INSTITUTO TECNOLOGICO DE TEPIC

LENGUAJES Y AUTOMATAS I

T6-ACTIVIDAD 5. PRESENTACIÓN Y ENTREGA FINAL DEL PROYECTO.

TEMA 6. MAQUINA DE TURING

LOS TILAPIOS
VALDEZ PONCE JOSE ALBERTO 20400833
SANDOVAL GARCIA YAEL ALEJANDRO 20400824
GARCIA TAPIA JOSE GABRIEL 20400741
CABRAL MACHUCA ALEJANDRO AARON 20400706
DIAZ CANALES JOSE LINO 20400727
PLATA CABELLO JOSE MANUEL 20400799

31 DE MAYO DEL 2024

Link del repositorio:
https://github.com/YASG07/LYA_HandTech
Link del video:
https://drive.google.com/file/d/1EPelIWQfQ1BnPbtc5n0u5Z5Ac2atRkZi/view?usp=sharing

Descripción de la actividad.

Actividad 5.Elaborar el Reporte Final del Proyecto.

Subtemas a desarrollar.

6.1 Definición formal MT.
6.2 Construcción modular de una MT.
6.3 Lenguajes aceptados por la MT.

Abstract.

This document contains the development of the preliminary entitled “HandTech”, which will allow the development of hand prostheses to persons that have a disability, or also for industrial use. This covers the design of the language that is going to be running on the future compiler for the project being developed under the name of “HandTech”. Remarking the fact of the compiler's development will be in JavaScript language and Electron.js framework.

Contenido.

1. Introducción.

En esta etapa, se establecen los fundamentos del proyecto, comenzando por comprender el alfabeto utilizado, que abarca desde letras y números hasta símbolos y caracteres especiales necesarios para la programación. Además, se definen tipos de datos para manipular información, junto como operadores aritméticos, relacionales y de asignación para realizar operaciones específicas.

Las palabras reservadas y la estructura del programa proporcionan un marco sólido para el desarrollo de funciones y algoritmos complejos en "HandTech". En resumen, este diseño detallado establece los cimientos para el sistema de prótesis robóticas, que promete revolucionar la vida de sus usuarios.

2. Antecedentes.

El desarrollo de prótesis ha crecido mucho a lo largo de los años, con la introducción de diferentes tipos de prótesis que llegan a reemplazar diferentes partes del cuerpo humano, la tecnología implementada en dichas prótesis a ido en aumento con el pasar de los años, y actualmente se maneja mucho la IA la cual se ha estado implementado en prótesis robóticas. En la actualidad existen diferentes tipos de prótesis, ya sean prótesis removible de resina, de codo, de hombro etc. las cuales se adaptan a las necesidades del usuario final.
Avances tecnológicos en el área de las prótesis humanas.
Como se mencionó antes a día de hoy ha habido muchos avances tecnológicos en prótesis y una de ellas es la IA la cual nos permite la adaptación a necesidades del usuario y el control mediante señales bioeléctricas. El uso de sensores que permitan la detección, posición, fuerza y presión que tendría nuestra prótesis. También el uso de bioingeniería que integra componentes mecánicos y eléctricos de la prótesis con el cuerpo humano. Otro avance tecnológico que facilita y economiza la creación de prótesis son las impresiones 3D, las cuales ayudan en la fabricación de prótesis personalizadas, livianas y por supuesto a un bajo costo.
Impacto social y económico del proyecto.
Las prótesis tienen un gran impacto sobre aquellas personas que carecen de una extremidad o amputación. Las prótesis le ayudan a la persona a recuperar su independencia y funcionalidad, así como a realizar tareas cotidianas con mayor facilidad y por último mejorar el autoestima y calidad de vida de los afectados.

Estas prótesis, ofrecen una solución a las personas que carecen de una extremidad del cuerpo humano. Sin embargo, la programación de dichas prótesis siguen resultando un desafío, ya que se requiere de un conocimiento técnico especializado el cual no siempre es accesible para los usuarios finales. Con esto en mente, este proyecto soluciona la necesidad de conocimientos tan avanzados y especializados, creando un lenguaje complejo pero fácil de entender para el usuario final.

Algunos casos de éxito de prótesis son la mano robótica DEKA Luke desarrollada por DEKA research and Development Corporation, las prótesis biónica de bebionic desarrollada por Ottobock, una empresa alemana líder en tecnología de rehabilitación y las prótesis mioeléctrica Michelangelo de la misma empresa anteriormente mencionada.

3. Justificación.

The implementation of this project aims to create a functional prototype of a hand prosthesis, which in the future could be used by individuals who have lost or need to recover part of the functionality and/or mobility of said limb

4. Objetivos.

4.1 General.

Desarrollar un nuevo lenguaje de programación, en específico diseñado para el desarrollo de prótesis de manos, con el fin de darle distintos usos, que a su vez permitirá al programador generar patrones de movimiento.

