

**Diseño de Compiladores**



Ing. Elda Guadalupe Quiroga

Dr. Héctor Gibrán Ceballos

     \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_                            \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

        Martha Lisett Benavides Martínez                                   José González Ayerdi

                       A01280115                                                                A01036121

Mayo 3, 2017

Contenido

[1. Descripción y Documentación del Proyecto 3](#_Toc481282341)

[1.1. Descripción del Proyecto 3](#_Toc481282342)

[1.1.1. Visión, Objetivo y Alcance del Proyecto 3](#_Toc481282343)

[1.1.2. Análisis de Requerimientos y Casos de Uso Generales 3](#_Toc481282344)

[1.1.3. Descripción de los Principales Test Cases 5](#_Toc481282345)

[1.1.4. Descripción del Proceso General Seguido para el Desarrollo del Proyecto 5](#_Toc481282346)

[1.2. Descripción del Lenguaje 12](#_Toc481282347)

[1.2.1. Nombre del Lenguaje 12](#_Toc481282348)

[1.2.2. Descripción Genérica de las Principales Características del Lenguaje 12](#_Toc481282349)

[1.2.3. Descripción de los Errores que Pueden Ocurrir (Compilación y Ejecución) 15](#_Toc481282350)

[1.3. Descripción del Compilador 15](#_Toc481282351)

[1.3.1. Equipo de Cómputo, Lenguaje y Utilerías Usadas en el Desarrollo del Proyecto 15](#_Toc481282352)

[1.3.2. Descripción del Análisis de Léxico 15](#_Toc481282353)

[1.3.3. Descripción del Análisis de Sintaxis 17](#_Toc481282354)

[1.3.4. Descripción de Generación de Código Intermedio y Análisis Semántico 17](#_Toc481282355)

[1.4. Descripción de la Máquina Virtual 21](#_Toc481282356)

[1.4.1. Equipo de Cómputo, Lenguaje y Utilerías Usadas 21](#_Toc481282357)

[1.4.2. Proceso de Administración de Memoria en Ejecución (Arquitectura) 21](#_Toc481282358)

[1.5. Pruebas del Funcionamiento del Lenguaje 21](#_Toc481282359)

[1.5.1. Pruebas que Comprueban el Funcionamiento del Proyecto 21](#_Toc481282360)

[1.6. Listados Documentados del Proyecto 36](#_Toc481282361)

[2. Manual de Usuario 36](#_Toc481282362)

# Descripción y Documentación del Proyecto

## Descripción del Proyecto

### Visión, Objetivo y Alcance del Proyecto

Hoy en día el uso de las Tecnologías de Información se encuentra en auge en distintas áreas de la vida diaria, tal como lo son los casos de grandes empresas de distintos giros, escuelas, aplicaciones web y móviles, así como videojuegos en 2D y 3D. Dado lo anterior, el propósito del presente proyecto es el de fomentar en jóvenes y adultos el aprendizaje de conceptos básicos de programación de manera sencilla y didáctica, en donde puedan tener la oportunidad de ver la perspectiva de desarrollo de los sistemas/programas que utilizan diariamente, de modo que obtengan una herramienta que les sirva como base para desarrollar sus propias ideas, que incluso, quizá en un futuro, este aprendizaje sea suficiente para buscar estudiar una carrera universitaria o técnica relacionada con las Tecnologías de Información, en donde la creatividad de cada persona es el límite.

El principal objetivo de APODORAX es el producir, como ya se mencionó con anterioridad, un lenguaje sencillo de utilizar, en donde las personas puedan aprender de forma simplificada y didáctica la lógica de la programación que se encuentra detrás de todas las aplicaciones que usan, dando paso a que con un poco de esfuerzo ellos mismos puedan generar instrucciones que les permitan observar de manera gráfica el resultado de las mismas (output gráfico).

De igual manera, dentro del alcance del proyecto se encuentra que el lenguaje tendrá como característica principal que estará basado en español, de modo que sea más sencillo para las personas que no tienen conocimiento del idioma inglés el declarar variables, ciclos, condiciones y funciones, ya que como es conocido, la mayoría de los lenguajes de programación usados a nivel mundial se encuentran escritos en inglés, logrando intimidar y limitar a aquellas personas con un bajo entendimiento del idioma.

### Análisis de Requerimientos y Casos de Uso Generales

Debido al hecho de que se mencionó que APODORAX estará desarrollado en español, la estructura del mismo, así como sus palabras clave, será representada en el mismo idioma.

#### Requerimientos Funcionales

Entre las principales funcionalidades que debe tener el compilador son las siguientes:

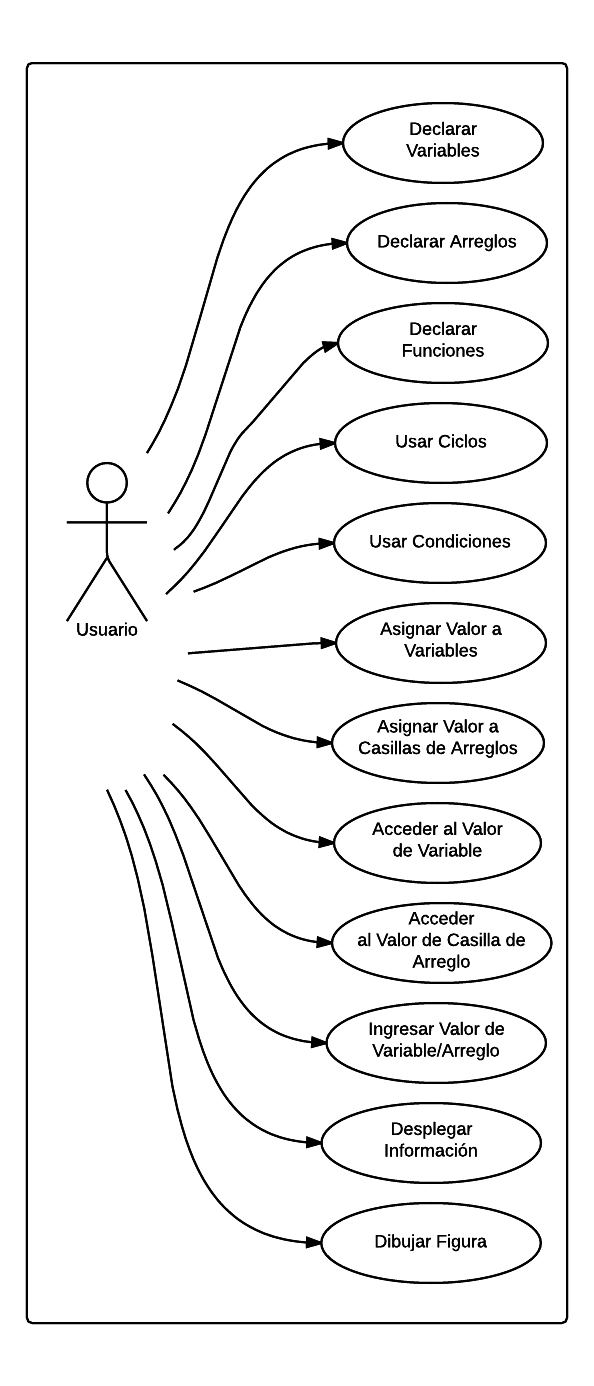
* RF.01. El sistema aceptará la declaración de variables globales.
* RF.02. El sistema aceptará la declaración de variables locales.
* RF.03. El sistema aceptará la declaración de arreglos globales.
* RF.04. El sistema aceptará la declaración de arreglos locales.
* RF.05. El sistema aceptará la declaración de funciones.
* RF.06. El sistema aceptará la declaración/uso de ciclos.
* RF.07. El sistema aceptará la declaración/uso de condiciones.
* RF.08. El sistema aceptará el uso de diversos tipos de datos (entero, flotante, cadena, carácter, bool).
* RF.09. El sistema permitirá asignarles valor a las variables.
* RF.10. El sistema permitirá asignarle valor a cada casilla de los arreglos.
* RF.11. El sistema permitirá tener acceso al valor de cada variable.
* RF.11. El sistema soportará recursividad.
* RF.12. El sistema soportará el acceso al valor de cada casilla de los arreglos.
* RF.13. El sistema mostrará en pantalla los datos solicitados por el usuario cuando él lo desee a través de la instrucción “desplegar”.
* RF.14. El sistema permitirá que el usuario ingrese en consola los datos que desee a través de la instrucción “entrada”.
* RF.15. El sistema permitirá mostrar de manera gráfica las figuras especificadas por el usuario.
* RF.16. El sistema deberá mostrar los errores que encuentre en compilación para notificarle al usuario.

