamos se centra en el diseño y desarrollo de un lenguaje de programación innovador, específicamente creado para mejorar la operación de las máquinas dispensadoras de bebidas. En un mundo donde la personalización y la eficiencia son cada vez más demandadas por los consumidores, nuestro lenguaje de programación ofrece soluciones avanzadas que permiten a las máquinas dispensar bebidas de manera más precisa y personalizada.

Con este proyecto, también buscamos demostrar la aplicabilidad del conocimiento de compiladores en entornos prácticos, más allá de su uso tradicional en la informática. En respuesta a la creciente demanda de opciones

personalizadas y experiencias únicas en la industria de las bebidas, nuestro lenguaje de programación permite una mayor flexibilidad y control en la adición de saborizantes y otros elementos personalizados.

2.2. Descripción de la empresa y área de trabajo

The project is based at the Instituto Tecnológico de Tepic (ITTEPIC) campus, located in Tepic, Nayarit, Mexico. The work area includes the classrooms and rooms within the campus, which are used by both teachers and students for various academic and research activities.

2.3. Problemas que resolver

El problema que abordaremos con este proyecto es la necesidad de mejorar la eficiencia y la personalización en la operación de las máquinas dispensadoras de bebidas. Actualmente, las máquinas tradicionales tienen limitaciones en términos de precisión en la distribución de bebidas y opciones de personalización para los consumidores. Esto puede llevar a experiencias insatisfactorias para los usuarios, así como a desperdicio de ingredientes y recursos.

Nuestro objetivo es desarrollar un lenguaje de programación específicamente diseñado para estas máquinas, que permita una operación más precisa y personalizable. Al integrar principios de compiladores y sistemas de ingeniería, buscamos abordar desafíos como la diversidad de hardware y la seguridad de las máquinas, con el fin de ofrecer una solución integral que

mejore la experiencia del usuario y promueva la eficiencia en la distribución de bebidas

2.4. Objetivos

2.4.1 General

The general objective of this project is to design and implement a beverage dispensing machine, applying principles of compilers and using the agile Scrum methodology.

2.4.2 Específicos

- Crear un sistema de control con lenguajes de programación y autómatas para gestionar la dispensación precisa de bebidas y la personalización de saborizantes.
- Aplicar técnicas de compilación para mejorar el rendimiento, reduciendo tiempos de espera y aumentando la velocidad de dispensación.
- Incorporar mecanismos para agregar saborizantes a las bebidas de manera controlada, asegurando precisión e higiene.

2.5 Justificación

La creación de una máquina dispensadora de bebidas representa una oportunidad significativa en el ámbito de la ingeniería de sistemas y la automatización de procesos. La justificación para este proyecto se fundamenta en varios factores clave.

Existe una creciente demanda por parte de los consumidores de opciones personalizadas y experiencias únicas en la industria de las bebidas. La capacidad de agregar saborizantes a las bebidas ofrece una oportunidad para diferenciarse en un mercado cada vez más competitivo.

La aplicación de principios de compilación en el diseño de la máquina dispensadora no solo mejorará su funcionalidad, sino que también demostrará la

utilidad de los conocimientos en compiladores en aplicaciones del mundo real, más allá de su ámbito tradicional.

Los consumidores se beneficiarán de una mayor variedad y personalización en sus bebidas, lo que mejorará su experiencia de consumo y aumentará su satisfacción.

En conclusión, la creación de una máquina dispensadora de bebidas con sabor no solo responde a las demandas del mercado y las tendencias tecnológicas actuales, sino que también representa una oportunidad para aplicar conocimientos en compiladores de manera innovadora. Los beneficios potenciales de este proyecto son tanto comerciales como tecnológicos, con el potencial de transformar la forma en que interactuamos con las máquinas expendedoras de bebidas en diversos entornos.

3. Marco teórico

Python es reconocido como un lenguaje de programación versátil y ampliamente utilizado en diversas áreas, incluyendo aplicaciones web, desarrollo de software, ciencia de datos y machine learning (ML). Su popularidad radica en su eficiencia, facilidad de aprendizaje y capacidad para ejecutarse en múltiples plataformas, lo que lo convierte en una herramienta fundamental para los desarrolladores en la creación de software.

En el contexto más amplio de la programación, los lenguajes de programación se consideran herramientas esenciales para diseñar y desarrollar programas informáticos que definen y gestionan el comportamiento de dispositivos físicos y lógicos en computadoras. Esto se logra mediante la creación e implementación de algoritmos precisos que actúan como una forma de comunicación entre humanos y computadoras.

La historia de las máquinas expendedoras se remonta a la antigüedad, con registros que datan de aproximadamente 2.500 años atrás en el antiguo Egipto. A lo largo del tiempo, las máquinas expendedoras han evolucionado desde ofrecer agua bendita hasta convertirse en sofisticados dispositivos digitales con pantallas táctiles y sistemas avanzados de reconocimiento facial.