4.2 Específicos.

- Facilitar la programación de patrones de movimiento.
- Desarrollar una sintaxis sencilla e intuitiva.
- Integrar capacidad de detección de objetos.

5. Metodología.

Diseño Detallado.

Alfabeto.

Para comenzar es importante señalar el alfabeto sobre el cual trabajara “HandTech”, el cual es el siguiente:
Letras mayúsculas: A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z.
Letras minúsculas: a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
Dígitos: 0,1,2,3,4,5,6,7,8,9.
Símbolos de agrupación: (), {}.
Operadores aritméticos: +, -, *, /, ++, --
Operador de asignación: =
Operadores relacionales:

- Operador relacional Mayor o Igual que: >=
- Operador relacional Menor o Igual que: <=
- Operador relacional Mayor >
- Operador relacional Menor <

Símbolo de fin de instrucción: $

Tipos de Datos.

Data types are fundamental to any programming language. They help us define the nature of values that can be stored and manipulated in a program. In 'HandTech,' we have defined several data types that cover a wide range of needs. Below is a table listing the available data types in 'HandTech'

Dato	Tipo	Descripción	Tamaño
Degree	Número en decimales	Representa números decimales	9 bits (de 0 a 360)
int	Número entero	Representa números enteros.	9 bits (de 0 a 360)
float	Número de puntos flotantes.	Representa números con parte fraccionaria.	9 bits (de 0 a 360)
bool	Valor booleano	Representa valores	1 bit (1 para verdadero 0 para falso)
none	Valor nulo.	Representa ausencia de valor	-
empty	Retorno de valor vacío.	Representa un tipo especial que indica la ausencia de un valor útil.	-

Operadores Aritméticos.

Los operadores aritméticos, son una parte fundamental de un lenguaje de programación, ya que son considerados como herramientas fundamentales para la realizar las operaciones aritméticas. A continuación, se presenta una lista de los operadores aritméticos disponibles en “HandTech” junto con sus descripciones:

Símbolo	Tipo
+	Suma
-	Resta
*	Multiplicación
/	División
++	Incremento en 1
–	Decremento en 1

Operadores Relacionales.

Este tipo de operadores los utilizaremos para comparar valores y realizar operaciones de comparación. A continuación, se presenta una lista de los operadores aritméticos disponibles en “HandTech” junto con sus descripciones:

Símbolo	Tipo
<	Menor que
>	Mayor que
>=	Mayor igual
<=	Menor igual
<>	Diferente de
=+=	Es igual a

Operadores de Asignación.

Es un símbolo o un conjunto de símbolos que se utiliza para asignar un valor a una variable o a una ubicación de memoria. La asignación implica tomar un valor y almacenarlo en una variable o en una ubicación de memoria específica para que se pueda utilizar en operaciones posteriores. En el caso de “HandTech”, los operadores de asignación permite asignar valores a variables de manera sencilla y eficiente que son los siguientes:

Símbolo.	Tipo.
=	Igual a

Operadores Lógicos.

Its use is to perform logical operations and make decisions based on conditions. In 'HandTech,' several logical operators have been implemented to evaluate logical expressions and determine the truth or falsehood of a statement. Below is a list of the available logical operators.

Símbolo.	Tipo.
AND	Operador AND
NOT	Operador NOT

Palabras Reservadas.

Son un conjunto de cadenas que tienen una función específica dentro de un programa, la cual puede ir desde definir una función hasta indicar que estructura se está usando. Su principal característica es que no pueden ser usadas como nombres de identificadores. Aquí tenemos un listado de palabras reservadas y su propósito o descripción.

Palabra Reservada	Descripción
if	La palabra “if” sirve para ejecutar un bloque de código si se cumple una condición.
then	“Then” tiene la función en la sintaxis de estructuras condicionales después de un “if”, para indicar el bloque de código a ejecutar si la condición es verdadera.
else	Se utiliza en estructuras condicionales para definir bloques de código que se ejecutan si la condición del "if" no se cumple.
for	Es utilizada para definir bucles "for" que permiten ejecutar un bloque de código un número determinado de veces.
while	Define bucles que ejecutan un bloque de código hasta que la condición dada deje de cumplirse.
stop	Detiene la ejecución de un ciclo (“for” o “while”).
method	Define un bloque de código como una función o método.
run	Método que ejecuta su contenido como el programa principal.
return	Es una palabra reservada que nos permite devolver el valor de una función o un método.
console	Es una palabra reservada que nos servirá para imprimir mensajes en consola.
export	Instrucción para exportar funciones y/o variables de archivo a otro..
import	Instrucción que nos servirá para importar archivos con configuraciones precargadas.
mbm	Define un objeto dentro del código del programa que asemeja una parte del

