

[Handwritten signatures]

INSTITUTO TECNOLÓGICO DE CELAYA

Lenguajes y Autómatas II

Equipo 2

Gomez Cabañas Anthony 20030057

Ortega Hernández Cristhian Alberto 20031091

Ponce Morales Alma Gabriela 20030274

Ruiz López Miguel Ángel 20030935

ACTIVIDAD 6:

MONOGRAFÍA ANÁLISIS SEMÁNTICO

I.S.C. Ricardo González González



DEPARTAMENTO DE SISTEMAS COMPUTACIONALES E INFORMÁTICA

ASUNTO: **SOLICITUD DE ACTIVIDADES**

Celaya, Guanajuato, 16 / octubre / 2023

LENGUAJES Y AUTÓMATAS II

DOCENTE DESIGNADO: ISC. RICARDO GONZÁLEZ GONZÁLEZ

SEMESTRE AGOSTO-DICIEMBRE 2023

ACTIVIDAD 6 (VALOR 44 PUNTOS)

LEA CUIDADOSAMENTE, Y REALICE LAS SIGUIENTE ACTIVIDADES, CONSIDERANDO LOS CRITERIOS DE CALIDAD PROPUESTOS EN LOS DOCUMENTOS DE LA [GUÍA TUTORIAL](#), Y LA [RÚBRICA DE EVALUACIÓN](#),

EL LECTOR DEBE TOMAR MUY EN CUENTA QUE ESTA ACTIVIDAD ES UN EXAMEN, Y NO UNA SIMPLE TAREA, PUES DEMANDA DEDICACIÓN PARA INVESTIGAR, LEER, ANALIZAR, REDACTAR, ILUSTRAR Y PROPOSER DE MANERA PROFESIONAL LOS TEMAS PROPUESTOS EN LA ESTRUCTURA TEMÁTICA DE ESTA ASIGNATURA.

4. ANÁLISIS SEMÁNTICO.

INVESTIGUE, LEA, COMPREnda Y ELABORE UNA **MONOGRAFÍA TÉCNICA** COMPLETAMENTE APEGADA A LO SOLICITADO EN LA [GUÍA TUTORIAL](#) (PUNTO 3, INCISO a) ACERCA DE LOS SIGUIENTES TEMAS :

- TEMA 4.1 ÁRBOLES DE EXPRESIONES.
- TEMA 4.2 ACCIONES SEMÁNTICAS DE UN ANALIZADOR SINTÁCTICO.
- TEMA 4.3 COMPROBACIONES DE TIPOS EN EXPRESIONES.
- TEMA 4.4 PILA SEMÁNTICA EN UN ANALIZADOR SINTÁCTICO.
- TEMA 4.5 ESQUEMA DE TRADUCCIÓN.
- TEMA 4.6 GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DE DIRECCIONES.
- TEMA 4.7 MANEJO DE ERRORES SEMÁNTICOS.

CONSIDERACIÓN :

DEBE USTED ENTENDER EL VALOR QUE TIENE ESTA ACTIVIDAD Y QUE LOS TEMAS ANTES REFERIDOS, PARA NADA DEBEN SER ABORDADOS COMO SIMPLES CONCEPTOS REDACTADOS CON LA LIGEREZA, PUES ESTA ACTIVIDAD ESTÁ CONSIDERADA COMO UN EXAMEN.

ANALICE CADA TEMA, SUS CARACTERÍSTICAS, SU IMPORTANCIA, SUS CONCEPTOS, SUS EJEMPLOS, SUS ILUSTRACIONES, Y LOS TIPOS DE EVIDENCIAS QUE USARÁ PARA DEMOSTRAR QUE USTED HA ADQUIRIDO UN VERDADERO CONOCIMIENTO ACERCA DE ÉSTOS.





A MODO DE PRÁCTICAS REALICE ESTE PUNTO Y ELABORE EJERCICIOS NECESARIOS CON LOS CUÁLES USTED DEMUESTRE

- ELABORE DOS VIDEOS (NO MÁS DE 25 MINUTOS) DISTRIBUIDOS DE LA SIGUIENTE FORMA Y EN LOS QUE EXPONGA SUS CONOCIMIENTOS ADQUIRIDOS. DESPUÉS COLOQUE SUS MATERIALES EN YOUTUBE E INCLUYA LAS LIGAS EN SU EXAMEN.

VIDEO 1 : TEMAS 4.1, 4.2, 4.3, 4.4

VIDEO 2 : TEMAS 4.5, 4.6, 4.7

MUY IMPORTANTE: SI ESTA ACTIVIDAD ES ENTREGADA EN EQUIPO, CADA UNO DE LOS INTEGRANTES DE ÉSTE DEBEN PARTICIPAR EN CADA VIDEO, EXPONIENDO JUNTO A SUS COMPAÑEROS CADA TEMA SOLICITADO.

POR FAVOR NO USE APUNTADORES O MATERIALES DE APOYO TAN SOLO LEER LOS CONCEPTOS. LA IMPORTANCIA Y EL VALOR DE LOS VIDEOS RADICA EN EXPRESAR Y EVALUAR CORRECTAMENTE SU CONOCIMIENTO EN ESTOS TEMAS.

IMPORTANTE: SI LO REQUIERE PUEDE CONSULTAR EL [SIGUIENTE DOCUMENTO](#) PARA ORIENTAR SU TRABAJO EN CONOCER QUÉ ES Y CÓMO HACER UNA MONOGRAFÍA CON EL RIGOR ACADÉMICO REQUERIDO.

POR ÚLTIMO, RECUERDE LEER LA [GUÍA TUTORIAL](#) PARA EL CORRECTO TRATAMIENTO DE ESTE INCISO.

¿QUÉ SE CALIFICARÁ ?

LA RÚBRICA PARA EVALUAR ESTA ACTIVIDAD ESTARÁ INTEGRADA POR LOS SIGUIENTES CRITERIOS.

- LA OPORTUNIDAD.** SI EL TRABAJO FUE ENTREGADO OPORTUNAMENTE.
- LA COMPRENSIÓN.** SE VALORARÁ EL GRADO DE COMPRENSIÓN DEL TEMAS ANALIZADOS.
- LA CALIDAD.** SI LAS EVIDENCIAS ENVIADAS CORRESPONDEN A LA CALIDAD ESPERADA PARA ESTE NIVEL PROFESIONAL QUE SE CURSA.
- LA CAPACIDAD DE SÍNTESIS.** SI LAS EVIDENCIAS ENTREGADAS TIENEN EL NIVEL DE DETALLE Y PROFUNDIDAD REQUERIDA, O EN BIEN SI SE OMITIERON CONCEPTOS CON EL AFÁN DE SIMPLIFICAR Y ENTREGAR UN MATERIAL ACADÉMICA Y TÉCNICAMENTE POBRE.
- LA CREATIVIDAD.** LA MANERA EN QUE SE EXPRESAN LOS CONCEPTOS Y EL TRATAMIENTO QUE SE DA A LA INFORMACIÓN ANALIZADA PARA QUE ÉSTA SEA COMPRESIBLE EN SU ESENCIA.

IMPORTANTE : CUENTA CON EL TIEMPO SUFFICIENTE PARA REALIZAR ESTA ACTIVIDAD Y SUMAR PUNTOS IMPORTANTES A SU CALIFICACIÓN DE ESTA EVALUACIÓN.

IMPORTANTE : TODO EL MATERIAL ESCRITO DEBERÁ SER HECHO A MANO.





CONSIDERACIONES.

CADA UNO DE LOS PUNTOS ANTERIORES DEBE SER DESARROLLADO CON LA PROFUNDIDAD ACORDE A UN NIVEL PROFESIONAL, Y APEGÁNDOSE COMPLETAMENTE A LAS DIRECTRICES DE LA GUÍA TUTORIAL.

NO CONCIBA ESTE TRABAJO, COMO UN SIMPLE RESUMEN O EJERCICIO DE TRANSCRIPCIÓN, PUES EL VALOR INDICADO AL INICIO DE ESTA ACTIVIDAD LE DARÁ A USTED UNA BUENA IDEA DE LO QUE SE ESPERA DE ELLA, EN CUANTO A CALIDAD Y EL APRENDIZAJE OBTENIDO, MISMO QUE SERÁ PUESTO A PRUEBA MEDIANTE UN EXAMEN ESCRITO O BIEN ORAL EN CLASE.

SI DECIDIÓ ELABORAR ESTA ACTIVIDAD EN EQUIPO, CADA INTEGRANTE DE ÉSTE DEBERÁ POSEER EL MISMO NIVEL DE CONOCIMIENTO, PUES TAN SOLO REPARTIR TEMAS ENTRE LOS INTEGRANTES DEL EQUIPO, SUPONDRIÁ UN GRAVE ERROR DE INTERPRETACIÓN A LA INTENCIÓN DIDÁCTICA REAL DE ESTA ACTIVIDAD.

POR ÚLTIMO, ESTA ACTIVIDAD SOLO SE PODRÁ DESARROLLAR EN EQUIPO, SI SE REGISTRÓ EN UNO PREVIAMENTE, UTILIZANDO EL FORMATO ENTREGADO EN LA ACTIVIDAD INICIAL. DE LO CONTRARIO DEBERÁ ELABORAR Y ENTREGAR LA ACTIVIDAD DE FORMA INDIVIDUAL.

LA ENTREGA DE DICHO REGISTRO SE HARÁ VÍA CORREO ELECTRÓNICO ENVIANDO ÉSTE AL PROFESOR DESIGNADO, Y POSTERIORMENTE EN CLASE ENTREGANDO LA HOJA EN FÍSICO.

OBSERVACIONES:

- CADA HOJA QUE ENTREGUE DE SU ACTIVIDAD, DEBERÁ ESTAR FIRMADA AL MARGEN DERECHO, INCLUIDA LA PROPIA SOLICITUD DE LA ACTIVIDAD.
- INTEGRE TODO SU TRABAJO EN UN SOLO ARCHIVO DE TIPO .PDE, Y ASIGNE EL NOMBRE QUE A CONTINUACIÓN SE INDICA.

NO OLVIDE ANEXAR LAS HOJAS DE ESTA ACTIVIDAD Y DE SU TRABAJO DESPUÉS DE SU PORTADA.

- UNA VEZ ELABORADA SU ACTIVIDAD, RECUERDE DIGITALIZARLA Y NOMBRARLA EN BASE A LA NOMENCLATURA QUE SE INDICA MÁS ADELANTE EN ESTE DOCUMENTO.
- SI SUS EVIDENCIAS ENVIADAS POR CORREO, NO CUMPLEN CON LA NOMENCLATURA SOLICITADA, NO SERÁN CONSIDERADAS COMO EVIDENCIAS PARA SU EVALUACIÓN.
- POR ÚLTIMO, POR FAVOR GESTIONE APROPIADAMENTE SU TIEMPO, Y SEA PUNTUAL EN SU ENTREGA Y ASÍ EVITAR PROBLEMAS DE NULIDAD POR EXTEMPORANEIDAD.





LA NOMENCLATURA SOLICITADA PARA ENVIAR SU TRABAJO ES LA SIGUIENTE :

AAAA-MM-
DD_TNM_CELAYA_MATERIA_DOCUMENTO_[EQUIPO]_NOCTROL_APELLIDOS_NOMBRE_SEM.PDF

(NOTA : * TODO DEBE SER ESCRITO USANDO LETRAS MAYÚSCULAS ***)**

DONDE :

TNM_CELAYA	:	INSTITUCIÓN ACADÉMICA
AAAA	:	AÑO
MM	:	MES
DD	:	DÍA
MATERIA	:	LAI _{II} , LI MÁS EL GRUPO (-A , -B, -C)
DOCUMENTO	:	A1-ACTIVIDAD 1, P1-PRACTICA 1, R1-REPORTE 1, T1-TAREA 1, PG1-PROGRAMA, ETC. (CAMBIANDO EL NÚMERO CONSECUTIVO POR EL QUE CORRESPONDA)
[EQUIPO]	:	NÚMERO DEL EQUIPO QUE CORRESPONDA SEGÚN INDICACIÓN DEL PROFESOR. [OPCIONAL]
NOCTROL	:	SU NÚMERO DE CONTROL
APELLIDOS	:	SUS APELLIDOS
NOMBRE	:	SU NOMBRE
SEM	:	EL PERIODO SEMESTRAL EN CURSO: AGO-DIC

EJEMPLO :

SI EL TRABAJO SE SOLICITÓ EN EQUIPO.

2023-10-16_TNM_CELAYA_LAI_{II}-A_A6_EQUIPO_99_9999999_PEREZ_PEREZ_JUAN_AGO-DIC23.PDF

DONDE EL NOMBRE DEBERÁ CORRESPONDER AL JEFE DE EQUIPO QUE HACE LA ENTREGA DEL TRABAJO.

SI EL TRABAJO SE SOLICITÓ INDIVIDUALMENTE.

2023-10-16_TNM_CELAYA_LAI_{II}-A_A6_9999999_PEREZ_PEREZ_JUAN_AGO-DIC23.PDF





FECHA Y HORA DE ENTREGA:

LA INDICADA EN LA PLATAFORMA VIRTUAL.

EN CASO DE QUE EL TRABAJO SE HAYA SOLICITADO EN EQUIPO, EL JEFE DEL MISMO SERÁ EL ÚNICO RESPONSABLE DE ENVIAR LA ACTIVIDAD EN LA PLATAFORMA VIRTUAL.

MUY IMPORTANTE:

1. DESPUÉS DE LA HORA INDICADA EN LA PLATAFORMA VIRTUAL (AÚN CUANDO SOLO SEA UN MINUTO O VARIOS), LA ACTIVIDAD SERÁ CONSIDERADA COMO EXTEMPORÁNEA Y NO CONTARÁ COMO EVIDENCIA PARA SU EVALUACIÓN.

SE LE SUGIERE ENVIAR CON ANTICIPACIÓN SU ACTIVIDAD A FIN DE EVITAR CONFLICTOS POR NO ENTREGAR ÉSTA A TIEMPO.

BAJO NINGÚN PRETEXTO O JUSTIFICACIÓN SE ACEPTARÁN LOS TRABAJOS EXTEMPORÁNEOS, EVITE LA PENA DE RECORDAR A USTED QUE EL VALOR DE LA PUNTUALIDAD ES PARTE IMPORTANTE DE SUS EVIDENCIAS Y ES EL PRIMER PUNTO QUE SE HA DE EVALUAR.

2. NO OLVIDE ANEXAR A SU ARCHIVO .PDF DE EVIDENCIAS UNA PORTADA PROFESIONAL, Y ESTA SOLICITUD DE ACTIVIDADES CON TODAS LAS HOJAS FIRMADAS EN EL MARGEN DERECHO.
3. POR ÚLTIMO, TODA EVIDENCIA GENERADA QUE CONTENGA AL MENOS UNA TRANSCRIPCIÓN DE CUALQUIER FUENTE Y DE CUALQUIER TIPO, ES DECIR CON MATERIAL PLAGIADO SERÁ ANULADA DE FORMA INCONTROVERTIBLE.



[Handwritten signatures]

**INSTITUTO
TECNOLÓGICO
DE
CELAYA**

Lenguajes y Autómatas II

Equipo 2

Gomez Cabañas Anthony 20030057

Ortega Hernández Cristhian Alberto 20031091

Ponce Morales Alma Gabriela 20030274

Ruiz López Miguel Ángel 20030935

**MONOGRAFÍA
ANÁLISIS
SEMÁNTICO**

I.S.C. Ricardo González González

ÍNDICE

Índice de recursos	4
Introducción	5
Generalidades	6
Desarrollo	7
4.1 Árboles de expresiones	7
4.1.1 Conversión notación infija a Prefija	10
4.1.2 Expresiones Binarias	12
4.1.3 Uso del árbol de expresiones	13
4.2 Acciones semánticas de un analizador sintáctico	15
4.2.1 ¿Qué es un analizador sintáctico?	15
4.2.2 Análisis Sintáctico	16
4.2.2.1 Funciones principales	18
4.2.3 Acciones Semánticas	20
4.3 Comprobaciones de tipos en expresiones	25
4.3.1 Comprobación estática	26
4.3.2 Ejemplo de verificación de tipos	27
4.3.3 Comprobación dinámica	29
4.3.4 Implicaciones de las comprobaciones de tipos en expresiones	31
4.4 Pila Sémantica en un analizador sintáctico	34
4.4.1 Operaciones básicas de las pilas	35
4.5 Esquema de traducción	42
4.5.1 Generación de un esquema de traducción	43

4.5.2 Esquema de traducción a partir de una definición L-atribuida	49
4.6 Generación de la tabla de símbolos y de direcciones	51
4.6.1 ¿Qué es una tabla de símbolos	51
4.6.2 ¿Por qué son necesarias las tablas de símbolos	52
4.6.3 Objetivos de la tabla de símbolos	53
4.6.4 Compiladores de una y varias pasadas	55
4.6.4.1 Compiladores de varias pasadas	55
4.6.4.2 Compiladores de una pasada	57
4.6.5 Contenido de la tabla de símbolos	58
4.6.5.1 Nombre del identificador	59
4.6.5.2 Atributos de los identificadores	60
4.6.6 Operaciones con la tabla de símbolos	67
4.6.6.1 Tablas de símbolos y declaración explícita y implícita	68
4.6.6.2 Operaciones con lenguajes estructurados en bloques	69
4.6.7 Organización de la tabla de símbolos	71
4.6.7.1 Tabla de símbolos como lista desordenada	71
4.6.7.2 Tabla de símbolos como lista ordenada	72
4.6.7.3 Tabla de símbolos como árboles binarios	72
4.6.7.4 Tabla de símbolos como tablas Hash	73
4.7 Manejo de errores semánticos	75
4.7.1 Definición	75
4.7.2 Formas de manejo	77

Conclusiones
Bibliografía

79
81

ÍNDICE DE RECURSOS

Figura 1	Árbol de expresiones $3 + ((5+9)*2)$	7
Figura 2	Notación infija, postfixa y prefija	9
Figura 3	Expresión Infija invertida	10
Figura 4	Vaciado de la pila	11
Figura 5	Ejemplo infijo a prefijo	12
Figura 6	Árbol de expresiones $(6+4)*8(7+4)$	12
Figura 7	Árbol de expresiones de una expresión binaria	13
Figura 8	Uso del comprobador de tipos	26
Figura 9	Ejemplificación del uso de la pila	35
Figura 10	Inserción en la pila	36
Figura 11	Pila resultante de la inserción	36
Figura 12	Ejemplo eliminación en la pila	37
Figura 13	Uso de la pila en la evaluación de expresión	39
Figura 14	Árbol sintáctico de la expresión $3*5+4$	44
Figura 15	Árbol sintáctico de la expresión $3*5+4$ Con acciones semánticas	45
Figura 16	Árbol sintáctico de la expresión $3*5+4$ con acciones semánticas y su recorrido	46
Figura 17	Esquema de traducción y árbol con reglas semánticas para la entrada "95-2+"	48
Figura 18	Sentencia de asignación	49
Figura 19	Se imprime $A_1.h$ y $A_2.h$ antes de evaluarlo	50
Figura 20	Se evalúa $A_1.h$ y $A_2.h$ antes de imprimirlo	50

Figura	21 Compilador de varias pasadas	56
Figura	22 Compilador de una pasada	57
Figura	23 Tabla de símbolos sencilla	58
Figura	24 Fragmento de programa en bloques	69
Figura	25 Subtabla de un programa en bloques	70
Figura	26 Estructura de lista desordenada	71
Figura	27 Estructura de lista ordenada	72
Figura	28 Estructura de árbol binario	73
Figura	29 Estructura de la tabla Hash	74

Introducción

En esta monografía, se va a incursionar en la fase de análisis semántico en el desarrollo de compiladores, un aspecto fundamental en la traducción de código fuente a un lenguaje intermedio o código objeto. A lo largo del documento, se van a abordar una serie de temas cruciales que abarcan desde la generación de cárboles de expresiones y las acciones semánticas de los analizadores sintácticos hasta la comprobación de tipos en expresiones. También se va a examinar la importancia de la pila semántica en la evaluación de expresiones y la gestión de la información contextual, el esquema de traducción y cómo se generan las tablas de símbolos y direcciones. Finalmente, se va a abordar en el manejo de errores semánticos, un componente crucial para garantizar que los desarrolladores puedan depurar y comprender los problemas en sus programas de manera efectiva. A lo largo de este trabajo entonces, se busca proporcionar una comprensión sólida de los principios y prácticas relacionadas con el análisis semántico en la construcción de compiladores.

GENERALIDADES

Esta monografía tiene como objetivo adentraros en los pilaresenciales del análisis semántico en el ámbito de la compilación de programas. Se explorarán con profundidad diversos componentes de este proceso, incluyendo la construcción de cintillas de expresiones, acciones semánticas, comprobación de tipos, gestión de pila semántica, mecanismos de traducción, generación de tablas de símbolos y manejo de errores.

Este análisis detallado permitirá comprender la importancia de estos conceptos en la generación de programas eficientes y confiables.

VIDEO NUMERO 1: [Link](#)

~~Aprendizaje~~ 4.1 Árboles de expresiones

Los árboles de expresiones son una representación de la estructura de las expresiones ya sean algebraicas o booleanas. Para contextualizar en realidad los árboles de expresiones son árboles binarios en los cuales cada nodo interno corresponderá al operador y cada nodo hoja corresponderá al operando donde dicho árbol se evaluará aplicando recursivamente los operadores a los operandos correspondientes.

Como ejemplo introductorio se utilizará el siguiente ejemplo que corresponde al árbol de expresiones para " $3 + ((5+9)*2)$ "

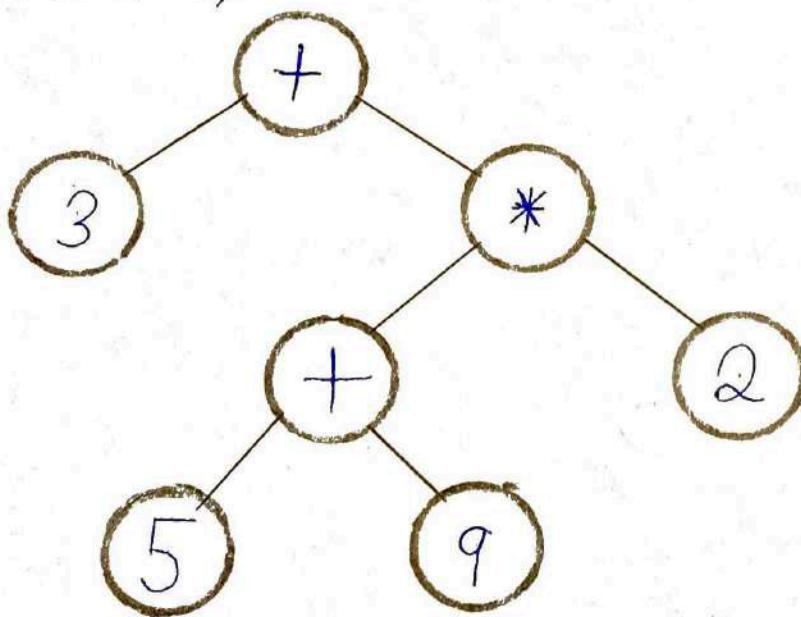


Figura 1 Árbol de expresiones $3 + ((5+9)*2)$

Alumno M. C. Gómez

Para recordar un poco y en base al ejemplo anterior, un árbol binario será todo aquel árbol en el cual cada nodo de este tendrá ya sea cero, uno o dos hijos como máximo, esta estructura restrictiva se aplica al árbol de expresiones, para simplificar el procesamiento de dichos árboles.