#### Requerimientos No Funcionales

Los Requerimientos No Funcionales esperados del compilador son los siguientes.

* RNF.01. El sistema deberá compilar sin ningún error.
* RNF.02. La extensión oficial del compilador será .apdx (APODORAX).
* RNF.03. El sistema deberá funcionar en sistemas operativos Windows, Mac OS y Ubuntu.

#### Casos de Uso

Los casos de uso que se esperan por parte del usuario están ilustrados a continuación:

### Descripción de los Principales Test Cases

Los principales casos de prueba realizados fueron aquellos que permitían comprobar el funcionamiento correcto de expresiones, ciclos, funciones, recursividad y arreglos.

*Arreglos*

|  |  |  |
| --- | --- | --- |
| Find de Posición | Buscar la posición de un valor dentro de un arreglo. | Prueba Exitosa |
| Find de Valor | Buscar el valor de un elemento dada una posición. | Prueba Exitosa |
| BubbleSort | Ordenar un arreglo acomodando sus valores de menor a mayor. | Prueba Exitosa |
| Valor en Casilla | Hacer que el valor de una casilla sea el valor de otra casilla. Ejemplo: x[y[5]] | Prueba Exitosa |
| Producto Punto | Hacer el producto punto entre dos vectores. | Prueba Exitosa |

*Ciclos*

|  |  |  |
| --- | --- | --- |
| Factorial Iterativo | Obtener el factorial de un número haciendo uso de un ciclo while (mientras). | Prueba Exitosa |
| Fibonacci Iterativo | Obtener el resultado final de la serie Fibonacci dado un número haciendo uso de un ciclo while (mientras). | Prueba Exitosa |

*Funciones*

|  |  |  |
| --- | --- | --- |
| Factorial Recursivo | Obtener el factorial de un número haciendo uso de recursividad | Prueba Exitosa |
| Fibonacci Recursivo | Obtener el resultado final de la serie Fibonacci dado un número haciendo uso de recursividad. | Prueba Exitosa |

### Descripción del Proceso General Seguido para el Desarrollo del Proyecto

De manera general, los integrantes del equipo estuvieron trabajando de manera conjunta para desarrollar todo el proyecto, por lo que se hizo uso de herramientas como Google Drive, Google Hangouts, Facebook y GitHub (<https://github.com/MarthaLisett/APODORAX>) para estar en constante contacto y tener un control de versiones.

#### Bitácoras de Avance

Para desarrollar el proyecto, el equipo de APODORAX se estuvo reuniendo semanalmente, incluyendo Semana Santa, para trabajar en los avances, por lo que en cada bitácora que se muestra a continuación se encuentra el resultado de los avances que se entregaron, incluyendo el progreso y las dificultades obtenidas.

