# Texto Descripción generada automáticamente con confianza baja

# Lenguaje Davidco

22 de Noviembre de 2022

David Sanchez Ortiz

A01283105

--David Sanchez Ortiz—

Diagrama

Descripción generada automáticamente

Contenido

[Descripción y documentación técnica del proyecto 2](#_Toc120053549)

[Propósito del Compilador: 2](#_Toc120053550)

[Alcance del Proyecto: 2](#_Toc120053551)

[Bitácora General: 2](#_Toc120053552)

[Lista de Commits del Repositorio: 5](#_Toc120053553)

[Descripción del Lenguaje: 5](#_Toc120053554)

[Nombre del lenguaje: 5](#_Toc120053555)

[Descripción genérica de las principales características del lenguaje 5](#_Toc120053556)

[Listado de errores que pueden ocurrir, tanto en compilación como en ejecución 6](#_Toc120053557)

[Descripción del Compilador 6](#_Toc120053558)

[Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto 6](#_Toc120053559)

[Descripción del Análisis Léxico: 6](#_Toc120053560)

[Descripción de Generación de Código Intermedio y Análisis Semántico. 10](#_Toc120053561)

[Código de operación y direcciones virtuales asociadas a los elementos del código: 10](#_Toc120053562)

[Diagrama de Sintaxis, Puntos neurálgicos, Semántica: 15](#_Toc120053563)

[Tabla de consideraciones semánticas (combinaciones factibles y errores de tipo). 20](#_Toc120053564)

[Descripción detallada del proceso de Administración de Memoria usado en la compilación. 21](#_Toc120053565)

[DESCRIPCION DE LA MÁQUINA VIRTUAL: 22](#_Toc120053566)

[Equipo de cómputo, lenguaje y utilerías especiales usadas (en caso de ser diferente que el compilador). 22](#_Toc120053567)

[Descripción detallada del proceso de Administración de Memoria en ejecución (Arquitectura) 22](#_Toc120053568)

[Especificación gráfica de CADA estructura de datos usada para manejo de scopes y su JUSTIFICACIÓN. (Memoria Local, global, etc..) 22](#_Toc120053569)

[Asociación hecha entre las direcciones virtuales (compilación) y las reales (ejecución). 26](#_Toc120053570)

[PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE: 26](#_Toc120053571)

[Pruebas: 26](#_Toc120053572)

[Serie Factorial 31](#_Toc120053573)

[Serie Fibonacci 32](#_Toc120053574)

# Descripción y documentación técnica del proyecto

## Propósito del Compilador:

El propósito de realizar el compilador davidco surgió debido a que primero se debía de hacer un compilador en la clase de Compiladores, pero además se trata de crear un lenguaje de programación que pueda realizar operaciones aritméticas para poder crear aspectos de estadística y operaciones varias.

## Alcance del Proyecto:

El alcance del proyecto llega hasta poder realizar funciones en la misma compilación para poder recoger diferentes variables para poder realizar operaciones aritméticas que les puede facilitar el aspecto de la estadística realizando diferentes operaciones.

## Bitácora General:

Semana 1: Decidí el nombre del proyecto y comencé a investigar de en que programa lo iba a realizar, teniendo en cuenta diferentes lenguajes como c, c++ y Python. Me puse a investigarlo porque no tenia idea de como realizar un lenguaje de programación.

--David S.--

Semana 2: No llegue a ninguna conclusión de en que lenguaje iba a realizar el compilador, comencé a realizar las tareas de la materia ya que desde el inicio de las tareas es de investigación y de ver en que lenguaje lo íbamos a realizar.

--David S.--

Semana 3: Después de la investigación de las tareas decidí realizar el lenguaje en Python ya que no conozco muy bien el lenguaje, pero me imagine que seria una buena idea comenzar a realizar el compilador mientras iba investigando la sintaxis de un nuevo lenguaje para mí.

--David S.--

Semana 4: Comenzamos a realizar las tareas de investigación de la librería que utilizaríamos para poder realizar el lexer y el parser de nuestro programa y decidí irme por ply, ya que vi diferentes proyectos que decían que la librería tenia diferentes libertades para poder realizar el compilador.

--David S.--

Semana 5: La semana estuvo muy complicada debido a que tuvimos el primer examen parcial y estuve estudiando para tener una buena calificación. Rindió frutos porque saque 92 en mi nota y ya habíamos visto los primeros aspectos de realizar un compilador.

--David S.--

Semana 6: En la semana 6 comencé a ver cuales caracteres iban a estar en mi lenguaje, todavía no tenia nada programado y lo tenia solo en mis libretas.

--David S.--

Semana 7: Termine de realizar los caracteres de mi lenguaje y a ver que diferentes palabras reservadas iban a tener. Todavía no tengo nada programado y lo tengo en papel.