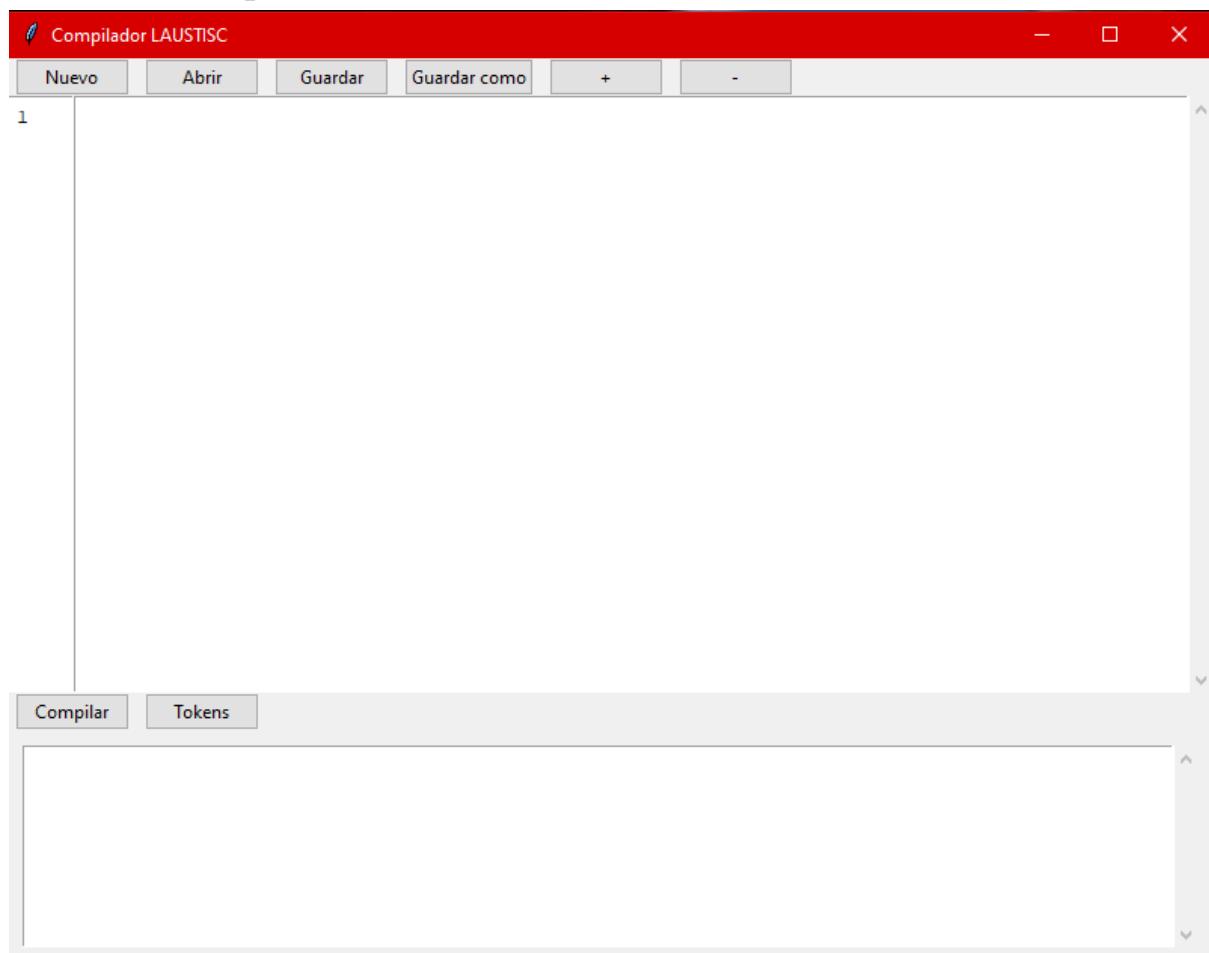
La expansión significativa de las máquinas expendedoras, especialmente en Japón, se produjo durante el siglo XX, con la llegada de máquinas de marcas reconocidas como Coca-Cola. Hoy en día, las máquinas expendedoras de última generación ofrecen una experiencia de usuario más interactiva y personalizada, con características como pantallas táctiles y sistemas de reconocimiento facial para recomendar bebidas según las preferencias del cliente.

4. Desarrollo

4.1. Manual de Usuario

Al iniciar el programa principal, lo primero que nos aparecerá será nuestro IDLE que se muestra a continuación y en breve se explicará más detalladamente cada uno de sus componentes.

