

TECNOLOGICO
NACIONAL DE MEXICO
EN CELAYA

Asignatura: LENGUAJES & AUTÓMATAS II

Docente: ISC. Ricardo González González

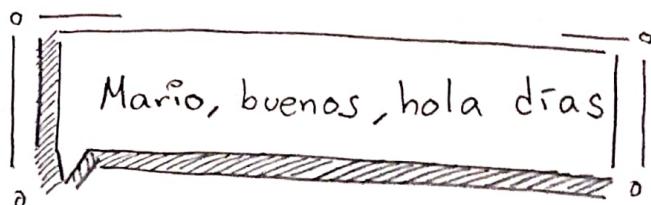
2023-10-16 LA-II-A

Actividad 6: Análisis Semántico

EQUIPO #1

INTEGRANTES:

- ARROYO GÓMEZ JOSÉ ALFREDO (20030029)
- GARCÍA HERNANDEZ CESAR (20030853)
- GASCA PALACIO JESÚS FERNANDO (20030606)
- GONZÁLEZ MANCERA CHRISTIAN MANUEL (20030115)



Léxico → ✓

Sintaxis → ✓

Semántica → X

Reglas
Semánticas

No tiene sentido
lo que dices

← Solución

Mario, hola, buenos días

R
e
s
u
l
t
a
c
o

Léxico
Sintaxis
Semántica { ✓



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.



César García Hernández



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.

- a. **LA OPORTUNIDAD.** SI EL TRABAJO FUE ENTREGADO OPORTUNAMENTE.
- b. **LA COMPRENSIÓN.** SE VALORARÁ EL GRADO DE COMPRENSIÓN DEL TEMAS ANALIZADOS.
- c. **LA CALIDAD.** SI LAS EVIDENCIAS ENVIADAS CORRESPONDEN A LA CALIDAD ESPERADA PARA ESTE NIVEL PROFESIONAL QUE SE CURSA.
- d. **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.
- e. **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.



César García Hernández



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.



César García Hernández



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_JUANAGO-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_JUANAGO-DIC23.PDF



César García Hernández



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.



César García Hernández

TECNOLÓGICO NACIONAL
DE MÉXICO EN CÉLAYA

Asignatura: Lenguajes y Autómatas II

Docente: Ricardo González González

2023 - 10 - 25 LA-II-A

Actividad: Monografía
Análisis Semántico

Equipo # 1

Integrantes

- Arroyo Gómez José Alfredo (20030029)
- García Hernández César (20030853)
- Gasca Palacio Jesús Fernando (20030606)
- González Mancera Christian Manuel (20030115)

Índice

Introducción	1
Generalidades	2
Título	5
Árboles de expresiones	5
Acciones semánticas del analizador sintáctico	12
Comprobaciones de tipos	17
Pila semántica	20
Esquema de traducción	28
Generación de la tabla de símbolos	34
Manejo de errores semánticos	41
Referencias bibliográficas	48
Anexos	51
Tarea individual : Arroyo Gómez	52
Tarea individual : García Hernández	61
Tarea individual : Gasca Palacio	67
Tarea individual : González Mancera	74

INTRODUCCIÓN

A este punto del semestre, nosotros como equipo perteneciente a la materia de Lenguajes y Autómatas II ya hemos abarcado y abordado temas del proceso de compilación y las principales fases de este. Hemos iniciado con el análisis léxico, el análisis sintáctico y ahora nos toca revisar los conceptos necesarios para comprender la fase del análisis semántico.

El análisis semántico es un aspecto muy importante que debería incluirse en un lenguaje. ¿De qué sirven todas esas palabras y caracteres que puedes usar y todas esas reglas gramaticales si tus oraciones o sentencias no tienen sentido. Podría ser que el mensaje o instrucción sea malinterpretado y así perder valiosa información. Es por ello que, en los lenguajes de programación, (y cualquier otro en realidad), se agregan aspectos semánticos a la definición y se plantean en el diseño del lenguaje para poder dar un sentido, una dirección y un propósito a las sentencias para alcanzar un objetivo. Séase validar tipos de datos, comprobar que el control de flujo es correcto, verificar que la información no sea ambigua o malinterpretada, etc.

GENERALIDADES

Contexto

¿Qué es un analizador semántico?

Un analizador semántico es parte de un compilador, más específicamente, es la 3era fase del proceso de compilación y en esta se encuentran las reglas semánticas que deben ser cumplidas en el código fuente ingresado al compilador.

¿Cuáles son los aspectos semánticos que debe cubrir el analizador semántico?

- Verificación de tipos de datos
- El control de flujo
- Resolución de nombre
- Certificar la coherencia del programa

Estos aspectos semánticos serán abordados en mayor detalle en las siguientes páginas del documento.

Como ya sabemos, el proceso de compilación consta de entradas y salidas, cada fase contiene su respectiva entrada y su salida, y la salida de una fase es la entrada de la siguiente fase. Entonces:

¿Cuál es la entrada de un analizador semántico?

- Árbol de sintaxis generado por el analizador sintáctico
- Tabla de símbolos generada por el analizador léxico
- Algunos compiladores pueden incluir información contextual adicional. Principalmente ~~se~~ reglas semánticas.

° ¿Por qué es importante conocer y aplicar la fase del análisis semántico?

Es importante porque el analizador semántico proporciona características que utilizamos en todos los lenguajes de programación como la validación de tipos.

Independientemente de si el lenguaje cuenta con inferencia de tipos o no, es necesario conocer el tipo de dato que se está utilizando ya que, gracias a esto, podemos realizar operaciones específicas dependiendo del tipo de dato. Una lista puede agregar elementos, un entero puede sumar, una cadena puede concatenar, etc. Por lo tanto, los aspectos semánticos nos permiten garantizar que nuestro lenguaje de programación sea coherente, integral, robusto y seguro gracias a esta verificación de tipos.

Los temas por abordar en este documento son los siguientes:

- 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 EQUHEMA DE TRADUCCIÓN
- TEMA 4.6 GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DE DIRECCIONES
- TEMA 4.7 MANEJO DE ERRORES SEMÁNTICOS

En cada uno de estos temas se verán ejemplos, ejercicios e ilustraciones creadas para reforzar el conocimiento de estos temas tan importantes para la materia.

Sin más dilación pasamos a presentar nuestro trabajo de investigación, análisis y redacción sobre el análisis semántico y, esperamos que este trabajo esté a la altura de la carrera, así como de los estándares impuestos en la rubrica de evaluación.

TITULO: ANÁLISIS SEMÁNTICO: ASPECTOS SEMÁNTICOS FUNDAMENTALES PARA CREAR UN ANALIZADOR SEMÁNTICO

TEMA 4.1 ÁRBOLES DE EXPRESIONES

Como recién vimos en el tema de análisis sintáctico, el uso de la estructura de datos. Los árboles, es muy utilizada en el diseño de un lenguaje de programación y de un compilador, ya que, esta estructura de datos discreta permite crear un mapa conceptual del conocimiento. Podríamos decir que es una pequeña base de datos que organiza y sabe donde está la información.

Asimismo, debemos recordar el uso de las expresiones para modelar comportamientos del mundo real con un lenguaje formal como son las matemáticas.

Una característica importante de las expresiones es la precedencia y prioridad de los operadores, esta permite asignar un peso entre las operaciones que deben realizarse en una expresión. Esto es útil cuando codificamos o modelamos expresiones o cálculos muy complejos.

Conociendo lo anterior, podemos introducir un tema bastante interesante el cual es el de los árboles de expresiones.

ÁRBOLES DE EXPRESIONES

Los árboles de expresiones son un tipo de árbol especial denominados árboles binarios, estos tienen la característica de que los nodos padres pueden tener dos hijos o cero. Esto es útil para representar expresiones aritméticas.

Las expresiones pueden ser representadas por medio de un árbol.

$x < y$ se representa como:

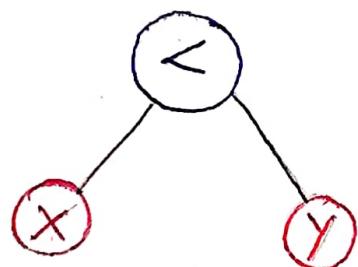


Figura 1. Árbol de expresión, $x < y$

Podemos observar lo siguiente de la figura anterior:

- Los nodos padre son operadores ($<$)
- Los nodos hijos son operandos (x, y)

Veamos otro ejemplo: $(a + b) - (c - d)$

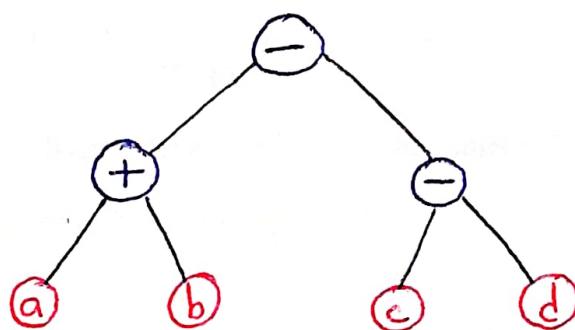


Figura 2. Árbol de expresión, $(a + b) - (c - d)$

De la figura 2 podemos concluir que:

- Los paréntesis de la expresión no se incluyen en el árbol explícitamente, pero sí proporcionan una guía de la precedencia e importancia de las operaciones.

Para poder evaluar las expresiones representadas por medio de un árbol es necesario realizar un recorrido de este. Para ello existen 3 tipos de recorridos que permiten recuperar la información proveniente del árbol.

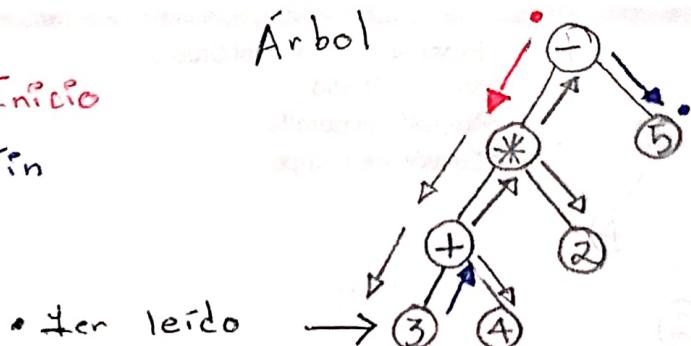
RECORRIDOS EN UN ÁRBOL

Recorrido en orden (inorder)

En un recorrido en orden, primero se reconoce el hijo izquierdo (subárbol izquierdo), luego se visita el nodo actual y, finalmente, se recorre el hijo derecho (subárbol derecho). Esto se utiliza comúnmente en árboles de expresiones donde se necesita evaluar la expresión en notación infija.

Notación infija: Forma tradicional de escribir expresiones matemáticas: $(3 + 4) * 2 - 5$

Árbol



• Inicio

• Fin

• Hacer leido

Figura 3. Recorrido

In-order

Recorrido:

1. 3
2. 3 +
3. $3 + 4 = 7$
4. $7 * 2$
5. $7 * 2 = 14$
6. $14 -$
7. $14 - 5 = 9$

Resultado = 9

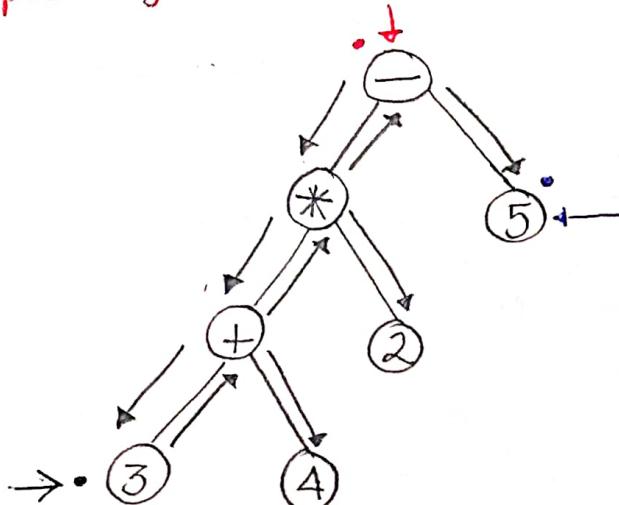
Recorrido en postorden (postorder)

Recorrido en el que primero se visitan los nodos hijos antes de visitar el nodo raíz. En otras palabras, se procesan los nodos de izquierda a derecha en un subárbol antes de procesar el nodo raíz del subárbol.

Notación Postfija. También conocida como notación polaca inversa o RPN. En la notación postfija, los operadores se colocan después de sus operandos. Esta notación elimina la ambigüedad en la secuencia de operaciones, ya que no es necesario utilizar paréntesis para indicar el orden. Para evaluar una expresión postfija, se utiliza una pila.

Expresión postfija: $3 \ 4 + 2 * 5 -$

- Inicio
- Fin
- 1er nodo en entrar



- Recorrido
1. 3
 2. 3 4
 3. 3 4 +
 4. 3 4 + 2
 5. 3 4 + 2 *
 6. 3 4 + 2 * 5
 7. 3 4 + 2 * 5 -

Figura 5. Recorrido en Postorden

Recorrido en preorder (preorder)

En un recorrido en preorder, primero se visita el nodo actual, luego se recorre el hijo izquierdo y finalmente, se recorre el hijo derecho. Esto se utiliza comúnmente en la evaluación de expresiones aritméticas y en la construcción de árboles de sintaxis abstracta. También se usa el recorrido en preorder cuando se necesita evaluar una expresión prefija.

Notación prefija: También conocida como notación polaca, en esta notación los operadores se colocan antes de sus operandos. No requiere paréntesis.

Para evaluar una expresión en notación prefija, también se utiliza una pila para llevar a cabo las operaciones en el orden correcto.

Expresión prefija: $- * + 3 4 2 5$

- Inicio
- Fin
- 1er nodo en entrar

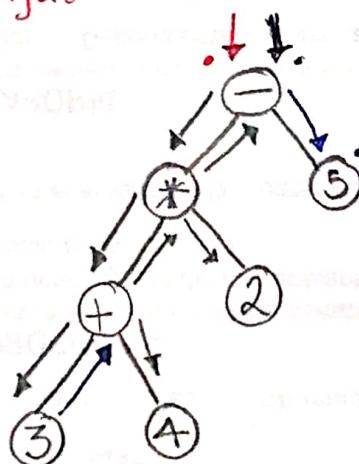


Figura 4. Recorrido en preorder

Recorrido

1. -
2. - *
3. - * +
4. - * + 3
5. - * + 3 4
6. - * + 3 4 2
7. - * + 3 4 2 5

Expresión final:

$- * + 3 4 2 5$

La notación postfija y la notación prefija son utilizadas comúnmente en calculadoras científicas y en la compilación de lenguajes de programación. Son especialmente útiles cuando se necesita evaluar expresiones de manera eficiente, ya que eliminan la necesidad de realizar un análisis de precedencia y de asociatividad de operadores.

Construcción de un árbol de expresiones a partir de una expresión infija

1. Crear una pila de operadores y una pila de nodos del árbol.
2. Inicializar un árbol vacío como el árbol de expresiones.
3. Tokenizar la expresión en notación infija en elementos individuales.
4. Para cada token en la expresión:
 - Si es un número, crear un nodo para ese número y colocarlo en la pila de nodos del árbol.
 - Si es un operador, seguir los pasos:
 1. Mientras la pila de operadores no esté vacía y el operador en la parte superior de la pila tiene mayor o igual precedencia que el operador actual, sacarlos de la pila y crear un nodo nuevo para el operador en la pila de nodos del árbol.
 2. Apilar el operador en la pila de operadores.
 - Si es un paréntesis de apertura "(":
 - Apilar el paréntesis en la pila de operadores

- Si es un paréntesis de cierre ")":
 1. Mientras el operador en la parte superior de la pila de operadores no sea un paréntesis de apertura "(":
 - Despila el operador y crea un nodo para él en la pila de nodos del árbol.
 2. Despila el paréntesis de apertura "(" de la pila de operadores
5. Después de procesar todos los tokens, si quedan operadores en la pila de operadores, sacarlos de la pila y crear nodos para cada uno de ellos en la pila de nodos del árbol.
6. El nodo en la parte superior de la pila de nodos de árbol es la raíz del árbol de expresiones.

El árbol resultante será un árbol de expresiones que representa la estructura de la expresión matemática en notación infija. A continuación, se puede hacer un recorrido en postorden para evaluar la expresión.

ACCIONES SEMÁNTICAS DE UN ANALIZADOR SINTÁCTICO

Durante el transcurso de redacción de trabajos anteriores, hemos mencionado la función del analizador sintáctico durante el proceso de compilado, aunque sus acciones no se limitan únicamente a comprobar y verificar que las cadenas de entrada estén compuestas en base a la gramática dada, para después dar y construir el árbol sintáctico; sino que este proceso dentro del compilado abarca acciones semánticas realizadas en esta parte descritas durante el desarrollo de esta investigación. A su vez nos arroja el analizador sintáctico nos arroja como salida un tipo de representación del árbol sintáctico que después va a reconocer las sucesiones de token que son dadas por el analizador léxico.