También vale la pena recordar los siguientes conceptos:

- * Los árboles binarios perfectos son aquellos en donde todos los nodos que lo compone tienen 0 o 2 hijos, y además que todas las hojas están en el mismo nivel.
- * Los árboles binarios equilibrados son árboles en donde los subárboles izquierdo y derecho no difieren en altura o profundidad en más de 1.

Así también es necesario considerar las siguientes características al introducir la expresión:

- * El nodo raíz siempre deberá ser un operador.
- * Las hojas siempre deberán ser operandos.
- * Los nodos se deberán etiquetar con operadores.
- * Si un operador tiene mayor prioridad que su raíz, este se colocara como su hijo.
- * Si un operador tiene igual o menor prioridad que un nodo, este se colocara como padre.
- * Un nodo podrá contener como hijo otro subárbol que contendrá una pequeña expresión.

~~Aprovechar~~ ~~para~~
Para abordar de mejor manera el tema de los árboles de expresiones, se ah de revisar el procedimiento o reglas para construir este mismo las cuales son las siguientes.

Primero la expresión necesita encontrarse en notación polaca o en otras palabras, en prefija (notación) donde es necesario resaltar que la notación utilizada comúnmente es la notación infija, recordando el ejemplo anterior dicha operación " $3 + ((5+9)*2)$ " se encuentra en notación infija en donde el operador se posiciona "entre" los operandos. Mientras que la prefija el operador "precede" o se encuentra antes de los operandos.

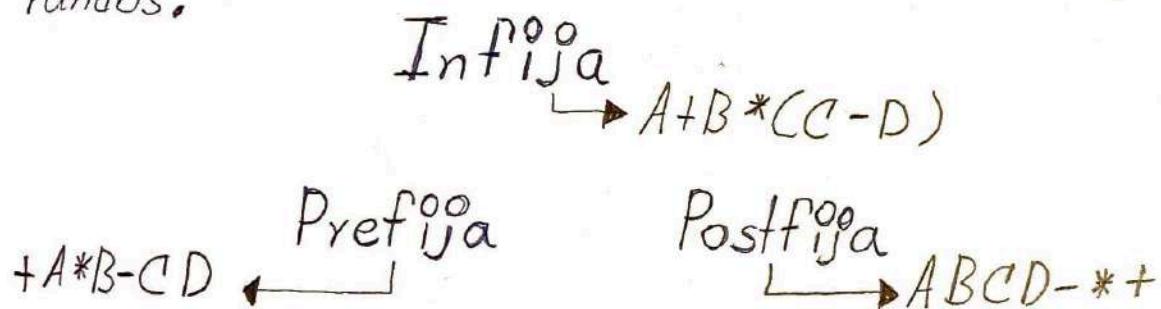


Figura 2 Notacion infija, postfija y prefija

Como se puede observar en la ilustración anterior, se puede apreciar que la principal diferencia después del orden de los operadores es que en prefija o postfija (no se utilizará en este, pero se colocara como referencia) no se cuenta con paréntesis.

~~Aprendizaje~~
El procedimiento general para realizar dicha conversión es el siguiente.

4.1.1 Conversión notación infija a prefija

Para llevar a cabo dicho proceso se tomará de base la expresión en orden infijo siguiente → " $(6+4)*8(7+4)$ ".

Paso inicial: Se deberá leer la expresión de derecha a izquierda en lugar de izquierda a derecha.

" $(6+4)*8(7+4)$
"Se lee"
↳) 4 + 7 (8 *) 4 + 6 C

Figura 3 Expresión Infija invertida

En el caso de los operadores estos se irán guardando en una pila que se vaciará en el arreglo final una vez se cierre un paréntesis (es decir ya figura el deertura como el de cierre) mientras que los operandos se irán vaciando a un arreglo que representará el resultado convertido.

Para operar en base a lo dicho anteriormente se leerá de izquierda a derecha la expresión previamente invertida si se encuentra un operando este directamente se pasará a salida, pero si se encuentra un operador, este procederá a guardarse en una pila (NOTA: aunque los parentesis no son

Aprobado

Operando, para dicha conversión se tomará como tal) y dicha pila será vaciada una vez se tenga tanto un paréntesis de abertura y uno de cierre (cabe destacar que al leerlo de forma invertida primero visualizaremos el de cierre y después el de abertura pero solo es una convención igual seguirá siendo el mismo) cabe también recordar que al hablar de una pila su vaciado será primero el último operador introducido y el último en salir será el primer operando ingresado para reforzar esto se incluye la siguiente ilustración.

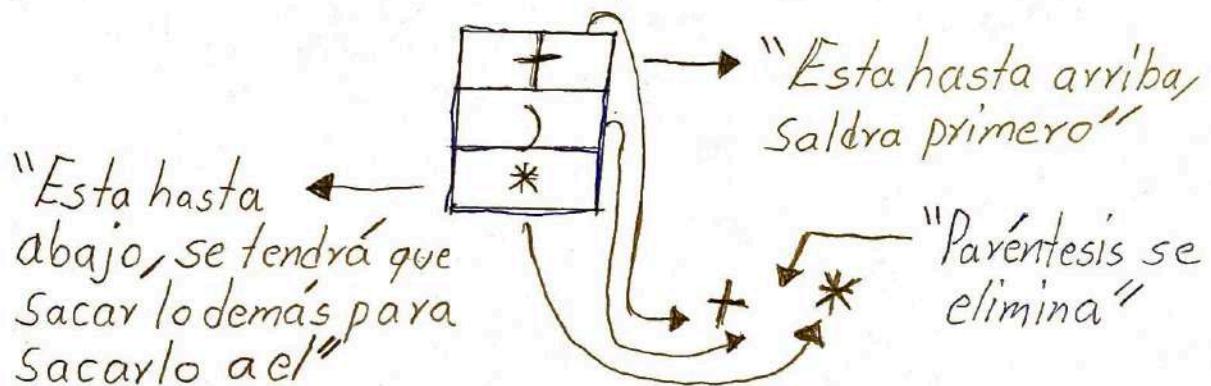


Figura 4 Vaciado de la pila

Siguiendo el algoritmo indicado anteriormente cuando se terminen todos los operandos o operadores disponibles para obtener finalmente la expresión en forma infija únicamente será de volverla a invertir.

Para indicar de mejor manera en la siguiente ilustración se realizará la conversión de la expresión en orden infijo "(6+4)*8(7+4)" a su equivalente en prefijo.

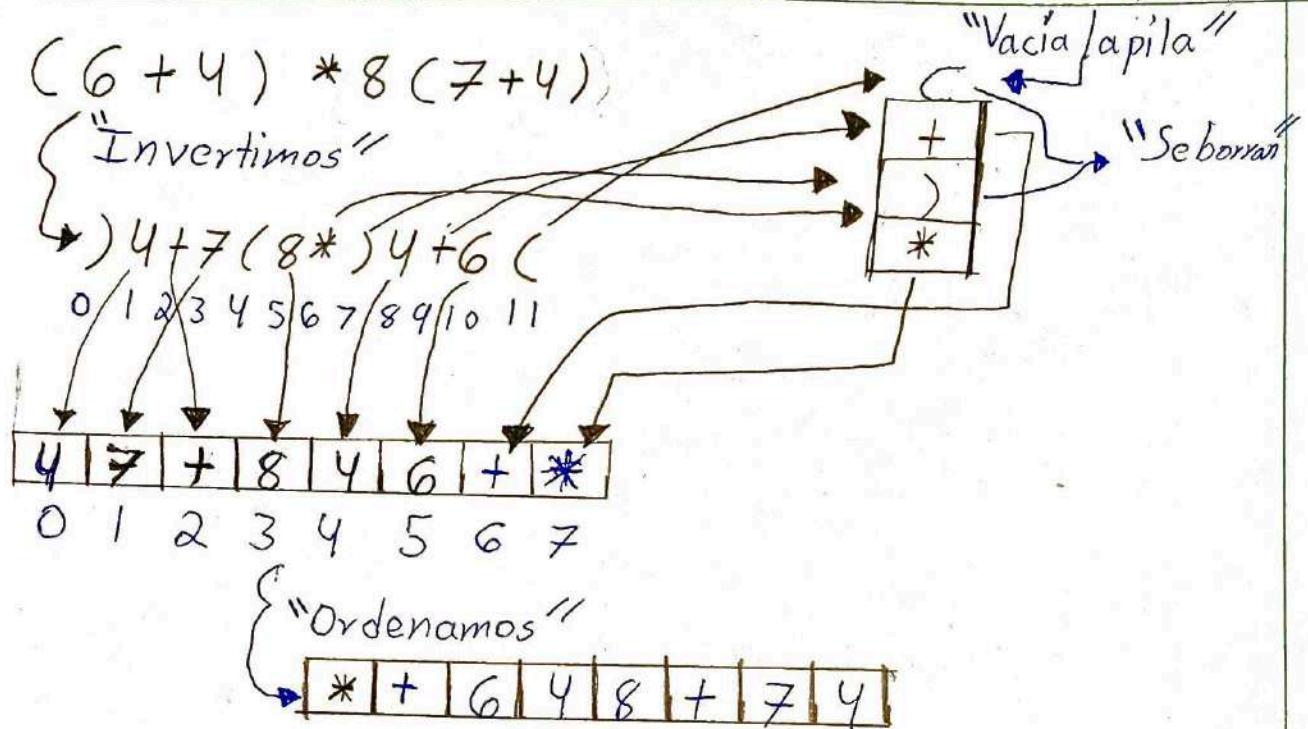


Figura 5 Ejemplo infijo a prefijo

Y ahora con el resultado anterior se tiene el siguiente árbol.

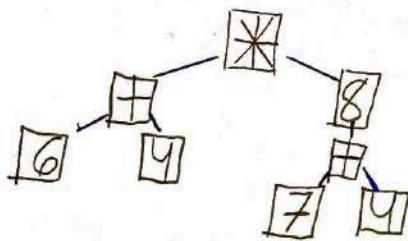


Figura 6 Árbol de expresiones $(6+4)*8(7+4)$

4.1.2 Expresiones Binarias

Aunque suele ser más común el manejo de expresiones algebraicas, también como se había planteado inicialmente se pueden representar de forma muy similar las expresiones binarias.

~~Algoritmo~~ ~~Diagrama~~
La única diferencia son los valores específicos y los operadores empleados, las expresiones booleanas utilizan verdadero (true) y falso (false) como valores constantes, y la lista de operadores incluyen a:

- * \wedge and (AND)
- \vee or (OR)
- \neg neg (NOT)

Para exemplificar de mejor manera lo anterior a continuación se mostrará el árbol de expresión correspondiente a la expresión binaria →
“((true \vee false) \wedge \neg false) \vee (true \vee false)”

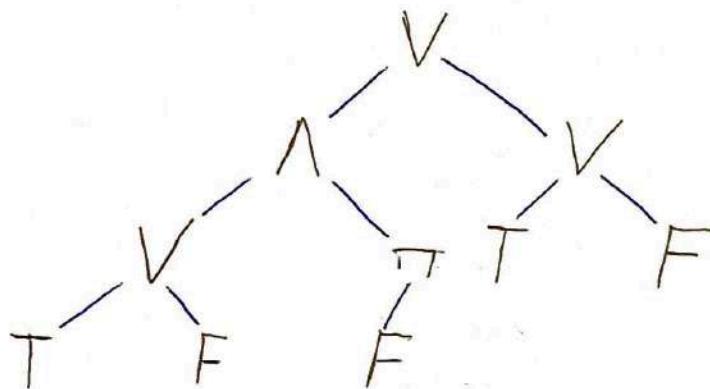


Figura 7 Árbol de expresiones de una expresión binaria

4.1.3 Uso del árbol de expresiones

Desempeñan un papel muy importante representando el código a nivel de lenguaje en la forma de los datos, los cuales como se ha revisado hasta el momento esta principalmente almacenado en estructuras de

~~Ambos~~ ~~Arboles~~
árboles. También se utiliza en la representación en memoria de las expresiones lambda ya que mediante la estructura de árbol de datos se puede expresar las expresiones lambda de una forma transparente y explícita. Pero a grandes rasgos el objetivo principal de los arboles de expresiones es la de hacer que expresiones complejas puedan ser fácilmente evaluadas a través del uso de estos. Además también permite encontrar la asociatividad de cada operador en la expresión así como también es utilizado para evaluar expresiones tanto en orden postfix, infix y prefix.

• 4.2 Acciones Semánticas de un Analizador Sintáctico

4.2.1 Analizador Sintáctico

Un analizador sintáctico es la parte del análisis de un compilador que fragmenta el código fuente en pequeñas secciones de cadenas de caracteres llamados tokens. Su función principal es determinar si las secuencias de tokens generadas por el analizador léxico cumplen con las reglas de gramática del lenguaje en cuestión. En caso de que el programa de entrada sea válido, se suministra el token al árbol sintáctico que lo reconoce.

Los analizadores sintácticos crean diagramas lógicos llamados árboles, cuyo propósito es ayudar al compilador a entender el contexto de cada token y su relación con el código fuente.

Podría decirse que el analizador sintáctico es alguna representación del árbol sintáctico que reconoce las secuencias de tokens creados por el analizador léxico.

4.2.2 Analizador Semántico

El analizador semántico suele trabajar simultáneamente con el analizador sintáctico y en estrecha cooperación.

Se entiende por semántica como el conjunto de reglas que especifican el significado de cualquier sentencia sintácticamente correcta y escrita en un determinado lenguaje. El análisis semántico dota de un significado con coherencia a lo que se ha hecho durante la etapas del análisis sintáctico.

Por ejemplo, en base a la reglas de un lenguaje, se requiere hacer la siguiente declaración:

$\langle \text{Asignación} \rangle \rightarrow \text{id} := \langle \text{Expresión} \rangle$

La sentencia tiene una sintaxis adecuada, pues para lograr llegar al analizador semántico, primero se tuvo que haber pasado por la

[Handwritten signatures]

prueba sintáctica.

Es común que se tengan ciertas restricciones, en el caso del ejemplo anterior, las restricciones podrían ser las siguientes:

- El identificador de la parte izquierda debe estar previamente declarado.
- El tipo de expresión debe ser compatible con el identificador.

Es el analizador semántico quien debe encargarse de comprobar que estas dos restricciones se cumplen antes de declarar que la sentencia de asignación está bien formada.

El chequeo semántico es una fase crítica del análisis semántico que se asegura de que las operaciones realizadas en el código tengan sentido desde el punto de vista del tipo de datos involucrados, es este mismo chequeo semántico el que se encarga de que los tipos que intervienen en las expresiones sean compatibles o que los

parámetros reales de una función sean coherentes con los parámetros formales.

4.2.2.1 Funciones Principales de un Analizador Semántico

→ Identificar cada tipo de instrucción y sus componentes
Aquí, se debe desglosar el código fuente en instrucciones individuales y comprender la estructura de cada una. Por ejemplo, identificar, declaracion, atribuções, estructuras de control, llamadas a funciones, entre otras, y analizar sus componentes como variables, operadores, argumentos y expresiones.

→ Completar la tabla de símbolos

Una tabla de símbolos (TS) es una estructura de datos utilizada en el análisis semántico para llevar un registro de las actividades, variables, funciones, entre otros elementos del programa, junto con su información relacionada, como tipos de datos y ámbito de visibilidad. Completar esta tabla implica registrar cada símbolo encontrado en

en el código y almacenar información relevante para su posterior uso en el análisis semántico.

→ Realizar distintas comprobaciones y validaciones:

En esta etapa se realizan distintas comprobaciones para garantizar la coherencia y validación del programa. Dentro de estas comprobaciones se pueden incluir:

- Comprobaciones de tipos:

Asegurarse de que las operaciones se realicen con tipos de datos compatibles

- Comprobaciones del flujo de control:

Verificar que las estructuras de control (como bucles y condicionales) se utilicen de manera coherente.

- Comprobaciones de unicidad:

Asegurarse de que no haya múltiples definiciones del mismo símbolo en el mismo ámbito.

- Comprobaciones de emparejamiento:

Garantizar que los paréntesis, corchetes, llaves y otros delimitadores se utilicen de manera adecuada y estén debidamente emparejados.

4.2.3. Acciones Semánticas

Las acciones semánticas son operaciones que se realizan durante el proceso sintáctico para asignar significado a las estructuras gramaticales reconocidas. Estas acciones pueden ser muy variadas y dependen del contexto.

Las acciones semánticas son agrupadas en distintos grupos dependiendo del tipo de sentencia que estén tratando, estos grupos pueden ser:

→ Sentencias de declaración:

Esto implica registrar la información relacionada con el tipo de dato de las variables o el valor de retorno de funciones/procedimientos en la "tabla de símbolos". Esto es crucial para el chequeo posterior de tipos y para garantizar que las operaciones posteriores con estas variables sean coherentes con sus tipos.

→ Sentencias "Ejecutables":

Las sentencias ejecutables son aquellas que realizan operaciones reales, como operaciones matemáticas, asignaciones, llamadas a funciones,

entre otras. En este contexto, las acciones semánticas implican verificar que los tipos de datos de los operandos involucrados sean compatibles con la operación. Por ejemplo, asegurarse de que una resta se realice entre dos variables del mismo tipo (por ejemplo, dos números enteros) y que el resultado sea del tipo correcto.

→ Funciones y procedimientos:

Cuando se realizan llamadas a funciones o procedimientos, las acciones semánticas se encargan de verificar que los argumentos pasados coincidan en número, orden, tipo, etcétera, con los tipos de parámetros formales de la función o procedimiento. Esto garantiza que las llamadas sean coherentes con la definición de la función o procedimiento.

→ Identificación de variables:

Una acción semántica importante es verificar si un identificador (como el nombre de una variable) ha sido previamente declarado en el ámbito actual antes de su uso.

Esto asegura que no se utilicen variables no declaradas, lo que sería un error.

~~BB~~ ~~AA~~
A. M. G.

→ Etiquetas:

En contextos donde se utilizan etiquetas (por ejemplo, en bucles o saltos condicionales), las acciones semánticas se aseguran de que las etiquetas sean únicas y, en el caso de saltos, que se realicen solo a lugares válidos dentro del programa.

→ Constantes:

En general, las constantes no pueden ser modificadas una vez que se les asigna un valor. Las acciones semánticas se encargan de garantizar que las constantes no sean utilizadas en el lado izquierdo de una asignación, ya que esto sería un intento de modificar su valor, lo cual no está permitido.

→ Conversiones y equivalencias de tipo.

Las conversiones de tipos o comprobaciones de equivalencia se realizan para garantizar que los tipos de datos sean coherentes en operaciones que implican tipos diferentes, como convertir un número entero en un número de punto flotante o comparar valores de diferentes tipos.

→ Sobre carga de Operadores y funciones:

La sobre carga de operadores o funciones implica que el mismo operador o función se pueda usar con diferentes tipos de datos.

Las acciones semánticas se encargan de detectar cuándo se usa una operación o función sobre cargada y determinar cual de las versiones es la adecuada para la situación actual.

Las acciones semánticas son cruciales para convertir una secuencia de tokens y una estructura gramatical en un programa o una representación semántica adecuada. Estas

acciones permiten interpretar el significado de la entrada y realizar las operaciones necesarias para su posterior procesamiento, ya sea la ejecución de un programa, la traducción a otro lenguaje o cualquier otra tarea relacionada.

Estas acciones son cruciales tanto en el análisis semántico como en el análisis sintáctico del proceso de compilación, ya que garantizan que el programa cumpla con las reglas semánticas del lenguaje y que las operaciones se realicen de manera coherente y correcta.

4.3 COMPROBACIONES DE TIPOS EN EXPRESIONES

La labor de comprobación de tipos es esencial en el proceso de compilación de un programa. Consiste en asignar y verificar los tipos de datos en las construcciones sintácticas del lenguaje de programación, lo que garantiza que las operaciones y las asignaciones de variables sean coherentes y cumplen con las reglas del lenguaje.

Esta tarea no se limita únicamente a verificar la semántica de tipificación, sino que también incluye la detección de errores de tipo, como intentar realizar operaciones no permitidas en el lenguaje o asignar valores incompatibles a variables. Por lo tanto, la comprobación de tipos es crucial para garantizar la integridad y la seguridad del programa.

Poder su complejidad, la comprobación de tipos se divide en dos fases principales en el proceso de compilación: la primera ocurre durante el análisis

semántico donde se comprueban los tipos ~~en~~ en tiempo de compilación, y la segunda se realiza durante la generación de código intermedio, donde se puede realizar una comprobación más profunda y refinada de los tipos en tiempo de ejecución. Esto asegura que cualquier error de tipo se detecte y se maneje adecuadamente antes de que el programa se ejecute, lo que contribuye a la robustez y la fiabilidad del software resultante.



Figura 8. Uso del comprobador de tipos

4.3.1 Comprobación estático

Las comprobaciones estáticas realizadas durante la fase de compilación, desempeñan un papel fundamental en la detección temprana de errores en el código fuente. Estas verificaciones garantizan que las variables y expresiones se ajusten a las reglas sintácticas y semánticas del lenguaje de

programación. Su importancia radica en que ~~que evitan~~ que los errores de tipo, entre otros, pasen desapercibidos y provoquen comportamientos no deseados o incluso fallos en tiempo de ejecución. La verificación de tipos es un ejemplo destacado de comprobación estática.

Cuando se realiza una comprobación de tipos, el compilador examina las expresiones en el código y se asegura de que los tipos de datos utilizados en operaciones y asignaciones sean compatibles, el compilador informará del error y proporcionará información detallada para que el programador pueda corregirlo antes de que el programa se ejecute. Esta anticipación de errores es esencial para la fiabilidad y la robustez del software.

4.3.2 Ejemplo de verificación de tipos.

Un ejemplo de verificación de tipos se puede observar en el contexto de un lenguaje de programación que verifica que los tipos de datos en

Una operación de suma sean compatibles. Al ~~imaginar~~^{Asumir} un lenguaje de programación hipotético en el que los tipos de datos son estrictos, lo que significa que no se permiten conversiones automáticas entre tipos. En este caso, se realizaría una verificación de tipos para asegurar que los operandos de una suma sean del mismo tipo.

$a = 5$ → 'a' se declara como un entero
 $b = "10"$ → 'b' se declara como una cadena de texto
 $c = a + b$ → Se intenta sumar 'a' y 'b'

Como 'a' se define como entero, mientras que 'b' se declara como una cadena de texto, si se intenta realizar una operación de suma entre 'a' y 'b', el sistema de comprobación de tipos detectará que se anda intentando combinar el número con la cadena de texto, lo que no constituye una operación válida. Como consecuencia, el compilador emitirá un error de tipo, advirtiendo que la adición no está permitida.

4.3.3 Comprobación dinámica

Las comprobaciones dinámicas son una parte vital del proceso de garantizar la integridad y el comportamiento correcto de un programa durante su ejecución. A diferencia de las comprobaciones estáticas, que se realizan en tiempo de compilación, las comprobaciones dinámicas ocurren en tiempo de ejecución, cuando el programa ya está en funcionamiento.

Estas comprobaciones pueden abarcar diversas áreas, como la verificación de límites de arreglos, la validación de entradas del usuario, la gestión de excepciones y la detección de condiciones de correr en hilos de ejecución, entre otras.