Pantalla Principal



Parte superior del IDLE

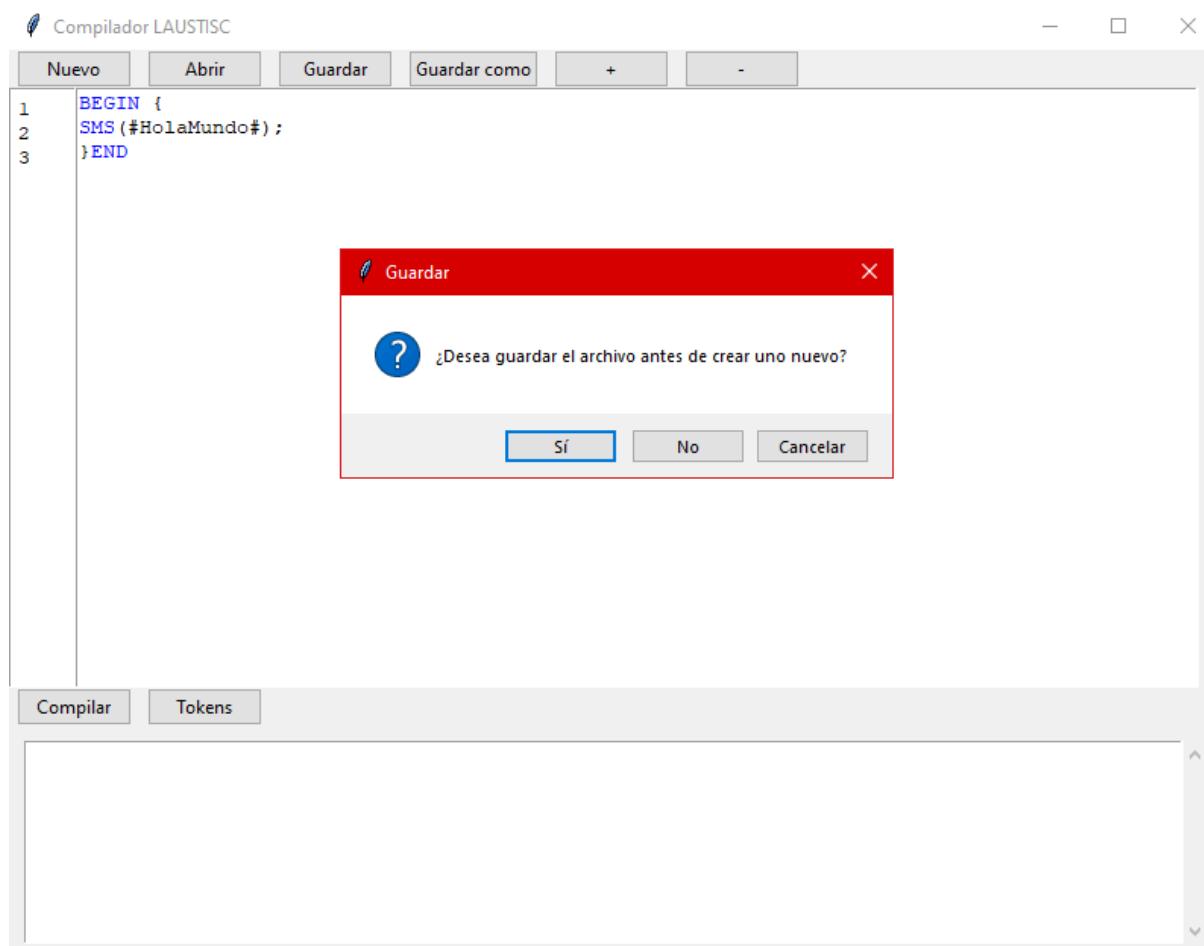


Botón Nuevo

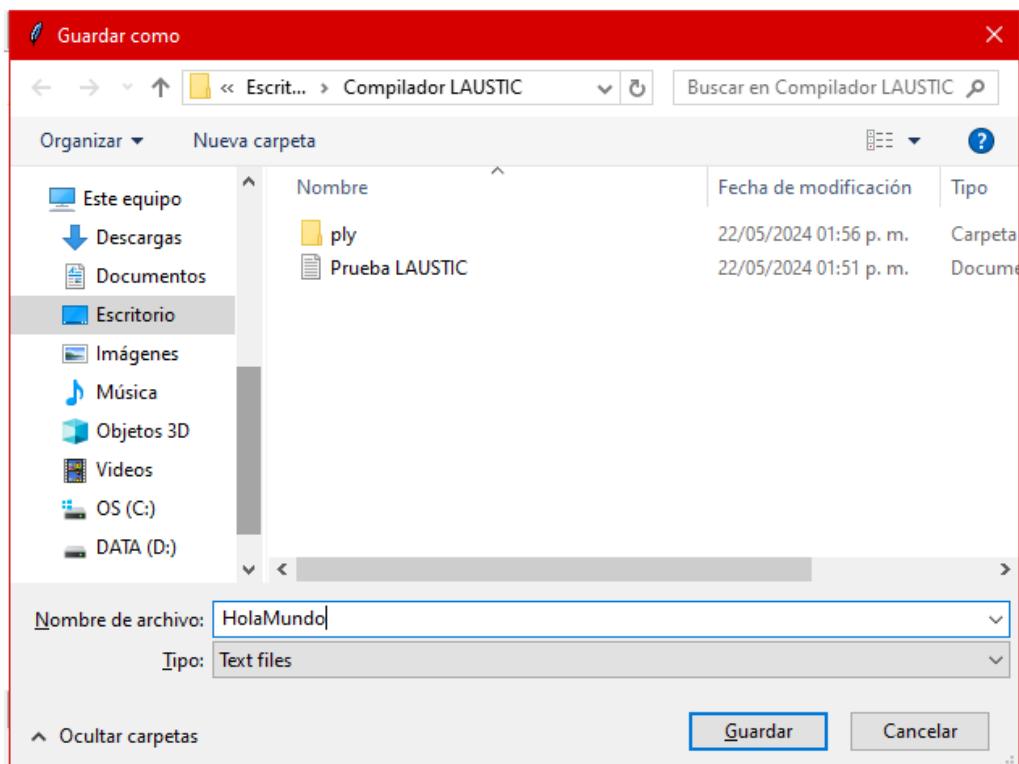
Al presionar el botón de “Nuevo” nos creará un nuevo entorno en blanco en el cual podremos iniciar a programar.



En caso de tener un archivo abierto nos mostrará un cuadro de diálogo en el cual nos advertirá si queremos guardar el archivo actual antes de crear uno nuevo, esto con el propósito de evitar que el usuario pierda su información al presionar el botón accidentalmente



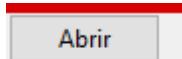
En caso de presionar la opción “Si” y no tenemos ningún archivo abierto, nos abrirá un cuadro en el cual podemos elegir la ruta donde queramos guardarla.



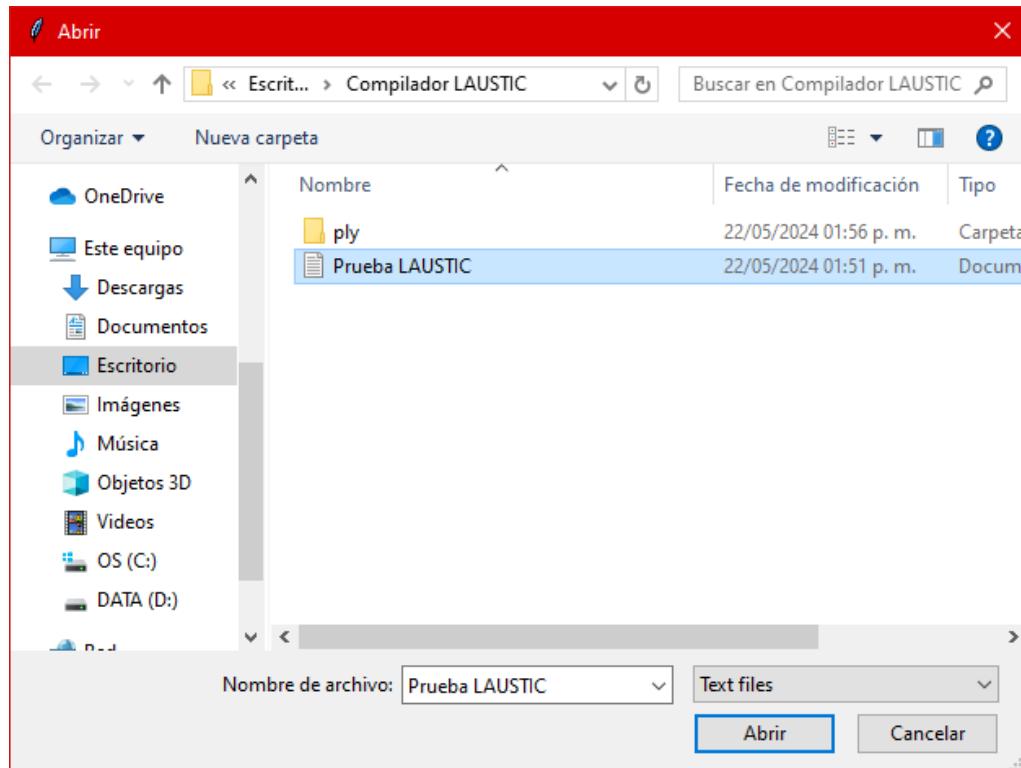
Y en caso contrario de seleccionar la opción de “No” el archivo actual se cerrará sin guardar los cambios.

Botón Abrir

Al presionar el botón de “Abrir”



Nos mostrará un cuadro en el cual podemos elegir el archivo a abrir y nos aparecerá en nuestra ventana de código.



(Archivo abierto correctamente).