--David S.--

Semana 8: No realice avance alguno solo algunas tareas que la maestra Elda nos había encargado para entender bien los conceptos del compilador.

--David S.--

Semana 9: Aquí le avance viendo cual iba a ser la gramática de mi lenguaje ya que quería comenzar con el lenguaje, pero en realidad no sabia como entonces en mis ratos libres comencé a hacer la gramática para ver como iba a funcionar mi nuevo compilador, además tuvimos nuestro examen del segundo parcial y también me puse a estudiar mucho, aunque los resultados no me favorecieron, hay algunas cosas que se me, pero se me olvidan y pues no saque buena calificación.

--David S.--

Semana 10: No le avance mucho al lenguaje ya que comenzaron mis parciales y estaba un poco estresado por ello.

--David S.--

Semana 11: Comencé a ver videos de cómo realizar el compilador, aunque muchos videos que encontré eran muy ineficientes y no entendía completamente de que estaban hablando.

--David S.--

Semana 12: En la semana termine de definir la gramática del compilador añadiendo varios aspectos que no tome en cuenta y agregando caracteres al leer sin haber programado mucho.

--David S.--

Semana 13: Tuvimos el tercer parcial y además revisión con el profesor donde le mostrábamos avances de nuestro compilador, yo solo mostré un programa que había realizado, pero no fue un avance en si porque todo lo que tenia en mis notas lo tenia que programar ya.

--David S.--

Semana 14: Ya comencé a realizar el lenguaje solo apoyándome de videos de youtube.

--David S.--

Semana 15: El lexer ya me funciona, pero tiene algunos errores por algunos aspectos que todavía no tomo en cuenta y pruebas que me faltan realizar. El parser ya esta en proceso de realizarse porque esa será mi parte importante del programa porque voy a poner todo sobre el parser. Mi gramática ya esta lista, pero le sigo añadiendo aspectos por cosas que llego a cambiar. Tengo una idea de como realizar la tabla de variables creando una clase, pero todavía no me decido como la voy a realizar. El cubo semántico si se cómo realizarlo, pero todavía no me decido si lo realizo con una clase con un archivo fuera o lo meto directamente al parser.

--David S.--

Semana 16: Ya me decidí de como dejar el cubo semántico, decidí dejarlo en mi parser y directamente ahí lo evaluó. Mi tabla de variables pensé en hacerlo en una clase, pero se me hizo un poco complicado entonces decidí ponerlo como un diccionario en mi parser. Comencé a realizar primero los stacks de mis variables y constantes que tenia en el parser, las declaré que después de que tuviera un operador, variable, constante, etc. Puse un punto neurálgico para poder realizar los movimientos de mis stacks y comenzar a realizar lo básico de mis cuádruplos. Después comencé a realizar lo de los cuádruplos y después ver que mis cuádruplos se me generaran de manera correcta. Luego ya comencé a realizar aspectos de precedencia y hacer varias pruebas para que mis cuádruplos estuvieran cada vez que se hiciera una operación. Después de varias pruebas ya me quedo bien mis cuádruplos bien implementados y comencé a realizar la máquina virtual, la información de mi parser la transformo en un json y luego lo recibo en la máquina virtual para comenzar a trabajar con la memoria. Despues de varias implementaciones mas mi memoria quedo lista y realice las funciones para poder ya compilar en mi nuevo compilador, despues de varias pruebas y muchos errores genere mi primer hello world en mi nuevo compilador davidco y quedo listo para la entrega.

--David S.--

## Lista de Commits del Repositorio:

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

# Descripción del Lenguaje:

## Nombre del lenguaje:

Davidco

## Descripción genérica de las principales características del lenguaje

En el lenguaje puedes declarar variables de tipo enteros y de tipo flotantes para tener ambos aspectos numéricos, esta seccionado en 3 partes diferentes donde se declarará lo deseado. La primera sección es var donde se pueden declarar las variables de todo el programa y donde únicamente puedes declararlas, la segunda sección es el de func de funciones para poder declarar las funciones tipo int, float o void para poder declarar funciones y el ultima sección esta main donde puedes declarar en un bloque lo que desees de asignar variables valores, escritura de variables con print en consola, un aspecto de if else de variables o un if singular también, también se pueden hacer whiles repitiendo ciclos teniendo la función while o do while y la llamada de funciones.

## Listado de errores que pueden ocurrir, tanto en compilación como en ejecución

* Si se declaran variables globales que ya tienen el mismo nombre va a haber un error.
* Si se llega a poner una variable que no se ha declarado hará error.
* Si hay algún valor que se quiere asignar siento el resultado un flotante y la variable a declarar un entero va a fallar.
* Si se trata de hacer diferentes tipos de print en el mismo lugar dará error es solo uno por línea.
* Es posible ciclar un while debido a que no se cambia el valor y no dará error, pero se ciclará y ya no funcionará.
* No puedes realizar una variable iniciando con un número, dará error.
* Si no sigues la gramática al pie de la letra dará error.
* Si al final de una acción no pones un ; dará error.
* Si al declarar una función no pones () y luego {} da error.
* Puede que haya algunos aspectos adicionales que estén implementados, pero no funcionan de la manera correcta.
* Si el string adentro del print finaliza en un espacio va a suceder un error.