Su objetivo es evitar que un programa se bloquee o proporcione resultados incorrectos en situaciones imprevistas.

Para llevar a cabo las comprobaciones dinámicas, el compilador agrega código adicional al programa que se ejecutará en tiempo de ejecución. Este código evalúa condiciones y verifica que se cumplan en tiempo real. Si una condición no se satisface, el programa puede tomar medidas específicas, como lanzar una excepción o informar un error.

Sin embargo, estas comprobaciones dinámicas también tienen implicaciones en el rendimiento y la seguridad del programa. El código adicional pueden rallentizar el programa y hacerlo menos eficiente. Además, si no se manejan adecuadamente, las comprobaciones dinámicas podrían presentar vulnerabilidades de seguridad.

EJEMPLO

Un ejemplo sencillo de comprobación dinámica se encuentra en la mayoría de los sistemas operativos, cuando se intenta eliminar un archivo. Cuando haces clic para eliminar un archivo en tu computadora, el sistema operativo verifica dinámicamente

mente si estás seguro de que deseas eliminar ese archivo. Si confirmas la eliminación, el archivo se borra. Esta es una comprobación dinámica porque ocurre en tiempo de ejecución, justo antes de que se realice la acción. La comprobación dinámica asegura que no elimines accidentalmente un archivo importante sin tener la oportunidad de confirmar tu decisión.

4.3.4. Implicaciones de las comprobaciones de tipos en expresiones.

Hay algunas implicaciones de las comprobaciones de tipos son:

- Validación de tipos en operaciones y operandos: cuando se realiza una operación dentro de una expresión, como sumar, restar o multiplicar, el compilador o intérprete examina minuciosamente que los operandos y operandos involucrados tengan tipos de datos compatibles con la operación específica.

Ejemplo: En un lenguaje de programación, cuando se desea realizar una operación de división ($/$) entre dos valores, el sistema verifica que ambos operandos sean números y que sean compatibles para la operación de división.

• Identificación de incompatibilidades: Esto actúa como un protector contra operaciones inválidas que podrían resultar en resultados incorrectos o fallas en la ejecución del programa.

• Facilitación de promoción de tipos y conversiones: La comprobación de tipos puede permitir la promoción de tipos o conversiones implícitas.

Ejemplo: En un lenguaje que admite la promoción de tipos, cuando sumas un entero y un número en punto flotante, como en la operación $3 + 2.5$, el sistema realiza una conversión implícita del entero a un número en punto flotante.

- Evaluación de expresiones condicionales: La ~~satisfacción~~ de tipos también se aplica a expresiones condicionales, como las utilizadas en sentencias "if" o "while". Se garantiza que estas expresiones condicionales se evalúen como valores booleanos o tipos compatibles.

Ejemplo: Dentro de una sentencia "if" en un lenguaje de programación, se examina una expresión condicional para determinar su veracidad, en este caso se tiene "if(edad>=18)" y se verifica si la variable "edad" cumple o no con la condición.

- Prevención de errores en tiempo de compilación, lo que significa que los errores de tipo se descubren antes de que el programa se ejecute.

4.4 Pilas semánticas en un analizador sintáctico

Las pilas y las colas son herramientas comunes en programación que simplifican diversas operaciones. Estas estructuras se pueden implementar usando matrices o listas enlazadas.

En el caso de una pila, se trata de una colección de datos donde el acceso se realiza mediante un extremo conocido como el "tope".

Un aspecto esencial de las pilas es que el elemento extraído siempre es el último en ser insertado, lo que las convierte en estructuras de datos LIFO. Si se implementan utilizando listas enlazadas, los elementos se agregan y retiran siempre desde el principio de la lista.

Las pilas tienen una amplia gama de aplicaciones en programación y se utilizan en muchas situaciones cotidianas.

En el contexto de un analizador sintáctico, se trata de un autómata de pila que se encarga de reconocer la estructura de una cadena de componentes léxicos.

Este analizador inicializa el compilador y, para cada símbolo de entrada, llama al analizador morfológico y proporciona el siguiente símbolo de entrada, lo que ayuda a procesar y comprender la estructura de un programa.

Cuando se menciona "pila semántica" se hace referencia a la necesidad de programar exclusivamente en un solo lenguaje, es decir, no se permite mezclar código de diferentes lenguajes. Esto asegura la coherencia y consistencia en el desarrollo de software.

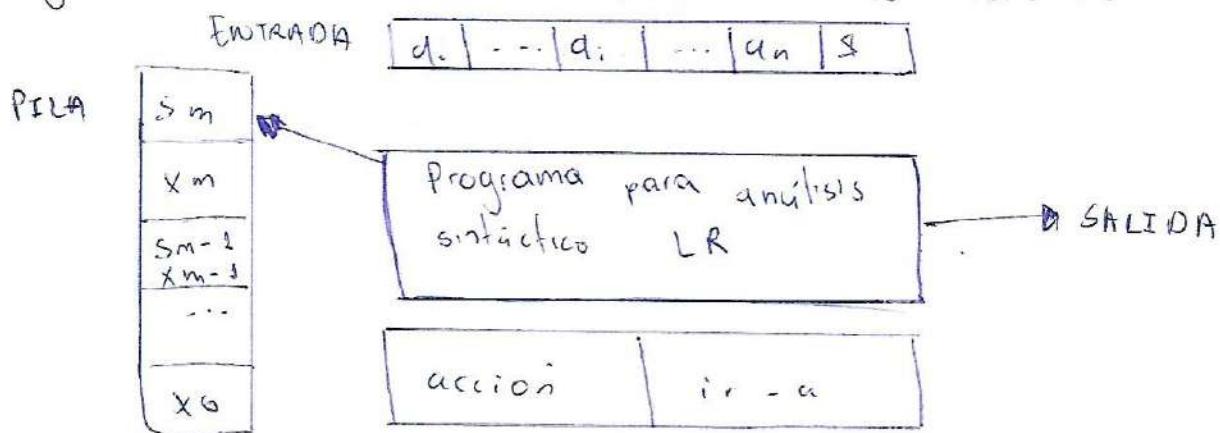


Figura 9.- Ejemplificación del uso de la pila

4.4.1. Operaciones básicas de las pilas

Insertar

Para comenzar, es importante señalar que esta operación es comúnmente conocida como "push".

La inserción en una pila ocurre en la parte superior como el último elemento insertado. Esto es una característica distintiva de las pilas.

Mientras que otras estructuras de datos lineales se representan horizontalmente, las pilas se representan de manera vertical. Por esta razón, se hace referencia a la "cima de la pila" en lugar de la "cola de la cima". A pesar de que, en esencia, se refiere al último elemento de la estructura de datos.

El proceso de inserción en la pila es bastante sencillo: se debe hacer que el nuevo nodo apunte al nodo previamente en la cima y, a continuación, establecer el nuevo nodo como la nueva cima de la pila.

Ahora, se puede observar un ejemplo de inserción:

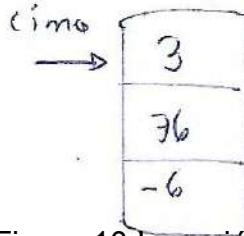


Figura 10 Inserción en la pila

Al insertar sobre esta pila el elemento 0, la pila resultante sería:

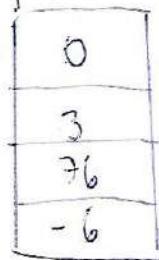


Figura 11 Pila resultante de la inserción

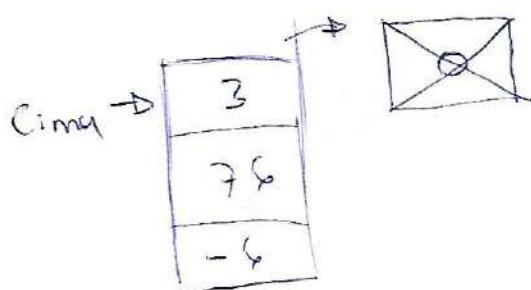
Eliminar

La operación de eliminación es comúnmente referida como "pop".

Cuando se lleva a cabo la eliminación de un elemento de la pila, se retira el elemento que ocupa la cima de la pila, es decir, el elemento que ha estado en la estructura durante menos tiempo.

El procedimiento para realizar esta operación es bastante sencillo: se avanza el puntero que apunta a la cima y se extrae el elemento que estaba en la cima previamente.

Si se aplica la operación "pop" a la pila de 4 elementos que se muestra arriba, el resultado sería:



4.4.2

Figura 12 Ejemplo eliminación en la pila

Ejemplo del uso de pila semántica

La pila semántica es una estructura de datos que se utiliza en la compilación y el análisis de código fuente para llevar a cabo operaciones rela-

cionesadas con la semántica del programa, como la evaluación de expresiones. En el contexto de la expresión $((1+2)*4)+3$, la pila semántica puede desempeñar un papel esencial en la evaluación de la expresión.

A continuación, se presenta una simplificación de cómo se podría usar la pila semántica en la evaluación de esta expresión:

1.- Cuando se encuentra el número 1 , se coloca en la pila semántica.

2.- Cuando se encuentra el operador $+$, se sabe que se necesita otro número para realizar la suma. Se coloca en la pila semántica.

3.- Cuando se encuentra el número 2 , se coloca en la pila semántica.

4.- Ahora, se puede aplicar la operación de suma a los dos números en la parte superior de la pila semántica ($1+2$). El resultado, 3 , se coloca nuevamente en la pila semántica.

5.- Cuando se encuentra el operador $*$, se coloca en la pila semántica.

6.- Cuando se encuentra el número 4 , se coloca en la pila semántica.

- 7.. Se realiza la operación de multiplicación entre los dos números en la parte superior de la pila semántica (4×3), y el resultado $= 12 =$ se coloca en la pila.
- 8.. Finalmente, cuando se encuentra el operador $= + =$, se coloca en la pila semántica.
- 9.. Cuando se encuentra el número $= 3 =$, se coloca en la pila semántica.
- 10.. Se realiza la última operación de suma en la pila ($12 + 3$), y el resultado $= 15 =$ se obtiene y se considera el resultado final de la expresión.

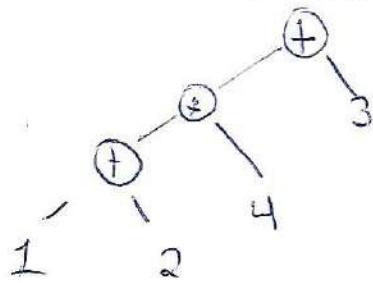


Figura 13 Uso de la pila en la evaluación de la expresión

El proceso anterior ilustra cómo la pila semántica puede utilizarse para evaluar expresiones, manteniendo temporalmente los operandos y operadores, aplicando las operaciones en el orden adecuado para obtener el resultado final de la expresión. Es importante destacar que, en la implementación real, la pila semántica es una estructura de datos que utiliza para gestionar y controlar este proceso de evaluación.

Al enfatizar lo anterior entonces se podría decir que el siguiente es el proceso general de cómo un analizador sintáctico utiliza la pila:

- Fase inicial: Al comenzar el proceso de análisis sintáctico, la pila se encuentra vacía.
- Análisis de Tokens: El analizador sintáctico escanea el código fuente y crea tokens.
- Incorporación de tokens en la pila: Conforme se generan los tokens, se incorporan en la pila siguiendo el orden en el que aparecen en el código fuente.
- Análisis Sintáctico: Durante el análisis sintáctico, el analizador sintáctico extrae tokens de la pila y los compara con las reglas de gramática del lenguaje de programación. Cuando un token y su contexto en la pila coinciden con una regla gramatical, se considera una coincidencia exitosa.
- Creación de estructura de datos: Al tener una coincidencia exitosa se crea una estructura de datos.

- Manejo de errores: Si en el proceso de análisis el analizador sintáctico extrae un token de la pila que no se ajusta a ninguna regla gramatical, se produce un error de sintaxis

VIDEO NUMERO 2: [Link](#)

4.5 Esquema de Traducción

Un esquema de traducción es una estructura basada en una gramática libre de contexto que incluye atributos asociados a los elementos gramaticales y acciones semánticas que se definen entre llaves, "{}" en las reglas de producción, indicando cuándo deben llevarse a cabo estas acciones.

En los compiladores de varias pasadas, existe una definida diferencia entre lo que se hace durante el análisis sintáctico y entre lo que hace un análisis semántico, donde son ejecutadas las rutinas semánticas. Sin embargo, en compiladores de una sola pasada, se requiere de un método en donde, a medida que se va ejecutando el análisis sintáctico, las acciones semánticas también se vayan ejecutando, esto se debe a que, en un compilador de una sola pasada, el compilador lee el código fuente una vez y genera el código objeto en esa única pasada.

Para ello existe el esquema de traducción, en donde una acción semántica se va a ejecutar cuando se reconozcan los tokens que le corresponden a esa acción semántica.

4.5.1. Generación de un Esquema de Traducción

EJEMPLO 1

Por ejemplo, se tiene la siguiente gramática libre de contexto definida, la cual permite sumas y multiplicaciones de números enteros:

Token o símbolo	Acción Semántica
$L \rightarrow E \ n$	<code>printf(E.val, "\n")</code>
$E \rightarrow E + T$	$\{E.val = E.val + T.val\}$
$E \rightarrow T$	$\{E.val = T.val\}$
$T \rightarrow T * F$	$\{T.val = T.val * F.val\}$
$T \rightarrow F$	$\{T.val = F.val\}$
$F \rightarrow \{\epsilon\}$	—
$F \rightarrow \text{dígito}$	$\{F.val = dígito.val\}$

En la tabla, '.val' es un atributo, el cual es el valor del símbolo o expresión correspondiente.

Al símbolo vacío, $\{F \rightarrow \{\epsilon\}\}$, no se le asigna una

~~Alumno~~
Nombre _____
Apellidos _____

acción semántica porque no requiere de una, sin embargo, podría añadírsela una como

printf("No hay símbolo \n")

pero como no es necesario, se omitirá (pues incluso, es algo que se pudo hacer en el análisis léxico). Además, se debe suponer que dígito es un token declarado cuyos valores son números enteros.

Ahora, para el siguiente ejemplo, se requiere evaluar la expresión $3 * 5 + 4$.

Para generar/hacer uso de su esquema de traducción, se tienen que realizar los siguientes pasos:

1. Construir el árbol de análisis sintáctico para la expresión $3 * 5 + 4$ en base a la gramática definida

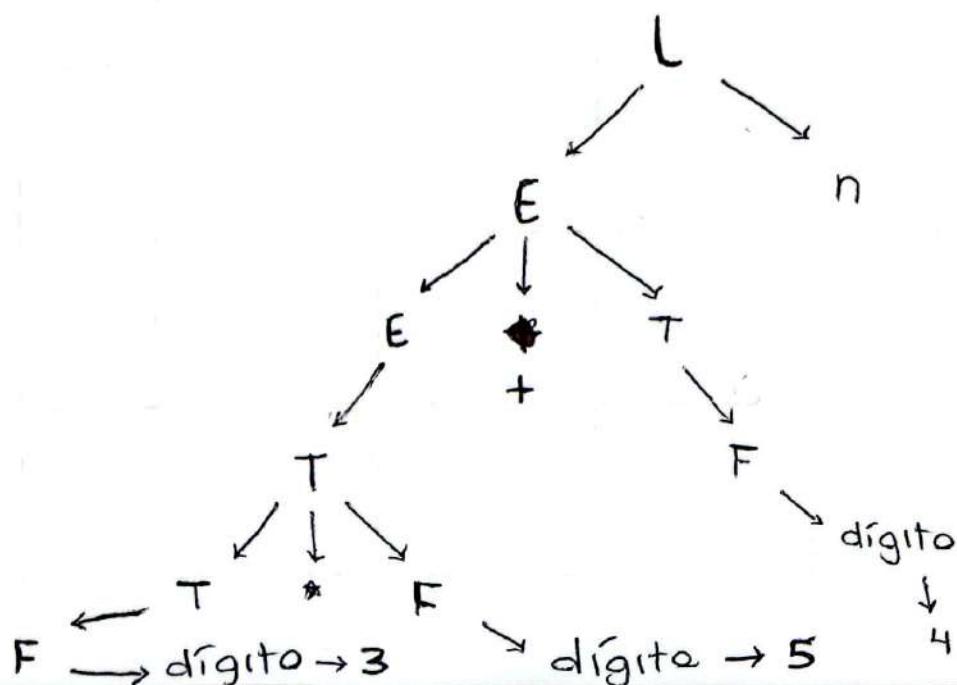


Ilustración 14 :
Árbol sintáctico de la
Expresión $3 * 5 + 4$

~~Árbol de interpretación~~
2. Recorrer el árbol y, a cada nodo, añadir la acción semántica correspondiente.

Este nodo nuevo se añadirá como si fuera un símbolo o token más.

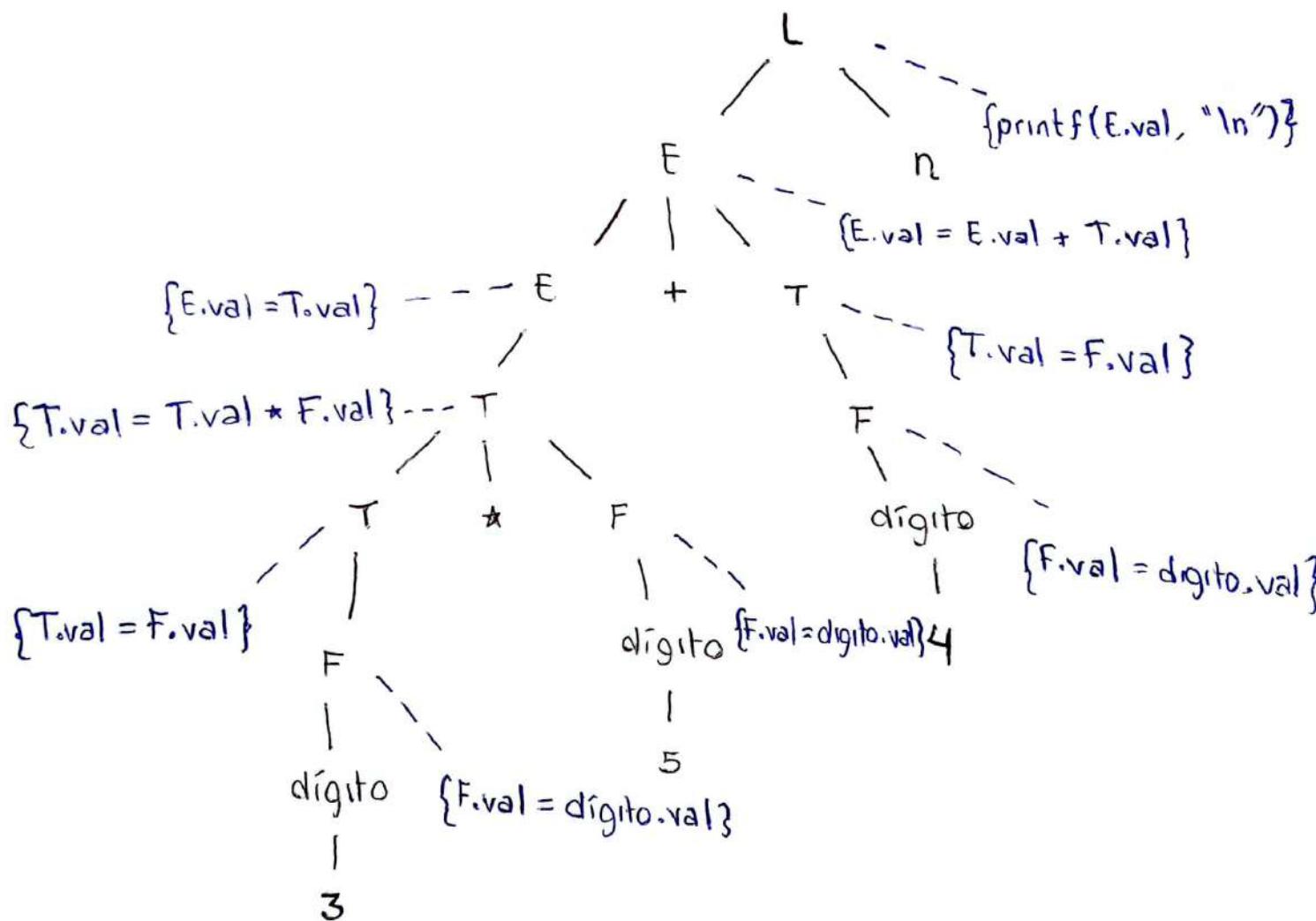


Ilustración 15:
Árbol sintáctico de la
Expresión $3 * 5 + 4$ con
Acciones Semánticas

3. Recorrer nuevamente el árbol, pero en preorden (desde la raíz, hacia la izquierda; después hacia la derecha).

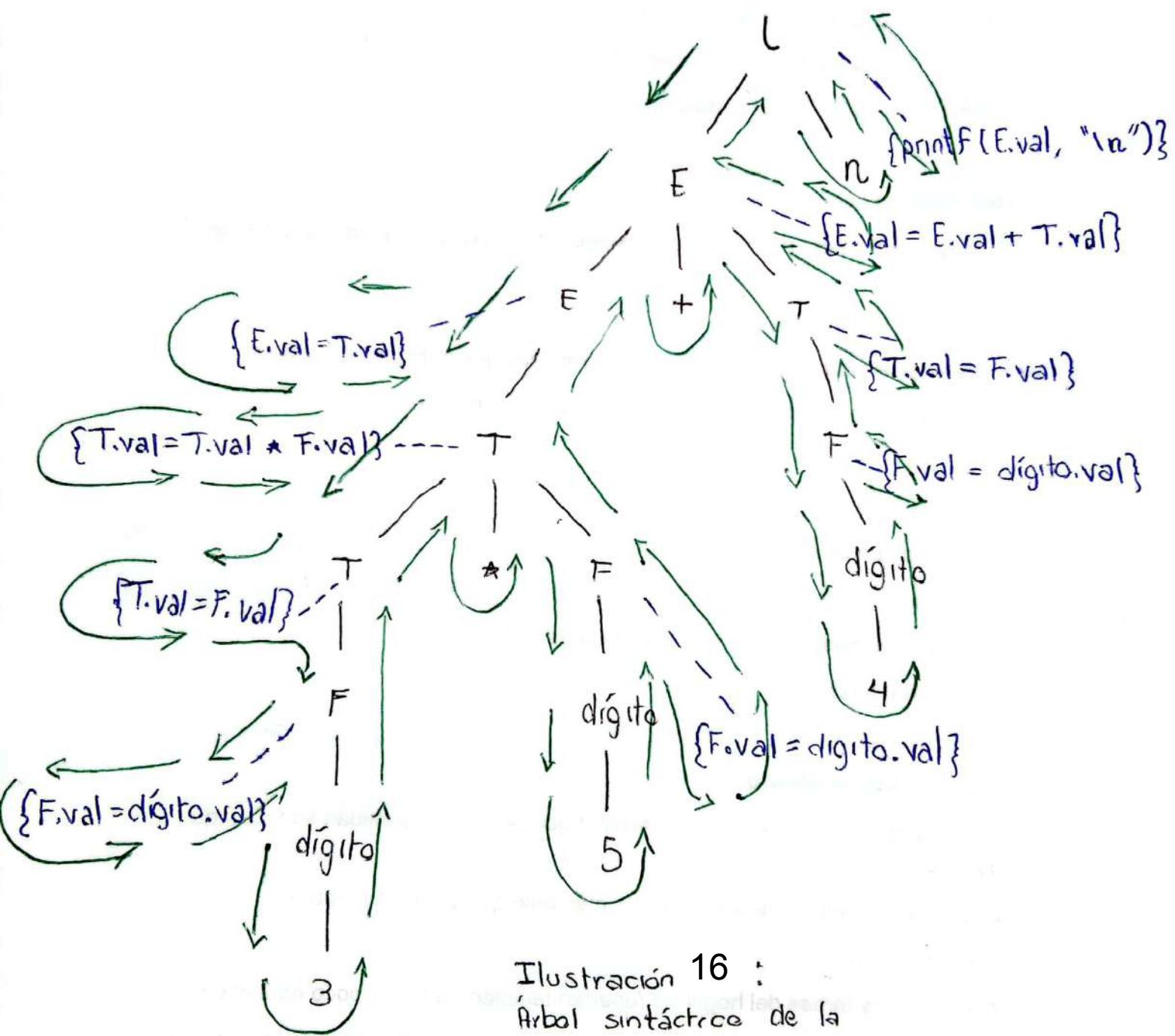


Ilustración 16 :
Árbol sintáctico de la
Expresión $3 * 5 + 4$
con Acciones Semánticas
y su Recorrido.