	cuerpo.
arm	“arm” es una palabra reservada que hace referencia al brazo.
hand	“hand” es una palabra reservada que hace referencia a toda la mano.
rotate	Método que nos sirve para rotar la muñeca con x cantidad de grados.
fng1 (finger)	“fng1” es una palabra reservada para definir lo que corresponde al dedo índice, para poder controlarlo individualmente.
fng2 (finger)	“fng2” es una palabra reservada para definir lo que corresponde al dedo anular, para poder controlarlo individualmente.
fng3 (finger)	“fng3” es una palabra reservada para definir lo que corresponde al dedo medio, para poder controlarlo individualmente.
fng4 (finger)	“fng4” es una palabra reservada para definir lo que corresponde al dedo meñique, para poder controlarlo individualmente.
fng5 (finger)	“fng5” es una palabra reservada para definir lo que corresponde al dedo pulgar, para poder controlarlo individualmente.
is	Método que devuelve una variable de tipo booleana en base al parámetro que recibe.
mov	Mover x cantidad de grados la muñeca.
force	Fuerza aplicada a un objeto medida en libras.
sensor	“sensor” es una palabra que nos servirá para instanciar un sensor para poder usarlo.
weight	Parámetro que devuelve el peso en libras de un objeto sostenido.
block	Es un parámetro ya definido, que nos retornará una variable de tipo booleana.
up	Palabra reservada para levantar alguna parte del brazo a cierta distancia del suelo u objeto, con cierta velocidad.
down	Palabra reservada para bajar alguna parte

	del brazo a cierta velocidad.
linkage (articulación)	“linkage” es la palabra reservada para referirse a las articulaciones, sea esta de los dedos o las articulaciones,.
HowardWolowitz	Chuck Lore.(2007-2009). *The Big Bang T4E01 “Manipulación Robótica”*.
wrist	“wrist” es una palabra reservada que hace referencia a la muñeca.
piRad	“piRad” es la palabra reservada que devuelve el valor de pi radianes.
degree	“degree” It is a reserved word that indicates how many degrees the components of the prosthesis should move.
wait	“wait” It is used to establish a delay between the operations performed by the prosthesis, represented in milliseconds.

Estructura del Programa.

Es importante definir cuál es la estructura base de “HandTech”, la cual es la siguiente.

```
method run () {

}
```

Micro-Estructuras

Definición de variables:

Una variable es una herramienta que nos permite reservar un espacio en memoria para almacenar algún dato para posteriormente poder trabajar con dicho dato.

HandTech posee una sintaxis muy sencilla para la declaración de variables, la cual hace uso del tipo de dato a declarar, enseguida del nombre de la variable, continuando con el operador de asignación y el valor asignar a esta variable.

```
Deegre start = 34.5$
```

Objetos:

Los objetos son abstracciones de objetos propios de la vida real, transportando sus características.
En HandTech la principal función de los objetos es asemejar a las partes del cuerpo, pero no se limita solo a eso. Para declarar una lo hacemos mediante la palabra reservada mbm.

```
mbm hand {
    Degree posStart = 0$
    Degree posEnd = 359$
    method mov (N: Degree){

        ;bloque de código
    }
}
```

Funciones/Métodos:

Las funciones son bloques de código que realizan alguna operación, pueden o no recibir parámetros, y pueden llegar a devolver valores de salida.

En HandTech las funciones se declaran mediante el uso de la palabra reservada method, esto luce de la siguiente manera.

```
method rotate (N: Degree){
    wrist.mov(down)$
    hand(20)$
        bool repeat = hand.block$
        while(repeat){
                hand.mov(0)$
                hand.mov(N)$
                repeat = hand.block$
                if(repeat){
                        stop$
                }
        }
}
```

Comentarios:

Comments are such an important part when it comes to documenting our code, due to their help, just like their name suggests, they allow us to leave a comment that will help understand the code we are reading.

```
method run () {
    ;comentario de una sola línea
    <- Comentario de
      múltiples líneas
    ->
}
```

Estructuras de control de flujo

Las estructuras de control son bloques de código que nos ayudan a que nuestro código pueda seguir cierto orden al momento de ejecutarse.