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 6 Marzo 2017 | |
| **Avance** | 1. Léxico y Sintaxis | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Léxico | Se incluyeron los tokens y palabras reservadas. | Funciona al 100% |
| Sintaxis | Se hizo la gramática correspondiente ya haciendo los cambios marcados a realizar en la propuesta. | Funciona al 100% |
| Pruebas | Se hicieron pruebas con todas las gramáticas, incluyendo declaraciones, asignaciones, ciclos, condiciones, comparaciones y uso de distintos tipos de datos. | Se hicieron pruebas tratando de afectar a la gramática, dando paso a que efectivamente lo detectara como error. |
| Ambigüedades | Solamente muestra 1 Warning de conflicto Shift-Reduce | No parece afectar al lenguaje. |
| **Cambios** | Se hicieron cambios a distintos diagramas de la gramática ya que habían elementos repetitivos que ocasionaban muchas ambigüedades (conflictos Reduce Reduce y Shift Reduce). | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 10 Marzo 2017 | |
| **Avance** | 1. Directorio de Funciones y Tablas de Variables | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Directorio de Funciones | Se creó un diccionario de funciones en donde se puedan agregar nuevas funciones si no existen previamente y se marque un error si existen funciones repetidas. Cada diccionario de funciones tiene su propio diccionario de variables locales. | Funciona al 100% |
| Tabla de Variables | Se creó un diccionario de variables con distinto scope, ya sea local y global, de modo que se asegure un acceso correcto a las mismas. | Funciona al 100% |
| Pruebas | Se hicieron pruebas de todos los distintos casos posibles: variables que no existen, funciones que no existen, variables globales/locales repetidas, funciones repetidas, acceso incorrecto a variables locales/globales. | Se pasaron todas las pruebas. |
| Ambigüedades | Ya no se muestra el Warning del conflicto Shift-Reduce, pero ahora sale 1 conflicto Reduce Reduce. | No parece afectar al lenguaje, ya que las pruebas funcionan al 100%. |
| **Cambios** | No se hicieron cambios a las gramáticas en general, sólo se agregaron nuevas reglas entre las gramáticas (intermedias) para poner hacer el directorio de funciones y la tabla de variables. | |
| **Dificultades** | Tuvimos dificultades para mostrar los errores, pudimos desplegarlos usando el “raise keyError”, por lo que con esto si despliega los errores pertinentes, pero además muestra un mensaje de error propio del raise keyError (no afecta al programa).  Al principio tuvimos el problema de que no registraba el scope global, pero se arregló después al insertar las reglas intermedias. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 19 Marzo 2017 | |
| **Avance** | 3. Semántica Básica de Expresiones: Tabla de Consideraciones Semánticas (Cubo Semántico), Generación de Código de Expresiones Aritméticas y estatutos secuenciales: Asignación, Lectura. | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Cubo Semántico | Se hizo una tupla de tuplas con operaciones lógicas/ aritméticas y los diferentes tipos de datos (enteros, flotantes, cadenas, caracteres, booleanos) para saber cuáles eran válidos y cuáles deberían mostrar error. | Se terminó el cubo semántico al 100%. |
| Código Asignación | Se codificaron las instrucciones para generar los cuádruplos de los estatutos. | Faltaron hacer las pruebas de su funcionamiento. Se declaró el orden de precedencias. |
| Pruebas | Se hicieron pruebas de operaciones entre diferentes tipos de operandos, los cuales mostraron resultados correctos. | Por el momento solamente se probó el cubo semántico con constantes, las cuales pasaron las pruebas de aceptación. |
| **Pendientes** | Código de Escritura y Lectura.  Aún no se manejan Direcciones Virtuales. | |
| **Adicionales** | Se agregaron las clases stack y queue para usarlas en el proyecto. | |
| **Dificultades** | Se tuvo la dificultad en saber en qué momento agregar un id a la pila de operandos y el asignar valores a los elementos del cubo. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 24 Marzo 2017 | |
| **Avance** | 4. Generación de Código de Estatutos Condicionales: Decisiones/Ciclos | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Cuádruplos Decisiones | Se recreó el código visto en clase para generar los cuádruplos de decisiones, por lo que ya se generan los mismos para if (si) y los else (sino). | Funciona al 100%, se hace uso de los GotoF y Goto. |
| Cuádruplos Ciclos | Se recreó el código visto en clase para generar los cuádruplos de ciclos, por lo que ya se generan los cuádruplos para los while (mientras). | Funciona al 100%, se hace uso de los GotoF y Goto. |
| Cuádruplos Entrada y Salida | Se aplicó el código de generación de cuádruplos para obtener los input (ingreso) y output (escritura). | Funciona al 100%, genera los cuádruplos de manera correcta. |
| Pruebas | Se hicieron pruebas para saber que se estuvieran generando los resultados correctos de expresiones (incluyendo paréntesis), que se hiciera una validación semántica correcta entre tipos, que el scope de variables fuera el correcto, que se estuvieran generando de manera correcta los cuádruplos. | Se pasaron todas las pruebas. Se pueden generar de manera correcta todos los cuádruplos descritos con anterioridad. |
| Ambigüedades | Sigue apareciendo el conflicto Reduce Reduce. | No parece afectar al lenguaje, ya que las pruebas funcionan al 100%. |
| **Cambios** | Se añadió una regla de Lógica (&&, ||) en la gramática (se removieron de Comparación para moverlos a otro nivel) y se decidió que para hacer los input y output solamente sería de un valor a la vez (no múltiples separados por comas). | |
| **Dificultades** | Se tomó mucho tiempo en desarrollar las pruebas para saber que se estuvieran generando de manera correcta las operaciones, validaciones de semántica y cuádruplos. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 2 Abril 2017 | |
| **Avance** | 5. Generación de Código de Funciones | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Cuádruplos Funciones | Se recreó el código visto en clase para generar los cuádruplos de funciones (void). | Funciona al 100% el ejercicio que está en la hoja que se nos dio en clase, las cuales son funciones void, sin embargo, aún no se implementan las funciones con retorno. |
| Pruebas | Se hicieron pruebas para saber que se estuvieran generando los cuádruplos correctos de funciones void. | Se pasaron todas las pruebas. Se pueden generar de manera correcta todos los cuádruplos descritos con anterioridad (sólo funciones void). Aún falta implementar las funciones con retorno. |
| Ambigüedades | Sigue apareciendo el conflicto Reduce Reduce. | No parece afectar al lenguaje, ya que las pruebas funcionan al 100%. |
| **Cambios** | No se hicieron cambios. | |
| **Dificultades** | Se estaban haciendo mal los cuádruplos de las funciones, pero se logró detectar que el problema estaba en una constante y se hizo la modificación. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 10 Abril 2017 | |
| **Avance** | 6. Mapa de Memoria de Ejecución para la Máquina Virtual. Máquina Virtual: Ejecución de Expresiones Aritméticas y Estatutos Secuenciales. | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Cuádruplos Funciones | Se recreó el código visto en clase para generar los cuádruplos de funciones tanto void, así como funciones con retorno. También se hicieron los cuádruplos de las funciones predefinidas. | Funciona al 100% el ejercicio que está en la hoja que se nos dio en clase. |
| Memoria | Se hicieron las clases memoria, temporal, local, global y constante, las cuales tienen direcciones para enteros, flotantes, booleanos, cadenas y caracteres. | Existen algunos problemas con la asignación correcta de los valores temporales, al parecer se están sobre-escribiendo. Se trabajará para resolverlo. |
| Pruebas | Se hicieron pruebas para saber que se estuvieran generando los cuádruplos correctos de funciones.  También se hicieron pruebas con la librería. | Se pasaron todas las pruebas. Se pueden generar de manera correcta todos los cuádruplos descritos con anterioridad.  Se generó un algoritmo para crear curvas usando puntos. |
| Ambigüedades | Se removió el conflicto Reduce Reduce. | Se hicieron pequeñas modificaciones en la gramática y se arregló el conflicto Reduce Reduce que provocaba tener la instrucción id[] o id(). |
| **Cambios** | Se hicieron cambios en la gramática para remover el conflicto Reduce Reduce, además a las funciones predefinidas se les cambió el “Args” por “llamadapar”, de forma que aceptara los argumentos específicos que deben recibir cada una de ellas.  Se agregaron los colores rosa y naranja. | |
| **Dificultades** | Había un error en el conteo de los cuádruplos, por lo que no encontrábamos el error, finalmente se resolvió notando que faltaba incrementar en 1 el contador en las condiciones. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 23 Abril 2017 | |
| **Avance** | 7. Generación de Código de Arreglos/Tipos Estructurados. Máquina Virtual: Ejecución de Estatutos Condicionales | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Cuádruplos Arreglos | Se recreó el código visto en clase para generar los cuádruplos de arreglos, por lo que ahora se pueden declarar y acceder a ellos. | Se probó con un BubbleSort y si funcionó. |
| Memoria | Se hicieron las clases memoria, temporal, local, global y constante, las cuales tienen direcciones para enteros, flotantes, booleanos, cadenas y caracteres. | Todas las variables se están asignando en sus respectivas secciones de memoria de manera correcta. |
| Máquina Virtual | Ya se tiene la máquina virtual para expresiones aritméticas, estatutos secuenciales, condiciones, ciclos, funciones, y funciones predefinidas (librería de output). | Recursividad ya funciona, se probó con un Fibonnaci.  También ya se puede dibujar el output gráfico. |
| Interfaz | Se creó una interfaz en donde se pueda escribir código, modificarlo y compilarlo. Además, tiene una sección en donde la persona puede ejecutar funciones de la librería (dibujar figuras) de modo que pueda observar cómo se dibujan las figuras haciendo cambios en colores, tamaños, etc. | La interfaz funciona de manera correcta, también hay una sección de debug en donde después de compilar se podrá ver si hay errores en el programa. |
| **Cambios** | A las funciones predefinidas (librería) se les hizo el cambio de que ahora en lugar de sólo mostrar que pediría Args, ahora se establecen el orden y el tipo de dato que tendrán los parámetros para llamarlas. | |
| **Dificultades** | Se tuvo mucha dificultad en poder generar los arreglos y llamadas recursivas, pero después de mucho tiempo y correcciones ya funcionaron. | |

|  |  |  |
| --- | --- | --- |
| **Participantes** | Martha Benavides | José González |
| **Fecha** | 23 Abril 2017 | |
| **Avance** | 8. Primera Versión de la Documentación. Generación de Código y Máquina Virtual para una Parte de la Aplicación en Particular. | |
|  | | |
| ***Componente*** | ***Descripción*** | ***Estatus*** |
| Documentación | Se comenzó a desarrollar la documentación del proyecto. | Se tienen hechas las secciones: a (completa), b (incompleta), c (incompleta), d (incompleta). |
| Máquina Virtual | La máquina virtual ya está completa. | Solamente falta la implementación del not. |
| Output Gráfico | Ya se tiene el funcionamiento completo para dibujar todas las figuras establecidas en la propuesta. | Funciona al 90%, se cierra la ventana cuando hay recursividad. |
| **Cambios** | Ahora los arreglos pueden ser anidados y se pueden asignar valores a los subíndices de arreglos a través del usuario (entrada).  Los colores ahora son puestos en la máquina virtual, ya no se hace uso de los implementados en la gramática. | |
| **Dificultades** | Se tuvo la dificultad en establecer la relación entre un return y un endproc.  También se tuvo la dificultar para usar arreglos de distintos tipos. | |

#### Reflexiones

##### Martha Benavides

El desarrollar este proyecto me dio la oportunidad de aplicar todo el conocimiento que he aprendido a través de toda la carrera, incluyendo estructura de datos, algoritmos, objetos, ingeniería de software (documentación), entre muchos otros, brindándome la posibilidad de desarrollar un lenguaje/compilador propio, siendo un sueño que no pensé que podría cumplir al ingresar a la carrera.