The screenshot shows the 'Compilador LAUSTISC' application window. The main area is a code editor with the following pseudocode:

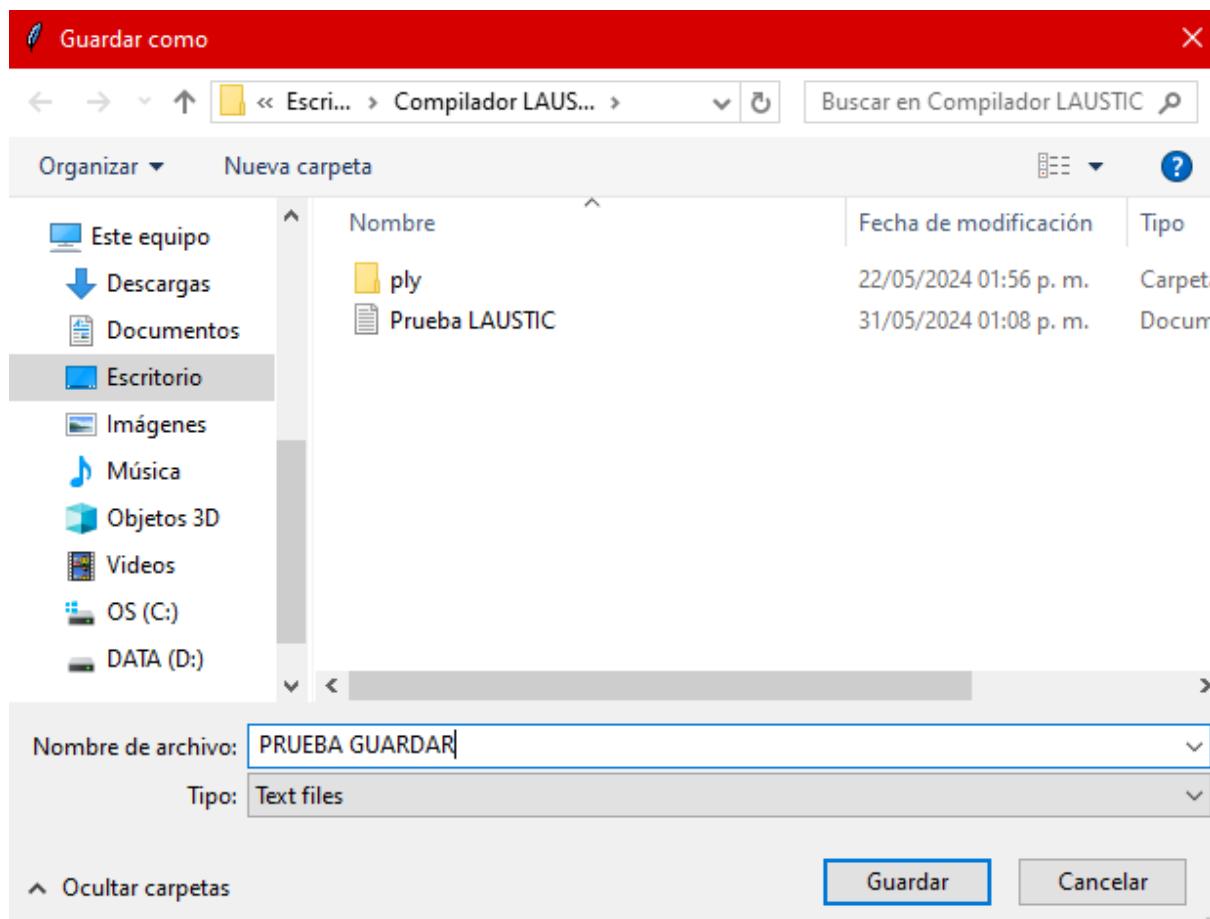
```
1 BEGIN {
2     ingrediente = CA[2];
3
4     // Creación de función para llenar //
5     FUN llenarBebida(int ingrediente) {
6         moveTo(ingrediente[1].compuerta);
7         int cantidadActual = 0;
8         int cantidad=0;
9         WHILE (cantidadActual <= cantidad) {
10             IF (glassPosition(ingrediente[2].compuerta) == True) {
11                 gateOpen(ingrediente[2].compuerta);
12
13             }
14         }
15     }END
16
17
18 }
```

Below the code editor are two buttons: 'Compilar' and 'Tokens'. The window has a standard Windows-style title bar with minimize, maximize, and close buttons.

Botón Guardar



Al presionar la opción de guardar y estamos manejando un archivo nuevo el cual nunca ha sido guardado, nos mandará al siguiente cuadro donde podemos guardarla en la ubicación y nombres deseados.