BB *des*
A. M. G.

Cuando se visita un nodo cuyo valor es una acción semántica, como una acción semántica no puede tener hijos, se ejecuta directamente. El resultado final de todo el recorrido es la acción semántica del símbolo inicial $L \rightarrow E_n$, la cual es la impresión del valor de E (el ~~valor~~ valor final) que, para este ejemplo, es 19.

$L \rightarrow E_n \quad | \quad \{\text{printf}(E.\text{val}, "\backslash n")\}$

Una característica de esto es que, al final, si se recorre el árbol en preorden, las acciones se ejecutarán como si se estuviera recorriendo en postorden, pues se está implementando como un analizador sintáctico ascendente, donde se empieza desde las hojas hacia la raíz.

Ejemplo 2

Otro ejemplo para ilustrar el concepto de precedencia con un ejemplo de traducción que convierte una expresión aritmética en notación polaca inversa, junto con un árbol de análisis para la entrada "9 - 5 + 2". Este ejemplo es útil para ver el análisis descendente y se ha simplificado eliminando la asociatividad por la igualdad y factorizando.

$$E \rightarrow T.R$$

$$R \rightarrow \text{opad } T \{ \text{print}(\text{opad}, \text{lexical}) \} R$$

$$T \rightarrow \text{NUM} \{ \text{print}(\text{NUM}, \text{lexical}) \}$$

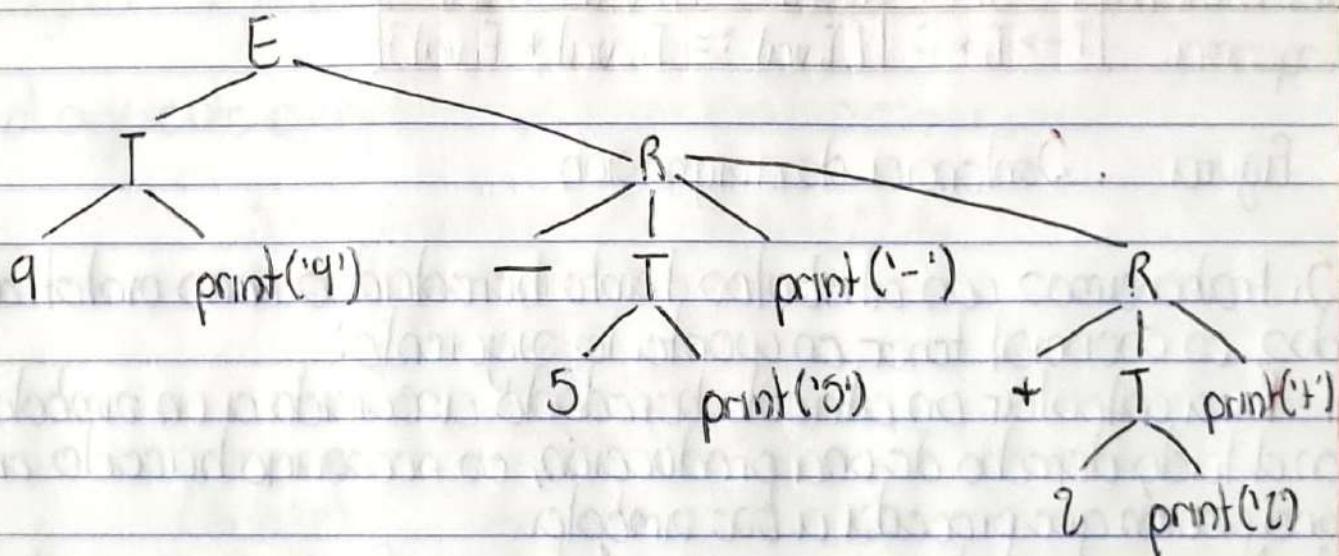


Figura 17. Ejemplo de traducción, árbol con reglas sintácticas para la entrada "9 - 5 + 2".

En este ejemplo, las acciones simbólicas se ubican en distintas partes de las reglas de producción, lo que significa que se ejecutarán en momentos específicos durante el análisis sintáctico. Al representar estos acciones simbólicas en forma de un árbol, se puede visualizar cuándo se activarán en un recorrido primario en profundidad del árbol de análisis.

5.2. Esquema de traducción a partir de una definición L-ATRIBUIDA

Las restricciones que se aplican en una definición L-garantizan que una acción no haga referencia a un atributo que aún no se ha calculado. En situaciones simples donde sólo se emplean atributos sintéticos, se sigue un proceso específico:

1. Se genera una sentencia de asignación para cada regla semántica.
2. Esta sentencia de asignación se coloca al final de la parte derecha de la producción gramatical asociada.

	Producción	Regla semántica
Definición:	$T \rightarrow T_1 * F$	$T_1.\text{val} := T_1.\text{val} * F.\text{val}$
Ejemplo:	$T \rightarrow T_1 * F$	$\{T.\text{val} := T_1.\text{val} * F.\text{val}\}$

Figura 18. Sentencia de asignación

Si trabajamos con atributos tanto heredados como sintéticos, es esencial tener en cuenta lo siguiente:

1. Para calcular un atributo heredado asociado a un símbolo en el lado derecho de una producción, es necesario hacerlo en una acción que prenda a ese símbolo.
2. No es apropiado que una acción haga referencia a un atributo sintético de un símbolo ubicado a la derecha de la propia acción.
3. En el caso de un atributo sintético relacionado con el símbolo que encabeza la producción, su cálculo sólo debe llevarse a cabo después de que todos los atributos a los que hace referencia hayan sido calculados, generalmente al final de la producción.

En el siguiente ejemplo no se respecta la anterior:

$$\begin{array}{l} \text{J} \rightarrow A_1, A_2 \quad \{A_1 \cdot h := 1; A_2 \cdot h := 2;\} \\ A \rightarrow a \quad \{\text{print}(A \cdot h);\} \end{array}$$

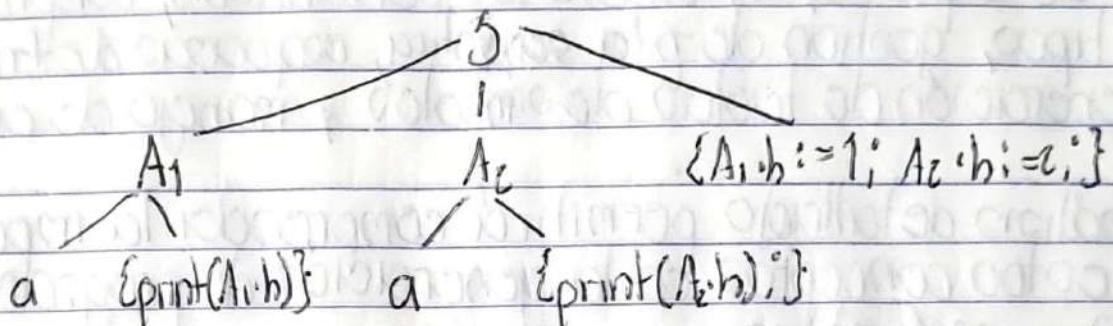


Figura 19. Se imprime $A_1 \cdot h$ y $A_2 \cdot h$ antes de calcularlo

En el siguiente ejemplo, si se respetan las condiciones:

$$\begin{array}{l} \text{J} \rightarrow \{A_1 \cdot h := 1;\} A_1 \quad \{A_2 \cdot h := 2;\} A_2 \\ A \rightarrow a \quad \{\text{print}(A \cdot h)\} \end{array}$$

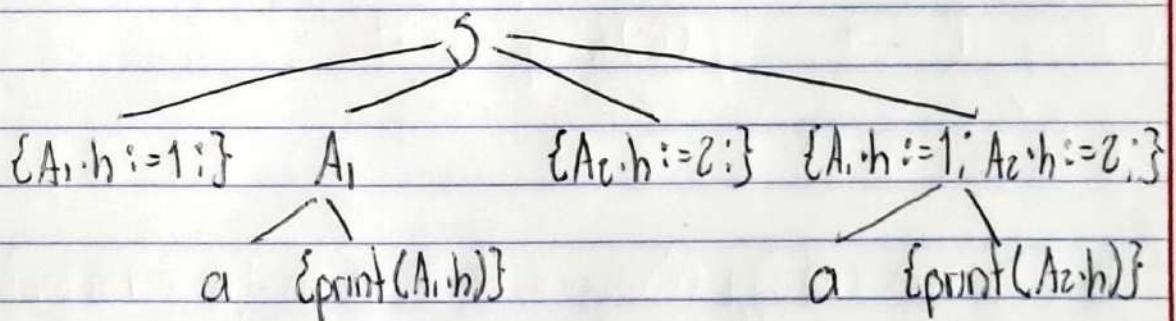


Figura 20. Se evalua $A_1 \cdot h$ y $A_2 \cdot h$ antes de imprimirlo

4.6. GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DE DIRECCIONES

4.6.1. ¿QUÉ ES UNA TABLA DE SÍMBOLOS?

Es una estructura de datos de alto rendimiento donde se almacena toda la información necesaria sobre los identificadores de variable. La tabla de símbolos tiene dos funciones principales:

- Recalcar checos semánticos
- Generar código

Esta estructura de datos permanece en la memoria sólo en tiempo de compilación y no en tiempo de ejecución (la única excepción a esto es cuando se compila con opción de depuración).

En esta tabla se almacena la información que se requiere sobre las variables del programa, tal como nombre, tipo, dirección de localización en memoria, tamaño, etc. Es importante que sea gestionada de una manera adecuada y eficaz, ya que su manipulación consume gran parte del tiempo de compilación.

En la tabla de símbolos también se puede guardar información de los tipos de datos que crean los usuarios, los tipos numéricos, y, en general, cualquier identificador que se crece. Aquí es donde el desarrollador decide si necesita tener distintas claves de identificadores en una sola tabla o usar varias, donde cada una tendrá una clave distinta de identificadores.

4.6.2. ¿POR QUÉ SON NECESARIAS LAS TABLAS DE SÍMBOLOS?

La fase de análisis semántico se llama así porque requiere información relacionada al significado del lenguaje, que se encuentra fuera del alcance de la representatividad de las grandes librerías de control y los algoritmos que existen para

el análisis; por ello, se dice que captura la parte de la frase de análisis que se considera que está fuera del ámbito de la sintaxis.

La mayoría de los compiladores utilizan gramáticas libres de contexto para poder describir la sintaxis del lenguaje. Posteriormente, utilizan la frase de análisis semántico para resaltar acepciones que semanticamente no son parte del lenguaje. Para exemplificar esto, supongamos que se ha declarado una variable en un lenguaje de programación, el analizador sintáctico se encargará de comprobar, mediante una gramática libre de contexto, que el identificador forma parte de una expresión, luego, el analizador semántico tendrá que comprobar que el identificador empleado como parte de una expresión se haya declarado anteriormente. El analizador semántico, para realizar esta tarea, se auxilia de la tabla de símbolos, la cual posee una entrada para cada identificador que se declara en el contexto que se está analizando.

1.6.3. OBJETIVO DE LA TABLA DE SÍMBOLOS

Las tablas de símbolos son estructuras de datos que almacenan la información de los identificadores del lenguaje fuente. Sus principales aportaciones en el proceso de traducción son realizar comprobaciones semánticas y ayudar en la generación de código.

Toda la información que se almacene en la tabla de símbolos depende del tipo de elementos y características del lenguaje a procesar. Los elementos que pueden utilizar la tabla de símbolos son los distintos tipos de identificadores del lenguaje, como los nombres de las variables, de objetos, de funciones,

de otiquotao, de dñaco, de mñodos, entre otros.

La información que corresponde a un elemento del lenguaje, se almacena en los atributos de ese elemento. Estos varían de un tipo de lenguaje a otro y de un elemento a otro. Así, atributos como nombre, tipo, dirección relativa en tiempo de ejecución, dimensiones de los arrays, número y tipo de los parámetros de métodos, entre otros, son recogidos y almacenados en la tabla de símbolos.

Los atributos son obtenidos ya sea a través del análisis del programa fuente (explícitamente) o a través del contexto en el que aparece el elemento en el programa fuente (implícitamente).

Cuando se realiza el proceso de compilación, se accede a la tabla de símbolos en puntos que dependen inicialmente del número y naturaleza de los parámetros del procesador del lenguaje y del propio lenguaje.

En este escenario, es posible que una declaración de una variable sea procesada por el generador de código antes de que se termine de analizar por completo el código fuente. Esto puede resultar beneficiado, ya que cualquier variable que se declare después de su declaración, puede tener sus atributos registrados en la tabla de símbolos por parte del generador de código.

4.6.4. COMPILADORES DE UNA Y VARIAS PASADAS

4.6.4.1. COMPILADORES DE VARIAS PASADAS

En un compilador de varias pasadas, la tabla de símbolos se genera en la primera etapa, que incluye el análisis léxico y sintáctico, conocido como pasada 1. En los compiladores modernos, la tabla de símbolos se construye en la segunda etapa, durante el primer recorrido del árbol de análisis sintáctico después de que se haya creado con éxito. En el inicio del proceso de traducción de un programa fuente, la tabla de símbolos generalmente está vacía o contiene información limitada. Las palabras reservadas del lenguaje están manteniéndose en una tabla separada y se usan sólo durante el análisis léxico.

El analizador léxico descompone el programa fuente en tokens y verifica si estos coinciden con las palabras reservadas del lenguaje. Si el token no coincide con ninguna palabra reservada, se considera un identificador y se agrega a la tabla de símbolos durante el análisis sintáctico. Si el token coincide con un identificador que ya está presente en la tabla, el analizador sintáctico accede directamente al índice que representa ese identificador.

A medida que avanza el proceso de compilación, solo se crea una entrada en la tabla de símbolos para cada identificador nuevo encontrado, y la tabla se consulta una vez por cada nueva aparición de un identificador para determinar su existencia o agregarlo si es nuevo.

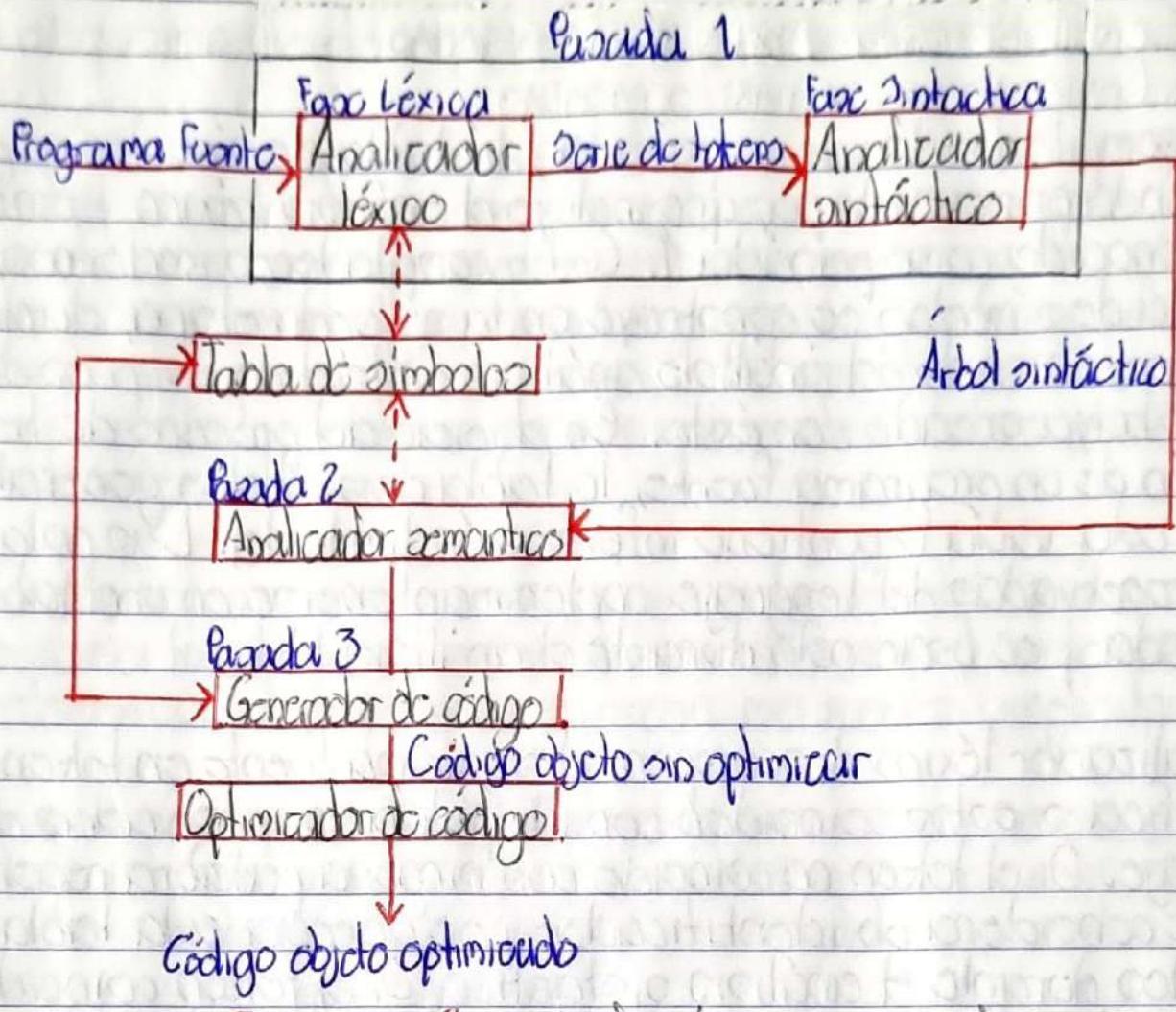


Figura 21 Compilador de varios pasados

Durante la etapa de análisis sintáctico, generalmente no se realizan operaciones que involucren el manejo de la tabla de símbolos, a menos que surgen situaciones ambiguas en la sintaxis que necesiten verificaciones semánticas para resolverse. No es hasta las fases de semántico y generación de código que la tabla de símbolos vuelve a ser utilizada.

La asignación de atributos a la tabla de símbolos podría intentarse en etapas del proceso de traducción distintas a la fase de análisis léxico. En cambio, esto requeriría realizar modificaciones en los analizadores sintácticos y semántico,

lo que conduciría a la creación de un compilador poco optimizado y desoptimizado.

4.6.4.2. Compiladores de una pasada

Otra manera de manejar las tablas de símbolos es cuando el análisis léxico, sintáctico, semántico y la generación de código se realizan en una pasada, esto quiere decir que se explora el texto fuente sentencia a sentencia o bloques de sentencias realizando los tres análisis y la generación del código.

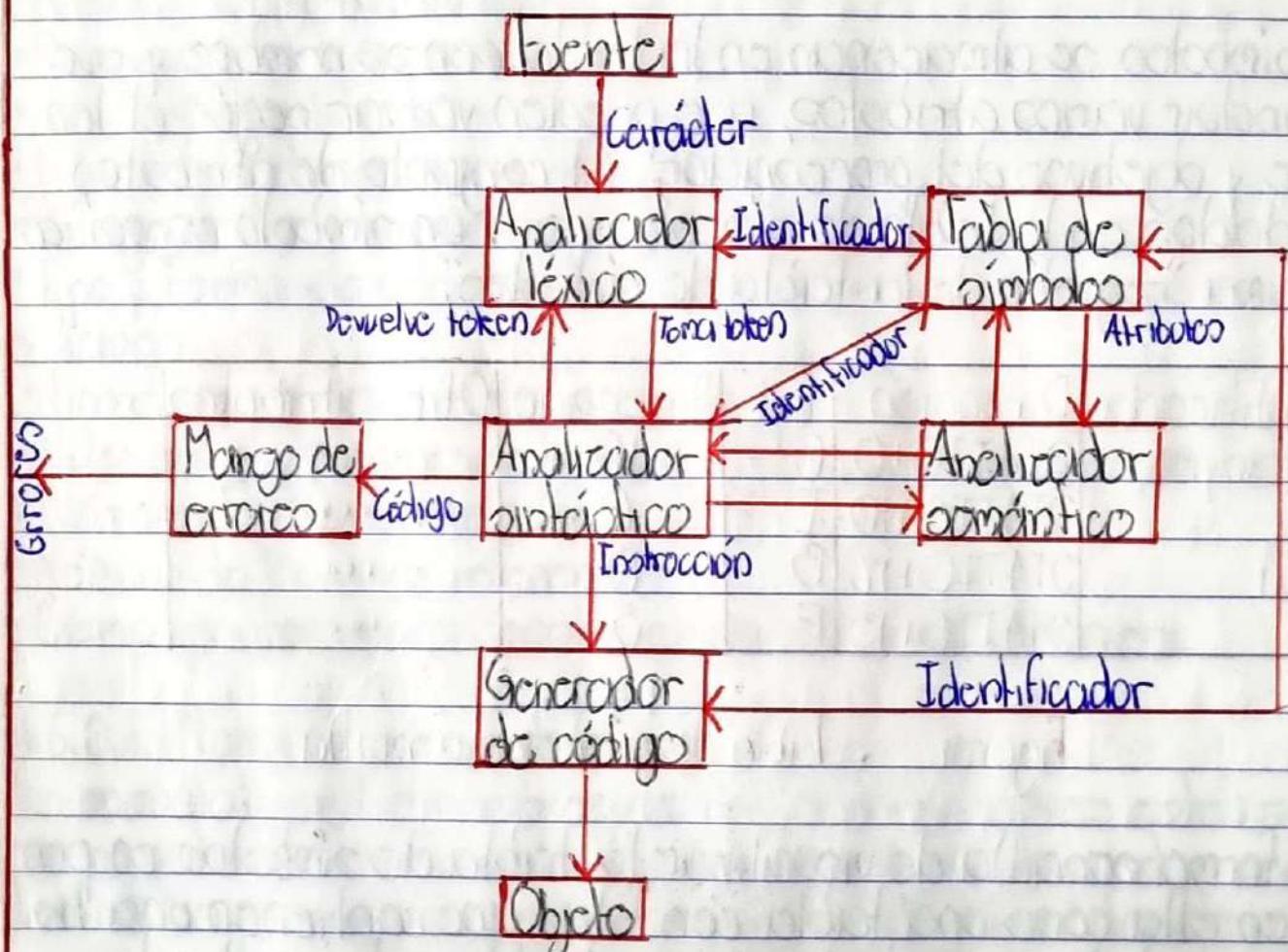


Figura 22. Compilador de una pasada

4.6.5. Contenido de la tabla de símbolos

Aunque el nombre "tabla de símbolos" sugiere una estructura de datos tabulares, no es la única forma en que se puede implementar. Lo fundamental es que la estructura permita establecer una relación entre los contextos de uso de los símbolos en el programa fuente y cómo se buscan en la tabla.