Fue muy interesante el poder generar un compilador desde cero, en donde se tenía que crear desde el léxico y la gramática, hasta generar los cuádruplos y la máquina virtual para que funcionara, teniendo que pasar, junto con mi compañero de equipo, por dificultades que nos costaron desveladas, estancias tardías en la escuela y muchas horas de trabajo, pero que finalmente nos deja una gran satisfacción el tener un producto 100% funcional que puede ser usado por jóvenes y adultos para iniciar su camino en la programación, y que mejor que hacerlo mediante el output gráfico. Me quedo con una grata experiencia con la realización de este proyecto, ya que incluso pudimos volvernos mejores testers.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Martha Lisett Benavides Martínez

##### José González

Sin dudarlo, este ha sido uno de los proyectos más retadores que he tenido que hacer en mi carrera universitaria, ya que además de ser sumamente demandante, implicaba un amplio conocimiento de materias de otros semestres, sobre todo de Estructuras de Datos y Programación Orientada a Objetos.

Aunque en un principio este proyecto comenzó siendo parte importante de la ponderación necesaria para acreditar el curso, terminó volviéndose el proyecto más valioso del semestre en cuanto al valor aportado por el mismo, ya que requirió mucho compromiso por ambos integrantes del equipo para poderlo sacar adelante, además de que se volvió muy significativo por el hecho de que puede ser de utilidad para aquellas personas principiantes en el área de programación que quieran aprender conceptos básicos de generación de código, y más aún, que incluso después ya de adquirir un conocimiento más profundo, puedan tener la oportunidad de crear su propio compilador al igual que nosotros.

En general, aprendí mucho con la realización de este proyecto, desde trabajar mejor en equipo, hasta utilizar nuevas funcionalidades de Python.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

José González Ayerdi

## Descripción del Lenguaje

### Nombre del Lenguaje

El nombre asignado para el compilador fue APODORAX, ya que Apodora es una especie de serpiente relacionada con el pitón (Python), por lo que debido a que usamos Python para desarrollar nuestro lenguaje, el equipo decidió darle un nombre similar.

### Descripción Genérica de las Principales Características del Lenguaje

APODORAX le da la oportunidad al usuario de declarar variables, arreglos y funciones; hacer uso de ciclos y condiciones; dibujar figuras con la ayuda de una librería gráfica (output gráfico), por lo que se le da la oportunidad de usar un lenguaje que tenga las herramientas básicas y esenciales para entender los conocimientos importantes sobre programación.

#### Elementos del Lenguaje

* *Tipos de Datos*: APODORAX soporta enteros, flotantes, cadenas, caracteres y booleanos.
* *Operaciones Aritméticas*: Se pueden usar sumas, restas, multiplicaciones y divisiones.
* *Operaciones Lógicas*: Se pueden usar && y ||.
* *Operaciones Relacionales*: Se pueden usar <, >, >=, <=, = =, !=.
* *Variables*: Se pueden declarar variables globales al inicio del programa y variables locales dentro de las funciones. Para la declaración de variables se necesita seguir la siguiente sintaxis: var tipo\_de\_dato nombre\_variable;

*var entero numero;*

* *Funciones*: Se pueden declarar funciones después de declarar todas las variables globales y antes de la función principal inicio. La sintaxis para declarar funciones es: funcion tipo\_de\_dato nombre\_funcion(declaración de parámetros){ }. Las funciones pueden tener o no retorno.

*funcion entero factorial (var entero x) {*

*resultado = 10;*

*regresar resultado;*

*}*

* *Arreglos*: Se pueden declarar arreglos siguiendo la siguiente sintaxis: var tipo\_de\_dato nombre\_arreglo[cantidad\_casillas];

*var entero clases[10];*

* *Condiciones*: Se pueden declarar condiciones para tener instrucciones más complejas: Condición simple: si (condición) entonces{ instrucciones }

Condición alternativa: si (condición) entonces{ inst } sino entonces { inst }

*si (x = 10) entonces {*

*y = 5;*

*} sino entonces {*

*y = 3;*

*}*

* *Ciclos*: Se pueden declarar ciclos para cuando se quiere hacer un procedimiento de manera repetitiva siguiendo la siguiente sintaxis: mientras (condición){ instrucciones }

*mientras (x < 10) {*

*x = x + 1;*

*}*

* *Escritura*: El usuario puede mostrar información en consola usando la instrucción “desplegar”:

*desplegar(“Hola Mundo”);*

* *Ingreso*: El usuario puede ingresar información en consola usando la instrucción “entrada”, la cual permite asignarle un valor a una variable:

*entrada(num);*

* Asignación: Se pueden asignar valores a las variables o subíndices de arreglos siguiendo la sintaxis: variable = valor;

*saludo = “hola”;*

*clases[3] = 5;*

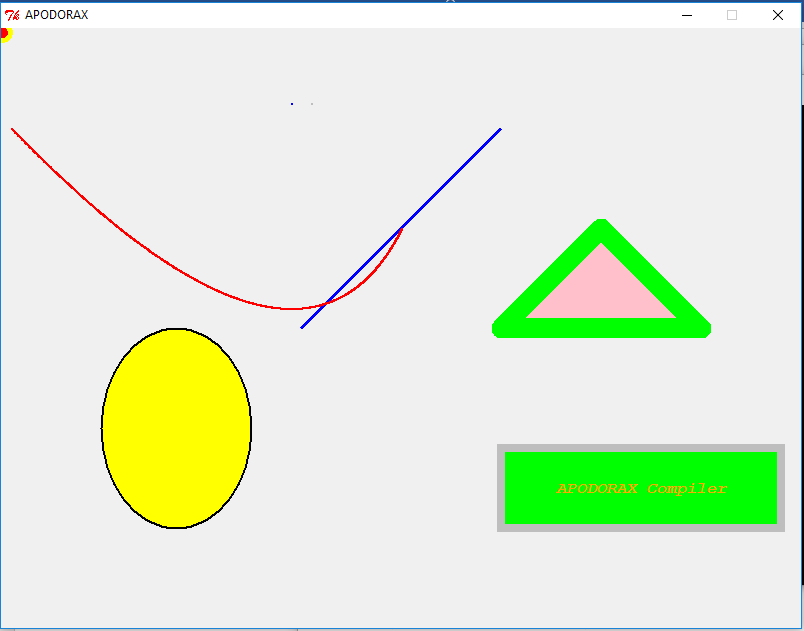
#### Elementos Gráficos

Asimismo, la librería a usar permite dibujar ciertas figuras, por lo que los datos necesarios para dibujar cada una de ellas son los siguientes:

* *Punto*: Permite dibujar un punto teniendo como datos la coordenadaX, coordenadaY, y el color del punto.
* *Texto*: Permite dibujar una línea teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, el texto a desplegar y el tamaño del texto.
* *Rectángulo*: Permite dibujar un rectángulo teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, el color del relleno, el color del contorno y el tamaño del grosor del contorno.
* *Triángulo*: Permite dibujar un triángulo teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, coordenadaX3, coordenadaY3, el color del relleno, el color del contorno y el tamaño del grosor del contorno.
* *Círculo*: Permite dibujar un círculo teniendo como datos la coordenadaX, coordenadaY, el tamaño del radio, el color del relleno y el grosor del contorno.
* *Óvalo*: Permite dibujar un óvalo teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, el color del relleno, el color del contorno y el tamaño del grosor del contorno.
* *Línea*: Permite dibujar una línea teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2, el color del relleno y el tamaño del grosor del contorno.
* *Curva*: Permite dibujar una curva teniendo como datos la coordenadaX1, coordenadaY1, coordenadaX2, coordenadaY2 y el color de la curva.

Cabe destacar que los colores disponibles son: rojo, amarillo, azul, rosa, naranja, verde, negro y gris.