Sentencia If

El if es una estructura de control que nos permite poner una condición para decidir si ejecutar un bloque de código o no.
Para lograr esto se hace uso de 3 palabras reservadas las cuales son if, then, else. La sintaxis es la siguiente.

```
if(condicion)then{
  ;bloque codigo
}else{
  ;bloque codigo
}
```

Sentencia For

El for en una estructura que sirve para poder repetir una acción un cierto número de veces, que será introducido según se requiera. Para poder usar esta sentencia tenemos que hacer una estructura que requiere de tres parámetros , primero declarar la variable que contiene el número de iteraciones que se vayan realizando puede ser dentro del misma estructura o fuera , después la comparación para la variable y las iteraciones totales una vez que se cumpla la comparación termina el bucle y por último los incremento de nuestra variable.

```
for (int nombreVa : NumeroDeIteraciones OperadoresRelacional Variable :  Variable
OperadorAritmetico)
{
 ;Bloque de código
}
```

Sentencia While

El while es otro ciclo que podremos encontrar en “HandTech”, este se trata de un ciclo que continúa siempre y cuando una condición se siga cumpliendo, esto se hace uso de la palabra reservada while y se implementa de la siguiente manera.

```
method captar(){
 sensor sn = true
 while(sn =+= true){
      arm.mov(45, 67)$
      wrist.rotate(30)$
     ;Recibe como parámetros X e Y
  }
}
```

Como se utiliza para el desarrollo de prótesis.

```
method run(){
  ;Aquí mandas a llamar los métodos que llegues a crear
  ;fng1 = 30$
  AgarrarSoltar()$
}

method AgarrarSoltar(){
  sensor sn = false$
  telefono tireloProfe = telefono$
  if(NOT sn) then{
    wrist.rotate(90)$ ;Cantidad de grados que rotará la muñeca
    wait(2000)$ ;Espera una cantidad de 2 segundos
    arm.mov(10)$ ;Cantidad de cm que se moverá la mano con respecto a X
    wait(2000)$
    hand.mov(tireloProfe.ancho)$ <- Cierra la mano en un valor de grados que indica el
                                    parámetro del objeto ->
    sn = true$
  } else {
    hand.mov(-tireloProfe.ancho)$ ;Abre la mano
    wait(2000)$
    arm.mov(-10)$
    wait(2000)$
    wrist.rotate(-90)$  ;Cantidad de grados que rotará la muñeca en -X
    sn = false$
  }
}

mbm telefono {
      int ancho = 15$ ;Cantidad en cm del ancho del obj
      int alto = 27$ ;Cantidad en cm del alto del obj
} ;Objeto teléfono nos ayudará a establecer los límites de dicho objeto
```

6.Herramientas

Este proyecto de analizador léxico, sintáctico y semántico para el lenguaje HandTech (.ht) está desarrollado en Python, utilizando Ply (Python Lex-Yacc) para la construcción de los analizadores léxico y sintáctico. El desarrollo se realiza en el entorno de desarrollo integrado (IDE) Visual Studio Code (VSCode), que es ideal por su soporte robusto para Python, extensiones útiles como Python, Pylint, y Jupyter, y su capacidad para facilitar el desarrollo, pruebas y depuración. El control de versiones se maneja con Git, y se recomienda utilizar GitHub para la colaboración y gestión del código. La documentación del proyecto se realiza en Markdown, y se utilizan herramientas como unittest o pytest para las pruebas unitarias y de integración, asegurando que cada componente del analizador funcione correctamente. La estructura del proyecto incluye módulos separados: lexico.py para el analizador léxico, sintactico.py para el analizador sintáctico, y semantico.py para el analizador semántico. Además, Ply se utiliza como la principal dependencia para la implementación de las reglas gramaticales y léxicas. Las pruebas se facilitan mediante scripts de ejecución y archivos de prueba que permiten validar el correcto funcionamiento del analizador con ejemplos de código en HandTech (.ht).

En el analizador léxico utilizamos las siguientes herramientas:

Ply (Python Lex-Yacc):

- Ply (Lex and Yacc para Python): Ply es una implementación de las herramientas lex y yacc en Python. En este caso, se utiliza para crear un analizador léxico, que es una parte fundamental del análisis del lenguaje.

Módulo difflib:

- Difflib: Este módulo de la biblioteca estándar de Python proporciona clases y funciones para comparar secuencias, en este caso, se utiliza para encontrar coincidencias cercanas de palabras reservadas incompletas.

Funciones y Variables Globales:

- Variables Globales (tabla_errores y tabla_simbolos): Se utilizan para almacenar errores léxicos y símbolos identificados durante el análisis del código fuente.
- Funciones Globales: Varias funciones se utilizan para manejar errores léxicos, encontrar la columna de un token, y reiniciar el analizador.

Expresiones Regulares:

- Expresiones Regulares: Se utilizan para definir los patrones de los tokens, incluyendo identificadores, números enteros, números decimales, comentarios, operadores y símbolos especiales.

Funciones de Manejo de Errores:

- Funciones de Manejo de Errores: Estas funciones se activan cuando se encuentra un token que no coincide con ninguna de las reglas definidas. Los errores se agregan a tabla_errores con información detallada sobre el tipo de error, valor, línea y columna.

Diccionarios:

- Diccionarios para Palabras Reservadas (reserved): Se utiliza para identificar y manejar palabras clave específicas del lenguaje de programación que se está analizando.
- Diccionarios para Descripciones (descriptions, symbols_descriptions): Se utilizan para proporcionar descripciones de palabras reservadas y símbolos, facilitando la interpretación y documentación del código.