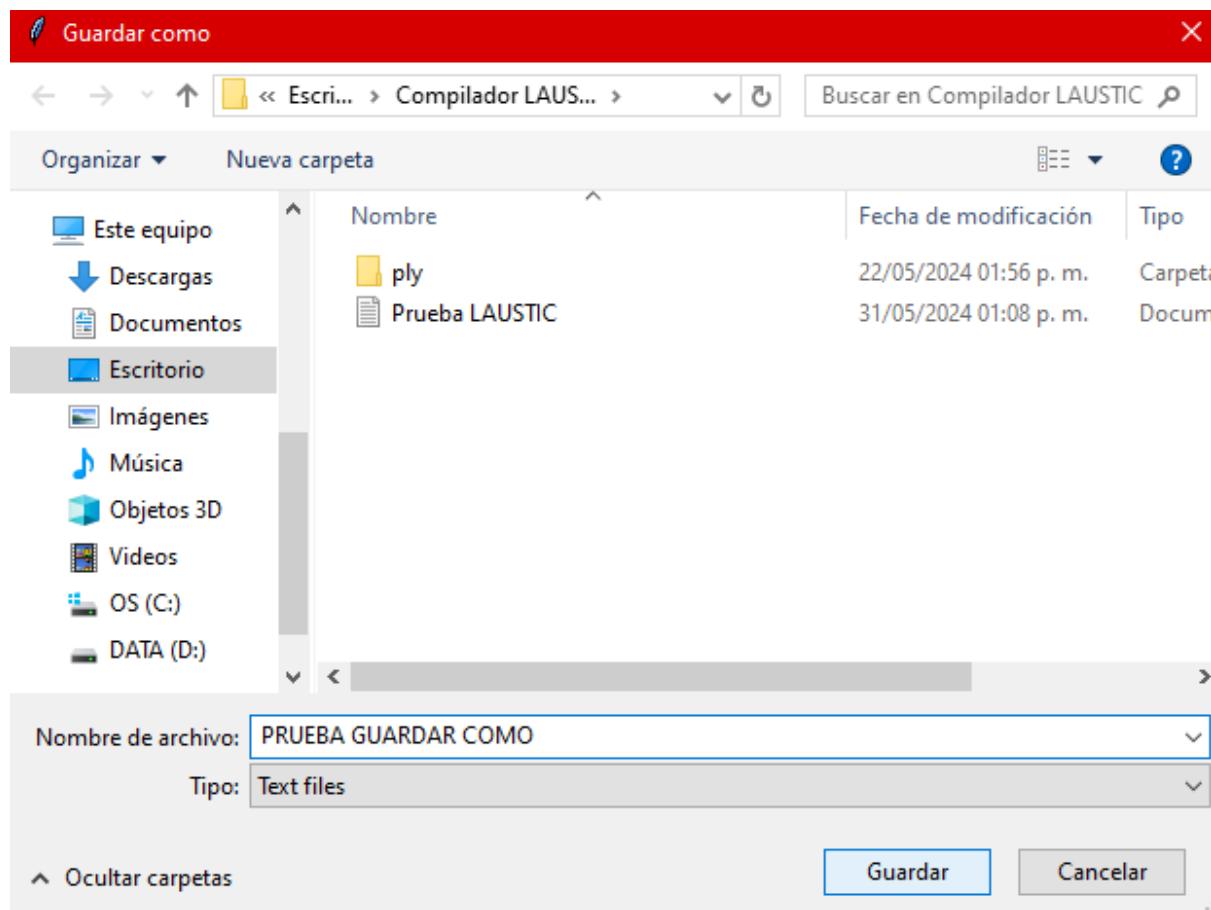


En caso de que estemos manejando un archivo el cual hemos abierto o ya ha sido guardado simplemente se guardará sin mostrar el cuadro anterior.

Botón Guardar como

Guardar como

Al realizar click en el botón “Guardar como” siempre nos mandará al siguiente cuadro donde nos pedirá la ubicación y nombre deseados a guardar, a diferencia del botón “Guardar” que solo la primera vez que manejamos el archivo nos pide ubicación y nombre del archivo a guardar.



Botones para Zoom



En este apartado tenemos los botones relacionados con el acercamiento o alejamiento de las pantallas, al presionar en el botón de “+”. realizaremos un acercamiento general del IDLE .

(Prueba de Acercamiento al presionar “+”)

The screenshot shows the 'Compilador LAUSTISC' window. The menu bar includes 'Nuevo', 'Abrir', 'Guardar', 'Guardar como', a separator, '+', a separator, and '-'. The code editor displays the following pseudocode:

```
1 BEGIN{  
2 //Prueba de Zoom  
3 SMS (#Texto A Ver#)  
4  
5 END}
```

Below the code editor are two buttons: 'Compilar' and 'Tokens'. The window has standard operating system window controls at the top right.

Mientras que al presionar el botón de “-” realizaremos un alejamiento general del IDLE.

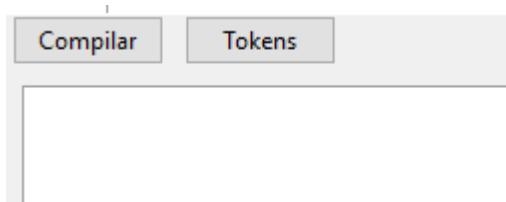
(Prueba de Alejamiento al presionar “-”)

The screenshot shows the 'Compilador LAUSTISC' window. The menu bar includes 'Nuevo', 'Abrir', 'Guardar', 'Guardar como', a separator, '+', a separator, and '-'. The code editor displays the same pseudocode as the previous screenshot:

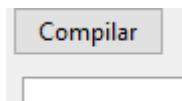
```
1 BEGIN{  
2 //Prueba de Zoom  
3 SMS (#Texto A Ver#)  
4  
5 END}
```

Below the code editor are two buttons: 'Compilar' and 'Tokens'. The window has standard operating system window controls at the top right. A red vertical line is drawn on the right side of the window, indicating the zoom level.

Parte Inferior del IDLE



Botón de Compilar



Dentro de la parte inferior del IDLE tenemos el botón de compilar, el cual es el encargado de analizar el código fuente para su respectivo análisis, en caso de detectar errores los mostrará en la sección de mensajes de consola.

(Salida en caso de presentar errores)

The screenshot shows the 'Compilador LAUSTISC' window. The code area contains the following pseudocode:

```
1 BEGIN {
2     ingrediente = CA[2];
3
4     // Creación de función para llenar //
5     FUN llenarBebida(int ingrediente) {
6         moveTo(ingrediente[1].compuerta);
7         int cantidadActual = 0;
8         int cantidad=0;
9         int datos=0;
10        WHILE (cantidadActual <= cantidad) {
11            IF (glassPosition(ingrediente[2].compuerta) == True) {
12                gateOpen(ingrediente[2].compuerta);
13            }
14        }
15    }
```

The status bar at the bottom shows:

- 'Compilar' button (highlighted)
- 'Tokens' button
- An error message: "Error sintáctico en la linea 9: Falta punto y coma cerca de '0'"
- "Se espera: tipo ID Asignacion expresion PUNTOCOMA" with a cursor pointing to the character '0'
- Suggestion: "Pruebe con: int datos = 0 ;"

Botón Tokens

Una vez que se haya compilado el programa, al presionar el botón de tokens nos aparecerá una ventana donde nos muestra la información correspondiente a dichas tokens generados durante el análisis con su respectiva localización de línea y columna.

Ventana Secundaria		-	□	×
Lexema	Token	Línea	Columna	
BEGIN	BEGIN	1	1	
{	LLAVE_A	1	7	
ingrediente	ID	2	3	
=	ASIGNACION	2	15	
CA	CA	2	17	
[CORCHETE_A	2	19	
2	NUMERO	2	20	
]	CORCHETE_B	2	21	
;	PUNTOCOMA	2	22	
FUN	FUN	5	3	
llenarBebida	ID	5	7	
(PARENTESIS_A	5	19	
int	int	5	20	
ingrediente	ID	5	24	
)	PARENTESIS_B	5	35	
{	LLAVE_A	5	37	
moveTo	moveTo	6	5	
(PARENTESIS_A	6	11	

En caso de no haber compilado el programa la ventana aparecerá vacía.

Ventana Secundaria		-	□	×
Lexema	Token	Línea	Columna	

4.2. Manual técnico

4.2.1. Requerimientos técnicos

- Procesador: Intel Core i3 o equivalente AMD.
- Memoria RAM: 4 GB.
- Espacio en disco duro: 2 GB disponibles.
- Tarjeta gráfica: Tarjeta gráfica integrada compatible con DirectX 10.
- Conexión a Internet: Se requiere para descargar e instalar actualizaciones y recursos adicionales.
- Puertos USB: Al menos 1 puerto USB disponible para conectar dispositivos periféricos como teclado y ratón.

4.2.2. Requerimientos mínimos de software

- Sistema operativo: Windows 10 o superior.
- Privilegios de administrador.