Por su parte, la función de un analizador semántico es dar un sentido coherente, es decir, que la relación entre todos los elementos tenga algún significado en el contexto llevado a cabo, que en este caso es coherencia respecto al análisis sintáctico.

He aquí porqué el analizador sintáctico puede llevar a cabo acciones semánticas, pues es la salida generada por el análisis sintáctico es usada por el semántico y, de igual manera asocia una acción en el código para que una vez se detecte una regla de análisis sintáctico se lleve a cabo una acción semántica.

Es importante comprender el concepto de acciones semánticas para entender qué función tienen dentro del

proceso de compilado; podemos entenderlas como tareas que son llevadas a cabo por un analizador sintáctico dentro del proceso de análisis del código a compilar y, gracias a ellas nos es posible asignar un significado a la estructura del programa, dentro de las tareas que pueden ser realizadas encontramos algunas como lo son la construcción de árboles de sintaxis abstracta o la comprobación de aspectos semánticos.

De esta manera, actúan y funcionan como un puente de comunicación a la gramática y la semántica del código, para así permitir al analizador sintáctico la correcta interpretación y procesamiento del programa a un nivel profundo.

Pero, ¿qué acciones semánticas lleva a cabo un analizador sintáctico?

- Acceso a la tabla de símbolos

Esta es una parte vital para el analizador sintáctico pues almacena datos sobre identificadores, variables o demás elementos del código, entonces, el analizador sintáctico verifica, por ejemplo, que estos identificadores estén declarados en la tabla de símbolos y sean empleados de manera correcta.

- Generación de código intermedio

Las acciones semánticas pueden cumplir la función de código intermedio (qué es la traducción de la estructura sintáctica a un formato más abstracto), lo que permite dar paso a optimizaciones antes de que se genere código máquina, este código se convierte en ejecutable.

• Inferencias de tipos

En este apartado, no se plantea ahondar a fondo pues en un siguiente punto se tratará más a detalle, aunque en resumen permite identificar posibles errores relacionados al tipado del lenguaje durante el proceso de compilación.

Gramática de un analizador sintáctico

A diferencia de los lenguajes naturales, que la gramática podemos mencionar no deja de crecer puesto que es el contexto que le rodea (llámese tiempo, lugar o situación) de tal manera que no deja de crecer al estar en constante interacción y retroalimentación con los individuos; los lenguajes de programación se caracterizan por seguir reglas estrictas formadas por gramáticas formales, donde se tiene que tener un planteamiento de carácter estricto y altamente detallado de tal manera que la comunicación humano-máquina sea efectiva y el procesador lleve a cabo las instrucciones indicadas por el programador.

Entonces, dentro del proceso de compilado, el analizador sintáctico acepta una gramática libre de contexto ya que gracias al uso de esta gramática nos libraremos de algunas problemáticas como lo puede llegar a ser un error de localización o ambigüedad de expresiones y sentencias.

La gramática libre de contexto se caracteriza por ser una cuádrupla como la siguiente:

$$G: \langle N, T, P, S \rangle$$

donde entendemos que:

N son los elementos ^{no} terminales.

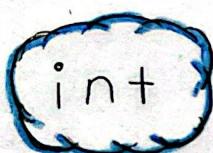
T son elementos terminales.

P. son reglas de producción.

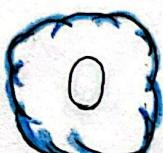
S es el axioma inicial.

GRAMÁTICA DE ATRIBUTOS

Entendemos como atributo la representación de una propiedad de símbolos (sea terminal o no terminal). Algunos ejemplos son los siguientes.



Tipo de dato



Valor



Posición en memoria

La gramática de atributos hace uso de la gramática libre de contexto, puesto que, una vez se valida la estructura del programa es necesario utilizar la información representada por los atributos que representan los símbolos terminal o no terminal asociada al lenguaje. Al usar esta gramática de atributos se dice que se hicieron traducciones dirigidas por sintaxis.

Al pasar valores a los atributos, cada una de las reglas gramaticales son asociadas a una operación, se conoce como regla semántica a esta característica.

Louden (2004) indica que las gramáticas de atributos son escritas en forma de tabla, como se representa a continuación.

Regla gramatical	Regla semántica
Regla 1	Regla de atributo asociada
:	:
Regla n	Regla de atributo asociada

De igual manera, existen dos tipos de atributos:

- Se considera heredado cuando el valor que se le es asignado es dependiente de los nodos hijos.
- Se considera sintetizado cuando el valor es dependiente de nodos hermanos y padre.

La escritura de un atributo es de la siguiente forma; cuando está asociado a un símbolo:

X: símbolo gramatical.

a: atributo de símbolo gramatical.

X.a: el valor de a está asociado al símbolo.

O visto de otra manera:

X.tipo Entero, booleano, real, ... Tipo de dato del símbolo X

X.valor Real Valor del símbolo X



Atributo



Posibles valores

COMPROBACIONES DE TIPOS

Cuando una persona se adentra en el mundo de la programación, uno de los conocimientos básicos que se deben manejar es sobre los tipos de datos existentes en los lenguajes de programación.

Los tipos de datos nos permiten como programadores manipular y manejar diferentes gamas de tipos de valores dentro de un programa, de forma general existen algunos como lo son el `integer` (para valores enteros) o `String` (valores de cadenas de texto).

Parte de la lógica de programación es entender que las operaciones realizadas con los datos deben de seguir una "misma línea" e incluso esto es aplicable a nuestra vida cotidiana, un ejemplo claro lo tenemos en la educación básica cuando nos decían que no podíamos mezclar peras con manzanas, ¿a qué nos referímos con esta expresión?

Extrapolando, podemos entender a las peras y a las manzanas como dos tipos de datos distintos, cada uno con sus diferentes y correspondientes características, por lo que realizar operaciones entre ellos resultaría en un grave error semántico.

A nivel sintáctico, las expresiones pueden superar las revisiones en base a la gramática y no detectar errores; como podemos ver en el siguiente ejemplo.

Float $x = 10.0f;$
tipo de dato variable valor asignado

Double $y = 50.0;$
tipo de dato variable valor asignado

$x = y;$ ¿Cuál es el
error?

Sintácticamente no hay errores. Entonces ¿a qué problemas nos enfrentamos?

CONSTRUCCIÓN DE TIPOS

Pues es de importancia resaltar que cada tipo de dato se basa en la construcción sintáctica del lenguaje, de manera que si los dos operandos de un operador aritmético son de tipo entero, pues este será de tipo entero igualmente.

Conocemos a esta comprobación como sistemas de tipos. Así mismo, la problemática del problema planteado anteriormente nos indica que queremos guardar un tamaño de información mayor a la capacidad máxima de otro tipo de dato, es decir, que queremos guardar 8 bytes de información (referente al tipo de dato double) en tan solo 4 bytes (referente al tipo de dato float). ¿Qué implicaría esto? Lleva a pérdida de información pues no se podría almacenar por completo, aunque también existe una herramienta en los compiladores que fuerzan a que esta información se guarde, a este proceso se le llama casteo.

VERIFICACIÓN DE TIPOS

En este proceso el compilador se asegura de que cada operador (llámese matemático o lógico) tenga valores permitidos de acuerdo con las reglas ya establecidas del lenguaje de programación en cuestión. En una basta cantidad de lenguajes no está permitido usar un número real como índice para acceder a un elemento en un arreglo, en caso que se proceda con esta acción, el compilador debe detectar ese error.

arreglo [4.8] ; ~~X~~ ierror!
est. de índice
datos

Entonces, se asegura que las operaciones tengan un tipo de dato idóneo para proceder con las operaciones requeridas y todo en base a las reglas del lenguaje, además de que lleva a cabo conversiones de datos automáticas para algunas operaciones.

4.4 Pila semántica en un analizador sintáctico.

Las pilas y colas son estructuras de datos que se utilizan generalmente para simplificar operaciones de programación. Estas estructuras pueden implementarse mediante arrays o listas enlazadas.

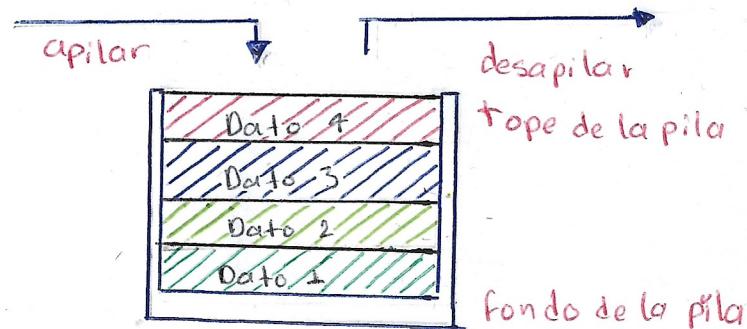
Una pila es una colección de datos a los cuales se les puede acceder mediante un extremo, que se conoce generalmente como tope.

Las pilas tienen dos operaciones básicas:

- push (para insertar un elemento).
- pop (para extraer un elemento).

Operaciones con pilas

- Insertar un elemento (push).
- Eliminar (pop).
- Pila vacía.
- Pila llena.



Su característica más notable es que, al extraer un elemento, siempre se obtiene el último que se ha insertado recientemente. Por esta razón también se les conoce como estructuras de datos LIFO (Last In First Out).

Una forma de implementarlas con listas enlazadas sería

insertar y extraer siempre desde el principio de la lista.

Tanto las pilas como las colas son estructuras de datos comúnmente empleadas para simplificar ciertas operaciones de programación utilizando matrices.

Al utilizar arreglos para implementar pilas se tiene la limitación de que se debe reservar el espacio en memoria con anticipación. Una vez dado un máximo de capacidad a la pila no es posible insertar un número de elementos mayor que el máximo establecido.

Si esto ocurre, en otras palabras, si la pila está llena y se intenta insertar un nuevo elemento se producirá un error conocido como desbordamiento (overflow). Una alternativa para abordar estas dificultades sería establecer pilas con una gran capacidad, pero esta aproximación resultaría ineficaz y costosa.

No siempre es posible prever con precisión la cantidad de elementos que se van a manipular, y siempre existe la posibilidad de que surja un problema de desbordamiento.

Otra alternativa son las colas también son llamadas FIFO (First In First Out), que quiere decir "el primero que entra es el primero que sale".

1. En colas simples, la inserción se realiza en un extremo y la extracción en el otro. En el caso de una cola simple, los elementos se agregan al final y se retiran del principio.

Para gestionar esta cola, es crucial llevar un registro del próximo elemento a ser leído y del último elemento que se ha introducido.

2. En colas circulares, se considera que después del último elemento se vuelve al primero. Esto permite la reutilización de las posiciones extraídas, ya que al final de la cola se convierte en el principio, creando un ciclo cerrado.
3. Las colas con prioridad se implementan utilizando listas o matrices ordenadas. En este caso, no importa el orden de entrada, sino que se asigna una prioridad. Puede ocurrir que varios elementos tengan la misma prioridad, y en este caso, se extraerá primero el que haya llegado primero FIFO.

El analizador sintáctico es un autómata de pila que reconoce la estructura de una cadena de componentes léxicos.

Por lo general, el analizador sintáctico inicia el proceso de compilación y, para cada símbolo de entrada, invoca al analizador morfológico para proporcionar el siguiente símbolo de entrada.

Se analiza el símbolo de la pila y el estado del autómata, el analizador sintáctico produce las estructuras necesarias para la siguiente etapa y en el caso de compilación dirigida por la sintaxis invoca llamadas directas al analizador semántico y al generador de código. Escribe mensajes de errores y trata de limitar el efecto de estos errores.

Al decir pila semántica no se refiere a que hay varios tipos de pilas; hace referencia a que se debe programar única y exclusivamente en un solo lenguaje, es decir, no podemos mezclar código de C++ con Visual Basic.

Ventajas

- Los problemas de integración entre los subsistemas son costosos y, en muchos casos, no se resuelven hasta que se acerca la fecha límite para la integración completa del sistema.
- Se requiere una memoria auxiliar que permita almacenar datos intermedios para su posterior comparación.

El objetivo teórico es la construcción de un árbol de análisis sintáctico, aun que este raramente se construye de manera explícita. En su lugar, las rutinas semánticas incorporadas van generando el árbol de sintaxis abstracta a medida que se procesa el código. Este proceso se define a través de una gramática libre de contexto.

La pila juega un papel fundamental en el desarrollo de cualquier analizador semántico. Dentro de cada elemento de la pila se guardan los valores que pueden tener una expresión.

Un analizador sintáctico ascendente utiliza una pila para guardar información acerca de los subárboles que ya han sido analizados. En un analizador sintáctico es posible utilizar campos adicionales en la pila para almacenar los valores de atributos sintetizados.

Para ilustrar este concepto, consideremos un ejemplo de una pila de analizador con espacio para un valor de atributo. Supongamos que esta pila se implementa mediante un par de matrices llamadas "estado" y "val". Cada entrada en la matriz "estado" es un puntero o índice que apunta a una tabla de análisis sintáctico.

En este escenario, la matriz "estado" podría utilizarse para llevar un seguimiento de los estados del analizador sintáctico a medida que procesa la entrada. Cada estado en la pila corresponde

a un estado particular en la tabla de análisis sintáctico.

La matriz "val" se utilizará para almacenar los valores de los atributos sintetizados a medida que se calculan durante el análisis sintáctico. Cada entrada en la matriz "val" estaría asociada con un estado específico en la matriz "estado".

A medida que se avanza en el análisis sintáctico, los valores de los atributos sintetizados se calcularían y se guardarían en la matriz "val" en la posición correspondiente.

Esta técnica permite mantener un seguimiento de los atributos sintetizados en función de los estados del analizador y facilita la generación de un árbol de análisis sintáctico o la realización de acciones semánticas a medida que se analiza la entrada.

tope →

estado	val
• • •	• • •
X	X X
Y	Y y
Z	Z z
• • •	• • •

El tope en curso de la pila se indica con el apuntador tope. Se supone que los atributos sintetizados se evalúan justo antes de cada reducción.

El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse conforme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Árbol de expresiones o árbol semántico

Se trata de un sistema jerárquico en el que se anotan las acciones llevadas a cabo por el programa original.

En cada rama del árbol se anota el valor o propósito que debe tener, y el proceso de análisis determina cuál de los valores anotados en las ramas es el adecuado.



Acciones Semánticas

Dependiendo del tipo de sentencias, las acciones semánticas pueden agruparse en:

- Sentencias de declaración: completar la sección de tipos basándose en la tabla de símbolos.
- Sentencias "ejecutables": realizar comprobaciones de tipos entre los operandos implicados.
- Funciones y procedimientos: comprobar el número, orden y tipo de los parámetros actuales en cada llamada a una función o procedimiento.
- Identificación de variables: comprobar si un identificador ha sido declarado antes de utilizarlo.
- Etiquetas: Comprobar si hay etiquetas repetidas y validación.
- Constantes: Comprobar que no se utilicen en la parte izquierda de una asignación.

- Conversiones y equivalencias de tipo: verificación.
- Sobrecarga de operadores y funciones: detectar y solventar.