Para lograr esto, la estructura debe manejar diferentes contextos de búsqueda que reflejen los diferentes tipos de bloques en el lenguaje fuente que se están compilando.

Los símbolos se almacenan en la tabla con su nombre y pueden incluir varios atributos, que pueden variar según el lenguaje y objetivo del procesador. El conjunto de atributos guardados en la tabla de símbolos para un símbolo específico se llama "registro de la tabla de símbolos".

Identificador	Dirección	Tipo	Dimensiones	Otras atributos
Compañia1	STATIC+0	C	10	...
x3	STATIC+10	I	0	...
format1	STATIC+12	B	0	...
b	STATIC+13	F	3	...

Figura 23. Tabla de símbolos sencilla

Una forma sencilla de visualizar la tabla de símbolos se pensaría en ella como una tabla con filas que contiene una lista de atributos vinculados a un identificador particular.

Los tipos de atributos presentes en una tabla de símbolos dependen de las características propias del lenguaje de pro-

gramación para el cual se coloca desarrollando el compilador.

La forma en que se organiza la tabla de símbolos variará en función de las restricciones de memoria y los requisitos de acceso a datos en el entorno en el que se implemente el compilador.

La siguiente lista de atributos no es requerida en todos los compiladores, ya que su necesidad depende de las particularidades de la implementación de un compilador específico:

- Nombre del identificador
- Dirección en tiempo de ejecución
- Tipo del identificador
- Número de dimensiones del array
- Rango de las dimensiones de los arrays
- Tipo y forma de acceso de miembros de estructuras, uniones o clases
- Tipos de parámetros de funciones
- Valor del descriptor de fichero, y tipo de elementos de fichero
- Número de línea de declaración
- Número de línea de referencia
- Campo puntero para construir una lista encadenada.

4.6.0.1. NOMBRE DEL IDENTIFICADOR

Esencial que los nombres de los identificadores contienen información registrados en la tabla de símbolos para que puedan ser controlados por el analizador semántico y el generador de código.

Un desafío en la organización de la tabla de símbolos es que los nombres de los identificadores pueden variar en longitud. La forma en que se almacenan estos nombres depende del lenguaje.

de programación en el que se implementa la tabla.

4.6.5.2. ATRIBUTO DE LOS IDENTIFICADORES

4.6.5.2.1. DIRECCIÓN DE MEMORIA

En los lenguajes de alto nivel se utilizan identificadores para referenciar variables y elementos, pero en el código solo se trabajan con direcciones de memoria donde estos elementos están almacenados.

Cuando el compilador genera un código objeto de muy bajo nivel, es necesario establecer una relación constante entre cada identificador y su dirección de inicio. En algunos casos, el código objeto puede estar en un nivel de lenguaje más alto. En ese caso, los identificadores pueden ser utilizados directamente sin necesidad de direcciones.

La tabla de símbolos asiste al generador de código al traducir los identificadores a sus direcciones de memoria correspondientes. Estas direcciones son relativos, es decir, desplazamientos desde una dirección base de referencia.

4.6.5.2.2. TIPO

El atributo tipo se registra en la tabla de símbolos cuando se trabaja con lenguajes de programación que tienen diferentes tipos de datos, ya sea que estos tipos estén definidos de manera explícita o implícita.

La información del tipo es fundamental en las verificaciones semánticas de las sentencias en el programa. Además, se emplea para determinar cuánta memoria debe asignarse en tiempo de ejecución para una variable.

4.6.5.2.3. NÚMERO DE DIMENSIONES, MIEMBROS O PARÁMETROS

El número de dimensiones de un array no solo es importante en términos de verificación semántica, sino que también se utiliza como un parámetro fundamental en la fórmula general para calcular la dirección de un elemento específico dentro de ese array.

En el caso de los llamados a procedimientos o funciones, es crucial que el número de parámetros proporcionados coincida con el número declarados en la definición del procedimiento o función. Dependiendo del lenguaje de programación en el que se implemente la tabla de símbolos, puede ser beneficiosa combinar todos los atributos en uno, dado que su verificación semántica es similar.

4.6.5.2.4. RANGOS DE ARREGLOS

En la tabla de símbolos, es necesario registrar el valor máximo al que puede llegar un array. Esto se debe a que, cuando se declara un array, su rango de valores comienza en cero, pero resulta esencial conocer su número total de elementos.

En situaciones en las que el lenguaje de programación no establece un valor máximo para las dimensiones de los arrays, es fundamental crear una estructura de datos dinámica que almacene los valores máximos de cada dimensión del array.

4.6.5.2.5. TIPO Y FORMA DE ACCESO DE LOS MIEMBROS DE ESTRUCTURA, REGISTROS, UNIONES Y CLASES

En la tabla de símbolos, es crucial registrar el tipo de cada miembro de una estructura, unión o clase, y en el caso de las clases, también es necesario almacenar cómo se accede a estos miembros. Para las funciones, su nombre debe estar vinculado a la clase.

En el contexto de tipos simples, se suele representar en la tabla de símbolos mediante constantes predefinidas. Sin embargo, si una estructura contiene otra estructura, se aplicará el identificador que representa a la estructura interna.

Cuando hay múltiples variables de un tipo de estructura que no tienen un nombre específico, se puede optar por un puntero que indique dónde está definida la estructura.

Los clásicos poseen datos y métodos. La información de datos, se organiza de manera similar a las estructuras, pero los métodos se almacenan de manera análoga a las funciones, con información sobre nombre, tipo de resultado, número y tipo de parámetros.

Además, se deben accesar, accesar, identificar funcións y almacenar información sobre constructores, destructores y métodos virtuales.

4.6.5.2.6. TIPO DE LOS PARÁMETROS DE LAS FUNCIONES O MÉTODOS DE LAS CLASES

El tipo de datos que una función devolverá generalmente se apoya al tipo básico del identificador. La información sobre los tipos de los parámetros se extrae de las declaraciones de las funciones.

Dado que los lenguajes de programación no imponen un límite en la cantidad de parámetros que pueden tener las funciones y procedimientos, se considera una lista dinámica para gestionarlos.

4.6.3.2.7. DESCRIPTOR DE FICHEROS

En la tabla de símbolos es posible incluir el descriptor del archivo o el "handle" de bajo nivel vinculado al identificador del archivo, el cual se usará en el proceso de generación de código.

4.6.3.2.8. OTROS ATRIBUTOS

Los descriptores de código y los analizadores de rendimiento han usado de la información contenida en la tabla de símbolos durante la ejecución del programa. Por este motivo, las distintas implementaciones de compiladores suelen complementar la tabla con información adicional que es útil para otras herramientas.

Las listas de referencias cruzadas son otro tipo de atributo que ofrece una valiosa apertura durante la fase de depuración. Esas listas incluyen algunos atributos mencionados anteriormente y, además, proporcionan información sobre el número de línea en el código fuente donde se declara una variable o donde se hace la referencia por primera vez. También indican los números de las líneas en las que se hace referencia a la variable en el código fuente.

También es factible incorporar un atributo en forma de puntero, cuya mención simplificaría la generación de una lista de nombres de variables organizados en orden alfabético.

4.6.6. OPERACIONES CON LA TABLA DE SÍMBOLOS

En las tablas de símbolos normalmente se realizan dos operaciones clave: la inserción y la búsqueda. La forma en que se ejecutan estas operaciones varía ligeramente dependiendo de si las declaraciones en el lenguaje son explícitas o implícitas.

Además, también se llevan a cabo acciones como habilitar (act) y deshabilitar (react) las tablas correspondientes a los identificadores locales o automáticos.

4.6.6.1. TABLA DE SÍMBOLOS Y DECLARACIÓN EXPLÍCITA E IMPLÍCITA

4.6.6.1.1. LENGUAJE CON DECLARACIONES EXPLÍCITAS OBLIGATORIAS

Ambas operaciones se realizan en momentos específicos del compilador. La operación de inserción ocurre al procesar una declaración, ya que una declaración proporciona una descripción inicial de los atributos de un identificador en el programa fuente.

Cuando la tabla de símbolos está organizada de manera ordenada, la operación de inserción involucra llamar a un procedimiento de búsqueda para determinar el lugar adecuado donde se deben ubicar los atributos del identificador que se está insertando.

En caso de que la tabla no esté organizada, la inserción es más simple, pero aún requiere un procedimiento para ubicar los atributos. Sin embargo, la operación de búsqueda es compleja, ya que debe examinar toda la tabla para encontrar la información necesaria.

4.6.6.1.2. LENGUAJE CON DECLARACIONES IMPLÍCITAS EN LOS IDENTIFICADORES

Las operaciones de inserción y búsqueda están intrínsecamente vinculadas. Cada vez que se encuentra un identificador en el código fuente, se debe tratar como una referencia inicial, ya

que no se puede determinar de antemano si los atributos de ese identificador ya han sido registrados en la tabla de símbolos.

En consecuencia, cuando se detecta un identificador en el código fuente, se activa el procedimiento de inserción para registrar los atributos del identificador en la tabla de símbolos.

4.6.6.2. OPERACIONES CON LENGUAJES ESTRUCTURADOS EN BLOQUES

4.6.6.2.1. OPERACIONES DE ACTIVACIÓN Y DEACTIVACIÓN DE TABLAS DE SÍMBOLOS

Los lenguajes de programación estructurados en bloques incluyen dos operaciones adicionales conocidas como act, react.

La operación act se emplea cuando se reconoce el inicio de un bloque en el cual se pueden declarar identificadores locales. La operación complementaria react se utiliza cuando se identifica el fin del bloque.

Figura 24 Fragmento de programa en bloques

BBLOCK

REAL X, Y; STRING NAME;

M1: PBLOCK (INTEGER IND);
INTEGER X;

1

CALL M2(IND+1);

END M1;

M2: PBLOCK (INTEGER J);

BBLOCK;

3

END;

ARRAY INTEGER F(J);

GND;

END M2;

CALL M(X/Y);

END;

Cuando comienza un bloque, la operación act da lugar a la creación de una subtabla adicional en la cual se almacenan los atributos de las variables declaradas en ese nuevo bloque.

La utilización de colas subtablas reduce la ambigüedad que puede surgir al buscar identificadores con el mismo nombre en distintos bloques, proporcionando un contexto claro y separado por cada bloque en el que se declaran las variables.

The diagram illustrates the scope of variables in nested blocks through three tables:

- Top Table (Variables in the current block):**

Nombre	Tipo	Otros Atributos
X	R	...
Y	R	...
NAME	S	...
M1	BL	...

- Left Table (Variables in the MAIN block):**

Número Bloque	Bloque	Subtabla
1	MAIN	-
2	M1	-
3	-	-
4	-	-

- Bottom Table (Variables in the M1 block):**

Nombre	Tipo	Otros Atributos
IND	I	...
X	I	...

Arrows indicate the relationship between the tables: from the 'Subtabla' column of the left table to the top table; and from the 'Subtabla' column of the left table to the bottom table.

Figura 25. Subtabla de un programa en bloques

Al finalizar un bloque, la operación rec act elimina las entradas en la subtabla correspondiente al bloque que se ha cerrado recientemente. Esta eliminación implica que las variables declaradas en ese bloque concluido ya no son válidas ni tienen alcance en ninguna otra parte del programa.

4.6.7. ORGANIZACIÓN DE LA TABLA DE SÍMBOLOS

Existen varias formas de organizar una tabla de símbolos, cada una con sus propias ventajas y desventajas en términos de eficiencia de tiempo y espacio.

4.6.7.1. TABLA DE SÍMBOLOS COMO LISTA DESORDENADA

Una lista no ordenada, insertaría cada nombre secuencialmente a medida que se declare. La operación de búsqueda tendría que buscar de manera lineal y, por lo tanto, en el peor caso, tendría que examinar todas las entradas y, en el caso promedio, aproximadamente la mitad de ellas. Por lo tanto, el tiempo de búsqueda es de $O(n)$.

No hay una buenas razón para usar una estructura de datos tan inefficiente a menos que se acepte que el número de entradas sea extremadamente pequeño.

Carácter	Clase	Alcance	Declarado	Referenciado
Main	Programa	0	Línea 1	
a	Variable	0	Línea 2	Línea 11
b	Variable	0	Línea 2	Línea 7
P	Procedimiento	0	Línea 3	Línea 11
x	Parámetro	1	Línea 3	
a	Variable	1	Línea 4	Línea 6

Figura 26. Estructura de lista desordenada

4.6.7.2. TABLA DE SÍMBOLOS COMO LISTA ORDENADA

Con un arreglo ordenado, se puede realizar una búsqueda binaria en tiempo $O(\log n)$ en promedio, pero la operación de inserción tomaría más tiempo ya que se debe mover elementos. Sin embargo, para realizar una búsqueda binaria, necesitamos implementar la lista ordenada como un arreglo.

La representación de la lista como una lista enlazada ordenada devolvería un tiempo de búsqueda $O(n)$, aunque la operación de inserción sería en promedio más sencilla, ya que solo se necesitan cambiar punteros en lugar de cambiar la posición de cada elemento.

Carácteres	Clases	Alcance	Declarado	Reforzamiento
a	Variable	0	Línea 2	Línea 11
a	Variable	1	Línea 4	Línea 6
b	Variable	0	Línea 2	Línea 7
Mann	Programa	0	Línea 1	
P	Procedimiento	0	Línea 3	Línea 11
x	Próximo	1	Línea 3	

Figura 27. Estructura de lista ordenada

4.6.7.3. TABLA DE SÍMBOLOS COMO ÁRBOLES BINARIOS

Los árboles binarios combinan la rapidez en la búsqueda de un arreglo ordenado con la facilidad de inserción de una lista enlazada.

Los árboles binarios son eficientes en términos de copia, ya que consumen una cantidad de copia proporcional al número de nodos.

sin embargo, si no permite que el árbol pierda su equilibrio, la búsqueda puede degradarse a una búsqueda lineal.

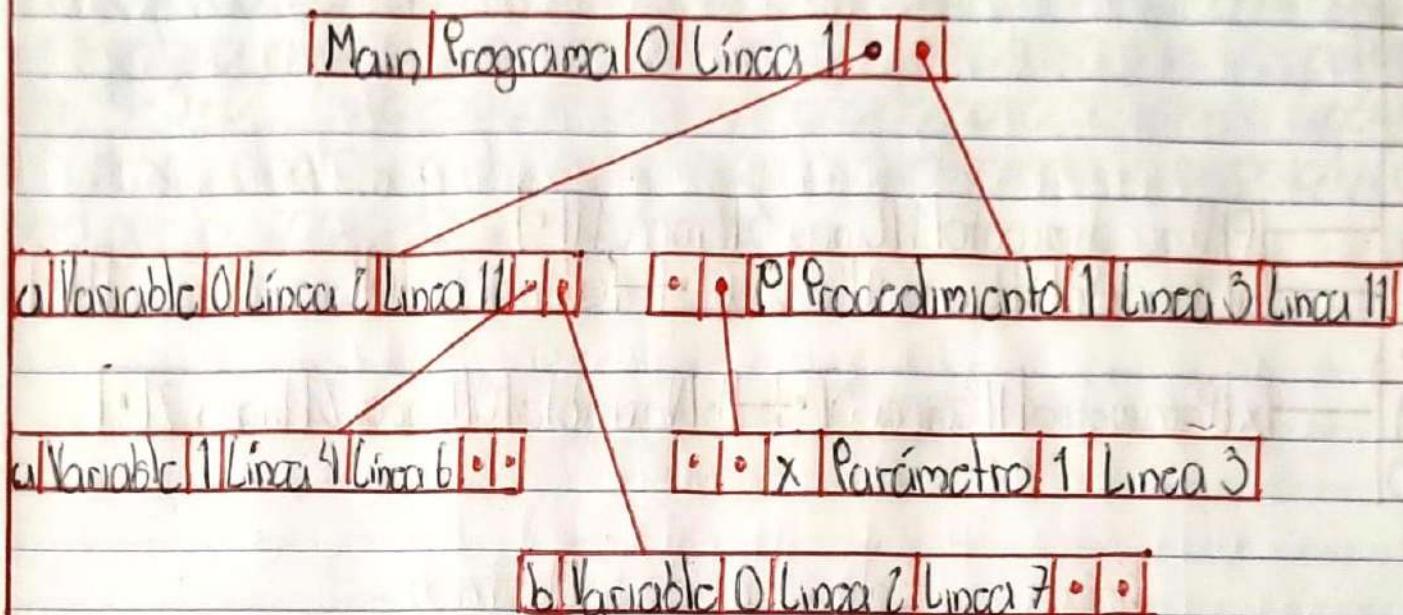


Figura 28. Estructura de árbol binario

4.6.7.4. TABLAS DE SÍMBOLOS COMO TABLAS HASH

Para lograr eficiencia, las tablas hash son el mejor método. La mayoría de los compiladores de alta calidad en producción utilizan el hashing para su estructura de tabla de tabla de símbolos.

Se han desarrollado muchas funciones de dispersión para nombrarlos. Estas funciones consisten en encontrar un valor numérico para el identificador, posiblemente alguna combinación del código ASCII como número o incluso su código de bits, y luego aplicar alguna de las técnicas utilizadas para la dispersión de números.

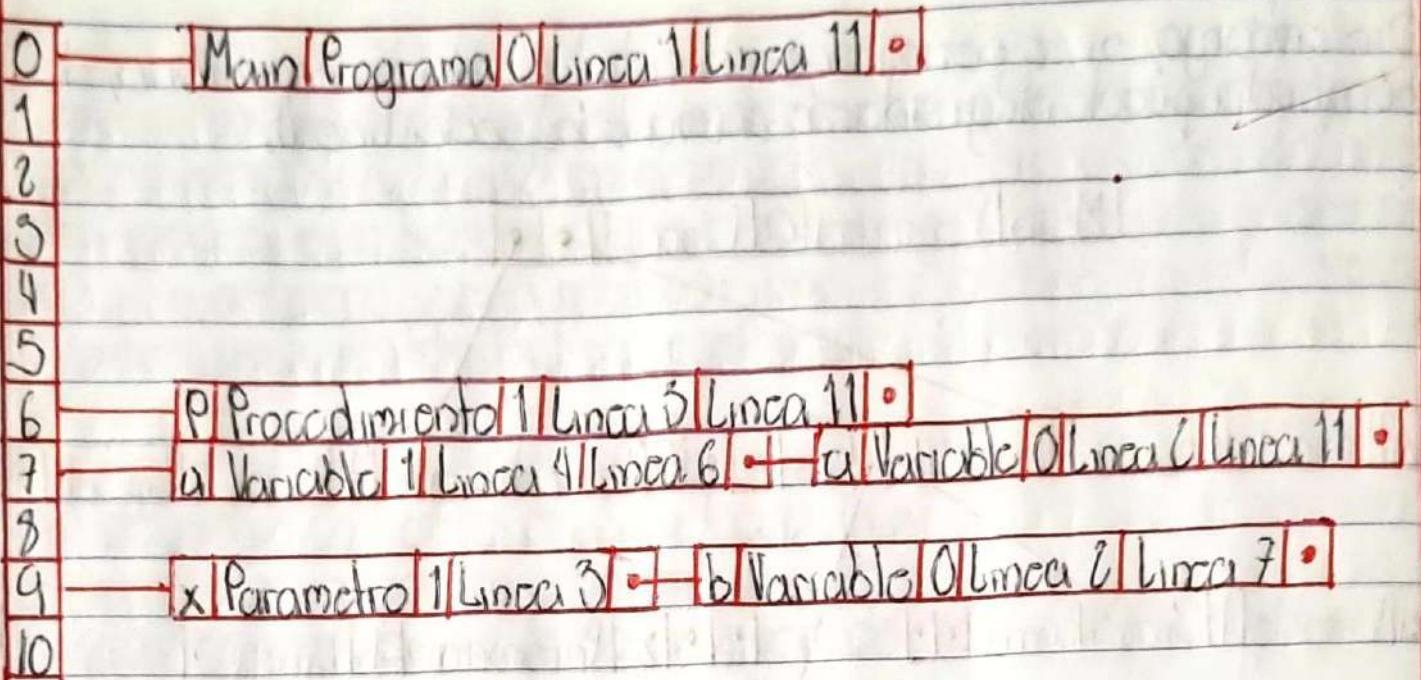


Figura 29. Estructura de tabla hash

5. La función de dispersión distribuye los nombres de manera uniforme, entonces la dispersión es muy efectiva, con un tiempo promedio de $O(1)$. El peor caso, en el que todos los nombres apuntan al mismo número, tiene un tiempo de $O(n)$. Sin embargo, esto ocurre con poca probabilidad.

Generalmente, se necesita memoria suficiente para una estructura de tabla de símbolos basada en dispersión, por lo que no es tan eficiente en términos de espacio.

4.7 Manejo de errores semánticos

El correcto manejo de errores semánticos representa un aspecto crítico en el proceso de traducción de código fuente a código ejecutable, siendo ésta la última oportunidad del compilador de descartar programas incorrectos.

Mientras que los errores léxicos y sintácticos se refieren principalmente a problemas de estructura y formato del código, los errores semánticos abordan problemas más profundos relacionados con el significado y la lógica subyacente en un programa. En esta sección, se explicará qué son dichos errores y cómo se gestionan o manejan en el proceso de compilación.

- 4.7.1 Definición

Como se mencionó anteriormente dichos errores de semántica son desviaciones en un programa que van más allá de las reglas de sintaxis y léxico.

Estos errores se presentan cuando un programa, aunque se encuentre escrito de forma correcta desde el punto de vista de la gramática y la estructura, este viola reglas o restricciones específicas impuestas por el lenguaje de programación en turno.

Dichos errores están directamente relacionados con la interpretación y significado del programa, esta característica los convierte en errores mucho más sutiles tanto de detectar como de corregir.

Estos se pueden dar de distintas maneras como lo podría ser la manipulación incorrecta de tipos de datos, el acceso a variables no inicializadas o la ejecución de operaciones incompatibles. A continuación se presentan algunos de los errores semánticos más comunes.

- * **Tipos de datos incompatibles:** Se da cuando se intenta realizar una operación entre dos variables o expresiones de tipos incompatibles, como lo podría ser sumar una cadena de texto con un número o multiplicar un booleano.
- * **Acceso a variables no declaradas:** Es intentar utilizar una variable que no ha sido declarada previamente en el alcance actual del programa (por ejemplo el intentar acceder a una variable local y que no esté especificada como global de una clase diferente en la que se encuentre donde se está haciendo referencia).
- * **Índices fuera de límites:** En el contexto de arreglos o listas, el intentar acceder a elementos fuera de los límites definidos para la estructura de datos. Por ejemplo intentar añadir un 4to elemento en un arreglo definido para exactamente tres elementos nos produciría dicho error. Esto también puede aplicar cuando se intenta modificar el valor de una variable estática.
- * **Conflicto de nombres:** Declarar dos variables o funciones con el mismo nombre en el mismo ámbito generará ambigüedad. Se genera aun mas ambigüedad cuando dicha variable está definida con el mismo nombre y con diferentes tipos de datos.