4.2.3. Herramientas utilizadas

- **Python:** Python es un lenguaje de programación de alto nivel, interpretado y multipropósito. Es ampliamente utilizado en el desarrollo de una variedad de aplicaciones debido a su sintaxis limpia y legible, así como a su amplia gama de bibliotecas y frameworks disponibles.
- **Visual Studio Code:** Visual Studio Code es un entorno de desarrollo integrado (IDE) ligero y altamente personalizable desarrollado por Microsoft. Destaca por su interfaz intuitiva, soporte para varios lenguajes de programación y extensiones, y funcionalidad de depuración y edición de código. Ofrece integración con herramientas de control de versiones como Git para facilitar el trabajo colaborativo en proyectos de software.

4.2.4. Instalación

Para instalar el lenguaje "Flavor Code" en un entorno de desarrollo, se deben seguir los siguientes pasos:

1. Descargar e instalar VisualStudio Code, preferentemente la versión más reciente.
2. Descargar el proyecto desde el repositorio de github.
3. Una vez instalado el proyecto lo descomprimimos del zip en que fue descargado y lo movemos a un directorio deseado (de preferencia a la carpeta de documentos).
4. Abrimos VisualStducio Code y damos click en: File > Open Folder
5. Si Python no está instalado en Visual Studio Code, siga estos pasos adicionales:
 - 5.1. Abra Visual Studio Code y vaya al menú "Extensions" en la barra lateral izquierda (ícono de cuatro cuadrados pequeños).
 - 5.2. En el campo de búsqueda, escriba "Python".
 - 5.3. De los resultados de búsqueda, seleccione "Python" desarrollado por Microsoft y haga clic en "Install" para instalar la extensión.
 - 5.4. Una vez instalada la extensión, es posible que deba reiniciar Visual Studio Code para que los cambios surtan efecto.
 - 5.5. Después de reiniciar, la extensión Python debería estar lista para usar. Puede verificarlo abriendo un archivo de código Python (.py) y verificando si hay sugerencias y características específicas de Python disponibles, como resaltado de sintaxis y autocompletado.
6. Una vez completado todos los pasos anteriores ya podremos crear el ejecutable de nuestro proyecto
 - Instala pyinstaller si aún no lo has hecho. Puedes instalarlo usando pip.
 - Navega hasta el directorio donde se encuentra tu script de Python

- Ejecuta y pasa el nombre de tu script como argumento.
- pyinstaller creará una carpeta dist en el directorio actual, que contendrá el ejecutable de tu script de Python junto con cualquier archivo adicional necesario.

5. Resumen

Resumen del video

En el video, comenzamos explicando la metodología que se utilizó para trabajar en nuestro compilador, llamada metodología scrum, que se caracteriza por permitir a los equipos trabajar de una manera eficiente y agilizada, se divide en sprints donde se enfocan en diferentes partes de lo que se requiere realizar, mencionamos cuantos sprints realizamos a lo largo del proyecto y las historias de usuario. Se mencionaron 6 sprints en total, cada uno enfocado a una parte importante del compilador, después, se prosiguió a presentar el diseño del compilador, donde se explicó que el objetivo era crear una interfaz amigable con el usuario y un lenguaje que sea sencillo y fácil de aprender para los clientes, utilizando palabras reservadas y estructuras de control para su mejor visualización, y al mismo tiempo que nos permita manejar diferentes tipos de archivos para su manejo, el manejador presenta una interfaz limpia, con un campo central donde va el texto, diferentes botones de funciones como agregar archivos, nuevo, aumentar tamaño de letra y disminuir y el compilar (también consta de un botón para visualizar los token) en la parte de abajo consta de un recuadro donde nos dirá si tenemos algún error a la hora de escribir el lenguaje.

Después se habló del analizador léxico, que es el analizador léxico, que es el primer paso del compilador de un lenguaje, donde el código fuente se transforma en tokens y palabras claves que se utilizarán, explicamos la sintaxis

a utilizar el lenguaje. Posteriormente se habló sobre el analizador sintáctico, que es el segundo paso del compilador, donde aquí se analizan una secuencia de tokens para determinar su estructura gramatical, o sea, posteriormente se habló del último paso de este, El analizador Semántico, que su función principal es encargarse de que lo que se escribió anteriormente, tenga un sentido lógico dentro del contexto. Luego presentamos un ejemplo de cómo funcionaba nuestro código para crear una bebida, explicamos las funciones que se crean y cual es su fin, luego el profesor trató de crear errores léxicos, sintácticos y semánticos editando el código en diferentes partes para ver si podía leer dichos errores, el cual fue un éxito, después intentaron hacer crashear el código pero por más que lo intentaron, no pudieron romper el código, también invitó a compañeros que lo ayudaran y tampoco pudieron, al final recibimos comentarios de mejora del profesor y nos comentó que nos faltaba agregar algunos pasos primero antes de lo que presentamos, como por ejemplo: código para encender la máquina antes de servir la bebida, pero en retrospectiva, nos comentó que el compilador está listo para seguir con la siguiente fase de lenguajes y autómatas 2.

6. Conclusiones

Después de un arduo trabajo y dedicación, hemos llegado al final de este proyecto de desarrollo de nuestro compilador "FlavorCode". A lo largo de este proceso, hemos enfrentado desafíos y superado obstáculos, pero también hemos aprendido y crecido como equipo y como individuos.

En primer lugar, hemos logrado diseñar y desarrollar un lenguaje de programación innovador que integra principios de compiladores para mejorar la operación de las máquinas dispensadoras de bebidas "FlavorCode".

Además, durante el desarrollo de este proyecto, hemos tenido la oportunidad de familiarizarnos con un nuevo lenguaje de programación, Python. Este lenguaje versátil y potente ha sido fundamental en la implementación de diversas funcionalidades de nuestro compilador, ampliando así nuestro conjunto de habilidades y conocimientos técnicos.

Hemos demostrado la aplicabilidad del conocimiento de compiladores en entornos prácticos, más allá de su uso tradicional en informática. Este proyecto no solo ha ampliado nuestros horizontes académicos, sino que también ha sentado las bases para futuras investigaciones y desarrollos en este campo.

En última instancia, queremos agradecer a todas las personas que han contribuido a este proyecto, desde nuestros compañeros de equipo hasta nuestro asesor, el Dr. Francisco Ibarra Carlos, cuyo apoyo y orientación han sido fundamentales para nuestro éxito. También queremos agradecer a nuestras familias y amigos por su apoyo inquebrantable a lo largo de este viaje.

En conclusión, estamos orgullosos del trabajo que hemos realizado. Este proyecto representa el resultado de nuestro esfuerzo colectivo y nuestro compromiso con la excelencia en la ingeniería de sistemas y la innovación tecnológica.