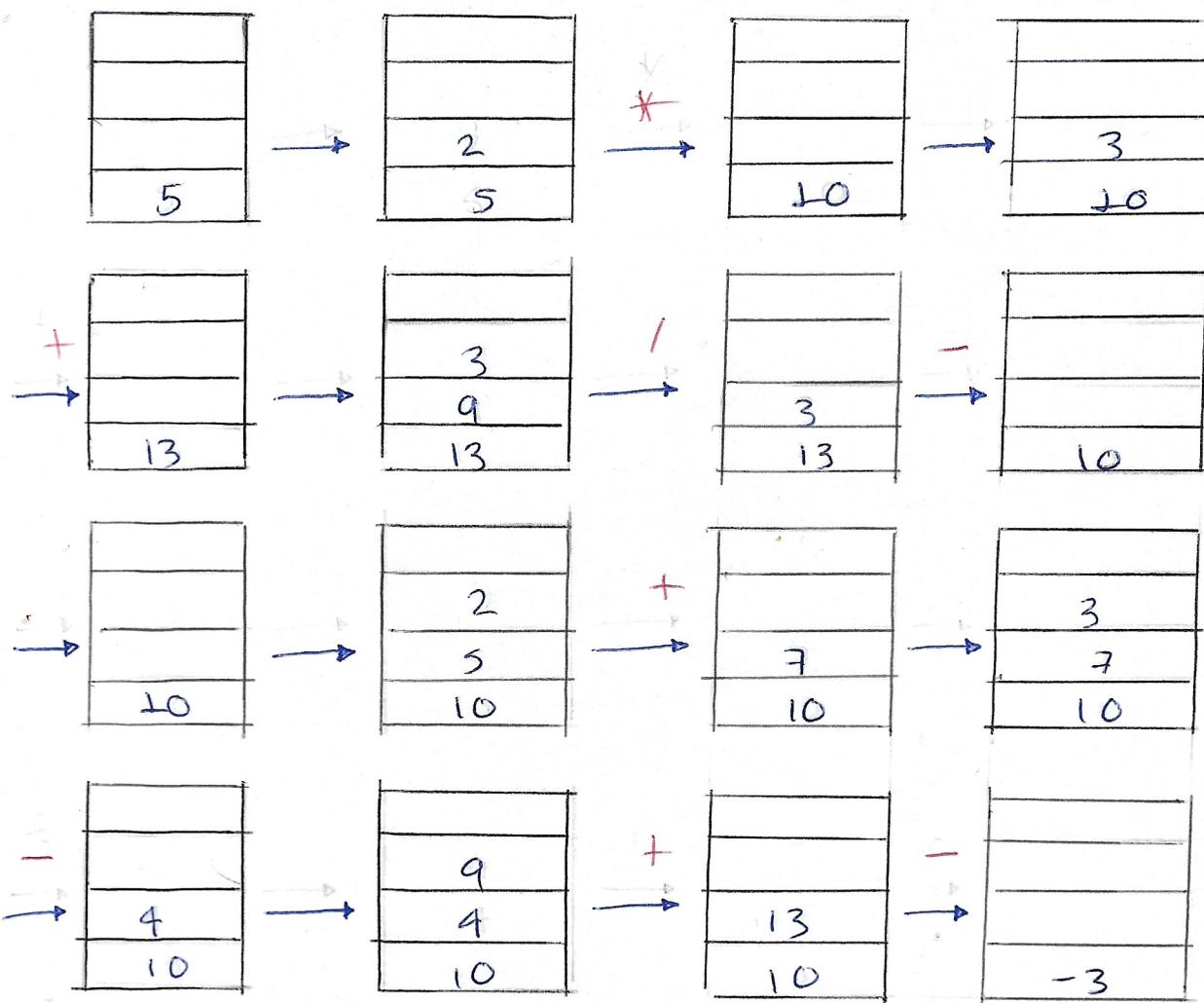
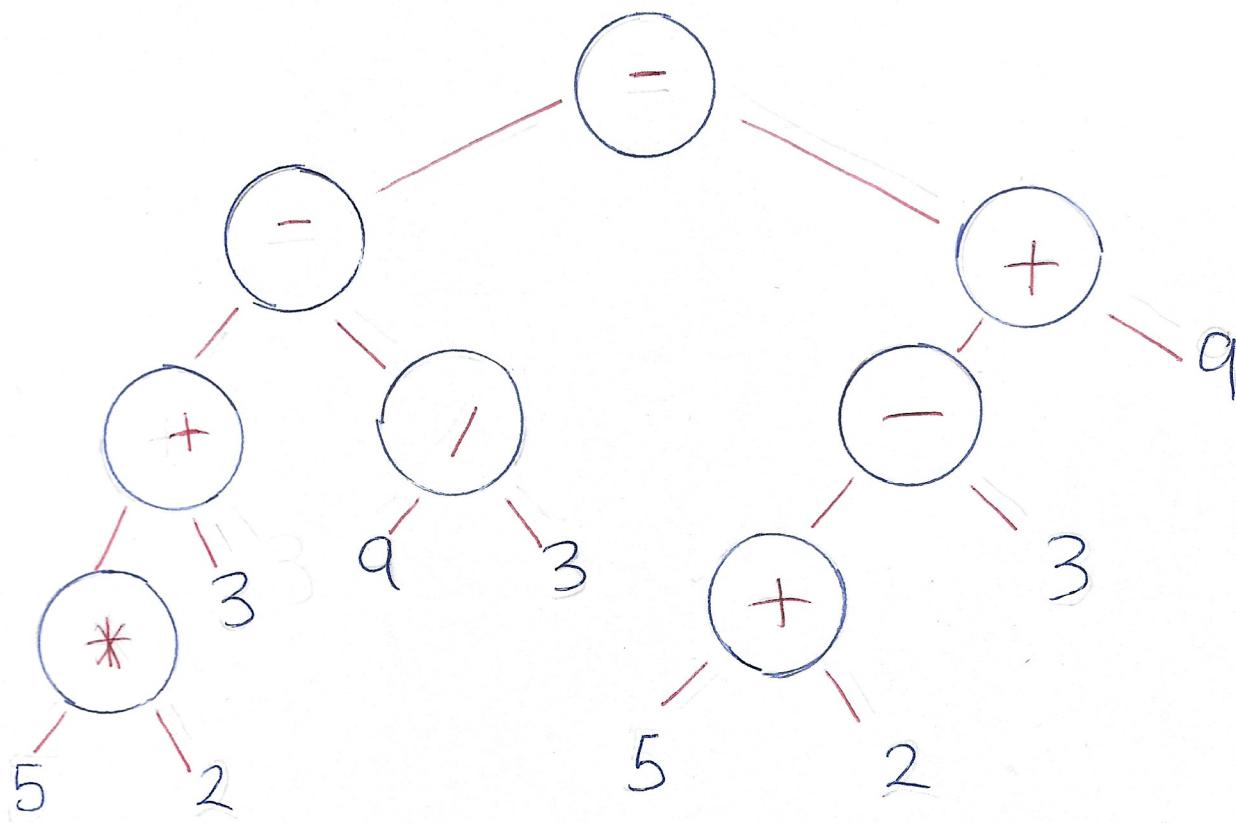
En el contexto del manejo de errores sintácticos, es esencial abordar estos desafíos para garantizar una compilación efectiva:

1. Indicar errores de manera clara y precisa: El objetivo es proporcionar mensajes de error que describan el tipo de error encontrado y su ubicación específica en el código fuente. Esto facilita a los programadores la identificación y corrección de los errores.
2. Recuperación de errores: cuando se detecta un error, es importante que el compilador se recupere y continúe analizando el código en busca de más errores. Esto evita que un solo error cause la terminación abrupta del proceso de compilación.
3. Mantener la eficiencia: aunque se deben abordar los errores, es esencial que el proceso de manejo de errores no ralentice significativamente la compilación. El compilador debe ser eficiente y no consumir recursos innecesarios en el manejo de errores.

Ejemplo demostrativo de pila.

Para la siguiente expresión $(5 * 2 + 3 - 9 / 3) - (5 + 2 - 3 + 1)$ se usa el siguiente algoritmo.

- Si encontramos un operando, lo metemos a la pila, si encontramos un operador los últimos 2 que están en la pila se sacan y se efectúa la operación y el resultado se reemplaza en la pila.



4.5 Esquema de traducción

Lo principal que tenemos que entender es que los lenguajes de programación tienen reglas específicas para su estructura, y a menudo, estas reglas no pueden ser totalmente descritas utilizando gramáticas simples. Por lo tanto, es necesario verificar la corrección de un programa en una etapa posterior después de analizar su estructura.

Por ejemplo, imagina que estás construyendo un rompecabezas. Las piezas son como las reglas de un lenguaje de programación, y cada pieza debe encajar de una manera particular. A veces, estas reglas no son tan simples como "las esquinas siempre deben tener 4 conexiones" y pueden requerir un análisis más detallado para asegurarte de que el rompecabezas (o programa) esté correctamente ensamblado.

Este proceso generalmente implica etapas posteriores durante la compilación o interpretación, en las cuales se necesita una representación de la entrada que permita que las funciones se ejecuten correctamente. Por esta razón, la detección de errores está estrechamente relacionada con la forma en que se representa la información.

Además, es común que se impongan ciertas restricciones, como la necesidad de que el identificador en la parte izquierda este previamente declarado y que el tipo de la expresión sea compatible con el identificador.

En otras palabras, asegurarse de que las partes del programa estén en su lugar adecuado y que tengan sentido es fundamental para el funcionamiento correcto del programa.

Entonces el analizador semántico tiene que comprobar que estas dos restricciones se cumplen antes de declarar que la sentencia de asignación se encuentra bien formada.

Un esquema de traducción consiste en una gramática independiente del contexto en la que se han insertado fragmentos de código en las partes derechas de sus reglas de producción.

Producción	Reglas Semánticas
$L \rightarrow E_n$	print(E.Val)
$E \rightarrow E_L + T$	E.Val := E_L.Val + T.Val
$E \rightarrow T$	E.Val := T.Val
$T \rightarrow T_L * F$	T.Val := T_L.Val * F.Val
$T \rightarrow F$	T.Val := F.Val
$F \rightarrow (E)$	F.Val := E.Val
$F \rightarrow \text{dígito}$	F.Val := dígito.lex

► Recuperado de: "Esquema de traducción ¡Fotos y guía 2021! (2020, 28 Sep). Esquema.net <https://esquema.net/traduccion/>

Los fragmentos de código que se insertan son denominados acciones semánticas por que se encargan de actuar, calcular y modificar los atributos que se encuentran asociados con los nodos del árbol sintáctico.

Para implementar un proceso de traducción, es necesario primero crear un árbol sintáctico y luego aplicar las acciones definidas en las reglas en un orden de recorrido de tipo primero en profundidad. Si la gramática es ambigua, es decir, si una frase puede tener dos árboles sintácticos diferentes, la ejecución de las acciones podría generar resultados distintos. Por lo tanto, para evitar confusiones en las interpretaciones semánticas, es crucial establecer cuál de los árboles sintácticos se está considerando.

En otras palabras, cuando la estructura de una frase puede ser interpretada de diferentes maneras, es necesario especificar claramente (cuál) interpretación se va a utilizar para garantizar que las acciones.

Semánticas se apliquen de manera coherente y se obtenga el resultado deseado.

Otra definición que podemos considerar importante es que un esquema de traducción es una gramática independiente de contexto en la que se asocian atributos con los símbolos gramaticales y se insertan acciones semánticas encerradas entre llaves {} dentro de los lados derechos de las producciones. Los esquemas de traducción pueden tener tantos atributos sintetizados como heredados

» Recuperado de: "De expresiones, A. (sif), Unidad I: Análisis semántico Ifpn.mx , Recuperado el 22 de octubre de 2023 <http://ifpn.mx/recursosisc/7semestre/lenguajesyautomas2/Unidad%20I.pdf>

Cuando se diseña un esquema de traducción, es importante respetar ciertas limitaciones para asegurarse de que el valor de un atributo esté disponible cuando una acción haga referencia a él.

Estas limitaciones son necesarias para evitar que las acciones hagan referencia a un atributo que aún no ha sido calculado. Un ejemplo simple de esto ocurre cuando solo se necesitan atributos sintetizados (el atributo sintetizado es el tipo de dato de una expresión. A medida que se analiza una expresión, se calcula su tipo apartir de los tipos de sus operandos, y este tipo se transmite hacia arriba en el árbol sintáctico).

Los atributos sintetizados son fundamentales en la construcción de árboles de análisis semántico y en la generación de código intermedio en el proceso de compilación). En este caso, se puede diseñar el esquema de traducción creando una acción de asignación para cada regla semántica y colocándola al final del lado derecho de la producción asociada.

En otras palabras, es esencial estructurar el esquema de traducción de manera que los cálculos de los atributos estén bien definidos y disponibles antes de que se utilicen en acciones.

Esto asegura un proceso de traducción coherente y sin problemas.

Algunos puntos importantes en los esquemas de traducción son que se debe tener un lenguaje, símbolos, acciones semánticas y reglas.

Ejemplos:

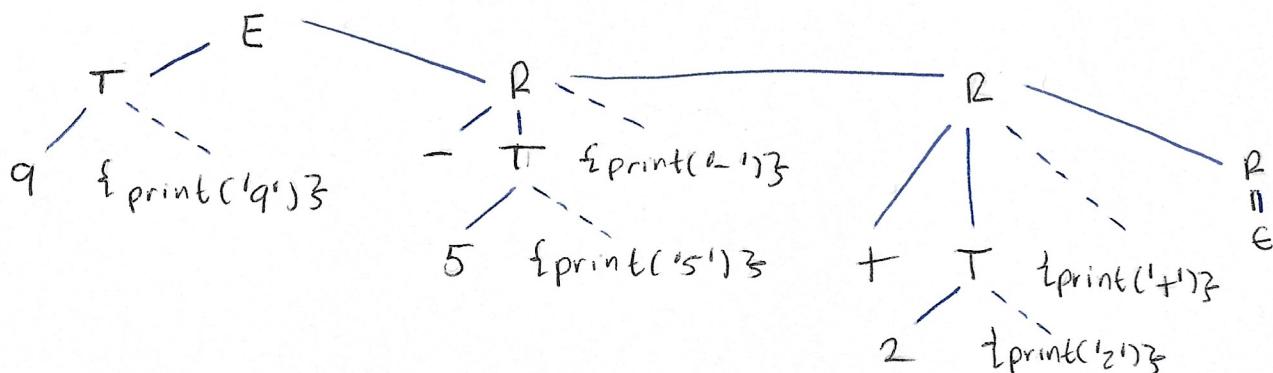
Un esquema de traducción simple que transforma expresiones infixas con suma y sustracción en las expresiones postfixadas correspondientes:

$$E \rightarrow TR$$

$$R \rightarrow \text{opsuma } T \{ \text{print}(\text{opsuma.lexema}) \} R \sqcup E$$

$$T \rightarrow \text{num} \{ \text{print}(\text{num.val}) \}$$

En el análisis semántico aplicado ya, se obtienen los valores de entrada $9 - 5 + 2$ con cada acción semántica asociada como el hijo adecuado del nodo que corresponde al lado izquierdo de su producción.



De este esquema obtenemos de acuerdo al orden de profundidad lo siguiente: $9 \ 5 \ - \ 2 \ +$

es la expresión obtenida del esquema de traducción.

Una traducción como lo indica la palabra traduce la expresión gramática a la expresión programable, asignando los términos de acción y las variables correspondientes ya declaradas.

$$E \rightarrow E + E$$

$$E \rightarrow E + E \{ \text{system.out.print}(\text{var} + (\text{var} - \text{val})) \}$$

$$E \rightarrow E \{ \text{system.out.print}("el valor es " + \text{cal}) \}$$

Ejemplo de implementación de gramática con pila de manera manual

Producción

$$L \rightarrow E_n$$

Acción

```
{ print(stack[top-1].val);  
  top=top-1; }
```

$$E \rightarrow E_L + T$$

```
{ stack[top-2].val = stack[top-2].val  
  + stack[top].val;  
  top=top-2; }
```

$$E \rightarrow T$$

$$T \rightarrow T_1 * F$$

```
{ stack[top-2].val = stack[top-2].val  
  * stack[top].val;  
  top=top-2; }
```

$$T \rightarrow F$$

$$F \rightarrow (E)$$

```
{ stack[top-2].val = stack[top-1].val;  
  top=top-2; }
```

$$F \rightarrow \text{digit}$$

	$3 + 5 \text{ n}$	shift
3	$+ 5 \text{ n}$	red
F	$+ 5 \text{ n}$	red
T	$+ 5 \text{ n}$	red
E	$+ 5 \text{ n}$	shift
$E+$	s n	shift
$E+5$	n	red
$E+F$	n	red
$E+T$	n	red
E	n	shift
$E\text{n}$		red
+		acept

Así se confirma el uso de la pila de manera manual.

acciones → retroceder en la pila

↳ es necesario conocer detalladamente la gramática

↳ si no se maneja bien la pila los valores esperados pueden ser incorrectos

elementos → n - simbolo para terminar la operación

↳ shift - desplazamiento

↳ red - reducción (reemplazar)

Shift → se toma un terminal y se agrega a la pila

red → reemplazar un simbolo de la pila por un no terminal en base a las reglas de la producción.

GENERACIÓN DE LA TABLA DE SÍMBOLOS Y DE DIRECCIONES

El análisis semántico es una de las fases cruciales en el proceso de compilación de un programa. Como hemos visto en esta etapa verificamos que el programa cumpla con las restricciones semánticas del lenguaje de programación y de construir estructuras de datos que representen la información semántica del programa. Dos de las estructuras más importantes en esta etapa son las tablas de símbolos y las tablas de direcciones, estas son esenciales para llevar a cabo la traducción del código fuente a código objeto y garantizar que el programa funcione.

Definición de la Tabla de Símbolos e Importancia

Podemos definir una tabla de símbolos como; una estructura de datos, usada en el proceso de traducción y centralización de un lenguaje de programación. Bien puede ser usado por un compilador o un intérprete.

Un compilador o intérprete usa esta estructura para llevar un registro sobre los atributos asociados a cada símbolo usado en el lenguaje de programación. Se examina el contenido de la tabla de símbolos cada que se encuentra un símbolo en el código fuente, así si se descubre un símbolo nuevo o nueva información sobre un símbolo existente se producen cambios en el contenido.

La implementación mas utilizado para una tabla de símbolos es la tabla hash, la cual detallaremos mas adelante.

Así pues una tabla de símbolos debe ser capaz de almacenar los atributos asociados a cada símbolo; dichos atributos son:

- Tipo de dato : entero, char, boolean, ... etc.
- Valor : 25, 4, cadena, 0, etc.
- Dirección de memoria.
- Número de línea : Usado en entornos integrados de desarrollo (IDE)
- Ámbito : función, main, método, etc..