A continuación se muestra una imagen en donde se aplicaron cambios en los colores, grosores y figuras, de forma que se puedan apreciar los posibles resultados.



***Curva***

***Punto***

***Línea***

***Óvalo***

***Triángulo***

***Rectángulo***

***Texto***

### Descripción de los Errores que Pueden Ocurrir (Compilación y Ejecución)

## Descripción del Compilador

### Equipo de Cómputo, Lenguaje y Utilerías Usadas en el Desarrollo del Proyecto

#### Equipo de Cómputo

Los Sistemas Operativos de las computadoras usados para la implementación y uso del lenguaje son Windows 10, Ubuntu Linux, Mac OS Sierra.

#### Lenguaje

El lenguaje en el que fue desarrollado el compilador es Python 2.7, usando la herramienta PLY como analizador de léxico y de sintaxis.

#### Utilerías

De igual manera, se hizo uso de la librería graphics.py como base para el desarrollo del output gráfico, con la cual se pueden mostrar puntos, líneas, círculos, óvalos, rectángulos, polígonos, texto e imágenes de manera gráfica.

### Descripción del Análisis de Léxico

#### Patrones de Construcción (Expresiones Regulares)

APODORAX mantiene un registro de tokens y palabras reservadas, las cuales se encuentran establecidas en el escáner del proyecto y se enlistan a continuación.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Token** | **Expresión Regular** | **Token** | **Expresión Regular** | **Token** | **Expresión Regular** |
| PUNTOYCOMA | r'\;' | CORCHETEIZQ | r'\[' | MAYORQUE | r'\>' |
| DOSPUNTOS | r'\:' | CORCHETEDER | r'\]' | MAYORIGUAL | r'\<\=' |
| COMA | r'\,' | SUMA | r'\+' | MENORIGUAL | r'\>\=' |
| PARENIZQUIERDO | r'\(' | RESTA | r'\-' | ASIGNACION | r'\=' |
| PARENDERECHO | r'\)' | MULTIPLICACION | r'\\*' | IGUAL | r'\=\=' |
| LLAVEIZQUIERDO | r'\{' | DIVISION | r'\/' | DIFERENTE | r'\!\=' |
| LLAVEDERECHO | r'\}' | MENORQUE | r'\<' |  | |
| CONJUNCION | r'\&\&' | DISYUNCION | r'\|\|' |
| CCADENA | r'\"(\\.|[^"])\*\"' | | CCARACTER | r'\'[a-z]\'|\'[A-Z]\'' | |
| ID | r'[a-zA-Z](\_?[a-zA-Z0-9]+)\*' | | CFLOTANTE | r'([\+|-]?[0-9]+[.])[0-9]+' | |
| CENTERO | r'[-]?[0-9]+' | | COMMENT | r'\#.\*' | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Tokens** | | | |
| PROGRAMA | ID | INICIO | FIN |
| FUNCION | VAR | CONJUNCION | DISYUNCION |
| CENTERO | ENTERO | CFLOTANTE | FLOTANTE |
| CCADENA | CADENA | CCARACTER | CARACTER |
| BOOL | SI | NO | SINO |
| ENTONCES | MIENTRAS | ENTRADA | DESPLEGAR |
| VERDADERO | FALSO | VACIO | REGRESAR |
| MENORQUE | MAYORQUE | MAYORIGUAL | MENORIGUAL |
| DIFERENTE | IGUAL | SUMA | RESTA |
| MULTIPLICACION | DIVISION | PARENIZQUIERDO | PARENDERECHO |
| CORCHETEIZQ | CORCHETEDER | LLAVEIZQUIERDO | LLAVEDERECHO |
| PUNTOYCOMA | COMA | DOSPUNTOS | ASIGNACION |

|  |  |  |  |
| --- | --- | --- | --- |
| **Palabras Reservadas** | | | |
| programa | entero | flotante | cadena |
| caracter | bool | verdadero | falso |
| inicio | fin | var | funcion |
| entrada | desplegar | mientras | regresar |
| si | sino | no | entonces |
| vacio | insertaTexto | insertaRectangulo | insertaTriangulo |
| insertaCirculo | insertaOvalo | insertaPunto | insertaCurva |
| insertaLinea |  |  |  |

#### Tokens del Lenguaje y su Código Asociado

De igual manera, para la formación de los cuádruplos se utilizó código de operación en lugar de los tokens, por lo a continuación se muestra cada token con su respectivo código.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tokens** | | | |
| PROGRAMA | 1 | ID | 2 |
| INICIO | 3 | FIN | 4 |
| FUNCION | 5 | VAR | 6 |
| CONJUNCIÓN | 7 | DISYUNCIÓN | 8 |
| CENTERO | 9 | ENTERO | 10 |
| CFLOTANTE | 11 | FLOTANTE | 12 |
| CCADENA | 13 | CADENA | 14 |
| CCARACTER | 15 | CARACTER | 16 |
| BOOL | 17 | SI | 18 |
| NO | 19 | SINO | 20 |
| ENTONCES | 21 | MIENTRAS | 22 |
| ENTRADA | 23 | DESPLEGAR | 24 |
| VERDADERO | 25 | FALSO | 26 |
| VACIO | 27 | REGRESAR | 28 |
| MENORQUE | 29 | MAYORQUE | 30 |
| MENORIGUAL | 31 | MAYORIGUAL | 32 |
| DIFERENTE | 33 | IGUAL | 34 |
| SUMA | 35 | RESTA | 36 |
| MULTIPLICACION | 37 | DIVISION | 38 |
| PARENIZQUIERDO | 39 | PARENDERECHO | 40 |
| CORCHETEIZQ | 41 | CORCHETEDER | 42 |
| LLAVEIZQUIERDO | 43 | LLAVEDERECHO | 44 |
| PUNTOYCOMA | 45 | COMA | 46 |
| DOSPUNTOS | 47 | ASIGNACIÓN | 48 |
| GOTOF | 49 | GOTO | 50 |
| ERA | 51 | PARAMETER | 52 |
| GOSUB | 53 | ENDPROC | 54 |
| RETURN | 55 | VER | 56 |

### Descripción del Análisis de Sintaxis

#### Gramática Formal Empleada para Representar las Estructuras Sintácticas

### Descripción de Generación de Código Intermedio y Análisis Semántico

#### Código de Operación y Direcciones Virtuales Asociadas a los Elementos del Código

#### Diagramas de Sintaxis con las Acciones Semánticas y de Código

**1**

**<PROGRAMA>**

**6**

**4 5**

**2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCacheContent.Word\PROGRAMA.PNG

**1)** Agrega a la tabla de símbolos las funciones predefinidas (figuras) y sus variables.

**2)** Inicializa la cantidad de variables de una función en cero.

**3)** Genera cuádruplo de Goto al main.

**4)** Llena el cuádruplo del Goto al main con el número del cuádruplo donde se encuentra.

**5)** Agrega la función inicio a la tabla de símbolos junto con su scope.

**6)** Generar los cuádruplos del programa y mandar a ejecutar la máquina virtual.

**<TRIANGULO>**

**5 6**

**4**

**1 2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\TRIANGULO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<TEXTO>**

**5 6**

**4**

**1 2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\TEXTO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<RECTANGULO>**

**1 2 3**

**4**

**5 6**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\RECTANGULO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<PUNTO>**

**5 6**

**4**

**1 2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\PUNTO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<OVALO>**

**5 6**

**1 2 3**

**4**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\OVALO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<LÍNEA>**

**1 2 3**

**4**

**5 6**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LINEA.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<CURVA>**

**5 6**

**4**

**1 2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CURVA.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

**<CIRCULO>**

**5 6**

**4**

**1 2 3**

C:\Users\César Benavides\AppData\Local\Microsoft\Windows\INetCache\Content.Word\CIRCULO.PNG