Funciones del Lexer:

- Definición de Tokens (tokens): Una lista de todos los tokens que el analizador léxico debe reconocer.
- Funciones de Tokenización: Cada token tiene una función asociada que define cómo se reconoce y procesa en el texto fuente. Algunas funciones también transforman los valores de los tokens (por ejemplo, convertir un número decimal de una cadena a un valor float).
- Funciones de Ignorar: Definen patrones que el analizador debe ignorar, como espacios en blanco y comentarios
.

Manejo de Palabras Reservadas Incompletas:

- Subcadenas de Palabras Reservadas (partial_reserved): Se utiliza para identificar subcadenas de palabras reservadas y proporcionar sugerencias cuando se encuentra una palabra reservada incompleta.

Proceso de Análisis Léxico

Inicialización del Lexer:

- El lexer se inicializa utilizando la función lex.lex(), que configura el analizador léxico basado en las reglas definidas.

Reinicio del Analizador:

- La función reiniciar_analizador_lexico se utiliza para reiniciar el analizador, limpiando la tabla de errores y restableciendo los contadores de línea y posición.

Análisis del Código Fuente:

- El código fuente se proporciona al lexer a través del método lexer.input(codigo).
- Se itera sobre los tokens generados por el lexer, imprimiendo o procesando cada token según sea necesario.

En el analizador sintáctico se utilizaron las siguientes herramientas:

Ply (Python Lex-Yacc):

- Ply (Lex and Yacc para Python): Ply es una implementación de las herramientas lex y yacc en Python, que se utilizan para crear analizadores léxicos y sintácticos. En este caso, Ply se utiliza para definir las reglas gramaticales y el manejo de errores sintácticos.

Imports y Configuración Inicial:

- Importaciones: Se importan los módulos ply.yacc y ply.lex para el análisis sintáctico y léxico. Además, se importan tokens y tabla_errores del módulo lexico.
- Inicialización de tabla_errores: La tabla de errores se inicializa con los errores léxicos obtenidos a través de obtener_errores_lexicos().

Manejo de Errores Sintácticos:

- Funciones para Agregar y Obtener Errores: Se definen funciones como agregar_error_sintactico para registrar errores sintácticos y obtener_errores_sintactico para obtener la lista de errores.
- Función find_column: Esta función se utiliza para encontrar la columna de un token en la línea, lo que es útil para reportar la ubicación exacta de los errores.

Precedencia de Operadores:

- Precedence: Una tupla que define la precedencia de los operadores. Esto ayuda a resolver ambigüedades en las expresiones que involucran múltiples operadores.

Producciones Gramaticales:

- Funciones de Producción: Se definen funciones que corresponden a las reglas gramaticales del lenguaje. Cada función tiene una cadena de documentación (docstring) que define la producción gramatical en BNF (Backus-Naur Form).
- Gramática Base y Estructuras: Se define la gramática base del programa, el método principal (main), y otras estructuras como bloques de código, expresiones, instrucciones, ciclos, y condiciones.

Manejo de Tokens y Símbolos:

- Tokens Importados: Se utilizan los tokens definidos en el analizador léxico para construir las producciones gramaticales.
- Valores y Operaciones: Se manejan diferentes tipos de valores (identificadores, números, booleanos) y operaciones (aritméticas y lógicas).

Errores Sintácticos:

- Funciones de Error: Se definen funciones específicas para manejar errores en diversas estructuras como programas, métodos principales, objetos, funciones, bloques de código, instrucciones, asignaciones, llamadas, condiciones y ciclos.
- Función Genérica p_error: Una función genérica para capturar errores sintácticos no especificados.

Instanciación del Analizador Sintáctico:

- Parser de Yacc: Se instancia el analizador sintáctico utilizando yacc.yacc().
- Configuración para Evitar Advertencias: Se configuran los logs para evitar la impresión de advertencias sobre tokens no utilizados.

Proceso de Análisis Sintáctico

Inicialización del Parser:

- Se inicializa el parser utilizando yacc.yacc(), que configura el analizador sintáctico basado en las reglas definidas.

Análisis del Código Fuente:

- El código fuente se analiza mediante la función analisisSintactico(src), que utiliza el parser para procesar el código y generar un árbol sintáctico abstracto (AST).

Manejo de Errores:

- Durante el análisis, cualquier error sintáctico se registra en tabla_errores con información detallada sobre el tipo de error, valor, línea y columna.

Herramientas utilizadas para el analizador semántico:

Ply (Python Lex-Yacc):

- Ply (Lex and Yacc para Python): Ply se utiliza para construir el analizador léxico y sintáctico que luego se utiliza en el análisis semántico.