La misión principal de las tablas de símbolos es asociar y colaborar en las comprobaciones del análisis semánticas. Las operaciones que se suelen utilizar son: inserción, búsqueda, actualización y eliminación. La eficiencia de cada una de estas operaciones depende de la estructura de datos elegida, así como su consumo de recursos, en tiempo de acceso a los datos.

Así pues la importancia de las tablas de símbolos radica en varios aspectos clave:

- Almacenamiento de información sobre símbolos: La tabla de símbolos se utiliza para almacenar información sobre todas las variables, constantes, funciones y otros identificadores utilizados en el código fuente.
- Comprobación de tipos: Durante el análisis semántico, se realiza una verificación de tipos para garantizar que las operaciones y asignaciones sean coherentes en cuanto a los tipos de datos involucrados.
- Resolución de referencias y ámbito de visibilidad: La tabla de símbolos ayuda a resolver referencias a identificadores en el programa. Asegura que las variables o funciones se utilicen dentro de su ámbito y referencias adecuadas.
- Optimización y captura de errores: Cuando se detectan errores semánticos, la tabla de símbolos puede proporcionar información sobre la ubicación del error y los tipos involucrados, lo que facilita la generación de mensajes de error claros y útiles.

Tipo de información que se almacena en la tabla de símbolos
La tabla de símbolos almacena una variedad de información sobre
cada identificador o símbolo; estos son utilizados a lo largo del
programa. Los tipos de información que se almacenan en la tabla
de símbolos pueden variar según el lenguaje y las necesidades del
compilador o intérprete, pero generalmente incluyen:

- **Nombres**: La tabla de símbolos almacena los nombres de todas las variables, símbolos y funciones definidos en el programa.
- **Tipo de datos**: Para cada uno de los símbolos, se registra su tipo de dato. Esto es esencial para realizar comprobaciones de tipos y asegurarse de que las operaciones se realicen de manera coherente.
Se pueden almacenar diferentes tipos:
 - **Tipos simples**: Enteros, Reales, Carácteres, etc.
 - **Tipos estructurados**: Arreglos (incluyendo dimensión), Estructuras, etc.
 - **Tipos archivo**: La representación interna de los archivos varía.
 - **Tipos función**: Funciones y procedimientos, parámetros, tipos que devuelven.
 - **Tipos clases**: Clases con sus atributos, métodos y relaciones.
- **Ambito de visibilidad**: Se registra el ámbito en el que se define una variable o función, lo que ayuda a determinar en qué partes del código pueden utilizarse.
- **Ubicación en memoria**: La tabla de símbolos puede mantener información sobre la ubicación en memoria de los símbolos, especialmente en lenguajes de bajo nivel.
- **Valores iniciales**: Se puede almacenar información sobre los valores iniciales o predeterminados.
- **Etiquetas y saltos**: En lenguajes de bajo nivel, se puede contener información sobre etiquetas y puntos de destino para saltos.
- **Otros**: Dependiendo del lenguaje puede haber atributos específicos.

Definición Tabla de direcciones y relación con la Tabla de direcciones

Una tabla de direcciones (también conocida como tabla de direcciones de memoria o tabla de direcciones de variables) es una estructura de datos utilizada en el contexto de la compilación o la interpretación de programas para llevar un seguimiento de la ubicación en memoria de variables y símbolos del programa.

Esta tabla contiene información sobre la dirección en la memoria en la que se almacenan valores de las variables y otros datos, lo que es fundamental para la generación de código y la gestión de la memoria.

Como se mencionó anteriormente la tabla de símbolos almacena información sobre los identificadores utilizados en un programa, como nombres de variables, funciones, tipos de datos, ubicación, etc. La tabla de símbolos se utiliza para mantener un registro de los símbolos. La tabla de direcciones, por otro lado, se utiliza para rastrear la ubicación en la memoria de las variables y otros datos. Contiene información sobre la dirección de memoria asignada a cada símbolo o identificador. Esto esencial para saber y recuperar los valores de cada símbolo.

Así pues podemos definir la relación entre estas dos estructuras:

- Durante la etapa de análisis semántico, se utiliza la información de la tabla de símbolos para asignar direcciones de memoria a las variables. La asignación de direcciones se realiza en función de factores como el tipo de dato y el ámbito de visibilidad.
- La tabla de direcciones se actualiza a medida que se asignan las direcciones de memoria; cada entrada está vinculada a un símbolo específico que se encuentran en la tabla de símbolos.
- En la etapa de generación de código, el compilador o intérprete utiliza la información de la tabla de direcciones para generar instrucciones que hagan referencia a las ubicaciones de memoria de cada variable y/o dato.

Estructura de la tabla de direcciones

La estructura y organización de una tabla de direcciones de memoria pueden variar según la implementación y las necesidades específicas de un compilador o intérprete, pero en general suele constar de una serie de entradas o registros que vinculan los símbolos con direcciones de memoria. Por lo regular consta de:

- **Identificador:** Cada entrada en la tabla de direcciones debe incluir un campo que identifique el símbolo asociado a la entrada. Este identificador suele ser el nombre de la variable, símbolo u objeto cuya dirección de memoria se está registrando.
- **Dirección de memoria:** Para cada identificador, la tabla de direcciones debe contener un campo que almacene la dirección de memoria en la que se encuentra la variable o el dato asociado. Esta dirección es un número entero que indica la ubicación específica en la memoria del sistema.
- **Otros atributos:** Dependiendo de los requisitos del compilador o intérprete, la tabla de direcciones puede incluir otros campos que almacenan información adicional (tipo de dato, ámbito de visibilidad, acceso, etc.).

Las estructuras comunes que se utilizan para organizar una tabla de direcciones incluyen:

- **Tabla hash:** Se puede utilizar una tabla hash para acceder eficientemente a las entradas de la tabla de direcciones. La función hash se aplica al identificador para determinar la ubicación.
- **Tabla de búsqueda binaria:** Se puede utilizar una tabla de búsqueda binaria cuando la tabla de direcciones se organiza de manera ordenada.
- **Lista enlazada:** Cada entrada en la tabla de direcciones se puede vincular a la siguiente, creando una lista enlazada.

Se recomienda el uso de tabla hash, ya que el objetivo es permitir un acceso rápido y eficiente a las direcciones de memoria asociadas a los identificadores y garantizar que la información se maneje adecuadamente.

Funcionalidad de Hash

Las funciones hash son algoritmos que toman una entrada (generalmente de longitud variable) y generan una salida de longitud fija, este valor es conocido como "valor hash" o "hash code". Estas funciones son ampliamente utilizadas en la búsqueda eficiente, seguridad, comprobación de integridad de datos y más.

Hash tiene características fundamentales:

- Los códigos hash identifican de manera inequívoca, por tanto nunca se generan dos hash idénticos si los datos de entrada son diferentes.
- Los códigos hash son únicos. Si aplicamos el algoritmo sobre un mismo conjunto de datos en varias ocasiones, siempre se obtiene la misma secuencia alfanumérica.
- Los funciones hash son unidireccionales. Partiendo del código hash, no se puede descifrar o inferir cuáles fueron los datos introducidos inicialmente.

La función hash funciona siguiendo:

- Entrada: La función hash toma una entrada, que puede ser de cualquier tipo de datos, como una cadena de texto, números, estructuras, etc. Esta entrada puede tener una longitud variable y ser de cualquier tamaño.
- Proceso hash: La función hash aplica un algoritmo matemático o lógico a la entrada para calcular un valor hash.
- Salida fija: La función hash produce una salida de longitud fija, lo que significa que el valor hash tiene una longitud constante.
- Determinista: La función hash es determinista, es decir para cada misma entrada siempre generará el mismo valor hash.
- Uniformidad: Un buen algoritmo hash distribuirá los valores hash de manera uniforme en su espacio de salida.
- Resistencia a colisiones: Aunque es poco probable que dos entradas diferentes generen el mismo valor hash, es importante que la función hash sea resistente a colisiones.

Ejemplos Tabla de símbolos y de direcciones.

Para ilustrar la generación de las tablas de símbolos y de direcciones, consideremos un programa simple:

```
int main() {  
    int x = 15;  
    float y = 2.3;  
    return x+y;}
```

El código simple anterior generaría la tabla de símbolos siguiente:

Identificador	Tipo de Dato	Ambito de Visibilidad
Main	Función	Global
x	int	main
y	float	main

En la tabla anterior se registran los nombres de la función y variables utilizadas, junto con su tipo de dato y ámbito de visibilidad.

La tabla de direcciones se utiliza para rastrear la ubicación en memoria de las variables y datos del programa.

Identificador	Dirección de Memoria
Main	0x1000
x	0x1004
y	0x1008

En esta tabla de direcciones, se registran los nombres de las funciones y variables utilizadas en el programa, junto con su dirección de memoria asignada.

MANEJO DE ERRORES SEMÁNTICOS

Definición de error Semántico

Un error semántico se produce cuando la sintaxis de un programa es correcto, pero la semántica (comportamiento que resulta de ejecutar el programa) no es lo que se esperaba. Se refieren en sí a problemas que afectan tanto la lógica, como el significado del código o programa.

Hay que tener en cuenta que la construcción del programa o código se rige por las reglas del lenguaje dado, por ello al realizarse un error semántico el compilador o intérprete no detecta estos errores semánticos; los compiladores e intérpretes solo se preocupan de la estructura del código ingresado y no de su comportamiento correcto o incorrecto deseado. Por consecuencia un error semántico puede ocasionar irregularidades en la ejecución del programa o código, puede dar como salida un mensaje de error o no puede dárlo, eso depende del compilador o intérprete.

Los errores semánticos más comunes son:

- **Uso incorrecto de variables:** Ocurre cuando se usa una variable de una manera que no tiene sentido en el contexto del programa. Por ejemplo:
`int n=5 y total=0;`
`total = total + "n";`
- **Condicionales incorrectos:** Estos errores semánticos surgen cuando las condiciones en las estructuras condicionales no reflejan correctamente su uso. Por ejemplo:
`float E=30.5`
`if (E = 10) { // uso incorrecto del operador de asignación
 System.out.println("Entró");`

- **Buclees infinitos o inalcanzables:** Este error semántico se da cuando un bucle nunca termina adecuadamente, esto debido a una condición incorrecta o una variable que no se actualiza. Por ejemplo:

```
int i=0;
while(i>0) { // Bucle infinito
    System.out.println("Valor de i=" + i);
}
```

- **Acceso a memoria no válida:** Por lo regular estos errores pueden llevar a la corrupción de la memoria o fallas en ejecución de estructuras.

Por ejemplo:

```
int arr[3]={1,2,3};
int x=arr[10]; // Acceso a una posición del arreglo fuera de límites.
```

- **Problemas de tipo de datos:** Asignar incorrectamente una variable un tipo incorrecto. Por ejemplo:

```
int n="H2";
```

- **Malentendidos de lógica empresarial:** Estos tipos de errores surgen cuando la implementación del código no coincide con los requisitos o la lógica empresarial. Por ejemplo, si un programa debe realizar cálculos incorrectos, esto sería un error semántico.

- **Manejo inadecuado de errores:** No manejar adecuadamente los errores o excepciones puede llevar a un comportamiento no deseado.

Importancia de detectar y manejar errores semánticos

Como se mencionó anteriormente los errores semánticos pueden causar resultados incorrectos o comportamientos inesperados en un programa o código. Detectar y manejar errores semánticos en el desarrollo de un compilador o intérprete es de suma importancia.

Si no se detectan los errores semánticos puede llevar a problemas de seguridad o daños en la organización de un código. Al detectar y corregir estos errores, se garantiza que un software funcione correctamente y cumpla con sus objetivos.

La presencia de errores semánticos disminuye la calidad del software.

César García Hernández

En resumen, la detección y gestión de errores semánticos son fundamentales para garantizar que un código sea confiable, seguro y cumpla con los requisitos comerciales y regulatorios. Abordar estos errores no solo mejora la calidad del software, sino que también ahorra tiempo y dinero a largo plazo, además de aumentar la satisfacción del usuario y la eficiencia del desarrollo.

Métodos y técnicas para detectar errores semánticos

Es más difícil introducir métodos formales para la recuperación de errores semánticos a comparación de otros errores como los sintácticos. No obstante, es esencial para un programador la obtención del error semántico.

La mayoría de los errores semánticos se pueden detectar con mayor facilidad mediante la revisión de la tabla de símbolos, esto mediante el campo del tipo de base en el contexto donde ocurre el error o un tipo universal permitiendo al identificador ser un símbolo de cualquier operador del lenguaje, así evitamos la producción de mensajes de error.

En cierta manera la tabla de símbolos es la manera más común que se utiliza, pero existen otros métodos y técnicas, como:

- **Análisis Estático de Código:** Utilizar herramientas de análisis estático de código, como linters, los cuales verifican el código en busca de patrones y prácticas de codificación incorrectas, estas herramientas pueden detectar desde variables no utilizadas hasta parámetros sin uso.
- **Sistemas de Tipos:** Los lenguajes de programación con sistemas de tipos fuertes pueden encontrar errores semánticos relacionados con el tipo de datos. Los sistemas de tipos ayudan a garantizar que las operaciones se realicen en tipos de datos compatibles.
- **Pruebas Unitarias:** Esta técnica sigue el flujo de datos a través del

1

2

3

César García Hernández

4

programa y busca inconsistencias o problemas, como variables no inicializadas o lecturas de variables antes de ser asignadas.

• Pruebas de aceptación: las pruebas de aceptación se centran en asegurarse de que el programa cumple con los requisitos del negocio y satisface las necesidades del usuario final.

Uso de tablas de símbolos y reglas de comprobación

El uso de tablas de símbolos y reglas de comprobación son técnicas esenciales para el manejo de errores semánticos. Como se vio en temas anteriores las tablas de símbolos son estructuras de datos que se utilizan para almacenar información sobre los identificadores o símbolos en un lenguaje de programación.

Durante el análisis semántico cuando se encuentra un nuevo símbolo en el código, se busca en la tabla de símbolos para su verificación, esto se usa para el rastreo del ámbito o scope de los símbolos, lo que es esencial para que las variables y símbolos se usen de manera coherente, si un símbolo se utiliza fuera de su ámbito, se detecta un error semántico. Así pues podemos decir que la información del tipo y ámbito en la tabla de símbolos se utiliza para verificar la coherencia de las operaciones en el código.

De mismo modo se aplican reglas de comprobación de las estructuras y operaciones en el código para garantizar que cumplen con la semántica del lenguaje de programación. Estas reglas se derivan de la especificación del lenguaje y son esenciales para prevenir errores semánticos. Algunas reglas de comprobación incluyen:

- Verificación de las asignaciones entre las variables y su tipo.
- Garantizar que se realice una llamada a una función con el número correcto de argumentos y tipos.
- Comprobar que las operaciones aritméticas y lógicas sean aplicables

- o los tipos de datos involucrados.
- Asegurarse de que los parámetros de una función sean coherentes con su definición.
- Verificar que las variables se inicialicen antes de ser utilizadas.

Las reglas de comprobación se aplican durante el análisis semántico y generan errores semánticos cuando se violan. Estas reglas se definen para garantizar que el código sea válido y cumple con la semántica del lenguaje de programación.

El uso de tablas de símbolos y reglas de comprobación es fundamental para detectar y gestionar errores semánticos, los tablas de símbolos ayudan a almacenar información sobre identificadores o símbolos, mientras que las reglas de comprobación aplican restricciones semánticas para garantizar que el código sea válido y cumple con la semántica del lenguaje.

Clasificación de errores semánticos

Desde el punto de vista del compilador, los errores semánticos se pueden dividir en dos categorías principales: Errores visibles y errores invisibles. Estos tipos de errores son conceptos utilizados para describir los errores semánticos en el desarrollo de código en función de si se manifiestan de manera evidente o no.

- **Errores visibles:** Son errores semánticos que se manifiestan de manera clara y evidente durante la ejecución. Estos errores producen resultados incorrectos o comportamientos inesperados que son fácilmente detectados por los usuarios.

Son problemas que afectan directamente la funcionalidad. Algunos ejemplos son, un cálculo aritmético que produce un resultado incorrecto, un programa se bloquea o se cierra inesperadamente, datos incorrectos o inconsistentes.

• **Errores Invisibles:** Son errores semánticos que no se manifiestan de manera obvia durante la ejecución del programa y pueden pasar desapercebidos. Estos errores pueden afectar el funcionamiento interno, la eficiencia, la seguridad o la calidad del código sin que los usuarios se den cuenta. Algunos ejemplos son: variables que no se liberan correctamente, errores de lógica que no afectan directamente al programa.

Los errores invisibles pueden ser más difíciles de detectar y corregir, ya que como se menciona el impacto no es muy evidente, esto provoca que se puedan acumular con el tiempo.

Generación de mensajes de error descriptivos

La generación de mensajes de error descriptivos en el contexto del análisis semántico en los compiladores o intérpretes es crucial para ayudar a los programadores a comprender y corregir problemas en el código fuente.

Antes de generar mensajes de error, un compilador o intérprete debe detectar y analizar los errores semánticos. Esto implica evaluar las construcciones del código en función de las reglas semánticas del lenguaje creado o especificado.