7. Competencias desarrolladas

1 Introducción a la Teoría de Lenguajes Formales

- 1.1 Alfabeto.
- 1.2 Cadenas.
- 1.3 Lenguajes, tipos y herramientas.
- 1.4 Estructura de un traductor

1.5 Fases de un compilador

2 Expresiones Regulares.

2.1. Definición formal de una ER.

2.2. Diseño de ER.

2.3. Aplicaciones en problemas reales.

3 Autómatas Finitos.

3.1 Conceptos: Definición y Clasificación de Autómata Finito (AF).

3.2 Conversión de un Autómata Finito No Determinista (AFND) a Autómata Finito Determinista (AFD).

3.3 Representación de ER usando AFND

3.4 Minimización de estados en un AF

3.5 Aplicaciones (definición de un caso de estudio).

4 Análisis Léxico.

4.1 Funciones del analizador léxico.

4.2 Componentes léxicos, patrones y lexemas.

4.3 Creación de Tabla de tokens.

4.4 Errores léxicos.

4.5 Generadores de analizadores Léxicos.

4.6 Aplicaciones (Caso de estudio).

5 análisis Sintáctico.

5.1 Definición y clasificación de gramáticas.

5.2 Gramáticas Libres de Contexto (GLC).

5.3 Árboles de derivación.

5.4 Formas normales de Chomsky.

5.5 Diagramas de sintaxis

5.6 Eliminación de la ambigüedad.

5.7 Tipos de analizadores sintáticos

5.8 Generación de matriz predictiva (cálculo first y follow)

5.9 Manejo de errores

5.10 Generadores de analizadores sintácticos

6 Máquinas de Turing.

6.1 Definición formal MT

6.2 Construcción modular de una MT

6.3 Lenguajes aceptados por la MT.

8. Fuentes de información

aws. 2023. ¿Qué es Python?. <https://aws.amazon.com/es/what-is/python/>

UNAM. 2017. Lenguajes de Programación.

https://repositorio-uapa.cuaied.unam.mx/repositorio/moodle/pluginfile.php/265/mod_resource/content/1/UAPA-Lenguajes-Programacion/index.html

nippon. 28 de junio de 2015. Máquinas expendedoras.

<https://www.nippon.com/es/features/jg00065/>

UAA - Sistemas electrónicos. SF. Compiladores [Archivo PDF].

<https://hopelchen.tecnm.mx/principal/syllabus/fpdb/recursos/r135301.PDF>

INAOE. SF. Autómatas Finitos [Archivo PDF].

<https://ccc.inaoep.mx/~emorales/Cursos/Automatas/Finitos.pdf>

IBM. 14 de abril de 2021. Expresiones regulares.

<https://www.ibm.com/docs/es/i/7.3?topic=expressions-regular>

Pontificia Universidad Católica de Perú. SF. IDENTIFICADORES.

[https://agora.pucp.edu.pe/inf2170681/2.htm#:~:text=Un%20identificador%20es,%20un%20nombre,que%20se%20tratar%C3%A1n%20m%C3%A1s%20adelante\).](https://agora.pucp.edu.pe/inf2170681/2.htm#:~:text=Un%20identificador%20es,%20un%20nombre,que%20se%20tratar%C3%A1n%20m%C3%A1s%20adelante).)

IBM. 28 de febrero de 2021. Código fuente y código objeto.

<https://www.ibm.com/docs/es/iis/11.5?topic=language-source-code-object-code>

onmex. 2023. ¿Qué son las palabras reservadas en Programación?.

<https://onmex.mx/business/que-son-las-palabras-reservadas-en-programacion/>

Definicion.DE. 14 de julio de 2022. TEOREMA. <https://definicion.de/teorema/>

UTN - Universidad Tecnológica Nacional Facultad Regional Rosario. SF.
Capítulo 2: Lenguajes.

https://www.fro.utn.edu.ar/repositorio/catedras/sistemas/2_anio/sintaxis/SSyL-cap2_2015_Lenguajes.pdf

GCFGlocal. SF. Operadores Lógicos o Booleanos.

<https://edu.gcfglobal.org/es/conceptos-basicos-de-programacion/operadores-logicos-o-booleanos/1/>

Glosario gráfico. 2023. Carácter. <http://www.glosariografico.com/caracter>

INAOE. SF.. Teoría de autómatas y lenguajes formales expresiones regulares y lenguajes.

https://posgrados.inaoep.mx/archivos/PosCsComputacionales/Curso_Propedeutico/Automatas/03_Automatas_ExpresionesRegularesLenguajes/CAPTUL1.PDF

FinePROXY. 21 de mayo de 2023. Código objeto..

<https://fineproxy.org/es/wiki/object-code/>

matesFacil. SF. Autómata Finito Determinista

<https://www.matesfacil.com/automatas-lenguajes/automata-finito-y-su-lenguaje.html>

CIDECAME .SF. 2.1. Función del Analizador Léxico.

http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro32/21_funcin_del_analizador_lxico.html

Del automata finito, 1. Una Vision Informal. (s/f). Propedéutico: Autómatas. Inaoep.mx., de

https://posgrados.inaoep.mx/archivos/PosCsComputacionales/Curso_Propedeutico/Automatas/02_Automatas_AutomatasFinitos/Capitulo_2_Automatas_finitos_Curso_Anterior.pdf

¿Qué es una cadena de caracteres? (s/f). Unam.mx. Recuperado de
https://arquimedes.matem.unam.mx/mati/actividades/info/info_que_es_una_cadena_de_caracteres/

Westreicher, G. (2021, marzo 2). Teoría de conjuntos. Economipedia.
<https://economipedia.com/definiciones/teoria-de-conjuntos.html>

Wikipedia contributors. (s/f). Gramática (autómata). Wikipedia, The Free Encyclopedia.

[https://es.wikipedia.org/w/index.php?title=Gram%C3%A1tica_\(aut%C3%B3mata\)&oldid=151483489](https://es.wikipedia.org/w/index.php?title=Gram%C3%A1tica_(aut%C3%B3mata)&oldid=151483489)

Wikipedia contributors. (s/f). Lenguaje regular. Wikipedia, The Free Encyclopedia.

https://es.wikipedia.org/w/index.php?title=Lenguaje_regular&oldid=150758673

KathleenDollard. (2023, 7 abril). *Tipo de datos - Visual Basic*. Microsoft Learn.
<https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/data-types/>

Patrones de diseño / Design patterns. (s. f.).