Manejo de Errores:

- Funciones para Agregar y Obtener Errores: agregar_error_semantico agrega errores semánticos a una lista, y obtener_errores_semanticos devuelve la lista de errores.
- Función find_column: Calcula la columna de un token en la línea para proporcionar detalles precisos sobre la ubicación de los errores.
- Función reiniciar_analizador_semantico: Reinicia el analizador semántico para un nuevo análisis.

Tabla de Símbolos:

- Tabla de Símbolos (tablaSimbolos): Un diccionario que almacena identificadores y su tipo de dato asociado.
- Funciones de Control de Simbolos: La tabla de símbolos se actualiza durante el análisis para asegurar que los identificadores se declaren y usen correctamente.

Árbol de Sintaxis Abstracta (ASA):

- Analizador Sintáctico: Se utiliza el resultado del analizador sintáctico (asa) para realizar el análisis semántico.
- Recorrido del ASA: La función analisis recorre el ASA y realiza verificaciones semánticas en cada nodo.

Funciones de Análisis Semántico:

- Funciones de Análisis: Cada nodo del ASA tiene una función correspondiente en el analizador semántico que realiza verificaciones y actualizaciones necesarias.
- Fases de Análisis: Se maneja una variable fase para controlar el recorrido del ASA en múltiples fases, si es necesario.

Proceso de Análisis Semántico

Inicialización y Limpieza:

- Función destructor: Limpia la tabla de símbolos y la lista de errores, y reinicia las variables de control de lectura de ASA.
- Función reiniciar_analizador_semantico: Reinicia los contadores de línea y posición del lexer.

Análisis del ASA:

- Función analisis: Recibe el ASA generado por el parser y recorre cada nodo, realizando las verificaciones semánticas correspondientes.

Verificaciones:

- Programas y Objetos: Se verifica que los métodos y objetos estén correctamente definidos.
- Funciones: Se asegura que las funciones no se redefinan y que los identificadores no se usen incorrectamente.
- Bloques e Instrucciones: Cada instrucción dentro de un bloque se analiza para verificar el uso correcto de variables y métodos.
- Asignaciones: Se comprueba que las variables se declaren antes de su uso y que los tipos de datos sean consistentes.
- Inicializaciones: Se asegura que los identificadores no se redefinan.
- Llamadas a Métodos: Se verifica que los métodos llamados estén previamente definidos.

Reporte de Errores:

- Función agregar_error_semantico: Agrega errores semánticos con detalles sobre el tipo, descripción, valor, línea y columna.
- Función obtener_errores_semanticos: Devuelve la lista de errores encontrados durante el análisis.

Función Principal para el Análisis:

- Función analizar: Realiza todo el proceso de análisis léxico, sintáctico y semántico, y devuelve un reporte de errores si los hay, o un mensaje indicando que no se encontraron errores.

7.Manual de usuario .

Declaraciones Básicas

En HandTech, puedes declarar variables de diferentes tipos: int, float, bool, sensor, degree.

```
int a = 5$
float b = 3.14$
bool c = true$
```

Estructura del Programa
El método principal del programa se define con method run():

```
method run(){
    ; Aquí van las instrucciones principales
}
```

Operaciones Aritméticas
Puedes realizar operaciones aritméticas básicas utilizando los operadores +, -, *, /.

```
int x = 10$
int y = 20$
int z = x + y$
```

Condicionales
HandTech soporta estructuras condicionales utilizando if, then, else.

```
if (x > y) then {
    ; Instrucciones si la condición es verdadera
} else {
    ; Instrucciones si la condición es falsa
}
```

Ciclos
El lenguaje soporta bucles for y while.

```
for (int i = 0 : i < 10 : i++) {
    ; Instrucciones del bucle for
}

while (x < 100) {
    ; Instrucciones del bucle while
}
```

Definición y Uso de Funciones
Puedes definir funciones adicionales utilizando la palabra clave method.

```
method myFunction() {
    ; Instrucciones de la función
}
```

Manipulación de la Prótesis
HandTech incluye comandos específicos para manipular la prótesis robótica, como mover partes específicas de la mano, aplicar fuerza y rotar.

```
hand.open()$
hand.close()$
wrist.rotate(45)$
fng1.mov(30)$
fng2.force(5.5)$
```

Declaraciones de Variables

Declaración de Enteros (int):
int a = 5$
Declara una variable entera a y la inicializa con el valor 5.

Declaración de Flotantes (float):
float b = 3.14$
Declara una variable flotante b y la inicializa con el valor 3.14.

Declaración de Booleanos (bool):
bool c = true$
Declara una variable booleana c y la inicializa con true.

Declaración de Sensores (sensor):
sensor s1 = false$
Declara un sensor s1 y lo inicializa con false.

Declaración de Grados (degree):
degree d = 45$
Declara una variable de tipo degree y la inicializa con 45 grados.