Es importante que para generar cada mensaje de error se deben especificar de manera preciso y claro la naturaleza del error, se debe mostrar el número de línea y si es posible, una descripción visual de la ubicación dentro de la línea. Se debe explicar porque el código es incorrecto y cómo debería corregirse, mediante ejemplos o sugerencias para su corrección.

Se debe mantener un formato consistente en todos los mensajes de error dentro del compilador, esto facilita que los usuarios se familiaricen con la estructura de los mensajes.

Es esencial generar mensajes de error descriptivos en el análisis semántico para facilitar la corrección de errores y mejorar la productividad de los desarrolladores.

Ejemplos de código para detección y manejo de errores semánticos
Los siguientes ejemplos se centran en la detección de errores de tipo
en expresiones matemáticas y en la identificación de errores de
ámbito.

```
%#t
#include <stdio.h>
#include <stdlib.h>
%#
%token NUM
%token STR
%token '+' '-'
%token '*' '/'
%token '='
%#
program: /*empty*/ | program statement '\n'
statement: expression printf ("Resultado: %d\n", $1); |
           | STR printf ("Error semántico: no se puede usar una cadena como expresión\n");
expression: expression '+' expression | $1 + $3; |
           | expression '-' expression | $1 - $3; |
           | expression '*' expression | $1 * $3; |
           | expression '/' expression | $1 / $3; |
           | NUM | $1;
%
int main()
{
    system();
    return 0;
}
int error (char* s)
{
    printf ("Error sintáctico: %s\n", s);
    return 0;
}
```

Referencias Bibliográficas

1.3.4. Análisis Semántico. (s/f). Edu.mx. Recuperado el 23 de octubre de 2023, de http://cidecame.uach.edu.mx/lcc/ma/PROYECTO/libro32/134-analisis_semantico.html

Laura, V., (s/f). ARBOLES DE EXPRESIONES. Blogspot.com. Recuperado el 23 de octubre de 2023, de <https://wilma-lauramoraleszunun.blogspot.com/2019/08/arboles-de-expresiones.html>

(s/f). Vam.es. Recuperado el 23 de octubre de 2023, de <http://arantxa.ii.vam.es/alfonsec/docs/compila5.htm>

(s/f). Studocu.com. Recuperado el 23 de octubre de 2023, de <https://www.studocu.com/in/document/itm-university/microeconomics/lenguajes-automatas-2-generalizado/1712657>

Universidad Europea de Madrid. (s/f). ANÁLISIS SEMÁNTICO. Academia Cartagena 99. Recuperado el 23 de octubre 2023, de <https://www.cartagena99.com/recursos/alumnos/apuntes/INFINA-M4-U5-T1.pdf>

Rodríguez, J. (2022). Acciones Semánticas de un analizador sintáctico. pdfslide.tips. <https://pdfslide.tips/documents/acciones-semanticas-de-un-analizador-sintactico.html?page=4>

Acosta, F. (2020, 25 noviembre). Comprobación de tipos [Vídeo]. YouTube. <https://www.youtube.com/watch?v=QXap-gSCKqY>

De expresiones, A. (s/f). Unidad Iº Análisis semántico. Itpn.mx
Recuperado el 22 de octubre de 2023, de http://itpn.mx/recursosesc/7_semestre/lenguajesyautomatas2/Unidad%201.pdf

Esquema de traducción. (2020, septiembre 28). Esquema.net.
<https://esquema.net/traducción/>

Esquemas de traducción. (s/f). Prezi.com. Recuperado el 22 de octubre de 2023, de <https://prezi.com/nbvubzlwe25f/esquemas-de-traducción/>

Guatemala, K. [@kunusoftguatemala2210]. (2020a, junio 7). OLC: 04 Esquemas de traducción orientada por la sintaxis. YouTube. <https://www.youtube.com/watch?v=5RUshWwCBag>

Guatemala, K [@kunusoftguatemala2210]. (2020b, agosto 9). Compiladores 2: 02 Aplicaciones y esquemas de traducción. YouTube. <https://www.youtube.com/watch?v=8I6uIVJu5Cg>

1.4 Pila Semántica en un Analizador Sintáctico. (s/f). Prezi.com. Recuperado el 22 de octubre de 2023, de <https://prezi.com/zogex9s-t2y8/14-pila-semantica-en-un-analizador-sintactico/>

Acosta, F. [@fernandoacosta7356]. (2020, noviembre 24). Pila semántica en un Analizador Sintáctico. YouTube. <https://www.youtube.com/watch?v=NPZWpIWshcM>

Gómez, José. (s/f). LENGUAJES Y AUTÓMATAS II. Blogspot.com. Recuperado el 22 de octubre de 2023, de <https://joseandresgomezlopez1997.blogspot.com/2019/09/pila-semantica-de-una-analizador.html>

Moralez, L. (s/f). Lenguajes y Autómatas 2. Blogspot.com. Recuperado el 22 de octubre de 2023, de <https://vilmalaura.moralezon.es.blogspot.com/2019/09/pila-semantica-de-un-analizador.html>

Unknown (1970). Crear tabla de símbolos y de direcciones. Recuperado el 17 de Octubre de 2023, de <https://gafetez.blogspot.com/2013/10/crear-tabla-de-simbolos-y-de-direcciones-18.html>

Tabla de Símbolos (s/f). Tabla de símbolos. Recuperado el 17 de Octubre de 2023, de <https://es.slideshare.net/lucita287/tabla-de-simbolos-5662345>

(s/f) 4.5 errores Semánticos. Recuperado el 17 de octubre de 2023, de <http://cidecame.uaeh.edu.mx/icc/mapa/PROYECTO/libro32/45-errores-semanticos.html>

Universidad. (s/f). Capítulo 12. Manejo de Errores y excepciones. Recuperado el 18 de octubre de 2023, de <https://uniwersidad.com/libros/1-algoritmos-python/capitulo-12>

Studocu. (s/f). 1 7 - manejo de errores semánticos - 1 manejo de errores semánticos un error Semántico. Recuperado el 18 de octubre de 2023, de <https://www.studocu.com/es-mx/document/instituto-tecnologico-de-oaxaca/lenguajes-y-automaticas-ii/1-7-manejo-de-errores-semanticos/8450915>

Málaga, E. J. (2008). Teorías de autómatas Y lenguajes formales. Cáceres: Universidad de Extremadura, Servicio de Publicaciones.

Anexos

Videos realizados para esta actividad:

2023-09-18_TNM_CELAYA_LAII-

A_A6_VIDEO_1_EQUIPO_01_20030029_ARROYO_GOMEZ_JOSE_ALFREDOAGO-DIC23

https://drive.google.com/file/d/1gKgdM9z9VN7DydJ35g_4-Fq-OSw_d94J/view?usp=sharing

2023-09-18_TNM_CELAYA_LAII-

A_A6_VIDEO_2_EQUIPO_01_20030029_ARROYO_GOMEZ_JOSE_ALFREDOAGO-DIC23

<https://drive.google.com/file/d/18Sf8jwKRsVTMkjKOPv1ccSP9FA-8qRkv/view?usp=sharing>



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

TEMA

“Orgullosamente Lince”

Nombre de la asignatura:	LENGUAJES Y AUTOMÁTAS II
Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES
Equipo:	1
Autores:	ARROYO GÓMEZ JOSÉ ALFREDO (20030029)
Fecha Realización / Fecha Entrega:	19/10/2023 – 19/10/2023

1

INTRODUCCIÓN

En este documento se muestran las conclusiones, observaciones y datos analizados del video ¿Qué pasa dentro de la mente de un procrastinador? (doblado al español) del canal de YouTube Unilingo.

Este video habla sobre que es lo que un procrastinador piensa en su mente durante el transcurso de los días que pasa cuando tiene que realizar tarea, trabajos, ensayos, proyectos, etc.

En general, un procrastinador piensa que tiene todo el tiempo del mundo y que podrá realizar todo en el ultimo minuto de la carrera. A veces nosotros como estudiantes de la carrera de ingeniería de sistemas llegamos a vivir esos momentos. Ya sea porque pensamos que tenemos mucho tiempo de sobra, porque consideramos que las cosas son fáciles de realizar, o simplemente no nos importa tanto el trabajo y decidimos que podemos dedicar nuestro tiempo a cosas diferentes, que sintamos que son mejores o más entretenidas para nosotros.

2

OBJETIVOS DE LA PRÁCTICA

- Analizar el comportamiento de un procrastinador
- Conocer que pasa por la mente de un procrastinador
- Realizar un análisis y compresión de un video de interés

3

MARCO TEÓRICO

¿Qué es un procrastinador?

Citando a la CIFPA:



TECNOLÓGICO
NACIONAL DE MÉXICO®



“PROCRASTINAR: Es un término que tiene su origen en la palabra latina «procrastinare» y significa «diferir, aplazar». Según la RAE, se trata de una voz creada en su origen a partir del adverbio «cras» (mañana, el día siguiente), del cual «procrastinar» toma su significado de «dejar para mañana, posponer, aplazar».”

(Sastre, 2022)

Podemos entender que un procrastinador es aquella persona que posterga, desplaza, atrasa, interrumpe, etc., sus actividades para otro momento. Probablemente esto lo realice de forma continua, por lo tanto, se llegaría a crear un hábito que consistiría en dejar las actividades a realizar para otro momento, a tal punto de perder el control y concluyendo en no terminar o ni siquiera iniciar la actividad, esto es algo muy grave si hablamos de que las personas tienen actividades importantes todos los días.

4

MATERIALES, EQUIPOS Y RECURSOS EN GENERAL A UTILIZAR

- **Equipos de cómputo (Nuestros equipos son de gama media)**
 - Procesadores Intel Core i3-i5
 - Memoria RAM de 8 – 16 GB
 - Discos duros de estado sólido con almacenamiento desde 180 GB hasta 500 GB
- **Internet (Dependiendo del proveedor y paquete contratado)**
- **Navegador web (Google, Opera, Mozilla, Edge, etc.)**
- **Editores de texto de ofimática (Word)**

5

DESARROLLO

El procrastinador promedio tiende a desplazar las actividades que debe realizar en un periodo de tiempo específico. ¿Como es posible que esto pase cuando a alguien se le indica?

Esto puede ser debido a diferentes factores como:

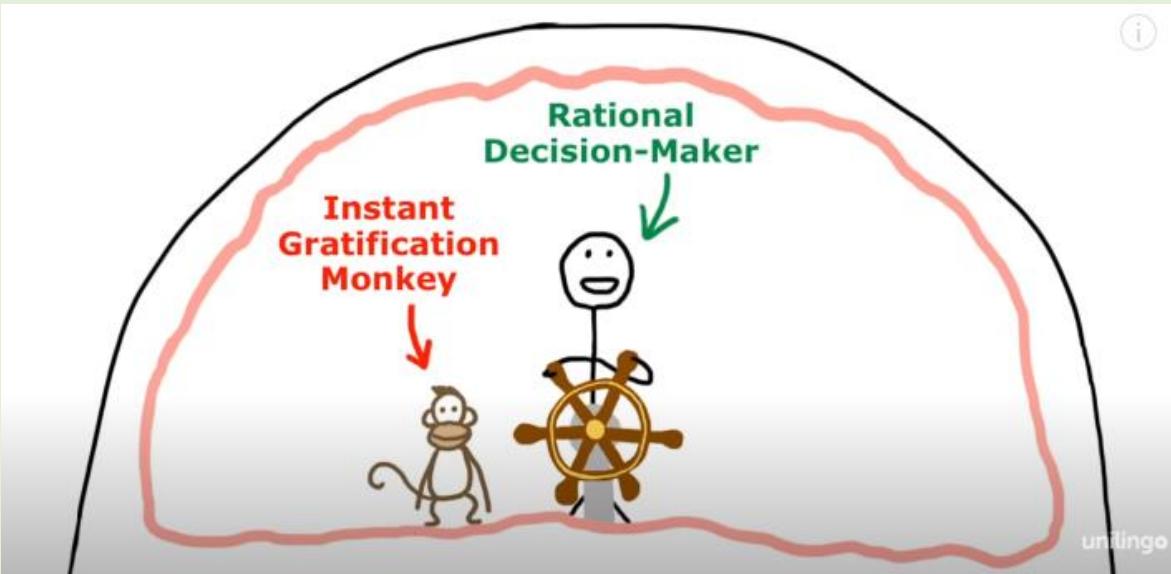
- **Distracciones**
- **Amenazas externas**
- **Trabajos más importantes**
- **Ocio**
- **Miedo al fracaso**
- **Flojera**
- **Cansancio**



TECNOLÓGICO
NACIONAL DE MÉXICO®



¿Cómo se vería el cerebro de un procrastinador?



- Observamos a nuestro comandante que es racional y toma nuestras decisiones
- Un mono que solo quiere vivir tranquilo y que nadie lo moleste, le gusta la diversión y el exceso.



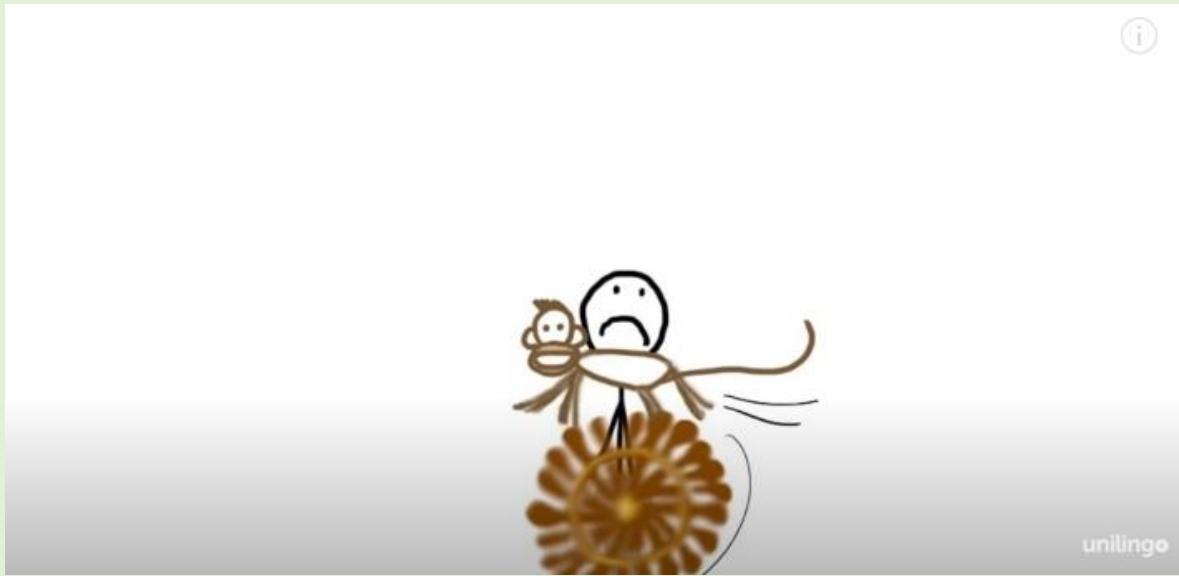
- No le importa si quieras trabajar, solo le interesa vivir la vida loca.
- Este monito evita que trabajemos en lo importante y empecemos a procrastinar



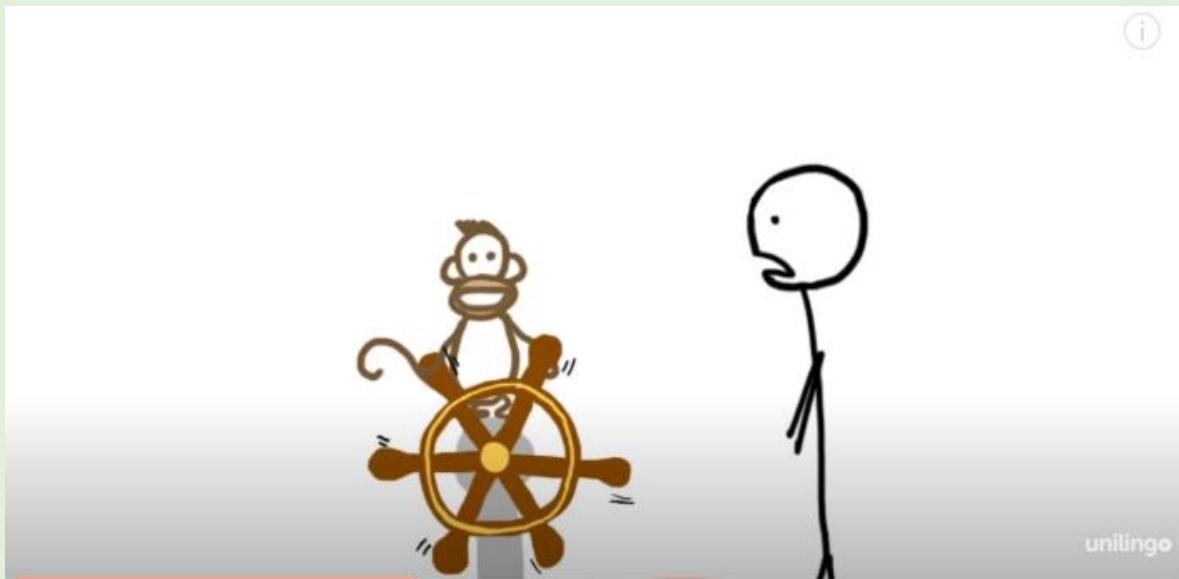
TECNOLOGICO
NACIONAL DE MEXICO®



César García Hernández



- Poco a poco el mono evita que nosotros podamos tomar las decisiones importantes y toma el control absoluto de nuestro raciocinio y toma de decisiones.