**1)** Obtiene el ID de la función actual.

**2)** Buscar que la función esté en la tabla de símbolos.

**3)** Guardar el ID de la función actual para hacer las llamadas posteriores.

**4)** Generar el cuádruplo de ERA.

**5)** Checar que la cantidad de argumentos sea la misma con la que está guardada esa función.

**6)** Generar el cuádruplo GOSUB.

#### Tabla de Consideraciones Semánticas

APODORAX toma en consideración algunos aspectos semánticos para ver la compatibilidad entre tipos, por lo que se formó un Cubo Semántico para mantener la relación entre los distintos tipos de datos.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **var1** | **var2** | **-** | **+** | **\*** | **/** | **&&** | **| |** | **>** | **<** | **>=** | **<=** | **!=** | **==** | **=** |
| entero | entero | ENT | ENT | ENT | FLOAT | ERROR | ERROR | BOOL | BOOL | BOOL | BOOL | BOOL | BOOL | ENT |
| entero | flotante | FLOAT | FLOAT | FLOAT | FLOAT | ERROR | ERROR | BOOL | BOOL | BOOL | BOOL | BOOL | BOOL | ENT |
| entero | cadena | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| entero | caracter | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| entero | bool | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| flotante | entero | FLOAT | FLOAT | FLOAT | FLOAT | ERROR | ERROR | BOOL | BOOL | BOOL | BOOL | BOOL | BOOL | FLOAT |
| flotante | flotante | FLOAT | FLOAT | FLOAT | FLOAT | ERROR | ERROR | BOOL | BOOL | BOOL | BOOL | BOOL | BOOL | FLOAT |
| flotante | cadena | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| flotante | caracter | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| flotante | bool | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| cadena | entero | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| cadena | flotante | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| cadena | cadena | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | BOOL | BOOL | CADENA |
| cadena | caracter | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| cadena | bool | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| caracter | entero | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| caracter | flotante | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| caracter | cadena | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| caracter | caracter | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | BOOL | BOOL | CHAR |
| caracter | bool | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| bool | entero | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| bool | flotante | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| bool | cadena | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| bool | caracter | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR | ERROR |
| bool | bool | ERROR | ERROR | ERROR | ERROR | BOOL | BOOL | ERROR | ERROR | ERROR | ERROR | BOOL | BOOL | BOOL |

## Descripción de la Máquina Virtual

### Equipo de Cómputo, Lenguaje y Utilerías Usadas

#### Equipo de Cómputo

Los Sistemas Operativos de las computadoras usados para la implementación y uso de la máquina virtual son Windows 10, Ubuntu Linux, Mac OS Sierra.

#### Lenguaje

El lenguaje en el que fue desarrollada la máquina virtual es Python 2.7, usando la herramienta PLY como analizador de léxico y de sintaxis.

#### Utilerías

De igual manera, se hizo uso de la librería graphics.py como base para el desarrollo del output gráfico, con la cual se pueden mostrar puntos, líneas, círculos, óvalos, rectángulos, polígonos, texto e imágenes de manera gráfica.

### Proceso de Administración de Memoria en Ejecución (Arquitectura)

#### Especificación Gráfica de Cada Estructura de Datos Usada (Memoria Local, global, etc..)

#### Asociación Hecha entre las Direcciones Virtuales (Compilación) y las Reales (Ejecución).

## Pruebas del Funcionamiento del Lenguaje

### Pruebas que Comprueban el Funcionamiento del Proyecto

#### Codificación de la Prueba (APODORAX)

##### Fibonacci Iterativo

programa fibonacciIterativo:

var entero result;

var entero term1;

var entero term2;

var entero i;

var entero num;

inicio {

result = 1;

term1 = 0;

term2 = 1;

i = 1;

entrada(num);

mientras (i < num){

result = term1 + term2;

term1 = term2;

term2 = result;

i = i + 1 ;

}

desplegar (result);

}

fin

##### Fibonacci Recursivo

programa fibonacciRecursivo:

var entero res;

var entero num;

funcion entero fibonacci(var entero x) {

var entero resultado;

var entero aux1;

var entero aux2;

si (x == 0) entonces {

resultado = 0;

}

si (x == 1) entonces {

resultado = 1;

}

si (x > 1) entonces {

aux1 = fibonacci(x - 1);

aux2 = fibonacci(x - 2);

resultado = aux1 + aux2;

}

regresar resultado;

}

inicio {

entrada(num);

res = fibonacci(num);

desplegar(res);

}

fin

##### Factorial Iterativo

programa factorialIterativo:

var entero num;

var entero cont;

var entero factorial;

inicio {

factorial = 1;

cont = 1;

entrada(num);

mientras (cont <= num) {

factorial = factorial \* cont;

cont = cont + 1;

}

desplegar(factorial);

}

fin

##### Factorial Recursivo

programa factorialRecursivo:

var entero res;

var entero num;

funcion entero factorial(var entero x) {

var entero resultado;

var entero aux;

si (x <= 1) entonces {

resultado = 1;

} sino entonces {

aux = factorial(x - 1);

resultado = aux \* x;

}

regresar resultado;

}

inicio {

entrada(num);

res = factorial(num);

desplegar(res);

}

fin

##### Bubble Sort

programa bubble\_sort:

var entero i[9];

var entero va;

var bool swapped;

var entero swap;

var entero j;

var entero w;

var entero z;

inicio {

# Inicialización de variables

w = 0;

z = 8;

# se piden los valores para las casillas del arreglo

mientras(z >= w) {

entrada(i[w]);

w = w + 1;

}

va = 0;

swapped = verdadero;

mientras (swapped == verdadero) {

swapped = falso;

va = 0;

mientras (va < 8) {

si (i[va] > i[va + 1]) entonces {

swap = i[va];

i[va] = i[va + 1];

i[va + 1] = swap;

swapped = verdadero;

}

va = va + 1;

}

}

j = 0;

mientras (j < 9) {

desplegar(i[j]);

j = j + 1;

}

}

Fin

##### Find

programa busqueda:

var entero lista[3];

var caracter opcion;

var entero i;

var entero objetivo;

var entero pos;

var bool encontrado;

inicio {

i = 0;

encontrado = falso;

lista[0] = 1;

lista[1] = 2;

lista[2] = 3;

opcion = 'h';

mientras(opcion != 'v' && opcion != 'i') {

desplegar("¿Quiere buscar valor o indice?[v/i]");

entrada(opcion);

si (opcion != 'v' && opcion != 'i') entonces {

desplegar("Ingrese un valor valido.");

}

}

si (opcion == 'i') entonces {

desplegar("Ingrese el valor que desea buscar:");

entrada(objetivo);

mientras(i < 3) {

si(lista[i] == objetivo) entonces {

encontrado = verdadero;

pos = i;

i = 3;

}

i = i + 1;

}

si (encontrado == verdadero) entonces {

desplegar(pos);

} sino entonces {

desplegar("No se encontró.");

}

} sino entonces {

desplegar("Ingrese la posición que desea buscar:");

entrada(objetivo);

si (objetivo > 2 || objetivo < 0) entonces {

desplegar("Esa posición no existe.");

} sino entonces {

desplegar(lista[objetivo]);

}

}

}

fin

##### Producto Punto

programa producto\_punto:

var entero vec1[5];

var entero vec2[5];

var entero res[5];

var entero acumulado;

var entero i;

var entero j;

inicio {

i = 1;

j = 1;

acumulado = 0;

mientras (i < 5) {

vec1[i - 1] = i;

i = i + 1;

}

Mientras (i < 10) {

vec2[i + -5] = i;

i = i + 1;

}

i = 1;

mientras (i < 5) {

res[i - 1] = vec1[i - 1] \* vec2[i - 1];

i = i + 1;