*Uso incorrecto de funciones o métodos: Es utilizar una función con el número incorrecto de argumentos o tipos de argumentos incompatibles.

Por ejemplo el enviar solo dos argumentos cuando se esperaban tres, o enviar una cadena cuando se esperaba un entero.

- 4.7.2 Formas de manejo

Para manejar los errores dichos anteriormente el compilador puede realizar ciertas acciones para ciertos tipos de errores, mientras que en otros por ser cuestión de lo que pida el programador no podrá hacer más que imprimir un error en pantalla.

Algunos de estos métodos se listan a continuación:

- Si el error de "Identificador no declarado" es encontrado, entonces el compilador para recuperarse puede crear una entrada a la tabla de símbolos para el correspondiente identificador es realizada.
- Si dos tipos de operandos son incompatibles pero pueden tener compatibilidad con una conversión, el compilador puede optar por hacer una conversión de tipos en automático. Donde más sentido tiene es por ejemplo a sumar dos números uno de tipo decimal con uno de tipo entero.

Los anteriores son algunas de las formas en las que se puede manejar los errores, no son demasiados ya que el implementar cambios o ajustes muy drásticos pueden

resultar en mayores problemas que dar a conocer que existe tal error.

Ya que por ejemplo si se da un conflicto de nombres y el compilador cambia el nombre por otro similar y dicha variable se vuelve a utilizar en un punto posterior se puede incurrir en más ambigüedad, o también cuando se tiene un uso no adecuado de parámetros en una función el compilador no puede simplemente poner un parámetro arbitrario a menos que se especifique previamente que puede recibir un parámetro nulo de este tipo.

Dicho lo anterior en cuanto errores semánticos se refiere, si el error es muy superficial o concreto el compilador puede realizar el pequeño ajuste pero cuando pueda incurrir en interferir directamente con la lógica del programa es más seguro simplemente avisar al programador que existe dicho error y en la práctica esta segunda aproximación es la más utilizada siendo la primera solo en casos muy concretos.

Conclusiones

En conclusión, esta monografía nos ha brindado una visión general sobre temas fundamentales relacionados con el análisis sintáctico y semántico en la compilación de programas. Hemos explorado conceptos clave que abarcan desde la estructura de árboles de expresiones hasta la gestión de errores semánticos. También logramos comprender la estrecha relación que existe entre el análisis semántico y sintáctico, y cómo juntos desempeñan un papel crucial en la interpretación y traducción de lenguajes de programación.

Estos conocimientos son esenciales para comprender cómo los lenguajes de programación son procesados y traducidos en código ejecutable.

La aplicación de estos conceptos es crucial para garantizar la coherencia, la seguridad y la eficiencia en la ejecución de programas informáticos, y son de relevancia tanto para

[Handwritten signatures]

desarrolladores de compiladores como para
ingenieros de software en la creación
de programas de alta calidad.

Bibliografía

Unidad I Análisis semántico. (n.d.). Studocu. Retrieved October 25, 2023, from

<https://www.studocu.com/es-mx/document/instituto-tecnologico-y-de-estudios-superiores-de-monterrey/lenguajes-y-traductores/unidad-i-analisis-semanitico/10903153>

(N.d.-b). Tecnm.Mx. Retrieved October 25, 2023, from

https://enlinea.zacatecas.tecnm.mx/pluginfile.php/10830/mod_resource/content/2/1.1%20Arboles%20de%20Expresion.pdf

Compiler Design - Semantic Analysis. (n.d.). Tutorialspoint.com. Retrieved

October 25, 2023, from

https://www.tutorialspoint.com/compiler_design/compiler_design_semantic_analysis.htm

Semantic analysis in compiler design. (2019, October 14). GeeksforGeeks.

<https://www.geeksforgeeks.org/semantic-analysis-in-compiler-design/>

(N.d.). Studocu.com. Retrieved October 25, 2023, from

<https://www.studocu.com/es-mx/document/instituto-tecnologico-de-oaxaca/lenguajes-y-automatas-ii/14-pila-semantica-en-analizador-sintactico/8456952>

Addison Wesley - Compiladores. Principios, Técnicas y Herramientas.pdf. (n.d.).

Google Docs. Retrieved October 25, 2023, from

[https://drive.google.com/file/d/oB_u1rzdqYCnzWFpqN3h1ZXZTRoE/view
?resourcekey=o-TNuuf8q7U5TTKLM41YsSvg](https://drive.google.com/file/d/oB_u1rzdqYCnzWFpqN3h1ZXZTRoE/view?resourcekey=o-TNuuf8q7U5TTKLM41YsSvg)

The-theory-and-practice-of-compiler-writing.Pdf. (n.d.). Google Docs. Retrieved October 25, 2023, from

[https://drive.google.com/file/d/1xxvoMFooFXPDr5hpWXgT6ikQ5sBwr2rU
/view?usp=drive_link](https://drive.google.com/file/d/1xxvoMFooFXPDr5hpWXgT6ikQ5sBwr2rU/view?usp=drive_link)

crafting-a-compiler_compress.pdf. (n.d.). Google Docs. Retrieved October 25, 2023, from

[https://drive.google.com/file/d/1vZ2oO2q_9XAwJwzLNLvkA14byrbbewgL
/view?usp=sharing](https://drive.google.com/file/d/1vZ2oO2q_9XAwJwzLNLvkA14byrbbewgL/view?usp=sharing)

5.0 Semantic Analysis Symbol Tables. (s/f). Wpi.edu. Recuperado el 18 de octubre de 2023, de

<https://web.cs.wpi.edu/~kal/courses/compilers/module5/myst.html>

(S/f). Informatica.uv.es. Recuperado el 18 de octubre de 2023, de
<http://informatica.uv.es/docencia/iiguia/asignatu/2000/PL/2005/tema6.pdf>

Aho, A. V., Sethi, R., Ullman, J. D., Principios, C., Addison-, W., & Tremblay, P. J. G. (s/f). Tecnm.mx. Recuperado el 18 de octubre de 2023, de

<https://hopelchen.tecnm.mx/principal/sylabus/fpdb/recursos/r97339.PDF>

Java a tope: Traductores y compiladores con Lex/Yacc, JFlex/Cup y JavaCC.

(s/f). Uma.es. Recuperado el 18 de octubre de 2023, de

https://ocw.uma.es/pluginfile.php/1027/mod_resource/content/o/Capitulo_6.pdf

(S/f). Cartagena99.com. Recuperado el 18 de octubre de 2023, de

https://www.cartagena99.com/recursos/alumnos/apuntes/ININF2_M4_U5_T2.pdf

Nº, C., Adolfo, A., Fuente, J., Manuel, J., Lovelle, C., Ortín, F., Raúl, S., Castanedo, I., Cándida, M., Díez, L., Emilio, J., & Gayo, L. (s/f). *Tablas de Símbolos de Procesadores de Lenguaje*. Uniovi.es. Recuperado el 18 de octubre de 2023, de

http://dio02.edv.uniovi.es/~cueva/publicaciones/monografias/41_TablasDeSimbolos.pdf

Tremblay, J. P., & Sorenson, P. G. (1986). *Theory and practice of compiler writing*. McGraw-Hill Education (ISE Editions).

Rouse, M. (2022). Parser. Disponible en: <https://www.techopedia.com/definition/3854/parser>

Rodríguez, J., E. (2022). Acciones semánticas de un analizador sintáctico. pdfslide.tips. <https://pdfslide.tips/documents/acciones-semanticas-de-un-analizador-sintactico.html?page=1>

Jiménez, V. (2018). Procesadores de Lenguaje: Análisis Semántico. Disponible en: <https://www3.uji.es/~vjimenez/AULASVIRTUALES/PL-0910/T4-SEMANTICO/semantico.apun.pdf>

Silvino, A. (s. f.). UNIDAD VI.-Análisis semántico. <http://cursocompiladoresuaeh.blogspot.com/2010/11/unidad-vi-analisis-semantico.html>

De expresiones, A. (n.d.). *Unidad I: Análisis semántico*. Itpn.Mx. Retrieved October 25, 2023, from
<http://itpn.mx/recursosisc/7semestre/leguajesyautomatas2/Unidad%20I.pdf>

1.4.- Pila semántica en un analizador sintáctico - Lenguajes y autómatas II. (s. f.). Lenguajes y Autómatas II. <https://5e344735705b1.site123.me/unidad-i-%C3%81nalisis-sem%C3%A1ntico/14-pila-sem%C3%A1ntica-en-un-analizador-sint%C3%A1ctico>

Simic, M. (2022, August 12). *From postfix expressions to expression trees*.

Baeldung on Computer Science. <https://www.baeldung.com/cs/postfix-expressions-and-expression-trees>

Expression tree. (2015, August 13). GeeksforGeeks.

<https://www.geeksforgeeks.org/expression-tree/>

Cooper, K., & Torczon, L. (n.d.). *Praise for Engineering a Compiler*. Acm.org.

Retrieved October 25, 2023, from
<https://dl.acm.org/doi/pdf/10.5555/2737838>

Eddy. (2023, May 21). *Árbol de expresión - Definición y explicación*. TechEdu.
<https://techlib.net/techedu/arbol-de-expresion/>

KathleenDollard. (n.d.). *Árboles de expresión - Visual Basic*. Microsoft.com.
Retrieved October 25, 2023, from <https://learn.microsoft.com/es-es/dotnet/visual-basic/programming-guide/concepts/expression-trees/>

Expression tree in data structure. (n.d.). Www.javatpoint.com. Retrieved October 25, 2023, from <https://www.javatpoint.com/expression-tree-in-data-structure>

Compiler design-Semantic Phase errors. (2021, October 6). I2tutorials.

<https://www.i2tutorials.com/compiler-design-tutorial/compiler-design-semantic-phase-errors/>

Lumunge, E. (2022, January 7). *Semantic analysis in compiler design.* OpenGenus IQ: Computing Expertise & Legacy. <https://iq.opengenus.org/semantic-analysis-in-compiler-design/>

Tostatronic, V. T. [@VideoTutorialesTostatronic]. (2019, July 7). *Conversiones de infija a prefija y postfija.* Youtube.

<https://www.youtube.com/watch?v=ps1YAnGv8BA>

Error detection and recovery in compiler. (2018, April 24). GeeksforGeeks.

<https://www.geeksforgeeks.org/error-detection-recovery-compiler/>

Compiler design-Semantic Phase errors. (2021, October 6). I2tutorials.

<https://www.i2tutorials.com/compiler-design-tutorial/compiler-design-semantic-phase-errors/>

Coding ninjas studio. (n.d.). Codingninjas.com. Retrieved October 25, 2023, from <https://www.codingninjas.com/studio/library/error-recovery-and-handling>

(2007). Stanford.edu. <https://suif.stanford.edu/dragonbook/lecture-notes/Stanford-CS143/14-Semantic-Analysis.pdf>

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTORES:**

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López

CARRERA	NOMBRE DE LA ASIGNATURA
INGENIERIA EN SISTEMAS COMPUTACIONALES	LENGUAJES Y AUTÓMATAS II

PRACTICA NO.	NOMBRE DE LA PRACTICA	FECHA DE ENTREGA
1	ANÁLISIS DEL VIDEO CON TÍTULO “¿QUÉ PASA DENTRO DE LA MENTE DE UN PROCRASTINADOR?”	19/10/2023

1	INTRODUCCION
Por definición la procrastinación es la acción de postergar actividades o situaciones que deben atenderse, sustituyéndolas más irrelevantes o agradables. Este es un término de uso un tanto reciente ya que no hace pocos años se empleó su uso para referirse a aquellas personas que postergan sus actividades u obligaciones remplazándolas o delegándolas a actividades más del agrado de dicha persona. Lo que esto termina provocando es retrasos considerables en las actividades que se tenían que realizar en dicho momento, en el desarrollo del video se detallara que es lo que fluye dentro de las mentes de estas personas que practican continuamente la procrastinación.	

2	OBJETIVO
Mediante la visualización del video y posterior análisis de este video se espera el comprender e informarnos acerca de la procrastinación y sus efectos que provoca realizar dicha acción tanto en nuestra mente como también influyendo en nuestro estilo de vida y posteriormente en nuestra disposición a las actividades que percibimos como no tan placenteras.	

3	FUNDAMENTO
La procrastinación es un efecto o a veces también planteado como vicio que puede ocurrir tarde o temprano en el lapso de nuestra vida, a veces con mayor o menor medida ya sea con menor o mayor influencia negativa. Claramente el que suceda de vez en cuando y en situaciones específicas no realizará grandes daños a nuestra vida o a nuestras actividades ya sean personales, sociales o académicas pero el problema radica en el aspecto que el procrastinar se vuelve una tendencia o un vicio que cuando empezamos a practicarlo de manera voluntaria o involuntaria después de un tiempo esto se volverá de forma descontrolada y terminaremos con muchos problemas y procrastinando aun cuando no queremos.	



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López

4

MATERIALES NECESARIOS

Hardware:

- CPU: Intel i3 8va generación 2.4 GHz
- RAM: 8GB
- SDD: 128 GB

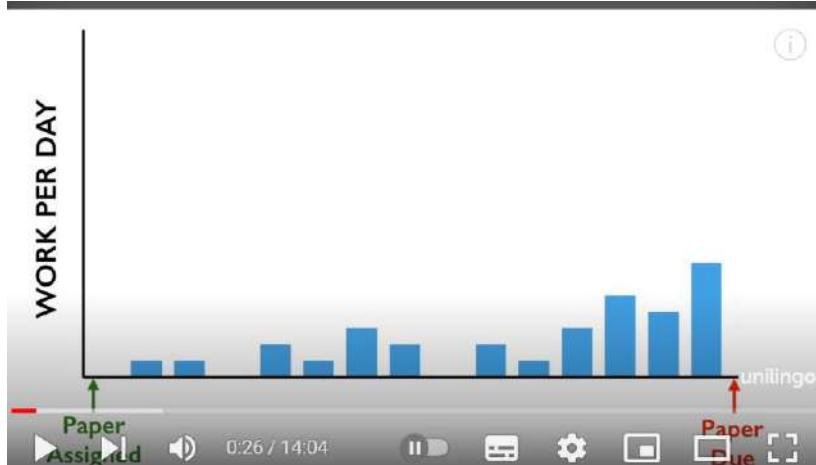
Software:

- SO: Windows 10 o alguna distribución de Linux.

Algún navegador de internet

5

DESARROLLO



Primero se comenzó hablando sobre que trabajar poco a poco permite tener grandes avances a la larga, esto es un trabajo continuo pero no es la situación normal ya que el flujo de trabajo en estudiantes es dejar todo al último.

Se planteo además el interpretar el trabajo de una tesis por ejemplo como una escalera de esta forma el avance será poco a poco y no se sentirá tan agresivo el trabajo.



TECNOLÓGICO
NACIONAL DE MÉXICO

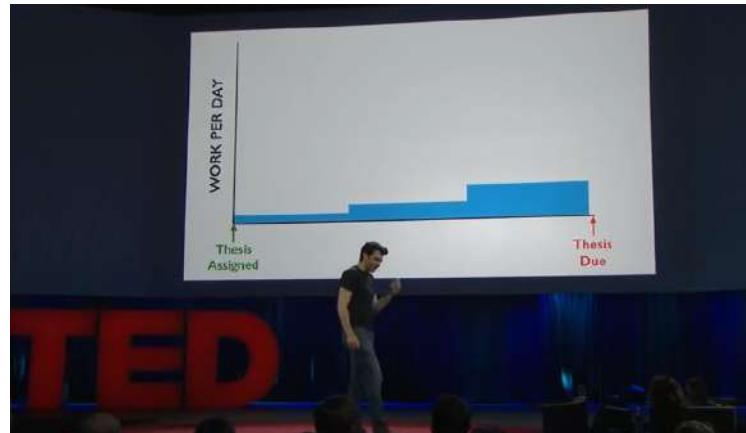
INSTITUTO TECNOLÓGICO DE CELAYA



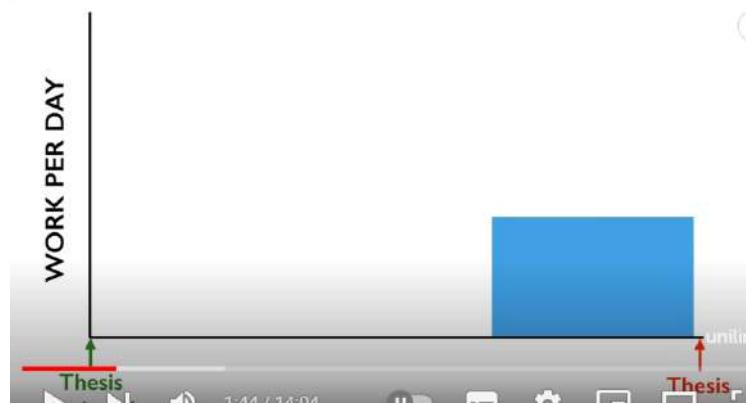
PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López



Pero esto de dividir el trabajo como una escalera no resultó como lo planeado y los primeros meses no se realizó nada, pero se propuso el adelantarse después de esto pero esto tampoco pasó, así que al final se terminó con la siguiente gráfica.



Después de eso el presentador habló de que tenía que realizar 90 páginas en solo algunas noches lo cual significaba el trabajar 24/7 sin parar y sucedió lo esperado le llamaron diciendo que era una de las peores tesis evaluadas.

Ahora se comenzó viendo cuáles son las diferencias entre la mente de un no procrastinador:



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

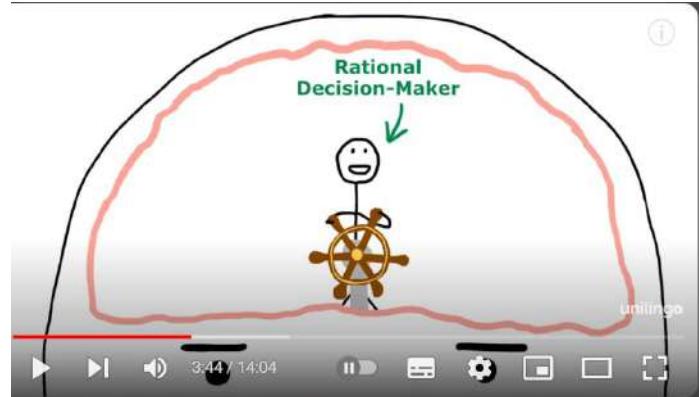


PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

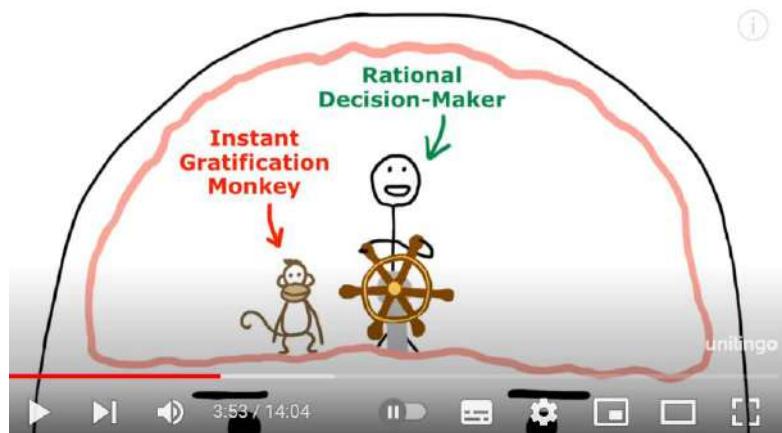
AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López

[Handwritten signatures]



Y un procrastinador:



Vemos que aparte del marinero se tiene un mono de gratificación instantánea que va limitando a la persona de decisiones racionales tomando el mando del timón y desviando el barco cuando el capitán quiere realizar algo productivo por ejemplo realizar una tarea y lo desvía por ejemplo a ver una película. Y esto comienza como una única acción pero termina como todo un círculo vicioso.



TECNOLÓGICO
NACIONAL DE MÉXICO

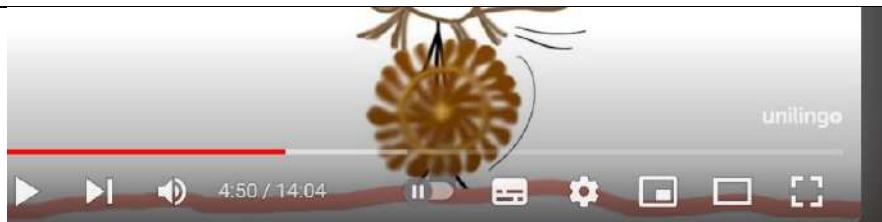
INSTITUTO TECNOLÓGICO DE CELAYA



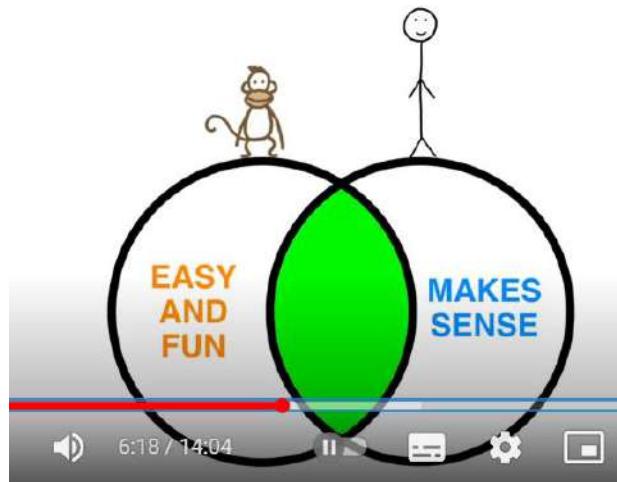
PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López



Y así es como el mono asume totalmente el mando, aunque pareciera que nadie quiere tenerlo al mando de la operación en realidad muchas veces se hace cargo de la situación ya que responde a uno de nuestros impulsos primitivos así como también un impulso de la naturaleza pero como seres racionales y civilizados debemos aprender a como manejar a este monito. Esto no significa quitar totalmente al mono sino controlarlo y dejarlo salir como por ejemplo en una reunión familiar o en el tiempo libre. Estos momentos en donde ambos están en armonía.



Pero esto es un ambiente de conflicto continuo en donde se pasa demasiado tiempo en el círculo del mono, pero también se puede caer demasiado en el otro círculo.

Pero el procrastinador no está solo lo acompaña el monstruo del pánico.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López



7:30 / 14:04



Este se despierta cuando se encuentra en peligro y es el único miedo del mono, ya que como le sucedió al presentador este acepto ser un orador para una plática de TED había planificado pero esta planificación se iba por las ramas o mejor dicho por las “lianás” y ya después de un tiempo determinado despertó dicho monstruo de pánico pues la plática se aproximaba.



9:14 / 14:04



Y después de eso todo el sistema entro en caos tanto la persona como el mono con esto el mono huyo y el capitán de las decisiones racionales pudo finalmente comenzar a trabajar en la tarea pendiente pero ahora constantemente perseguido y presionado por el monstruo.

El monstruo corresponde a todos esos pensamientos de culpa con los que contamos por ejemplo el decir “Tuve 10 días para realizar este trabajo pero pase 8 días haciendo puras tonterías” y este también nos fuerza a trabajar bajo presión y por largas sesiones continuas.

No es una situación muy agradable pero es un sistema que regularmente funciona pero lo ideal sería que no fuera de esta manera.