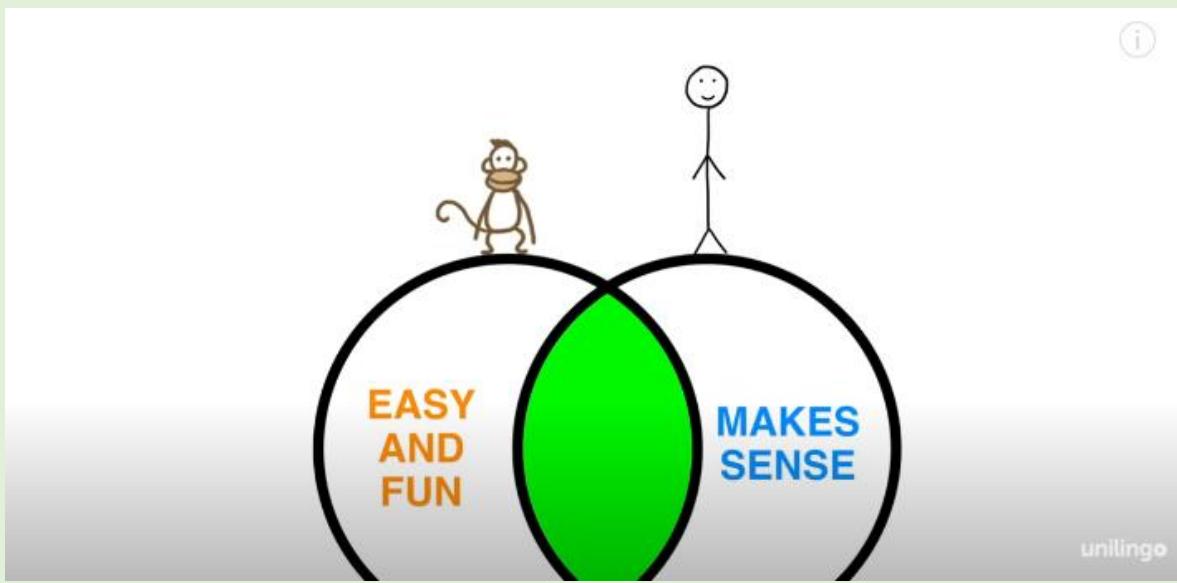


- El mono: “No time for work just for fun”.
- No hay que trabajar solo jugar y dormir je, je.



- A veces quisiera ser el mono, pero la vida no es así.
- Tenemos que trabajar para poder sacar nuestras vidas a delante para poder tener sustento y con que vivir una vida plena.

Uno pensaría que esta completamente mal disfrutar y pasar un rato de ocio y entretenimiento, sin embargo, lo importante es hallar un equilibrio entre tomar decisiones correctamente pensadas y momentos de descanso. No podemos trabajar duro si es que estamos cansados, y tampoco podemos dejar de trabajar por 2 días completos simplemente porque estamos cansados.



Si nos vamos por el lado del ocio y la vagancia obtendremos diversión, entretenimiento a costa de frustración, miedo, ansiedad, cansancio y estrés. Finalmente, no completaríamos nuestro trabajo, es por ello que debemos evitar este tipo de comportamiento.

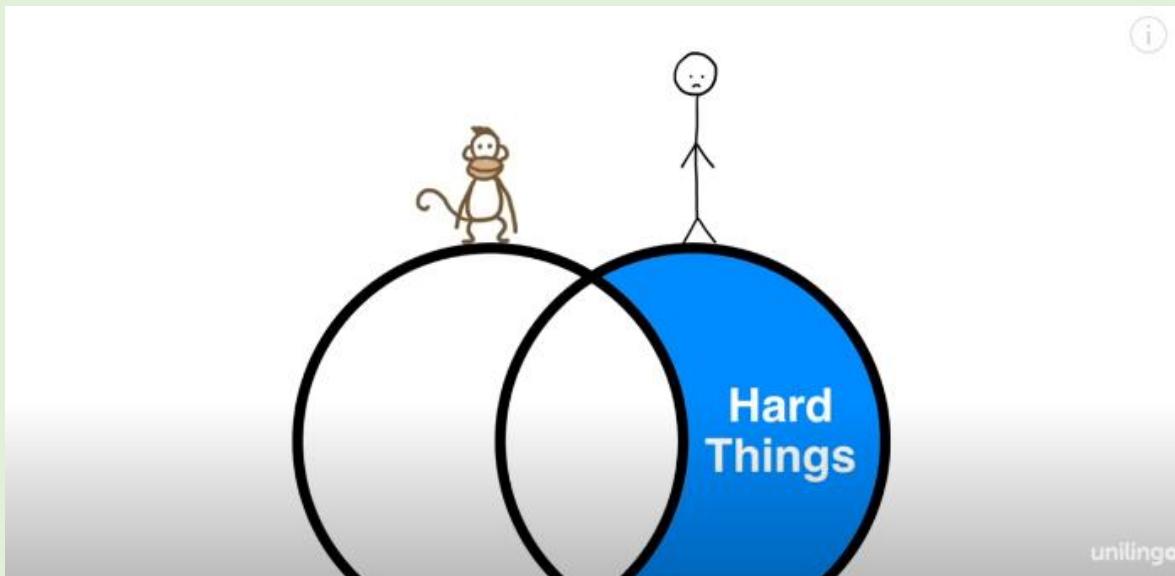


Si seguimos este camino nos encontraremos con el monstruo del pánico:

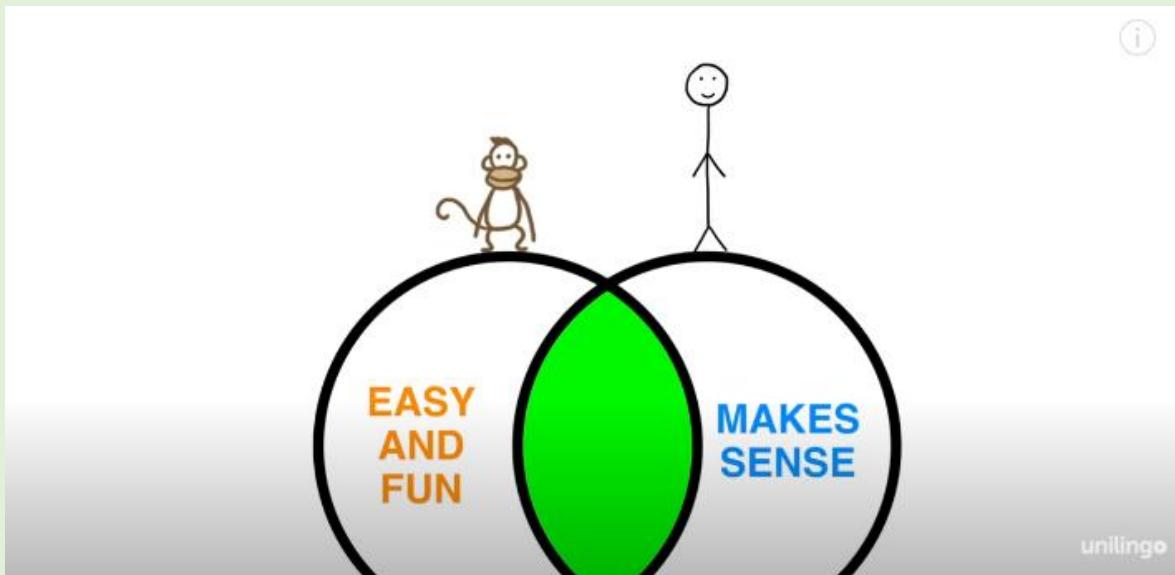


- Este monstruo provoca que el mono se vaya a un árbol lejos de nosotros
 - Nosotros empezamos el trabajo, pero ya es muy tarde, tenemos todo el peso del mundo y poco tiempo para terminar el trabajo.
 - Ahora estamos estresados y con mucho miedo

Curiosamente si nos vamos por el camino de hacer las cosas difíciles nosotros obtendremos cansancio, esfuerzo, dolor, sudor, pero a costa de terminar y obtener un trabajo completo y merecedor de sentir orgullo y tranquilidad de que nosotros hemos completado una tarea o proyecto de forma satisfactoria.



Por lo tanto, es importante centrarnos en encontrar un buen equilibrio que nos permita descansar un poco y avanzar más en nuestros trabajos.



- Esta imagen me gusto bastante porque representa precisamente eso.

6

BITÁCORA DE INCIDENCIAS

Fecha	Hora	Descripción de la incidencia	Solución
19/10/2023	9:00	Teníamos clase a la hora que se asignó la tarea	La hicimos en la otra clase ja, ja.

7

OBSERVACIONES

Creo que el video me ayudo a reflexionar sobre mi comportamiento a la hora de hacer tarea, trabajos o proyectos. Usualmente trato de no procrastinar, y siento que quiero hacer otras cosas, no siempre son cosas que no tengan que ver con entretenimiento u ocio, a veces es porque quiero aprender otra cosa, como programación en otro lenguaje, técnicas de diseño de interfaces u otros temas relacionados. Pero, aun así, tiendo a procrastinar si dejo lo importante al final, lo cual, trato de que no pase, porque no me gusta sentirme frustrado por no avanzar al menos un poco cada día en mis tareas, creo que de mi parte y de mi equipo lo hemos demostrado con los trabajos que realizamos ya que no son trabajos que salgan de 1 día para otro. Son trabajos que son resultado de varios días de estudio y aplicación. Aun así, quiero seguir mejorando para poder aprender más y terminar todas mis tareas de la mejor manera posible.

8

CONCLUSIONES

En conclusión, dentro de la mente de un procrastinador, se entrelazan complejas emociones como el miedo al fracaso, el perfeccionismo, la falta de motivación y problemas de autocontrol. La procrastinación surge como una respuesta a estas tensiones emocionales y puede llevar a una constante postergación de tareas importantes. Superar este hábito requiere enfrentar estos miedos y aprender a administrar el tiempo de manera efectiva, estableciendo metas claras y manejando las expectativas, para finalmente recuperar el control y la productividad en la vida diaria.

9

REFERENCIAS BIBLIOGRÁFIA

- Sastre, T. A. (2022, enero 28). Procrastinar... ¿De qué me estás hablando? CIFPA; CENTRO DE INNOVACIÓN PARA LA FORMACIÓN PROFESIONAL DE ARAGÓN. <https://cifpa.aragon.es/procrastinar-de-que-me-estas-hablando-2-2/>

- 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>

10

ANEXOS

Video proporcionado por el profesor:

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

César García Hernández



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CELAYA

TEMA

“Orgullosamente Lince”

Nombre de la asignatura:	LENGUAJES Y AUTOMÁTAS II
Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES
Equipo:	1
Autores:	GARCÍA HERNÁNDEZ CÉSAR (20030853)
Fecha Realización / Fecha Entrega:	19/10/2023 – 19/10/2023

1

INTRODUCCIÓN

Durante el desarrollo del video, pudimos analizar la exposición que el ponente que me parece se llamaba Tim Urban, mediante el canal de Youtube Unilingo, hace sobre qué pasa en la mente de un procrastinador, pues a pesar de que este individuo tiene que cumplir con diversas tareas (llámese trabajos, tareas, estudio, etc.) prefiere destinar su tiempo a otras actividades de manera que impide pueda cumplir satisfactoriamente con ellas.

Un procrastinador a mi punto de vista es una persona que carece de responsabilidad y compromiso con sus deberes, además que tiene una escasa planificación de sus labores. Me siento medianamente identificado pues si bien, logro cumplir con las obligaciones, la planificación y destinación de tiempos ocupada no es la óptima y aquí encuentro un área de oportunidad en mi persona.

2

OBJETIVOS DE LA PRÁCTICA

- Comprender qué pasa por la mente de un procrastinador
- Sintetizar la información contenida en el video
- Aplicar el conocimiento a tu vida diaria

3

MARCO TEÓRICO

¿Qué es un procrastinador?

Entendemos a un procrastinador como aquella persona que demora las actividades o compromisos con los que tiene que cumplir al destinar el tiempo que le tomaría hacerla para hacer otras actividades (sea por placer y entretenimiento, ocio vaya).



TECNOLÓGICO
NACIONAL DE MÉXICO®



Es una problemática en los tiempos actuales pues vivimos en un mundo lleno de distracciones, donde además el acceso al fácil entretenimiento nos ha “educado” para tomar los caminos fáciles y creación de malos hábitos. La procrastinación en complemento con este estilo de vida ha llevado a que nosotros como nuevas generaciones seamos víctimas de “ser robados por nuestro tiempo”.

Aunque no todo está perdido, pues siempre se puede retomar hábitos de planificación que lleven a disminuir el desperdicio de tiempo en actividades meramente de ocio que retrasan con la entrega de obligaciones.

4

MATERIALES, EQUIPOS Y RECURSOS EN GENERAL A UTILIZAR

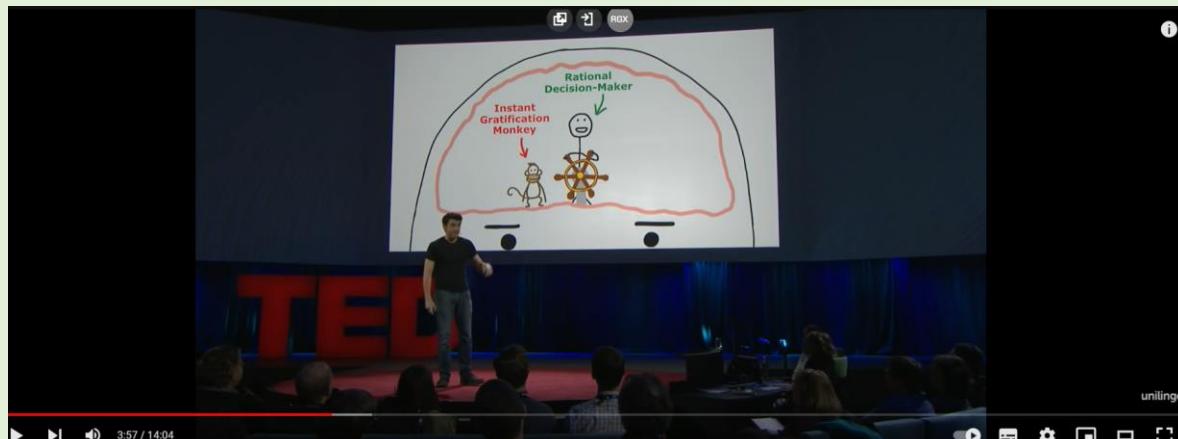
- **Equipos de cómputo (Nuestros equipos son de gama media)**
 - Procesadores Intel Core i5
 - Memoria RAM de 8 GB
 - Discos duros de estado sólido con almacenamiento de 500 GB
- **Internet (Dependiendo del proveedor y paquete contratado)**
- **Navegador web (Google, Opera, Mozilla, Edge, etc.)**
- **Editores de texto de ofimática (Word)**

5

DESARROLLO

¿Cómo se ve interíormente el cerebro de un procrastinador?

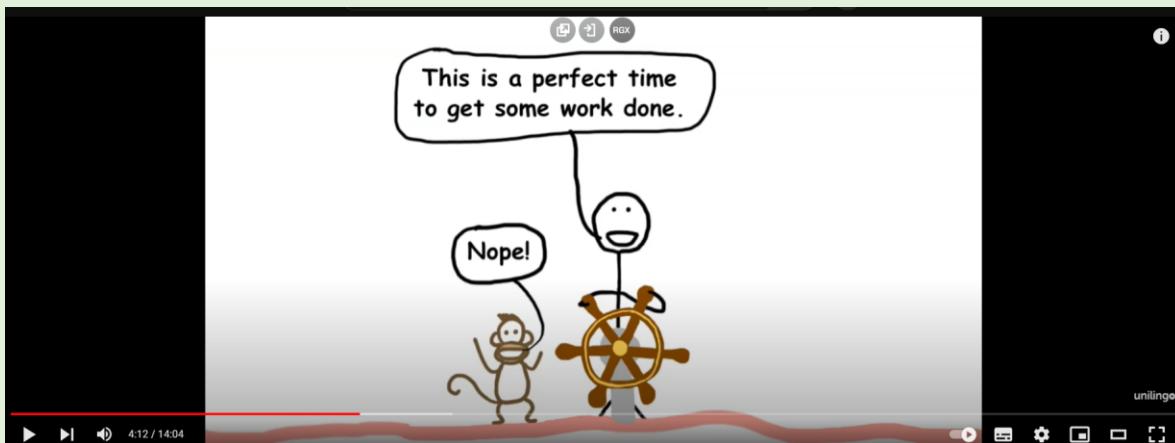
Mediante algunas capturas de pantalla de imágenes mostradas en el video se explica el cómo funciona el cerebro de un procrastinador.