Estructuras de Control

Condicionales (if, then, else):

```
if (x > y) then {
    ; Instrucciones si la condición es verdadera
} else {
    ; Instrucciones si la condición es falsa
}
```

Ejecuta un bloque de código si la condición es verdadera y otro si es falsa.

Bucles (for, while):

```
for (int i = 0 : i < 10 : i++) {
    ; Instrucciones del bucle for
}

while (x < 100) {
    ; Instrucciones del bucle while
}
```

Ejecuta un bloque de código repetidamente según la condición especificada.

Funciones y Métodos

Definición de Funciones:

```
method myFunction() {
    ; Instrucciones de la función
}
```

Define una función llamada myFunction.

Llamada a Funciones:
myFunction()$
Llama a la función myFunction.

Manipulación de la Prótesis

Abrir y Cerrar la Mano:
hand.open()$
hand.close()$
Abre y cierra la mano de la prótesis.

Rotación de la Muñeca:
wrist.rotate(45)$
Rota la muñeca 45 grados.

Movimiento de Dedos:
fng1.mov(30)$
fng2.force(5.5)$
Mueve el dedo índice 30 grados y aplica una fuerza de 5.5 unidades al dedo anular.

Ejemplo Completo de un Programa en HandTech

```
method run() {
   int t = 0$
   int b = 5$
   t = b$
   int f = 0.0$
   bool flag = false$
   if (t > b) then {
      hand.close()$
   } else {
      hand.open()$
   }
   for (int i = 0 : i < 10 : i++) {
      wrist.rotate(10)$
   }
   while (flag == false) {
      flag = true$
   }
}

mbm objeto1 {
   ; int a = 0$
}

method checkSensor() {
   sensor s1 = true$
   if (s1) then {
```

```
      ; Instrucciones si el sensor está activo
   }
}
```

Interfaz del Compilador

Secciones de la Interfaz

1. *Menú Superior:*

   File: Permite abrir y guardar archivos.
   Analizar: Opción para iniciar el análisis del código fuente.
   Tablas: Acceso a tablas de símbolos y errores.
   Tamaño de la Letra: Permite ajustar el tamaño del texto en el editor.
   Compilar: Botón para compilar el código.

2. *Editor de Código:*

   Un área principal donde se escribe y edita el código fuente. Soporta resaltar la sintaxis del lenguaje HandTech.
   Numeración de Líneas: Muestra el número de cada línea para facilitar la referencia.

3. *Consola de Salida:*

   Área inferior donde se muestran los mensajes de salida del compilador, incluyendo errores léxicos, sintácticos y semánticos, así como el resultado de la compilación.

Funcionalidades Clave

Resaltado de Sintaxis: El editor resalta diferentes componentes del lenguaje, como palabras clave, identificadores y operadores, para mejorar la legibilidad del código.
Reporte de Errores: Los errores encontrados durante el análisis se muestran en la consola de salida con detalles sobre el tipo de error, la ubicación (línea y columna), y una descripción del problema.
Compilación Directa: El botón de compilar permite ejecutar el compilador y ver los resultados directamente en la consola de salida.

Ejemplo de Uso

Escribir Código: Escribe el código en el área de edición. El ejemplo mostrado en la imagen es un programa que define el método principal run y una función adicional AgarrarSoltar.

Guardar el Archivo: Guarda el archivo con extensión .ht para asegurarte de que el compilador lo reconozca.

Compilar el Código: Haz clic en "Compilar" para iniciar el proceso de compilación. Los errores y mensajes se mostrarán en la consola de salida.

Corrección de Errores: Si hay errores, corrige el código según los mensajes mostrados en la consola y vuelve a compilar.

8. Cronograma de actividades.

Cronograma de actividades.																								
Mes		Enero				febrero				Marzo				Abril				Mayo				Junio			
Semas del mes		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Plantemiento inicial de nuesto proyecto, invetigaciones basicas: leguajes, cadenas, etc.																								
2	Inicio de la creación de nuestro lenguaje, asi como el plantamineto del proyecto.																								
3	Creación de nuestra propuesta final del proyecto, asi como la selección de donde se desarrollara.																								
4	Inicio del desarrollo del analizador lexico ,semantico y sintáctico.																								
5	Implementación del analizador semántico.																								
6	Implementación del analizador sintáctico.																								
7	Implementar el codigo intermedio																								
9	Optimización																								
8	Reportes de avances del proyecto , asi como su documentación.																								

9. Resultados esperados.

After the project is completed, HandTech can be used for the programming of hand prosthetics, which individuals can use in their daily lives. Simultaneously, it can also be employed to program these prosthetics for industrial purposes.

Dentro de lo que sería el compilador del proyecto, que es donde se llevará a cabo la programación para las prótesis, este mismo deberá ser capaz de generar programas que optimicen el consumo de recursos de la prótesis, lo cual implica el desarrollo de algoritmos de optimizacion y gestion de recursos.