La dilación se presenta en varias formas ya sea por plazos de tiempo como por ejemplo el tiempo para entregar una tarea o trabajo pero también existe el caso en el dónde no se tienen plazos como ambiciones personales como aprender un idioma o ir al gimnasio, no existe un plazo en estas situaciones no existe el monstruo del pánico pues no hay un peligro en el horizonte próximo, siendo la dilación a largo plazo que no se nota muy a corto plazo pero tiene efectos muy notorios a largo plazo.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

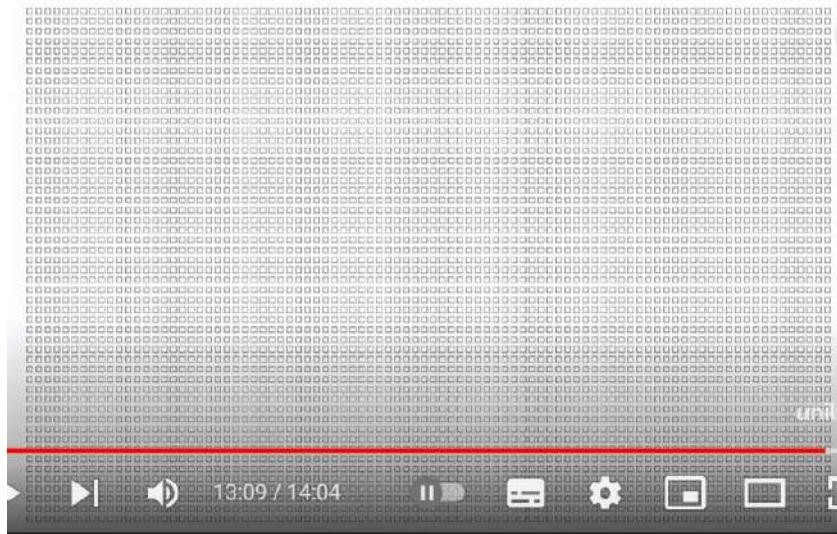
AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López



Pero esta dilación a largo plazo es la principal culpable de desesperación y lamentos de cosas que pudieron haber pasado o sueños que pudieron haberse cumplido.

Al final se llegó a la reflexión que no existe tal cosa como no procrastinadores pues todos lo somos en mayor o menor medida. Ahora se presenta un calendario de vida.



En este existe una caja por cada semana de una vida de 90 años que parecen muchas pero en realidad no son muchas, sobre esto podemos ver que cosas postergamos en la vida. Igual también es muy importante tener en mente el mono y sus impactos sobre dicha caja.

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTORES:**

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López

6 BÍTACORA DE INCIDENCIAS		
Fecha	Problema encontrado	Solución
19/10/23	Tiempo limitado a la realización de la actividad ya que se tenía una actividad de clase	Se pudo terminar de forma rápida la actividad planteada dando margen a un espacio de tiempo para la realización de la presente actividad.

7 CONCLUSIÓN	
La procrastinación es una espada de doble filo pues nos da satisfacción y felicidad al momento ya que nos permite realizar las cosas que queremos en el momento que queramos sin importar que tengamos pendientes de por medio. Pero esta felicidad viene a un costo, costo que se pagará una vez que sea demasiado tarde para empezar a realizar lo pendiente ya que se podría decir que el procrastinar significa el tomar prestado un lapso, ya después tendrás que realizar el pago el préstamo pero el prestamista te pedirá a cambio estrés, frustración y desesperación y también en gran medida calidad de lo que ibas a realizar. En realidad la persona no procrastinadora no existe y es muy difícil el no sucumbir en las garras de este pero lo que nos debemos plantear realmente es el saber controlar estos impulsos y dejarlos salir en momentos muy concisos en donde el liberarlo no hará daño pues también es dañino el dejarlo reprimirlo siempre.	

8 REFERENCIAS	
Unilingo. (2017b, junio 15). ¿Qué pasa dentro de la mente de un procrastinador? (Doblado al español) [Vídeo]. YouTube. https://www.youtube.com/watch?v=PG6oFK0a1NA	



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Alma Gabriela Ponce Morales
Anthony Gómez Cabañas
Cristhian Alberto Ortega Hernández
Miguel Ángel Ruiz López

Guerri, M. (2023, 18 mayo). Procrastinación: qué es y cómo superarla (2023).

PsicoActiva.com: Psicología, test y ocio Inteligente.

<https://www.psicoactiva.com/blog/la-procrastinacion/>

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTOR:**

Cristhian Alberto Ortega Hernández

CARRERA	NOMBRE DE LA ASIGNATURA
INGENIERIA EN SISTEMAS COMPUTACIONALES	LENGUAJES Y AUTÓMATAS II

ACTIVIDAD NO.	NOMBRE DE LA ACTIVIDAD	FECHA DE ENTREGA
2	¿Qué pasa dentro de la mente de un procrastinador?	19/10/2023

1**INTRODUCCION**

En esta actividad se realizará un resumen sobre el video titulado “¿Qué pasa dentro de la mente de un procrastinador?” Con este video se podrá comprender un poco más lo que es la procrastinación y las consecuencias que esta puede traer.

2**OBJETIVO**

- Realizar un resumen sobre el video.
- Analizar con profundidad y detalle el contenido del video.
- Realizar una reflexión sobre el contenido y sobre mi propia persona.

3**FUNDAMENTO**

La procrastinación es la acción o hábito de retrasar actividades que deben atenderse para realizar otras actividades que son más agradables o irrelevantes. Esta forma de actuar, si se convierte en un patrón, puede interferir de manera significativa en aspectos personales, laborales, sociales, etc., y esto puede traer consecuencias al bienestar emocional, pues también trae consigo emociones como ansiedad, estrés, rabia, culpa, entre muchas otras.

Algunas de las causas de la procrastinación pueden ser:

- Baja autoestima
- Falta de confianza
- Exceso de planificación
- Dificultades de autorregulación
- Sentimiento de insatisfacción
- Perfeccionismo

4**MATERIALES NECESARIOS**

- Computadora
- Internet
- Navegador Web

5**DESARROLLO**



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

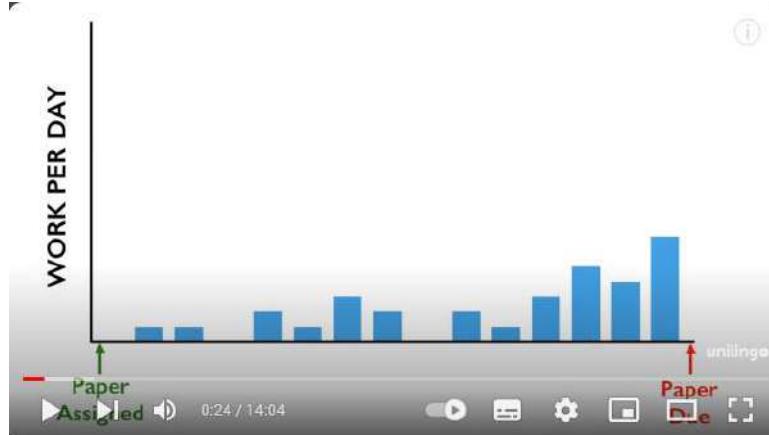


**PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II**

AUTOR:

Cristhian Alberto Ortega Hernández

El ponente comienza explicando que el realizaba muchos escritos en la universidad y muestra una gráfica donde se puede observar cómo distribuía el trabajo desde que se le asignaba la actividad hasta la fecha de entrega.



Lo ideal es que el trabajo sea distribuido entre la semana: algunos días se realiza algún avance significativo y otros un avance más pequeño. La realidad es muy diferente, generalmente se deja todo el trabajo cuando ya está cerca la fecha de entrega.



Muestra otro ejemplo relacionado a la elaboración de una tesis, y menciona que su forma de planificar su flujo de trabajo durante el año fue iniciando poco, a la mitad realizaba un avance más grande, y ya cerca de la fecha de entrega dedicaría mucho más tiempo.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

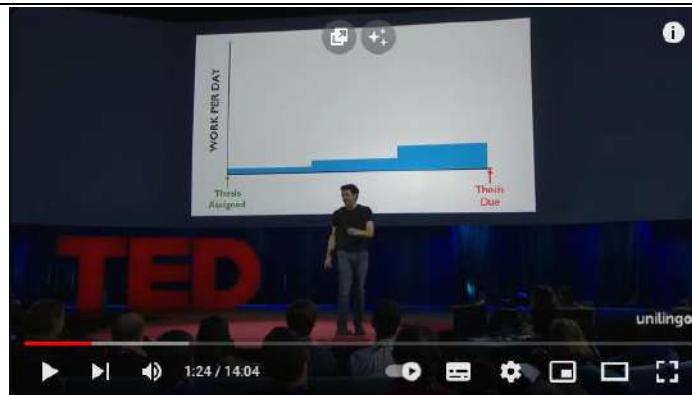


Alfredo M. Cárdenas
Hernández

PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

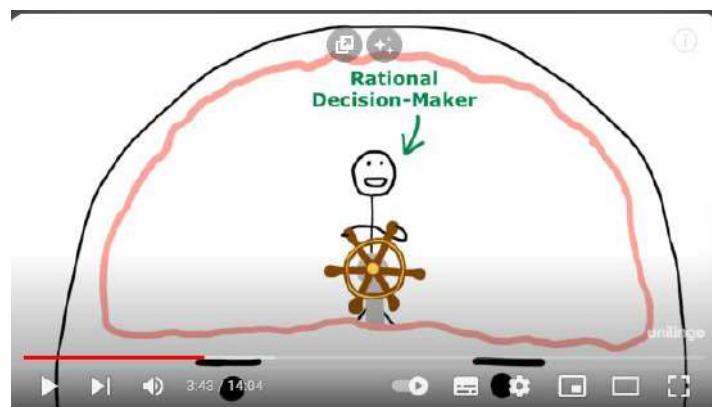
AUTOR:

Cristhian Alberto Ortega Hernández

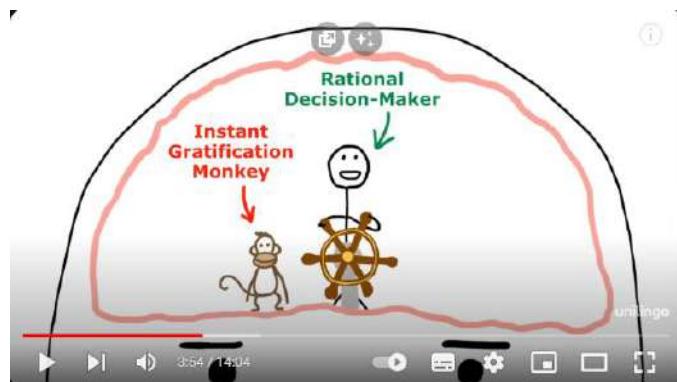


A pesar de que tenía un plan ya elaborado, menciona que no lo siguió; fueron pasando los meses, semana y días hasta que sólo le quedaban tres días para elaborar su escrito.

Cuando termina de contar su experiencia, comienza a hablar sobre su hipótesis sobre las diferencias entre el cerebro de una persona procrastinadora y una no procrastinadora. Para exemplificar el cerebro de una persona no procrastinadora, muestra el siguiente dibujo:



Y luego muestra el de una persona procrastinadora:



Menciona que las similitudes que hay entre los dos cerebros es que los dos tienen una persona que



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



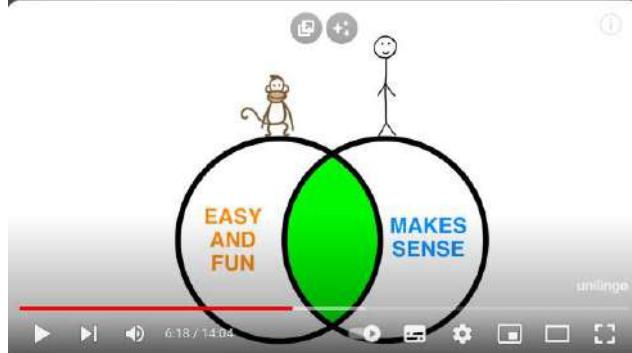
**PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II**

AUTOR:

Cristhian Alberto Ortega Hernández

realiza decisiones racionales, pero el de la persona procrastinadora tiene, además, a un mono de la gratificación instantánea, quien se encarga de tomar las decisiones que llevan a la persona procrastinadora a realizar actividades que sean más de su agrado.

Posteriormente menciona que, para un animal, realizar actividades fáciles y divertidas es sinónimo de éxito, pero nosotros como humanos necesitamos hacer otras actividades que no sean fáciles y divertidas, pero sí necesarias. Muchas veces se puede encontrar un equilibrio entre no hacer nada y también trabajar de forma activa.



El procrastinador prefiere pasar más tiempo en la zona de actividades fáciles y divertidas, el ponente lo llama "El patio oscuro". Ahí es donde ocurren las actividades de ocio en momentos donde no deberían ocurrir y es por eso que la diversión que ahí se obtiene no es "diversión merecida", y generalmente llegan sentimientos como miedo, temor, ansiedad.

Para poder salir de ese lugar, se menciona que el procrastinador tiene un "ángel guardián" llamado "El monstruo de pánico".



Este monstruo está inactivo la mayoría del tiempo, pero se vuelve activo cuando una fecha de entrega está cerca o puede ocurrir alguna otra consecuencia desagradable.

Cuenta otra experiencia que tuvo cuando recién lo habían invitado a realizar la plática. Menciona que su parte racional le decía que, desde el momento en que lo invitaron, debía comenzar a trabajar en preparar todo lo necesario, pero el mono le decía que no, que faltaba mucho tiempo. El ponente decidió hacerle caso al mono y dejó de lado la preparación de la práctica por mucho tiempo hasta que

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTOR:**

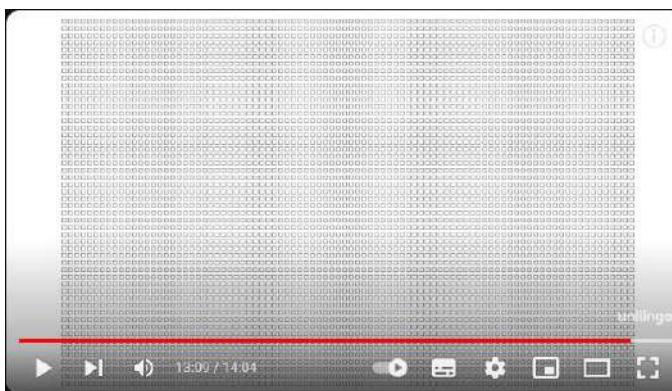
Cristhian Alberto Ortega Hernández

faltaba 1 mes aproximadamente, ahí fue cuando decidió empezar a escribir un artículo que hablaba sobre el tema de su plática. Gracias a este artículo, recibió muchos correos y comentarios de gente que pasaba por la misma situación pero que se encontraba muy frustrada.

Aquí es donde menciona que existen actividades que no tienen un plazo definido como algunas otras actividades, y es esa falta de plazo lo que hace que nunca tengan la importancia debida esas actividades y nunca se despierte ese monstruo de pánico. Esta dilación a largo plazo es la que genera frustración en la vida de las personas, pues hace que se sientan como espectadores de sus propias vidas al no poder tomar la acción para poder comenzar algo.

El ponente llega a la conclusión de que todos somos procrastinadores, pero unos son más afectados que otros.

Por último, menciona el calendario de vida, un calendario diseñado para una vida de 90 años. Con este calendario nos podemos dar una idea de que es lo que estamos postergando en nuestras vidas y así ser más conscientes sobre qué aspectos deberíamos de mejorar.



6

BÍTACORA DE INCIDENCIAS

Fecha	Problema encontrado	Solución
10/19/2023	Falta de tiempo	La actividad fue realizada entre clases.

7

CONCLUSIÓN

El video me pareció muy interesante, ya que explicó el concepto de la procrastinación de una manera clara y entretenida. Se describieron situaciones que considero que todas las personas hemos experimentado, pero también incitó a la reflexión sobre las consecuencias que se pueden tener.

El video me ayudó a reflexionar sobre como afecta la procrastinación en mi vida, y me he dado cuenta que es un aspecto que debo de mejorar.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



[Handwritten signatures]

**PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II**

AUTOR:

Cristhian Alberto Ortega Hernández

8

REFERENCIAS

Unilingo [@unilingo]. (2017, junio 14). *¿Qué pasa dentro de la mente de un procrastinador? (doblado al español)*. Youtube.

<https://www.youtube.com/watch?v=PG6oFKoa1NA>

de Lauracoach, A. (2019, octubre 1). *Procrastinación. Qué es y cómo se vence*. Blog de Psicología del Colegio Oficial de la Psicología de Madrid.

<https://www.copmadrid.org/wp/procrastinacion-que-es-y-como-se-vence/>

¿En qué consiste la procrastinación? (s/f). Psiquiatriapsicologia-dexus.com.

Recuperado el 19 de octubre de 2023, de <https://www.psiquiatriapsicologia-dexus.com/es/unidades.cfm/ID/9734/ESP/-que-consiste-procrastinacion-.htm>

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTOR:**

Alma Gabriela Ponce Morales

CARRERA	NOMBRE DE LA ASIGNATURA
INGENIERIA EN SISTEMAS COMPUTACIONALES	LENGUAJES Y AUTÓMATAS II

TAREA INDIVIDUAL NO.	NOMBRE DE LA TAREA	FECHA DE ENTREGA
1	¿Qué pasa dentro de la mente de un procrastinador?	10 de octubre de 2023

1	INTRODUCCION
La procrastinación es un fenómeno común que afecta a muchas personas en su vida diaria. ¿Qué sucede dentro de la mente de un procrastinador? Esta pregunta intrigante es el enfoque central de nuestro video, donde exploraremos las complejidades de la postergación y sus efectos en la productividad y el bienestar. A medida que nos adentramos en las profundidades de la mente de un procrastinador, desvelaremos los pensamientos intrusivos, las luchas con la planificación y el aterrador "patio oscuro". En este documento se van a abordar los temas tratados en el video para comprender mejor este fenómeno y descubrir estrategias para superar la procrastinación.	

2	OBJETIVO
El objetivo de este estudio es investigar y comprender en profundidad el fenómeno de la procrastinación, centrándonos en lo que sucede dentro de la mente de un procrastinador.	

3	FUNDAMENTO
La procrastinación es un desafío común que enfrentan muchas personas en la actualidad. Postergar tareas importantes puede tener consecuencias significativas en términos de productividad, calidad de trabajo y bienestar emocional. Comprender las causas y los mecanismos subyacentes de la procrastinación es esencial para abordar este problema de manera efectiva.	

Este video y posterior resumen se justifica en el interés de mejorar la calidad de vida de las personas, ayudándolas a gestionar su tiempo de manera más efectiva y a alcanzar sus objetivos personales y profesionales sin caer en la trampa de la procrastinación. Al comprender y abordar lo que sucede dentro de la mente de un procrastinador, Se ofrece una contribución significativa a la gestión del tiempo y al bienestar de quienes luchan contra este fenómeno.

4	MATERIALES NECESARIOS
<ul style="list-style-type: none">- Conexión a internet.- Editor de texto como Word.- Un navegador.	



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II

AUTOR:

Alma Gabriela Ponce Morales

5

DESARROLLO

Durante el video se ejemplifican mediante graficas las diversas maneras de trabajar para escribir un artículo. Ejemplificando con la Figura 1 una manera de repartición de trabajo óptima para mantener un flujo de trabajo constante.



Figura 1. Ejemplo sobre una repartición óptima de trabajo.

De esta manera se busca ejemplificar el cómo se puede planear una distribución y al final del día no se sigue la distribución llevando a que la carga de trabajo sea mayor para los últimos días. De manera posterior se ejemplifican los pensamientos intrusivos mediante la Figura 2. que impiden de diversas maneras seguir una planeación.



Figura 2. Caricatura sobre los pensamientos intrusivos de un procrastinador.

Este tipo de pensamientos alimentados por la búsqueda de una satisfacción inmediata llevan a que no se diminue la responsabilidad que está siendo dejada de lado, entrar en esta área el orador le

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTOR:**

Alma Gabriela Ponce Morales

da el nombre de "El patio oscuro" y pese a que las actividades puedan parecer divertidas como no están recompensadas por un trabajo constante están cargadas de un estrés mental. Pasar mucho tiempo en este lugar hace que se acerque silenciosamente el monstruo del pánico, el único personaje al que le tiene miedo el mono de la procrastinación.

Donde de manera increíble con la presencia del monstruo empiezas a ser consciente del tiempo desperdiciado y de que con una mejor organización podrías haber realizado de mejor manera las actividades necesarias.

Dejar mucho tiempo al mono en control del timón la vida puede llevar a la percepción de que no se tiene un control sobre la propia existencia y pensar que no se está realizando nada con la misma. Lo que lleva a que tareas con plazos definidos puedan ser ejecutadas con algunos errores, pero entregadas en el tiempo especificado, sin embargo, si solamente con el pánico pueden ser realizadas las actividades entonces representa un problema en tareas a largo plazo donde no hay una fecha definida decidiendo dejar de lado esos objetivos.

Lo que nos lleva a que reflexionemos sobre las tareas que se están postergando y que acciones estamos dispuestos a realizar para que alcancemos nuestros objetivos.

6**BÍTACORA DE INCIDENCIAS**

Fecha	Problema encontrado	Solución
10/19/2023	Se disponía de poco tiempo para la realización de la práctica.	Se decidió disponer de un poco de tiempo libre entre el cambio de materias.