[https://refactoring.guru/es/design-patterns#:~:text=Los%20patrones%20de%20dise%C3%B1o%20\(design,dise%C3%B1o%20particular%20de%20tu%20c%C3%B3digo.](https://refactoring.guru/es/design-patterns#:~:text=Los%20patrones%20de%20dise%C3%B1o%20(design,dise%C3%B1o%20particular%20de%20tu%20c%C3%B3digo.)

Porto, J. P., & Merino, M. (2023, 19 enero). *Lexema - Qué es, definición, ejemplos y tipos*. Definición.de. <https://definicion.de/lexema/>

KathleenDollard. (2023b, mayo 10). *Operadores aritméticos - Visual Basic*. Microsoft Learn.

<https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/language-features/operators-and-expressions/arithmetic-operators>

9. Glosario

Componente léxico. Es la secuencia lógica y coherente de caracteres relativo a una categoría: identificador, palabra reservada, literales (cadena/numérica), operador o carácter de puntuación, además de que un componente léxico puede tener uno o varios lexemas.

Diagrama de transición. El diagrama de transición es una representación gráfica de un autómata finito donde cada estado se muestra como un nodo, las transiciones entre estados se muestran como arcos etiquetados con símbolos, el estado inicial se etiqueta con una marca especial, y los estados finales se destacan con un doble círculo. Es una herramienta visual que simplifica la comprensión de los estados, las transiciones y las condiciones de un autómata finito

Expresión regular. Es una secuencia de caracteres que actúa como modelo para la coincidencia y manipulación de series.

Identificador. Un identificador es un nombre, que define el programador, que sirve para denotar ciertos elementos de un programa. Estos elementos pueden ser las denominadas variables, constantes y funciones.

Código fuente. El código fuente es el formato de entrada original de la rutina escrita por el programador.

Palabra reservada. Es un término que fue, como su nombre lo indica, reservado por el lenguaje de programación para realizar funciones específicas. Estas palabras tienen un significado predefinido y no pueden ser utilizadas

como identificadores, como nombres de variables o funciones, ya que podrían interferir con la lógica del programa.

Teorema. Consiste en una proposición que puede ser demostrada de manera lógica a partir de un axioma o de otros teoremas que fueron demostrados con anticipación. Este proceso de demostración se lleva a cabo mediante ciertas reglas de inferencia.

Símbolo. Es una entidad indivisible, que no se va a definir.

Operador Booleano. Se utilizan para conectar conceptos entre sí, con el fin de comprobar su veracidad dentro de un enunciado.

Token. Son las palabras reservadas de un lenguaje, secuencia de caracteres que representa una unidad de información en el programa fuente.

Carácter. Caracteres son las letras, cifras, signos de puntuación e ideogramas en aquellos sistemas de escritura que sean ideográficos

Cerradura de Kleene. La cerradura de Kleene de un lenguaje L se denota como L^* , representa el conjunto de cadenas que puede que pueden formarse tomando cualquier número de cadenas de L, posiblemente con repeticiones y concatenando todas ellas.

Código Objeto. El código objeto es un conjunto de instrucciones que puede ser ejecutado directamente por el procesador sin ninguna traducción posterior, o puede ser la forma intermedia de las instrucciones del ordenador, que necesita ser procesada posteriormente antes de ser ejecutada por el procesador.

AFN. Un Autómata está formado por un conjunto de estados, uno de los cuales es el estado en el que la máquina se encuentra inicialmente. Recibe como entrada una palabra (una concatenación de símbolos del alfabeto del autómata) y según esta palabra la máquina puede cambiar de estados.

Alfabeto.

Analizador léxico: El analizador léxico es la primera fase de un compilador. Su principal función consiste en leer los caracteres de entrada y elaborar como salida una secuencia de componentes léxicos que utiliza el analizador sintáctico para hacer el análisis.

Autómata Finito Determinista: El término determinista se refiere al hecho de que para cada entrada hay uno y solo un estado al que el autómata puede cambiar. Por el contrario, un autómata no determinista, puede estar en varios estados al mismo tiempo.

Cadena. En computación, una cadena de caracteres o cadena de texto o simplemente cadena (string en inglés) es una secuencia ordenada de símbolos, con una longitud arbitraria (con tantos símbolos como queramos).

Conjunto. Es una agrupación de elementos, ya sean números, letras, palabras, funciones, símbolos, figuras geométricas u otros.

Gramática Equivalente. Una gramática ("G") desde el punto de vista de la teoría de autómatas es un conjunto finito de reglas que describen toda la secuencia de símbolos pertenecientes a un lenguaje específico L. Dos gramáticas que describen el mismo lenguaje se llaman gramáticas equivalentes.

Lenguaje. Un lenguaje regular es un lenguaje formal que puede ser definido por una expresión regular, generado por una gramática regular, y reconocido por un autómata finito.

Lexema.

Se llama lexema a la unidad mínima que dispone de significado léxico pese a no contar con morfemas gramaticales o prescindiendo de ellos mediante un mecanismo de segmentación, tomemos el caso de zapato y de palabras vinculadas como zapatería, zapatero o zapatazo. El lexema, en este caso, es zapat- (zapat-o, zapat-ería, zapat-ero, zapat-azo).

Operador aritmético.

Los operadores aritméticos se usan para realizar muchas de las operaciones aritméticas conocidas que implican el cálculo de valores numéricos representados por literales, variables, otras expresiones, llamadas a funciones y propiedades y constantes. También clasificados con operadores aritméticos son los operadores de desplazamiento de bits, que actúan en el nivel de los bits individuales de los operandos y desplazan sus patrones de bits a la izquierda o derecha.

Patrón.

Los patrones de diseño (design patterns) son elementos reutilizables creados para resolver problemas comunes. Es decir que con su aplicación y utilización podremos corregir diferentes problemas que presenta nuestro código de una manera segura, estable y testeada por cientos de programadores de todo el mundo.

Tipo de dato.

El tipo de datos de un elemento de programación hace referencia al tipo de datos que puede contener y cómo almacena los datos. Los tipos de datos se aplican a todos los valores que se pueden almacenar en la memoria del equipo o participar en la evaluación de una expresión. Cada variable, literal, constante, enumeración, propiedad, parámetro de procedimiento, argumento de procedimiento y valor devuelto de un procedimiento tiene un tipo de datos.

Lema de Arden.

El Lema de Arden, en lenguajes formales, indica una solución particular a la ecuación con expresiones regulares: $x = rx + s$ (en donde r y s son expresiones regulares conocidas, y x es desconocida). Esta solución provee un Algoritmo sistemático y metódico para la conversión de Autómata finito a Expresión regular.