Del mismo modo, el compilador permitirá la fácil adaptación del código a diferentes componentes con los cuales puede estar creada la prótesis, esto implica la inclusión de características de modularidad y de configurabilidad que permitan la personalización de los diferentes componentes.

Por último, este mismo deberá ser integrable con las tecnologías de desarrollo ya existentes de desarrollo y simulación en el ámbito de la robótica y de la ingeniería de software.

10. Conclusiones

Para completar este proyecto hemos llevado a cabo el análisis e investigación en la profundidad suficiente para entender las necesidades, que conlleva el desarrollo del lenguaje de programación propuesto, así como la manera en que este cubrirá su propósito con el se ideó en un principio.

Como parte integral del desarrollo de un nuevo lenguaje, este proyecto también conlleva una fase de desarrollo para el programa que será el compilador del lenguaje a desarrollar, siendo de gran apoyo tanto este documento como los objetivos fijados para una finalización correcta.

Agregando como último punto la importancia de la investigación llevada a cabo en la materia de prótesis para el reemplazo de extremidades perdidas, y los conocimientos previos en diversos lenguajes de programación, como lo son: Java, JavaScript, C#, C, Assembler, etc. Reconociendo su influencia en el diseño de este proyecto.

11. Collage.


Good morning
Ciao!
Hello
JS
JavaScript
PROGRAMMING LANGUAGE
</></?></{;}

12. Glosario.

Prótesis: Sustituto artificial de una parte del cuerpo faltante (tanto en singular como en plural; se llama prótesis).

IA: Inteligencia Artificial (IA) es la combinación de algoritmos planteados con el propósito de crear máquinas que presenten las mismas capacidades que el ser humano.

Extremidad: Pies y manos del ser humano.

Bioeléctrica: Todos los impulsos que genera un cuerpo biológico se denominan impulsos bioeléctricos.

Robótica: Ciencia que aglutina varias disciplinas o ramas de la tecnología con el objetivo de diseñar máquinas programadas para realizar tareas de forma automática o para simular el comportamiento humano o animal.

Prototipo: Modelo más representativo de algo.

Movilidad: Cualidad de movible.

Desarrollar: Realizar o llevar a cabo algo.

Sintaxis: Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación.

Detección: Descubrir la existencia de algo que no era patente.

Alfabeto: Sistema de escritura formado por signos

Operadores: Símbolo matemático que indica que debe ser llevada a cabo una operación especificada

Relacionales: Conexión, correspondencia de algo con otra cosa.

Símbolo: Elemento u objeto material que, por convención o asociación, se considera representativo de una entidad, de una idea, de una cierta condición, etcétera.

Decimal: Aquel que se compone de unidades enteras y de una fracción

Degree: A unit of measurement of angles, one three-hundred-and-sixtieth of the circumference of a circle.

Booleano: Aquel que puede representar valores de lógica binaria, esto es 2 valores, que normalmente representan falso o verdadero.

Palabra Reservada: Es una palabra que tiene un significado gramatical especial para ese lenguaje y no puede ser utilizada como un identificador

Cadena: Secuencia ordenada de elementos que pertenecen a un cierto lenguaje formal o alfabeto análogas a una fórmula o a una oración.

Fuerza: Magnitud vectorial que mide la intensidad del intercambio de momento lineal entre dos cuerpos.

Microestructura: Es una estructura que se integra a otra de mayor tamaño o magnitud.

Variable: Formada por un espacio en el sistema de almacenaje y un nombre simbólico que está asociado a dicho espacio.

Memoria: Dispositivo que retiene, memoriza o almacena datos informáticos durante algún periodo de tiempo.

Objeto: Ente orientado a objetos que consta de un estado y de un comportamiento

Función: Bloque de código que realiza alguna operación.

Operación: Ejecución de algo.

Documentar: Añadir suficiente información como para explicar lo que hace, punto por punto, de forma que no sólo los ordenadores sepan qué hacer, sino que además los humanos entiendan qué están haciendo y por qué.

Estructura: Modo de representar información en una computadora, aunque además, cuentan con un comportamiento interno.

Código: Conjunto de instrucciones que un desarrollador ordena ejecutar a un computador.

Condición: Instrucción de que algo debe ser cierto para que algo pueda pasar.

Resultado: Se refiere a salidas de programas, valores de expresiones, estados de autómatas, códigos generados en compilación, entre otras interpretaciones.

Compilador: Es un programa informático que traduce un código fuente escrito en un lenguaje de programación de alto nivel a un código ejecutable de bajo nivel o a otro lenguaje de programación.

Integrable: Se refiere a la capacidad de combinar o conectar diferentes componentes o módulos de manera efectiva para construir un sistema más grande y complejo.

Modularidad: Se refiere a la capacidad de descomponer un sistema en módulos o componentes independientes que realizan funciones específicas.