7**CONCLUSIÓN**

En conclusión, el video proporciona una valiosa perspectiva sobre la planificación y la gestión del trabajo al ilustrar, a través de gráficos y caricaturas, la importancia de una distribución óptima de tareas para mantener un flujo de trabajo constante. Además, al abordar el concepto del "patio oscuro" y los pensamientos intrusivos, nos alerta sobre cómo la búsqueda de satisfacción inmediata puede socavar nuestra responsabilidad y llevarnos al estrés mental. La metáfora del "monstruo del pánico" nos recuerda que un control inadecuado de nuestras tareas puede llevar a la percepción de falta de dirección en la vida. Por último, nos insta a reflexionar sobre nuestras propias tendencias de procrastinación y a considerar las acciones necesarias para alcanzar nuestros objetivos. En resumen, el video nos brinda valiosas lecciones sobre la gestión del tiempo y la importancia de la planificación en nuestras vidas.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



A. Gabriela Ponce Morales

**PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II**

AUTOR:

Alma Gabriela Ponce Morales

8

REFERENCIAS

Unilingo. (2017, 15 junio). ¿Qué pasa dentro de la mente de un procrastinador? (Doblado al español) [Vídeo]. YouTube. <https://www.youtube.com/watch?v=PG6oFK0a1NA>

TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTORES:**

Miguel Ángel Ruiz López

CARRERA	NOMBRE DE LA ASIGNATURA
INGENIERIA EN SISTEMAS COMPUTACIONALES	LENGUAJES Y AUTÓMATAS II

PROYECTO NO.	NOMBRE DEL PROYECTO	FECHA DE ENTREGA
1	Desarrollo de un compilador	19/10/2023

1	INTRODUCCION
	<p>El siguiente trabajo es un resumen de un video sobre la procrastinación, una conferencia donde exploran el proceso de la procrastinación que muchos pasamos al hacer trabajos o tareas importantes. Se explorará el vínculo entre la procrastinación y la búsqueda de gratificación inmediata, destacando la necesidad de tomar decisiones conscientes para abordar tareas con beneficios a largo plazo sin demora. El video examina cómo este comportamiento es común a todos los seres humanos y promueve la importancia de la autoconciencia para mejorar la eficacia en proyectos y metas personales.</p>

2	OBJETIVO
	<p>El objetivo de este resumen es proporcionar una visión general de la relación entre la procrastinación y la búsqueda de gratificación instantánea, destacando la importancia de la toma de decisiones conscientes y resaltando cómo la autoconciencia puede mejorar la eficacia en la realización de proyectos y metas personales.</p>

3	FUNDAMENTO
	<p>La procrastinación es un desafío común que afecta a muchas personas en diversas esferas de su vida, desde la educación y el trabajo hasta la consecución de metas personales. Entender las causas de la procrastinación y cómo está relacionada con la búsqueda de gratificación instantánea es esencial para abordar este comportamiento y mejorar la productividad y el bienestar personal.</p>

4	MATERIALES NECESARIOS
	<ul style="list-style-type: none">• Dispositivo con acceso a YouTube (Laptop)• Internet• Editor de Texto para redactar (Word)



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

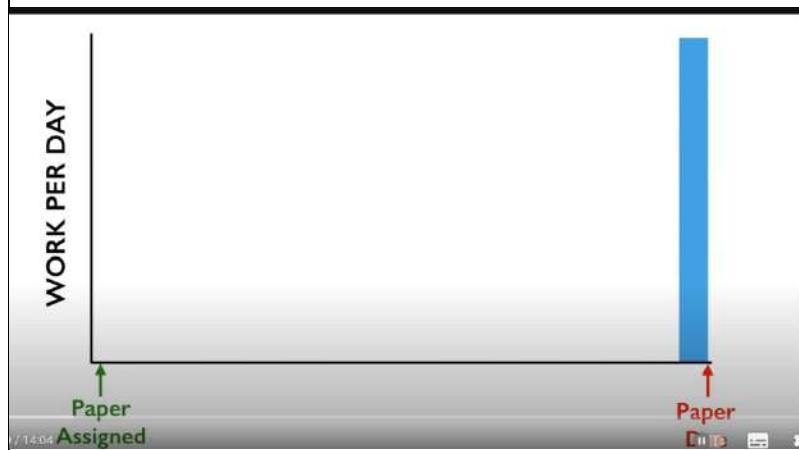
AUTORES:

Miguel Ángel Ruiz López

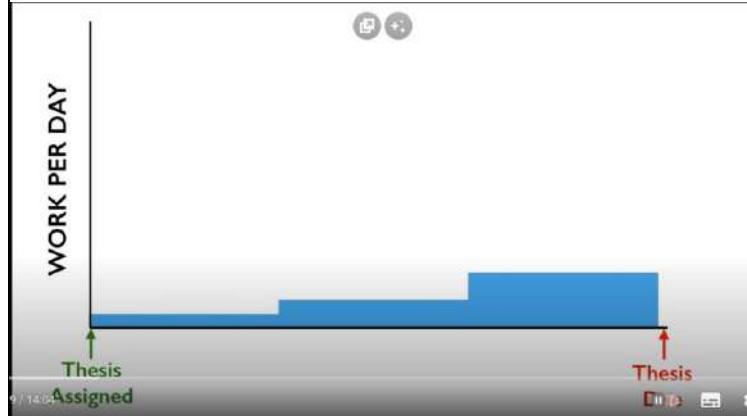
5

DESARROLLO

Esto pasa usualmente, las personas tenemos una tarea asignada y sin embargo, pese a que tenemos bastante tiempo para hacerlas, nos esmeramos para hacerlas justo al final, bajo presión.
Esto se puede representar en la siguiente gráfica.



El expositor, tomando como ejemplo las tesis que hizo, planificó mejor el tiempo de desarrollo de su tesis.



Al principio inició, con un trabajo relativamente bajo por día, para finalmente no tener tanta carga de trabajo, trabajar más tranquilo y menos presionado, además, asegurando que su tesis tendría mejor calidad.

Sin embargo, pese a tener todo planeado, no hizo nada los primeros meses, así que tuvo que reajustar su plan, una vez más.



TECNOLÓGICO
NACIONAL DE MÉXICO

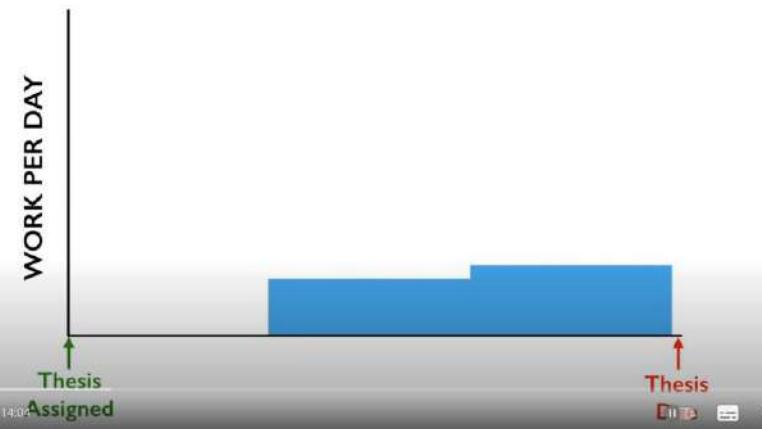
INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

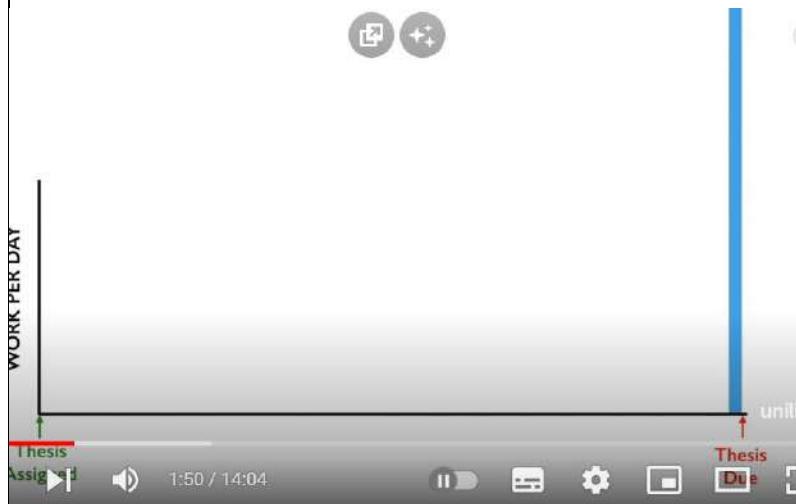
AUTORES:

Miguel Ángel Ruiz López



Reajustó su planificación para trabajar cantidad moderada de tiempo los penúltimos meses.

Luego, pasaron los meses de en medio y el expositor siguió sin escribir absolutamente nada, pues procrastinó. Finalmente, al faltar tres días para la entrega, escribió 90 páginas de tesis.



Él se siente satisfecho con la tesis que hizo, pese al hecho de que la hizo en tan solo tres días, pero la universidad le da un toque de realidad al decirle que la calidad de su trabajo era mala.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



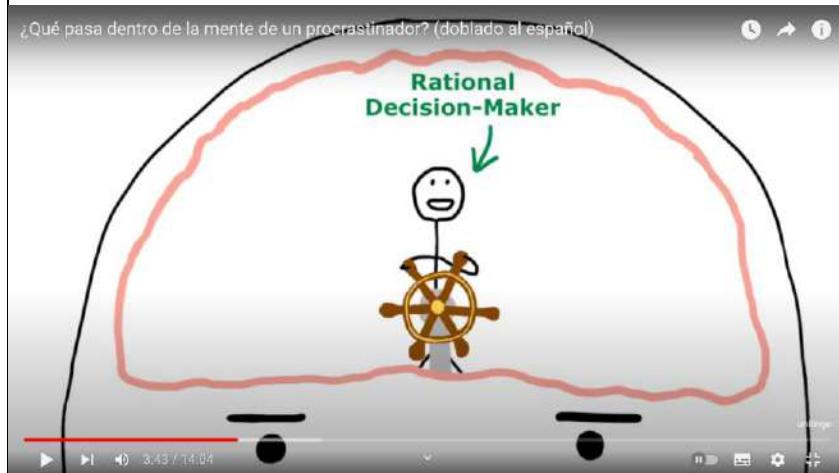
**PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II**

AUTORES:

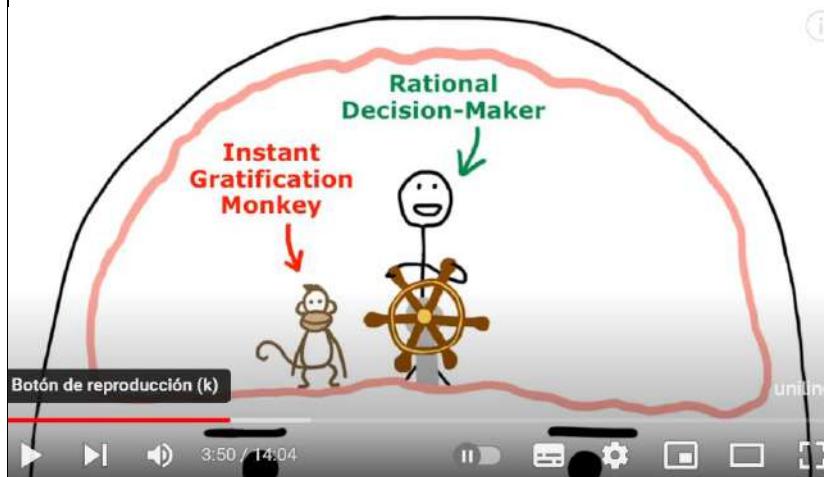
Miguel Ángel Ruiz López

Diferencias entre Procrastinadores y No Procrastinadores.

Según el expositor, este es el cerebro de un NO procrastinador:



Y este es el de un procrastinador (él mismo):



Ambos cerebros tienen a alguien que toma decisiones racionales, pero el del procrastinador tiene un mono de la gratificación inmediata.

Esto, según el expositor, significa que todo está bien hasta que sucede lo siguiente:

Cuando el racional está decidido a ponerse a trabajar, al mono de la gratificación instantánea no le gusta la idea y le arrebata el timón:



TECNOLÓGICO
NACIONAL DE MÉXICO

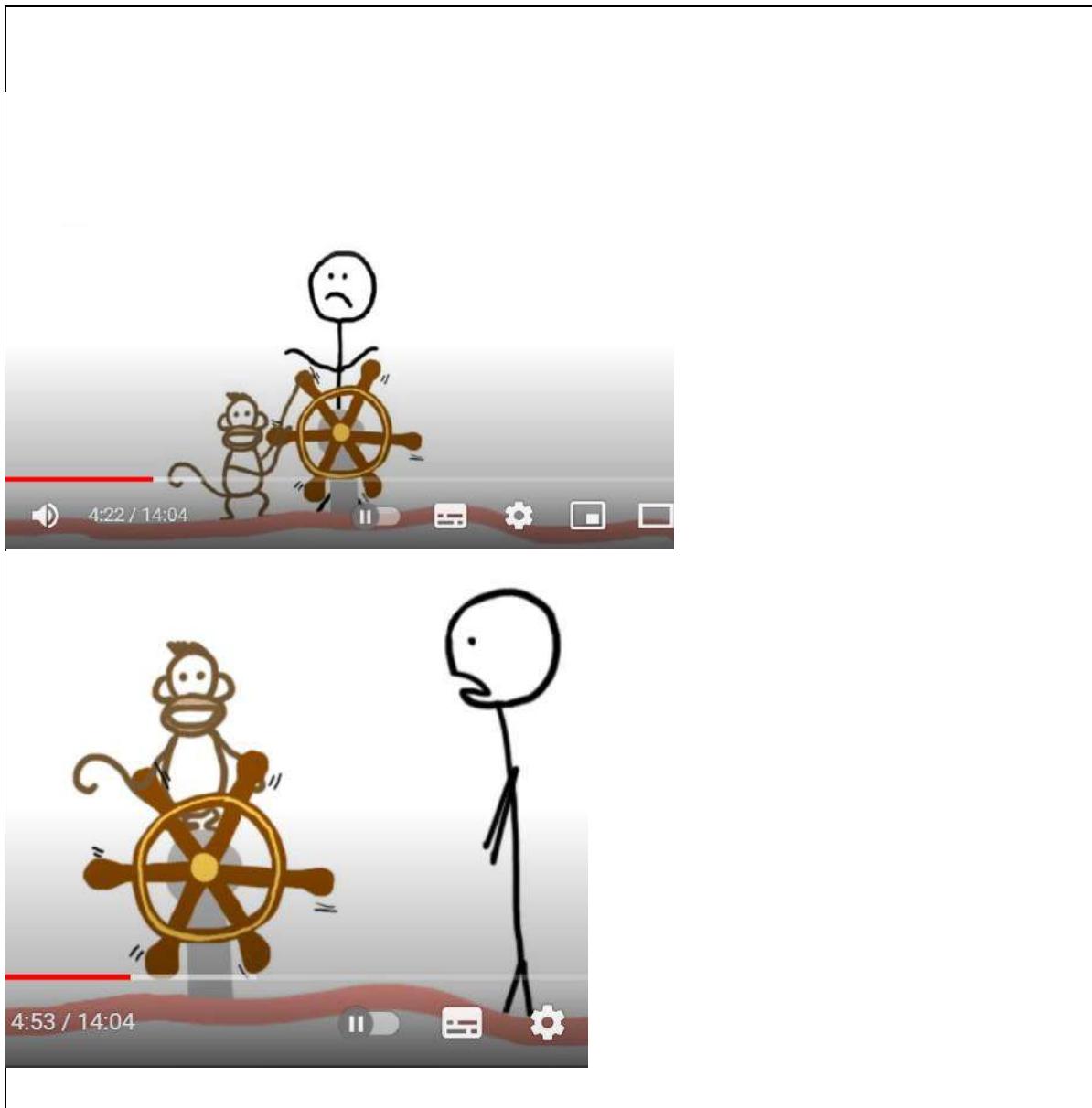
INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Miguel Ángel Ruiz López



Al mono de la gratificación inmediata solo le importan dos cosas, lo fácil y lo divertido, en el mundo animal esto funciona bien, pues es finalmente lo que hacen los animales, cosas a corto plazo. Los humanos en cambio, podemos hacer planes a largo plazo, y podemos hacer lo que tenga sentido hacer en ese preciso momento. Sin embargo, a veces también tiene sentido hacer cosas fáciles y divertidas como tomarnos un tiempo libre o cenar. Ahí es donde hay concordancia, ambas partes (la racional y la divertida) pueden ponerse de acuerdo.



TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

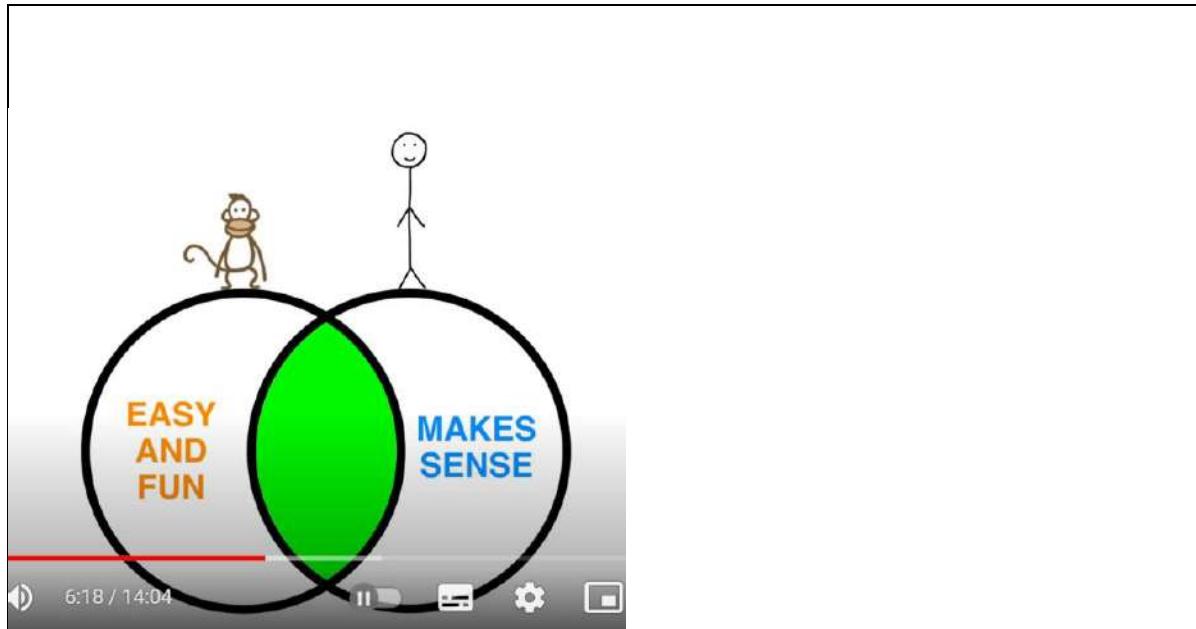


PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

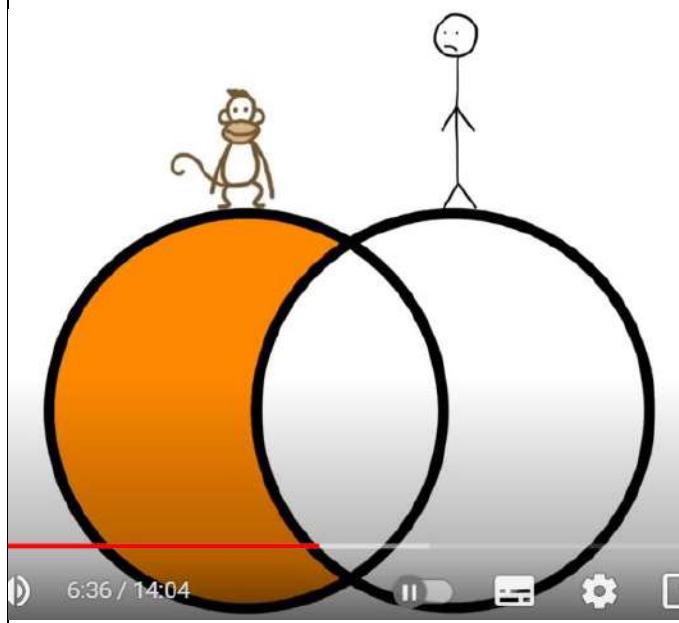
AUTORES:

Miguel Ángel Ruiz López

[Handwritten signatures]



Otras veces tiene más sentido hacer cosas más difíciles o menos placenteras, por el bien del panorama general, y es ahí donde entra el conflicto para el procrastinador, pues ese conflicto para el procrastinador siempre suele acabar en cosas divertidas y fáciles, haciéndole perder mucho tiempo.





TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

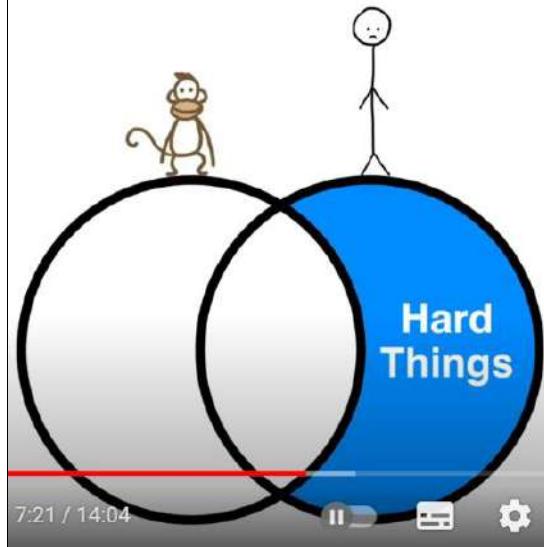
AUTORES:

Miguel Ángel Ruiz López

[Handwritten signatures]

Este círculo naranja es llamado el patio oscuro por el expositor, porque está alejado del círculo del sentido. La diversión que uno tiene en el “patio oscuro” no es genuina, porque no está bien merecida y el aire está cargado de culpa, temor, ansiedad, etcétera.

El procrastinador quiere salir de ese círculo naranja y pasarse al azul, donde ocurren cosas menos placenteras y más difíciles pero mucho más relevantes e importantes.



El procrastinador, según el expositor, tiene un “ángel de la guarda” que está inactivo la mayor parte del tiempo, llamado el Monstruo de Pánico.

The Panic Monster





TECNOLÓGICO
NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA



PRÁCTICA DE LABORATORIO - LENGUAJES Y AUTÓMATAS II

AUTORES:

Miguel Ángel Ruiz López

Se activa cuando una entrega se aproxima demasiado, un desastre en la carrera, si hay riesgo de vergüenza pública, etcétera; lo más importante de todo es que, este “monstruo” es a lo único a lo que le teme el Mono de la Recompensa Inmediata.

El expositor pone de ejemplo de cuando fue invitado a dar la CharlaTED, estaba emocionado porque siempre fue su sueño dar una plática de estas, sin embargo, no sabía dimensionar la importancia de ello, por lo que siguió procrastinando la preparación de su presentación.

Cuando faltaba a penas un mes, el equipo de TED anunció a los expositores, donde se encontraba el expositor actual, lo que hizo despertar a su Monstruo de Pánico, lo que hizo que todo su “sistema” estuviera en “caos”.



El Mono de las Recompensas Inmediatas huyó al ver al Monstruo y entonces las decisiones racionales retomaron el timón.



TECNOLÓGICO
NACIONAL DE MÉXICO**INSTITUTO TECNOLÓGICO DE CELAYA****PRÁCTICA DE LABORATORIO - LENGUAJES
Y AUTÓMATAS II****AUTORES:**

Miguel Ángel Ruiz López

Toda esta situación, con los tres personajes explicados, es el sistema de dilación, no es agradable pero finalmente, funciona.

El video presenta la idea de que todos los seres humanos son procrastinadores en mayor o menor medida, en el sentido de que todos dejamos las cosas para después. Se discute la diferencia entre el arduo trabajo de quienes tienen plazos y la falta de progreso de quienes no los tienen. Concluye con la idea de que todos debemos ser más conscientes de la gratificación instantánea que disfrutamos y comenzar a trabajar en tareas que tienen beneficios a largo plazo ahora, en lugar de esperar para más tarde.

6**BÍTACORA DE INCIDENCIAS**

Fecha	Problema encontrado	Solución
19/10/2023	El profesor asignó la actividad en hora asíncrona, por lo que tuve dificultades para entregar a tiempo la actividad, pues tenía clases.	Pensé en decirle al profesor que lo que hizo me parecía un poco injusto pero retador, sabía que si le decía lo consideraría, sin embargo, decidí tomar el reto y hacerlo durante otras clases, no perdí contenido importante de las clases en las cuales realicé la actividad.

7**CONCLUSIÓN**

En resumen, el video enfatiza la relación entre la procrastinación y la búsqueda de gratificación instantánea, subrayando la importancia de tomar decisiones conscientes para abordar tareas con beneficios a largo plazo de inmediato. La procrastinación es un comportamiento común a todos los seres humanos, pero reconocerlo y actuar en consecuencia puede mejorar la eficacia y el éxito en proyectos y metas personales.

8**REFERENCIAS**

- [1] Unilingo. (2017, 15 junio). ¿Qué pasa dentro de la mente de un procrastinador? (Doblado al español) [Vídeo]. YouTube. <https://www.youtube.com/watch?v=PG6oFK0a1NA>