}

i = 1;

mientras (i < 5) {

acumulado = acumulado + res[i - 1];

i = i + 1;

}

desplegar(acumulado);

}

fin

#### Resultados Arrojados por la Generación de Código Intermedio y por la Ejecución

##### Fibonacci Iterativo

0:['Goto', '', '', 1]

1:['=', 15000, '', 0]

2:['=', 15001, '', 1]

3:['=', 15002, '', 2]

4:['=', 15003, '', 3]

5:['entrada', '', '', 4]

6:['<', 3, 4, 12000]

7:['GotoF', 12000, '', 15]

8:['+', 1, 2, 10000]

9:['=', 10000, '', 0]

10:['=', 2, '', 1]

11:['=', 0, '', 2]

12:['+', 3, 15004, 10001]

13:['=', 10001, '', 3]

14:['Goto', '', '', 6]

15:['escritura', '', '', 0]

##### Fibonacci Recursivo

0:['Goto', '', '', 23]

1:['==', 5006, 15000, 12000]

2:['GotoF', 12000, '', 4]

3:['=', 15001, '', 5007]

4:['==', 5006, 15002, 12001]

5:['GotoF', 12001, '', 7]

6:['=', 15003, '', 5007]

7:['>', 5006, 15004, 12002]

8:['GotoF', 12002, '', 21]

9:['ERA', 'fibonacci', '', '']

10:['-', 5006, 15005, 10000]

11:['PARAMETER', 10000, '', 'arg #0']

12:['GOSUB', 'fibonacci', '', 'dir']

13:['=', 2, '', 5008]

14:['ERA', 'fibonacci', '', '']

15:['-', 5006, 15006, 10001]

16:['PARAMETER', 10001, '', 'arg #0']

17:['GOSUB', 'fibonacci', '', 'dir']

18:['=', 2, '', 5009]

19:['+', 5008, 5009, 10002]

20:['=', 10002, '', 5007]

21:['RETURN', 5007, '', '']

22:['ENDPROC', '', '', '']

23:['entrada', '', '', 1]

24:['ERA', 'fibonacci', '', '']

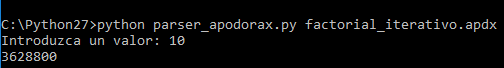
25:['PARAMETER', 1, '', 'arg #0']

26:['GOSUB', 'fibonacci', '', 'dir']

27:['=', 2, '', 0]

28:['escritura', '', '', 0]

##### Factorial Iterativo

0:['Goto', '', '', 1]

1:['=', 15000, '', 2]

2:['=', 15001, '', 1]

3:['entrada', '', '', 0]

4:['<=', 1, 0, 12000]

5:['GotoF', 12000, '', 11]

6:['\*', 2, 1, 10000]

7:['=', 10000, '', 2]

8:['+', 1, 15002, 10001]

9:['=', 10001, '', 1]

10:['Goto', '', '', 4]

11:['escritura', '', '', 2]

##### Factorial Recursivo

0:['Goto', '', '', 14]

1:['<=', 5006, 15000, 12000]

2:['GotoF', 12000, '', 5]

3:['=', 15001, '', 5007]

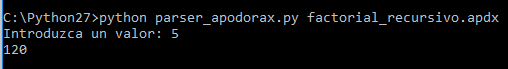
4:['Goto', '', '', 12]

5:['ERA', 'factorial', '', '']

6:['-', 5006, 15002, 10000]

7:['PARAMETER', 10000, '', 'arg #0']

8:['GOSUB', 'factorial', '', 'dir']

9:['=', 2, '', 5008]

10:['\*', 5008, 5006, 10001]

11:['=', 10001, '', 5007]

12:['RETURN', 5007, '', '']

13:['ENDPROC', '', '', '']

14:['entrada', '', '', 1]

15:['ERA', 'factorial', '', '']

16:['PARAMETER', 1, '', 'arg #0']

17:['GOSUB', 'factorial', '', 'dir']

18:['=', 2, '', 0]

19:['escritura', '', '', 0]

##### Bubble Sort

0:['Goto', '', '', 1]

1:['=', 15001, '', 12]

2:['=', 15002, '', 13]

3:['>=', 13, 12, 12000]

4:['GotoF', 12000, '', 12]

5:['VER', 12, 0, 8]

6:['+', 12, '0\_', 10000]

7:['+', 10000, '0\_', 10000]

8:['entrada', '', '', '\_10000']

9:['+', 12, 15003, 10001]

10:['=', 10001, '', 12]

11:['Goto', '', '', 3]

12:['=', 15004, '', 9]

13:['=', 17000, '', 2000]

14:['==', 2000, 17001, 12001]

15:['GotoF', 12001, '', 51]

16:['=', 17002, '', 2000]

17:['=', 15005, '', 9]

18:['<', 9, 15006, 12002]

19:['GotoF', 12002, '', 50]

20:['VER', 9, 0, 8]

21:['+', 9, '0\_', 10002]

22:['+', 10002, '0\_', 10002]

23:['+', 9, 15007, 10003]

24:['VER', 10003, 0, 8]

25:['+', 10003, '0\_', 10004]

26:['+', 10004, '0\_', 10004]

27:['>', '\_10002', '\_10004', 12003]

28:['GotoF', 12003, '', 47]

29:['VER', 9, 0, 8]

30:['+', 9, '0\_', 10005]

31:['+', 10005, '0\_', 10005]

32:['=', '\_10005', '', 10]

33:['VER', 9, 0, 8]

34:['+', 9, '0\_', 10006]

35:['+', 10006, '0\_', 10006]

36:['+', 9, 15008, 10007]

37:['VER', 10007, 0, 8]

38:['+', 10007, '0\_', 10008]

39:['+', 10008, '0\_', 10008]

40:['=', '\_10008', '', '\_10006']

41:['+', 9, 15009, 10009]

42:['VER', 10009, 0, 8]

43:['+', 10009, '0\_', 10010]

44:['+', 10010, '0\_', 10010]

45:['=', 10, '', '\_10010']

46:['=', 17003, '', 2000]

47:['+', 9, 15010, 10011]

48:['=', 10011, '', 9]

49:['Goto', '', '', 18]

50:['Goto', '', '', 14]

51:['=', 15011, '', 11]

52:['<', 11, 15012, 12004]

53:['GotoF', 12004, '', 61]

54:['VER', 11, 0, 8]

55:['+', 11, '0\_', 10012]

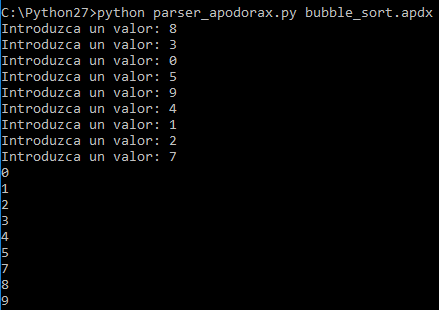
56:['+', 10012, '0\_', 10012]

57:['escritura', '', '', '\_10012']

58:['+', 11, 15013, 10013]

59:['=', 10013, '', 11]

60:['Goto', '', '', 52]



##### Find

0:['Goto', '', '', 1]

1:['=', 15001, '', 3]

2:['=', 17000, '', 2000]

3:['VER', 15002, 0, 2]

4:['+', 15002, '0\_', 10000]

5:['+', 10000, '0\_', 10000]

6:['=', 15003, '', '\_10000']

7:['VER', 15004, 0, 2]

8:['+', 15004, '0\_', 10001]

9:['+', 10001, '0\_', 10001]

10:['=', 15005, '', '\_10001']

11:['VER', 15006, 0, 2]

12:['+', 15006, '0\_', 10002]

13:['+', 10002, '0\_', 10002]

14:['=', 15007, '', '\_10002']

15:['=', 19000, '', 4000]