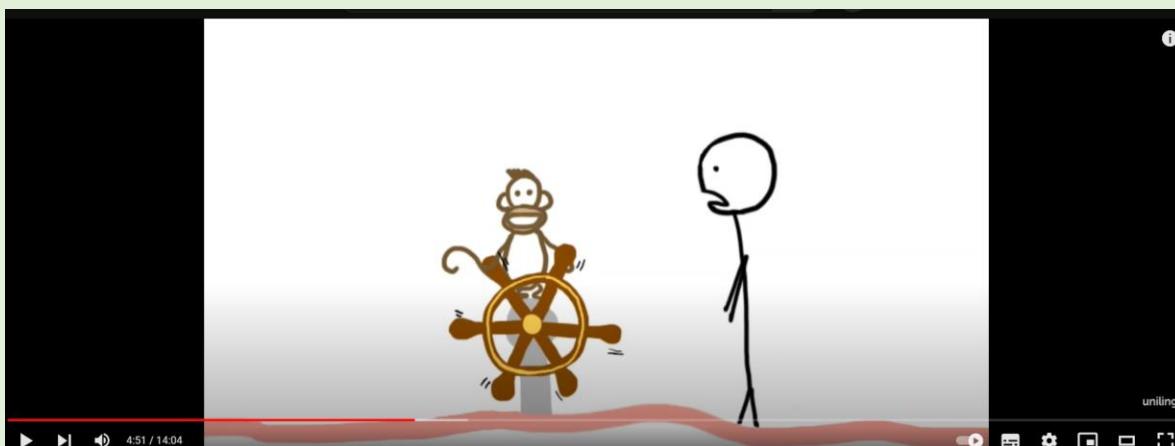
TECNOLÓGICO
NACIONAL DE MÉXICO®



- Dualidad de emociones, una persona racional y un mono que prefiere la diversión fácil.



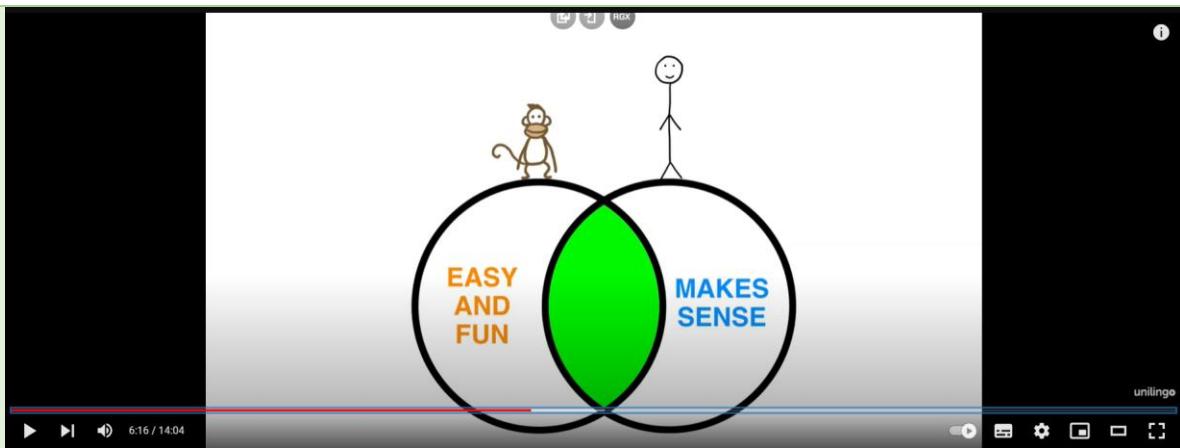
- En vez de ocupar el tiempo de manera productiva, este mono te lleva a malgastar tiempo.



- Cuando menos lo esperas, el mono poco chambeador tiene por completo el control.

La vida no se trata de diversión y libertinaje, como seres pensantes gozamos de derechos que te permiten llevar a cabo actividades para tu entretenimiento, pero no todo es eso. Uno piensa que el placer solo se encuentra en actividades como estas, pero, dentro de las responsabilidades también encontramos satisfacción al cumplir con lo que nos toca, además de que se toma un aprendizaje significativo.

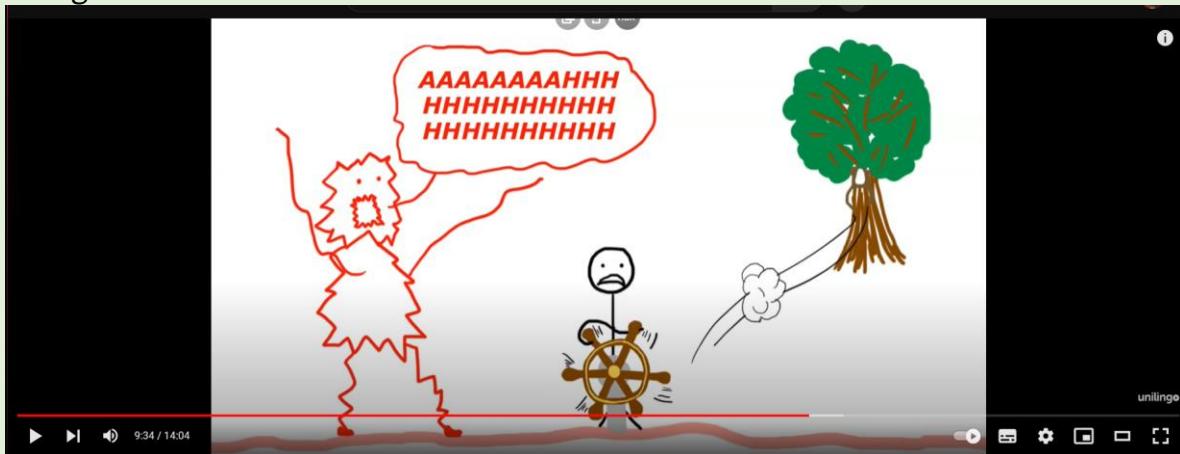
No es malo divertirte, no es malo pasar entretenimiento fácil, pero debemos regular los tiempos que tomamos en cada una de nuestras actividades. En el desarrollo del video se muestra una imagen que me queda como aprendizaje, la siguiente.



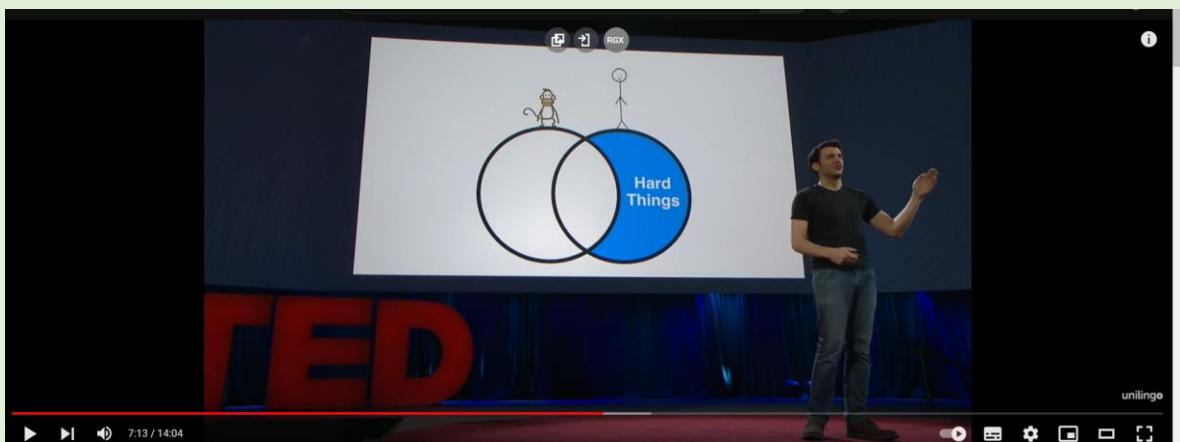
Este es el equilibrio perfecto entre las actividades de diversión y las de toma de decisiones pensada.

Después, se muestra que tanto mucha diversión como mucha carga de trabajo, nos hace la vida más pesada.

La siguiente imagen muestra el caos de mucha diversión:



Y la siguiente una sobre carga de trabajo:



TECNOLÓGICO
NACIONAL DE MÉXICO®



Me imagino la representación del pensamiento chino del Ying y el Yang, donde todo se encuentra en equilibrio. Ni mucho de algo, ni poco de lo otro. Todo debe estar en perfecto equilibrio, como debe ser.

6

BITÁCORA DE INCIDENCIAS

Fecha	Hora	Descripción de la incidencia	Solución
19/10/2023	8:50-12:00	La tarea se dejó a las 8:50, mi horario estudiantil presenta otras materias en el horario a la cual se pidió la asignación. La asignación de la hora de entrega de la tarea fue incorrecta a mi parecer.	La actividad se realizó durante el horario de otra asignatura.

7

OBSERVACIONES

Se llevó a cabo una reflexión y análisis personal sobre el comportamiento que tengo, tiendo a reemplazar horas en las cuales puedo llevar a cabo actividades por entregar, sin embargo, prefiero dedicar tiempo a otros pasatiempos como ver alguna serie o jugar videojuegos.

No hago referencia a que todo el tiempo tiene que ser trabajo, pues parte de la salud mental es dedicar tiempo libre a actividades que liberen la carga emocional para evitar problemas, pero, en ocasiones estas actividades toman un tiempo que no es correcto.

Se planea tomar compromiso a la realización de tareas de mejor manera en cuestión de administración de tiempo.

8

CONCLUSIONES

Para superar la procrastinación, es importante identificar patrones de pensamiento y emociones, y desarrollar estrategias para cambiarlos. Esto puede incluir el establecimiento de metas que no superen la realidad, la gestión y administración del tiempo, la eliminación de distracciones, la práctica de la autorregulación y la búsqueda de apoyo o motivación externa. Somos individuos únicos, por lo que las estrategias para superar la procrastinación pueden variar según sus circunstancias y necesidades individuales, en mi caso la autorregulación de fuentes de entretenimiento y la administración del tiempo creo que sería un buen inicio



César García Hernández

para tomar mejor hábitos.

9

REFERENCIAS BIBLIOGRÁFIA

- 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>

10

ANEXOS

Video a consultar brindado por el maestro:
<https://www.youtube.com/watch?v=PG6oFKoa1NA>

César García Hernández



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CELAYA

TEMA

“Orgullosamente Lince”

Nombre de la asignatura:	LENGUAJES Y AUTOMÁTAS II
Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES
Equipo:	1
Autores:	GASCA PALACIO JESUS FERNANDO (20030606)
Fecha Realización / Fecha Entrega:	19/10/2023 – 19/10/2023

1

INTRODUCCIÓN

En el siguiente documento realizare mi análisis del video ¿Qué pasa dentro de la mente de un procrastinador? (doblado al español) del canal de YouTube Unilingo.

En el video se trata el tema de que es lo que pasa en la mente de una persona que procrastina, a la hora de llevar a cabo actividades como tareas, trabajos, ensayos, proyectos, etc.

Podemos pensar que un procrastinador no tiene una noción cierta del tiempo y que las cosas pueden llevarse a cabo en un tiempo muy corto, como al último momento.

Esta reflexión pienso que queda bien a cualquier persona, ya sea estudiante, trabajador, etc., porque pensamos que el tiempo es muy largo, y las actividades se llevan a cabo rápido, por eso se llevan acabo otras actividades antes que la prioridad.

2

OBJETIVOS DE LA PRÁCTICA

- Comprender el comportamiento y las tendencias de una persona procrastinadora
- Que elementos afectan el comportamiento de procrastinar
- Que pasa en la mente durante este fenómeno.

3

MARCO TEÓRICO

El procrastinador es una persona que deja de lado las actividades que tiene que cumplir, ya sea por comodidad, falta de motivación, u otros factores, la idea es dejar de lado la actividad que mas va a demandar y en su lugar llevar a cabo alguna otra que no tenga un propósito o un objetivo que cumplir, tareas que no dejan nada de valor.

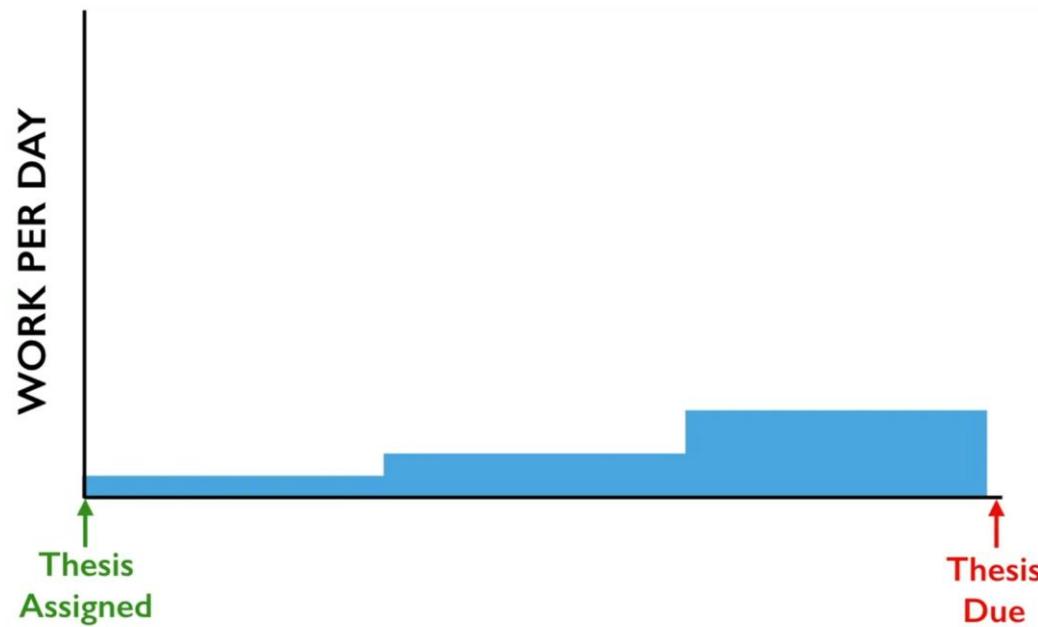


TECNOLÓGICO
NACIONAL DE MÉXICO®



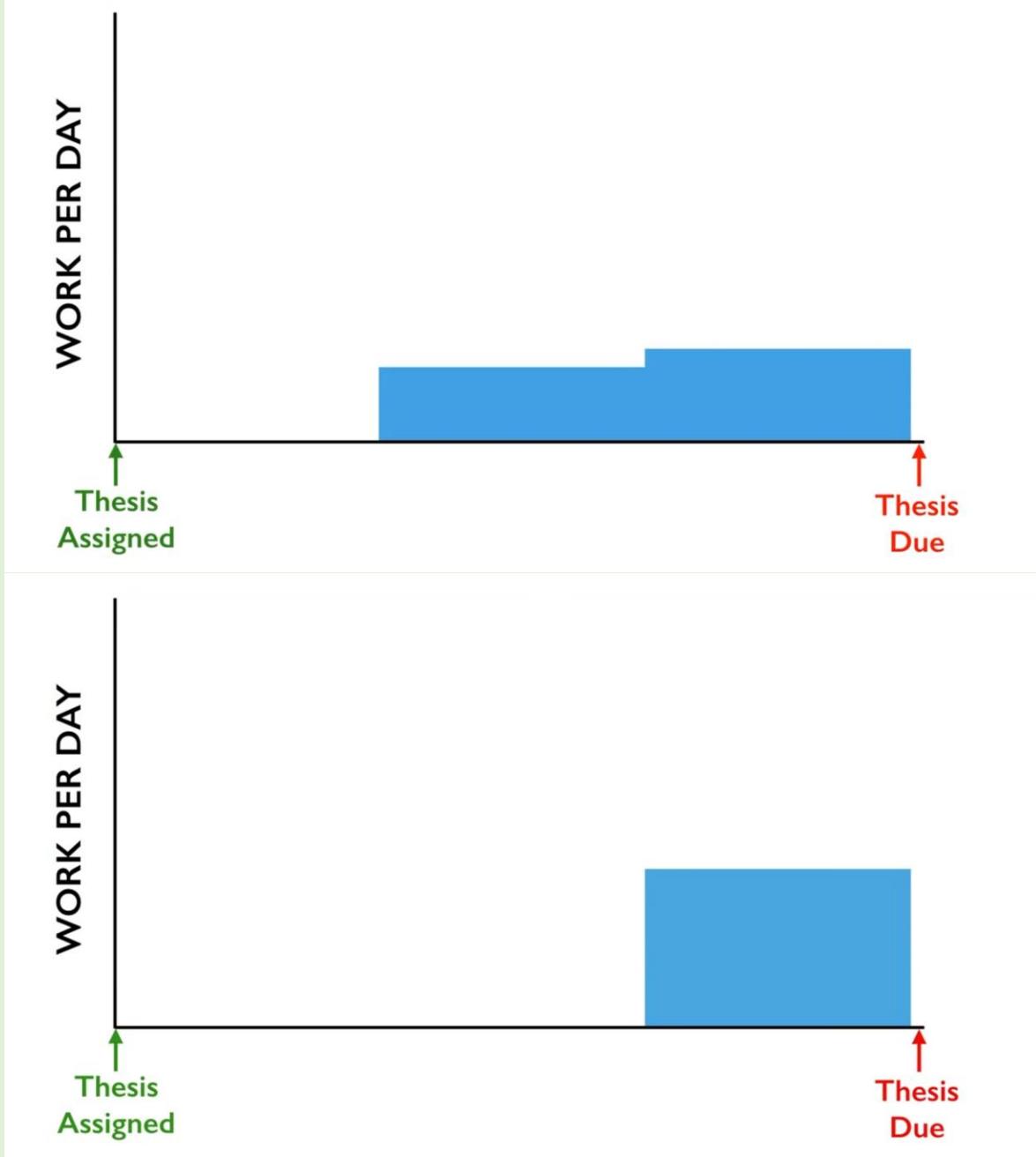
- **Equipos de cómputo**
 - Procesador Ryzen 7
 - Memoria RAM de 16 GB
 - Discos duros de estado sólido con almacenamiento desde 180 GB hasta 500 GB
- Internet (Dependiendo del proveedor y paquete contratado)
- Navegador web (Google, Opera, Mozilla, Edge, etc.)
- Editores de texto de ofimática (Word)

Durante la preparación para llevar a cabo alguna actividad, este es el ejemplo de una planeación adecuada.