# Descripción del Compilador

## Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto

Se utilizo una computadora de escritorio y una laptop para realizar el proyecto, usando el IDE de Visual Studio Code. Se utilizo el lenguaje de programación de Python añadiendo la herramienta de PLY para Python para desarrollar el principio del proyecto con el Lexer y el parte del Parser. Además, se utilizaron las notas de los profesores de la clase de Diseño de Compiladores para realizar el compilador.

## Descripción del Análisis Léxico:

Diagrama

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Tokens:

* 'ID' = '[a-zA-Z][a-zA-Z\_0-9]\*'
* 'NUMINT' = '[-]?[0-9]+'
* 'NUMFLOAT' = '[+-]?[0-9]+\.[0-9]+'
* 'STRING' = “(.\*?)”
* 'PLUS' = +
* 'MINUS' = -
* 'MULT' = \*
* 'SLASH' = /
* 'AEQL' = =
* 'EQL' = ==
* 'LESSTHAN' <
* 'NOTEQL' = !=
* 'MORETHAN' = >
* 'AND' = and
* 'OR' = or
* 'LPAR' = (
* 'RPAR' = )
* 'LKEY' = {
* 'RKEY' = }
* 'LBRK' = [
* 'RBRK' = ]
* 'COMMA' = ,
* 'SEMICOLON' = ;

Palabras Reservadas:

* 'int':'INT'
* 'float':'FLOAT'
* 'if':'IF'
* 'else':'ELSE'
* 'while':'WHILE'
* 'do':'DO'
* 'func':'FUNC'
* 'print':'PRINT'
* 'var':'VAR'
* 'func':'FUNC'
* 'main':'MAIN'
* 'void' : 'VOID'
* 'return' : 'RETURN'
* 'and' : 'AND'
* 'or' : 'OR'

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

### Código de operación y direcciones virtuales asociadas a los elementos del código:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

### Diagrama de Sintaxis, Puntos neurálgicos, Semántica:

1. resetearwatcher = Es la funcion de mi watcher para avisarme que ahora estamos en global
2. debug = imprime tablas de variables y constantes, los stacks y los cuadruplos para ver que sucede
3. salidadoc = crea mi archivo json para poder mandar mis tablas y cuadruplos a la maquina virtual y ahí recibe la informacion
4. var id = ve en donde va a guardar la variable si en la funcion de globales o en alguna funcion y tambien identifica el tipo que es si int o float.
5. Cuadexp = identifica si es + o – y le hace pop a todos mis stacks para generar un cuadruplo tambien identificando en la tabla de tipos si es correcto.
6. terminoexp = identifica si es \* o / y le hace pop a todos mis stacks para generar un cuadruplo tambien identificando en la tabla de tipos si es correcto.
7. Meteopdores = hace push a mi stack de oepradores
8. Infondo = crea un fondo falso en mi stack de operadores
9. Outfondo = hace pop al fondo falso de mi stack de oepradores
10. Meteconst = mete la constante en mi tabla de constantes viendo si es int, flaot o string, tambien haciendo push en mi stack de operandos y de tipos
11. Funciontipo = pone mi watcher diciendo que ahora las variables declaradas son globales
12. Funcionid = mete el nombre de la funcion a mi tabla de variables como una nombre de una funcion
13. Resetmem = resetea mis temporales porque termino una funcion, ademas de reinicia todos mis contadores a sus valores iniciales excepto el contador global de memoria
14. Cuadexpresionandor = identifica si es “and” o “or” y le hace pop a todos mis stacks para generar un cuadruplo tambien identificando en la tabla de tipos si es correcto.
15. cuadass = realiza un cuadruplo en asignacion y hace pop en todos los stacks
16. cuadescritura = realiza un cuadruplo de print a expresiones
17. Meteopandos = mete a los stacks de operadores y de tipos la info del id
18. Cuadescriturastring = realiza un cuadruplo de print a strings
19. Cuadcondif = realiza un cuadruplo de gotoF para el if
20. Cuadrellenoif = rellena el gotoF con el cuadruplo donde temina
21. Cuadconifelse = realiza un cuadruplo de goto para el ifelse
22. Guardarsal = guarda el salto del while para regresar y evaluar la expresion
23. cuadcondwhile = realiza un cuadruplo gotoF y falta el relleno
24. rellenoconwhile = rellena el gotoF del while
25. guardardowhile = guarda el salto del dowhile para regresar en algun punto
26. quadconddowhile = realiza un cuadruplo gotoV para regresarse en caso de ser verdadero
27. cuadexpresion = identifica si es < o > != o == y le hace pop a todos mis stacks para generar un cuadruplo tambien identificando en la tabla de tipos si es correcto.

Diagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteImagen en blanco y negro

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteImagen en blanco y negro

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente Diagrama, Esquemático

Descripción generada automáticamente

Texto

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza mediaUn dibujo de una persona

Descripción generada automáticamente con confianza mediaUn dibujo de una persona

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamente Diagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaUn dibujo de una persona

Descripción generada automáticamente con confianza bajaUn dibujo de una persona

Descripción generada automáticamente con confianza mediaUn dibujo de una persona

Descripción generada automáticamente con confianza mediaDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteTexto

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

Diagrama

Descripción generada automáticamenteUn dibujo de una persona

Descripción generada automáticamente con confianza bajaDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

### Tabla de consideraciones semánticas (combinaciones factibles y errores de tipo).

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

### Descripción detallada del proceso de Administración de Memoria usado en la compilación.

Especificación gráfica de CADA estructura de datos usada y su JUSTIFICACIÓN. (Dir.Func., Tablas de Var's, Cuádruplos, Pilas, etc.…)

Se utilizo el uso de directorios de Python para la creación de la tabla de variables y constantes, debido a que tiene una facilidad de poder manipular la información del directorio. Se puede acceder a los valores de manera sencilla y el proceso de búsqueda de un valor, variable o constante es eficiente.



Para los cuádruplos se utiliza una clase debido a que se busca poder almacenar muchos valores y al realizar una clase hay una facilidad de poder meter valores a la clase y se vayan almacenando los valores.

Texto

Descripción generada automáticamente

Se utilizaron 5 stacks para poder realizar los cuádruplos y los saltos de estos. El stack de operandos se utilizaba para almacenar los operandos del cuádruplo. El stack de operadores se utilizo par almacenar los operadores. El stack de saltos se uso para guardar los saltos y regresarse o ir al cuádruplo que estuviera arriba en el stack. El stack de tipos se utilizaba para igualar los tipos y verificar si eran compatibles las operaciones que realizaban cada uno. En el stack de cuádruplos almacenábamos los cuádruplos que íbamos creando para poder después mandarlos a la memoria virtual. Se decidió usar stacks debido a su función de almacenar la información y poder meter y sacar información y poder acceder a la información que esta siempre arriba, también por su facilidad de uso y lo conveniente que era para este proyecto.

Texto

Descripción generada automáticamente

# DESCRIPCION DE LA MÁQUINA VIRTUAL:

## Equipo de cómputo, lenguaje y utilerías especiales usadas (en caso de ser diferente que el compilador).

Se utilizo un archivo .json para generar y guardar la información del código que analizo el parser, en la máquina virtual se agarra y transforma esta información para poder ser utilizada.

## Descripción detallada del proceso de Administración de Memoria en ejecución (Arquitectura)

### Especificación gráfica de CADA estructura de datos usada para manejo de scopes y su JUSTIFICACIÓN. (Memoria Local, global, etc..)

Se copiaron los valores utilizados en el parser para después realizar una operación y dejar la memoria real en el arreglo correspondiente. Se decidió usar diferentes arreglos para poder almacenar la información correspondiente y almacenar la información sin desperdiciar memoria de más. En una función después se realiza la operación para poder saber los espacios de memoria que necesitan cada uno de los arreglos y luego asignárselas. La memoria local se reiniciara después de que pase por un punto neurálgico pero la memoria global se mantendrá ahí por el resto del programa.

Texto

Descripción generada automáticamente

Se realizan iteraciones para almacenar la información en cada lugar correspondiente

Texto

Descripción generada automáticamente

Con la función se lee y regresa en que valor tiene que estar

Texto

Descripción generada automáticamente

Se tiene una función para poder escribir la memoria y quitarle los valores que se pusieron en el parser y ya se queda con el espacio de memoria final

Texto

Descripción generada automáticamente

### Asociación hecha entre las direcciones virtuales (compilación) y las reales (ejecución).

En la función anterior se llega a quitar los valores que se asignaron en la dirección virtual y después se puede acceder a la dirección real. Se realiza esta función de quitar la memoria porque debido a que en un punto nosotros le brindamos direcciones virtuales en un programa muy largo se llega a utilizar mucha memoria debido a que va de dirección a dirección a dirección viendo donde esta el valor real y al quitarle al final la dirección virtual que le asignamos se queda la dirección real y ya va directamente a donde está la dirección de la variable.

# PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE:

## Pruebas:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente Texto

Descripción generada automáticamente

### Serie Factorial

Texto

Descripción generada automáticamente

### Serie Fibonacci

Captura de pantalla de computadora

Descripción generada automáticamente