16:['!=', 4000, 19001, 12000]

17:['!=', 4000, 19002, 12001]

18:['&&', 12000, 12001, 12002]

19:['GotoF', 12002, '', 28]

20:['escritura', '', '', 18002]

21:['entrada', '', '', 4000]

22:['!=', 4000, 19003, 12003]

23:['!=', 4000, 19004, 12004]

24:['&&', 12003, 12004, 12005]

25:['GotoF', 12005, '', 27]

26:['escritura', '', '', 18006]

27:['Goto', '', '', 16]

28:['==', 4000, 19005, 12006]

29:['GotoF', 12006, '', 51]

30:['escritura', '', '', 18008]

31:['entrada', '', '', 4]

32:['<', 3, 15008, 12007]

33:['GotoF', 12007, '', 45]

34:['VER', 3, 0, 2]

35:['+', 3, '0\_', 10003]

36:['+', 10003, '0\_', 10003]

37:['==', '\_10003', 4, 12008]

38:['GotoF', 12008, '', 42]

39:['=', 17001, '', 2000]

40:['=', 3, '', 5]

41:['=', 15009, '', 3]

42:['+', 3, 15010, 10004]

43:['=', 10004, '', 3]

44:['Goto', '', '', 32]

45:['==', 2000, 17002, 12009]

46:['GotoF', 12009, '', 49]

47:['escritura', '', '', 5]

48:['Goto', '', '', 50]

49:['escritura', '', '', 18017]

50:['Goto', '', '', 63]

51:['escritura', '', '', 18018]

52:['entrada', '', '', 4]

53:['>', 4, 15011, 12010]

54:['<', 4, 15012, 12011]

55:['||', 12010, 12011, 12012]

56:['GotoF', 12012, '', 59]

57:['escritura', '', '', 18022]

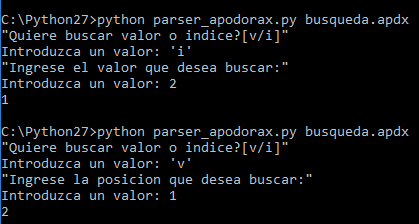
58:['Goto', '', '', 63]

59:['VER', 4, 0, 2]

60:['+', 4, '0\_', 10005]

61:['+', 10005, '0\_', 10005]

62:['escritura', '', '', '\_10005']

**

0:['Goto', '', '', 1]

1:['=', 15001, '', 3]

2:['=', 17000, '', 2000]

3:['VER', 15002, 0, 2]

4:['+', 15002, '0\_', 10000]

5:['+', 10000, '0\_', 10000]

6:['=', 15003, '', '\_10000']

7:['VER', 15004, 0, 2]

8:['+', 15004, '0\_', 10001]

9:['+', 10001, '0\_', 10001]

10:['=', 15005, '', '\_10001']

11:['VER', 15006, 0, 2]

12:['+', 15006, '0\_', 10002]

13:['+', 10002, '0\_', 10002]

14:['=', 15007, '', '\_10002']

15:['=', 19000, '', 4000]

16:['!=', 4000, 19001, 12000]

17:['!=', 4000, 19002, 12001]

18:['&&', 12000, 12001, 12002]

19:['GotoF', 12002, '', 28]

20:['escritura', '', '', 18002]

21:['entrada', '', '', 4000]

22:['!=', 4000, 19003, 12003]

23:['!=', 4000, 19004, 12004]

24:['&&', 12003, 12004, 12005]

25:['GotoF', 12005, '', 27]

26:['escritura', '', '', 18006]

27:['Goto', '', '', 16]

28:['==', 4000, 19005, 12006]

29:['GotoF', 12006, '', 51]

30:['escritura', '', '', 18008]

31:['entrada', '', '', 4]

32:['<', 3, 15008, 12007]

33:['GotoF', 12007, '', 45]

34:['VER', 3, 0, 2]

35:['+', 3, '0\_', 10003]

36:['+', 10003, '0\_', 10003]

37:['==', '\_10003', 4, 12008]

38:['GotoF', 12008, '', 42]

39:['=', 17001, '', 2000]

40:['=', 3, '', 5]

41:['=', 15009, '', 3]

42:['+', 3, 15010, 10004]

43:['=', 10004, '', 3]

44:['Goto', '', '', 32]

45:['==', 2000, 17002, 12009]

46:['GotoF', 12009, '', 49]

47:['escritura', '', '', 5]

48:['Goto', '', '', 50]

49:['escritura', '', '', 18017]

50:['Goto', '', '', 63]

51:['escritura', '', '', 18018]

52:['entrada', '', '', 4]

53:['>', 4, 15011, 12010]

54:['<', 4, 15012, 12011]

55:['||', 12010, 12011, 12012]

56:['GotoF', 12012, '', 59]

57:['escritura', '', '', 18022]

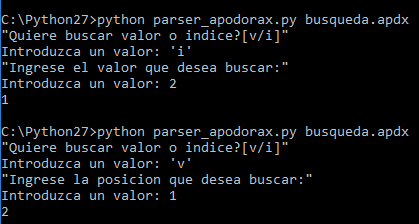
58:['Goto', '', '', 63]

59:['VER', 4, 0, 2]

60:['+', 4, '0\_', 10005]

61:['+', 10005, '0\_', 10005]

62:['escritura', '', '', '\_10005']

**

##### Producto Punto

0:['Goto', '', '', 1]

1:['=', 15003, '', 16]

2:['=', 15004, '', 17]

3:['=', 15005, '', 15]

4:['<', 16, 15006, 12000]

5:['GotoF', 12000, '', 14]

6:['-', 16, 15007, 10000]

7:['VER', 10000, 0, 4]

8:['+', 10000, '0\_', 10001]

9:['+', 10001, '0\_', 10001]

10:['=', 16, '', '\_10001']

11:['+', 16, 15008, 10002]

12:['=', 10002, '', 16]

13:['Goto', '', '', 4]

14:['<', 16, 15009, 12001]

15:['GotoF', 12001, '', 24]

16:['+', 16, 15010, 10003]

17:['VER', 10003, 0, 4]

18:['+', 10003, '0\_', 10004]

19:['+', 10004, '5\_', 10004]

20:['=', 16, '', '\_10004']

21:['+', 16, 15011, 10005]

22:['=', 10005, '', 16]

23:['Goto', '', '', 14]

24:['=', 15012, '', 16]

25:['<', 16, 15013, 12002]

26:['GotoF', 12002, '', 44]

27:['-', 16, 15014, 10006]

28:['VER', 10006, 0, 4]

29:['+', 10006, '0\_', 10007]

30:['+', 10007, '10\_', 10007]

31:['-', 16, 15015, 10008]

32:['VER', 10008, 0, 4]

33:['+', 10008, '0\_', 10009]

34:['+', 10009, '0\_', 10009]

35:['-', 16, 15016, 10010]

36:['VER', 10010, 0, 4]

37:['+', 10010, '0\_', 10011]

38:['+', 10011, '5\_', 10011]

39:['\*', '\_10009', '\_10011', 10012]

40:['=', 10012, '', '\_10007']

41:['+', 16, 15017, 10013]

42:['=', 10013, '', 16]

43:['Goto', '', '', 25]

44:['=', 15018, '', 16]

45:['<', 16, 15019, 12003]

46:['GotoF', 12003, '', 56]

47:['-', 16, 15020, 10014]

48:['VER', 10014, 0, 4]

49:['+', 10014, '0\_', 10015]

50:['+', 10015, '10\_', 10015]

51:['+', 15, '\_10015', 10016]

52:['=', 10016, '', 15]

53:['+', 16, 15021, 10017]

54:['=', 10017, '', 16]

55:['Goto', '', '', 45]

56:['escritura', '', '', 15]



## Listados Documentados del Proyecto

# *Manual de Usuario*