Pero lo mas acercado a la realidad son los siguientes casos:

Cesar García Hernández

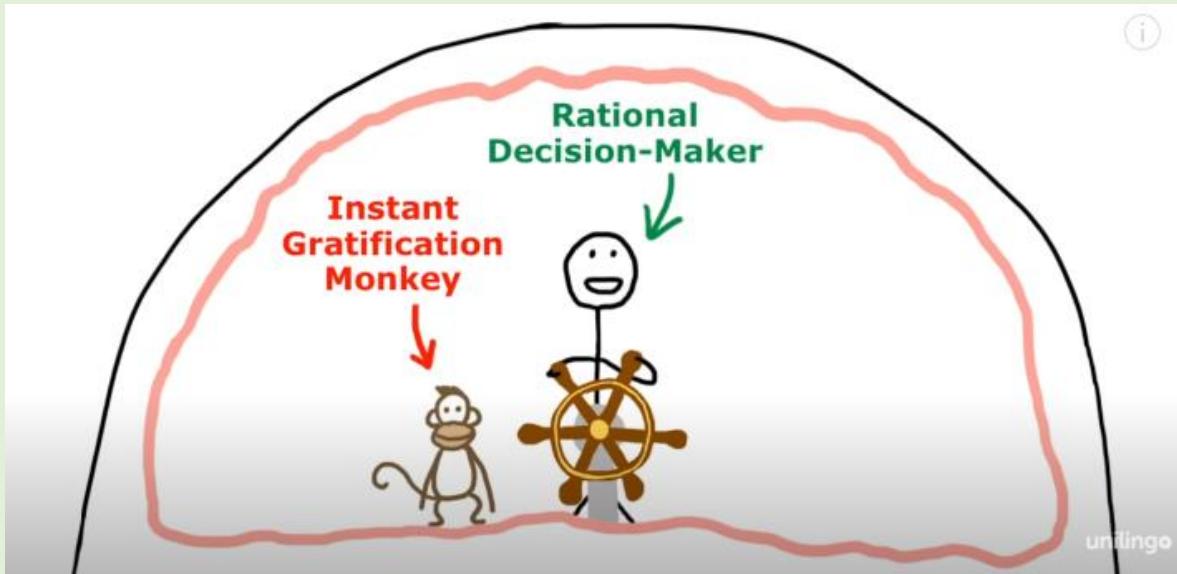


Desafortunadamente este es un efecto muy común ya que la planeación cumple con criterios para llevar un orden y una organización, pero es difícil cumplir con esa planeación y en su lugar se lleva acabo el trabajo de último momento.

¿Hay diferencias entre las personas que procrastinan y las que no?

La respuesta está en la siguiente imagen





Tenemos dos personajes un comandante que es racional y toma nuestras decisiones y un mono que solo quiere vive el ahora, no se preocupa por el pasado ni futuro .



Podemos ver que el mono que nos representa mejor, se encuentra sereno y tranquilo ya que las actividades que tenemos que llevar acabo por el momento no se cuentan cerca, asi que el pienso que aun hay tiempo, pero cuando se acerca el momento de cumplir con las actividades el mono entra en modo de alerta como en la siguiente figura:

i



unilingo

Este representa que este momento exacto advierte el mono que debe de llevar a cabo ya de manera desesperada las actividades que antes pudo a haber llevado a cabo de manera más tranquila.

Pero ¿Qué lo activo a tomar acción?

Pues la personificación de la preocupación, el tiempo de entrega, entre otros factores, todo esto lo podemos ver como el monstruo de pánico, el mono le teme a este monstruo .

i



unilingo

Este monstruo representa nuestro subconsciente de que somos capaces de en los últimos momentos de preocupación darnos una alerta de que ya es hora por que ya no hay más oportunidad.

Es como esa ultima alarma que nos advierte que ya es tarde y que si no es ahora ya no se solucionara.

La idea es que este monstruo no nos asuste cuando nos representamos como el mono, pero lograr esto es difícil, es mucho trabajo.

S. Hernández

César García Hernández

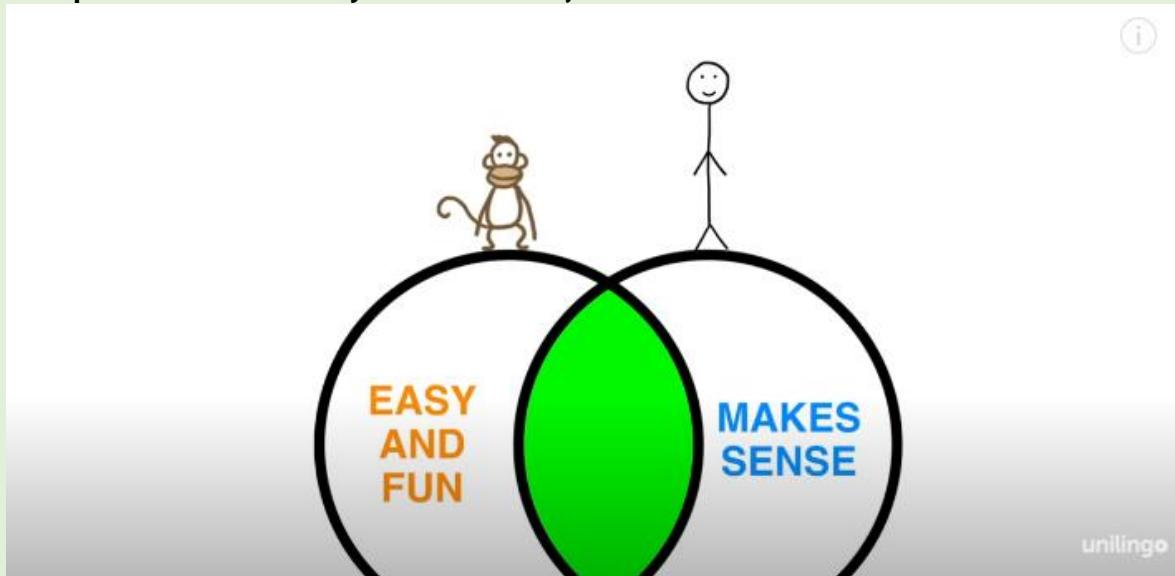


TECNOLÓGICO
NACIONAL DE MÉXICO®



Lo ideal sería que tengamos un equilibrio entre el mono y el comandante, pero repito es una tarea difícil de lograr, como todo se debe practicar y practicar hasta conseguirlo.

Evitando ya sea procrastinar mucho y también trabajar en exceso en los últimos momentos.



6

BITÁCORA DE INCIDENCIAS

Fecha	Hora	Descripción de la incidencia	Solución
19/10/2023	9:00	El tiempo de la actividad no fue el correcto.	Realizarlo con otras actividades en otras materias.

7

OBSERVACIONES

El video representa de forma muy exacta y entendible todo el proceso que se lleva a cabo de manera mental, de el proceso que se lleva a cabo cuando una persona procrastina, los ejemplos que se llevan acabo son lo suficiente útiles y demostrativos.

Las representaciones me parecieron agradables, para conceptualizar los términos y definiciones ya que de esta forma se hace más eficiente la comprensión.

8

CONCLUSIONES

El comportamiento del que se habla en el video, a mi punto de vista es un efecto generacional, ya que desde que existen las redes sociales, hay un gran efecto de procrastinación, ya que es más cómodo no llevar acabo las actividades urgentes y en su lugar hacer algo que no aporta ningún valor, todo lo que se menciona en el video me representa en algunos momento ya que en ocasiones sin darme cuenta postergo actividades pensando que las puedo realizar rápido y cuando llega el momento crítico de entrega ahí si realizo todo.

Me ayudo a reflexionar sobre como combatir ese efecto que no lo hago tan seguido, pero en ocasiones si me representa.



- 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>

Video de referencia:

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



TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE CELAYA

TEMA

“Orgullosamente Lince”

Nombre de la asignatura:	LENGUAJES Y AUTOMÁTAS II
Carrera:	INGENIERÍA EN SISTEMAS COMPUTACIONALES
Equipo:	1
Autores:	GONZÁLEZ MANCERA CHRISTIAN MANUEL (20030115)
Fecha Realización / Fecha Entrega:	19/10/2023 – 19/10/2023

1

INTRODUCCIÓN

El video nos cuenta sobre un tema general de lo que es un procrastinador, dicha explicación es explicada de manera un poco “cómica” al poner imágenes y diseños como lo es un mono y un monstruo. Se nos dice que un procrastinador es una persona que tiene el hábito de posponer tareas o actividades importantes, a menudo en favor de realizar actividades menos urgentes o más placenteras.

2

OBJETIVOS DE LA PRÁCTICA

Ver el video y obtener la idea general de lo que es un procrastinador.

3

MARCO TEÓRICO

4

MATERIALES, EQUIPOS Y RECURSOS EN GENERAL A UTILIZAR

Celular
YouTube
Computadora

5

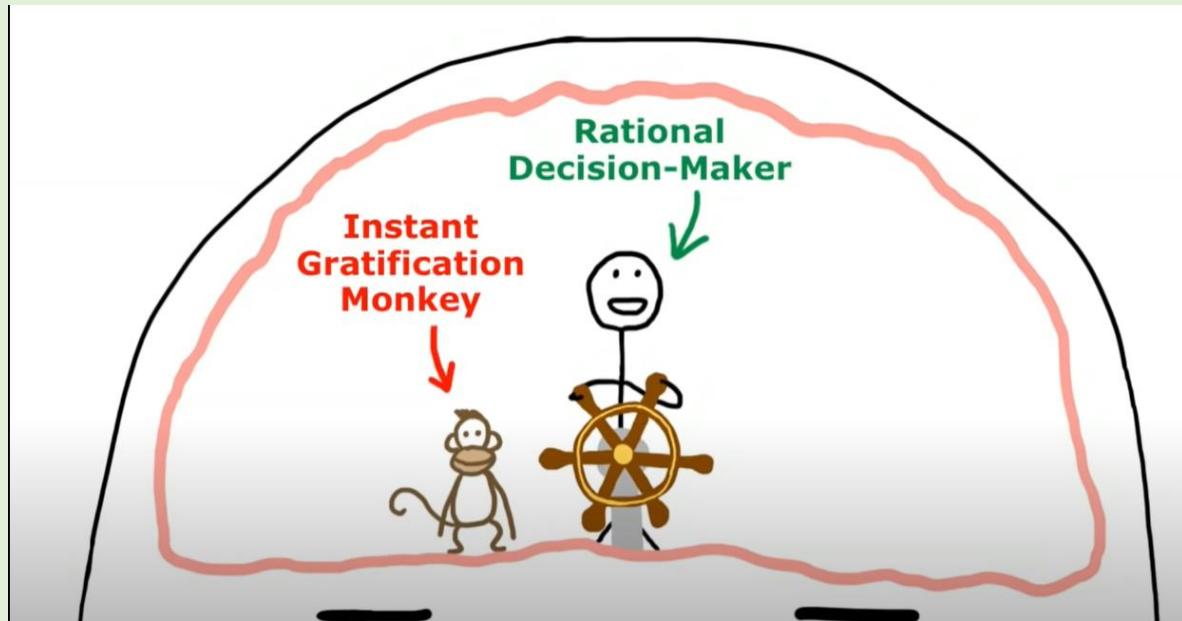
DESARROLLO

Se nos dice como un procrastinador tiene en su mente dos atributos esenciales a la hora de un trabajo un manejador de decisiones racional y uno de instinto de gratificación. Por lo regular este ultimo representado como un mono toma el control en muchas ocasiones.

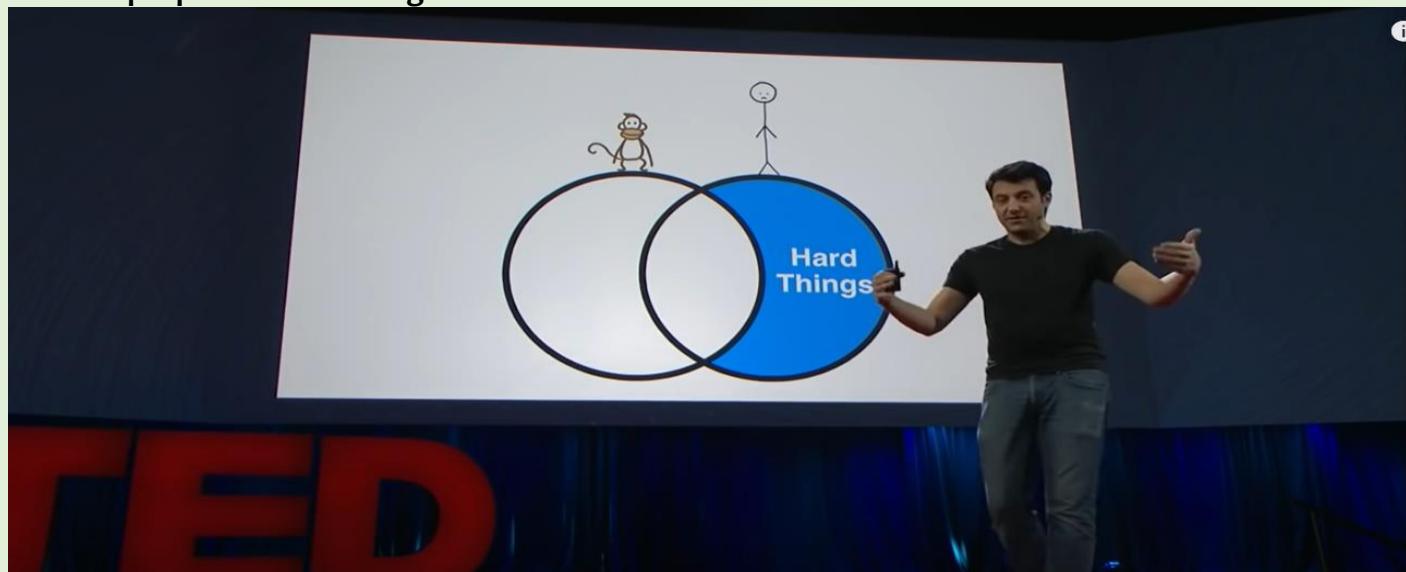


TECNOLÓGICO
NACIONAL DE MÉXICO®





La procrastinación puede ser perjudicial en la vida cotidiana, ya que puede llevar a retrasos en la finalización de proyectos, aumentar el estrés y reducir la productividad. Muchas personas luchan con la procrastinación en algún momento de sus vidas, pero para algunas personas, se convierte en un patrón de comportamiento crónico que puede afectar negativamente su calidad de vida.



Pero los procrastinadores tienen alguien que los ayuda en la toma de decisiones y hacer las cosas, el monstruo del pánico.

Este monstruo actúa por lo regular en el ultimo momento y es una especie de amigo.





César García Hernández



6

BITÁCORA DE INCIDENCIAS

Como el video duro 14 minutos tuve que tomar tiempo de la siguiente clase.

7

OBSERVACIONES

8

CONCLUSIONES

Superar la procrastinación a menudo implica desarrollar habilidades de gestión del tiempo, establecer metas claras, dividir las tareas en pasos más pequeños y encontrar formas de motivarse a uno mismo para comenzar y completar las tareas importantes.

9

REFERENCIAS BIBLIOGRÁFIA

Mi escape de Corea del Norte | Hyeonseo Lee | Doblado al Español
<https://youtu.be/wiwsA9atRbg>

10

ANEXOS

César García Hernández