

**Diseño de Compiladores**

**Descripción y Documentación Técnica Compilador**

**Mayka**

**Guillermo Enrique Valles Villegas – A01561722**

**25 Noviembre 2020**

**Índice**

**a) Descripción del Proyecto****……………………………………………………………………...2**

a.1) Propósito y Alcance…………………………………………………………………………..2

a.2) Análisis de Requerimientos y Casos de Uso generales..……………………………………..2

a.3) Descripción de los principales Test Cases. …………………………………………………..2

a.4) Bitácora y Reflexión………………………………………………………………………….3

**b) Descripción del Lenguaje…****…………………………………………………………………..6**

b.1) Nombre del programa………………………………………………………………………...6

b.2) Descripción Genérica del lenguaje…………………………………………………………...6

b.3) Errores que pueden ocurrir dentro del programa……………………………………………..6

**c) Descripción del Compilador………………………………………………………………….7**

c.1) Equipo de Computo, lenguaje y herramientas.……………………………………………….7

c.2) Descripción del Análisis de Léxico…………………………………………………………..8

c.3) Descripción del Análisis de Sintaxis………………………………………………………..11

c.4) Descripción de Generación de Código Intermedio y Análisis Semántico…………………..11

c.5) Descripción detallada del proceso de Administración de

Memoria usada en la compilación... …………………………………………………………….19

**d) Descripción de la Maquina Virtual…………………………………………………………21**

d.1) Equipo de Computo, lenguaje y herramientas. ……………………………………………..21

d.2) Descripción detallada del proceso de

Administración de Memoria en ejecución. ……………………………………………………...21

**e) Pruebas del funcionamiento del lenguaje…………………………………………………..23**

1. **Descripción del Proyecto.**

**a.1)** **Propósito y Alcance.**

**Propósito**:

El propósito de este proyecto es crear un compilador de lenguaje funcional poniendo en practica los conocimientos vistos en la materia de Diseño de Compiladores, cumpliendo con ciertas características propias del proyecto.

**Alcance:**

Crear un compilador que cumpla con los requerimientos del proyecto, poniendo en practica lo que se ha ido aprendiendo previamente en la carrera, así como lo nuevo que se ira aprendiendo a lo largo del semestre. El proyecto incluirá su Lexer, Parser y Maquina Virtual, que leerá los cuádruplos generados, así como un output grafico.

**a.2)** **Análisis de Requerimientos y Casos de Uso generales.**

1. Poder declarar variables de tipos (Enteras, Flotantes y Caracteres).
2. Expresiones aritméticas en orden de prioridad correcto.
3. Uso de ciclos.
4. Uso de condicionales.
5. Crear funciones y que se puedan llamar desde el main.
6. Entrada y Salida de Datos.
7. Uso de Herramienta Grafica.
8. Cubo Semántico para validación de tipos.
9. Lexer para los tokens y palabras reservadas.
10. Parser para la estructura del lenguaje.
11. Cuádruplos para las instrucciones del lenguaje.
12. Maquina virtual que pueda leer los cuádruplos,
13. Uso apropiado de memoria.

**a.3)** **Descripción de los principales Test Cases.**

|  |  |
| --- | --- |
| Prueba | Descripción |
| Cálculo Factorial | Calcular el factorial de un numero con ciclos.  Sirve para validar las expresiones, entradas, salidas, ciclos, uso de funciones, parámetros y decisiones. |
| Serie de Fibonacci | Calcular la serie de Fibonacci con ciclos.  Sirve para validar las expresiones, entradas, salidas, ciclos, uso de funciones, parámetros y decisiones. |
| Cálculo Factorial | Calcular el factorial de un numero con recursión.  Sirve para validar las expresiones, entradas, salidas, ciclos, uso de funciones, parámetros y decisiones. |
| Serie de Fibonacci | Calcular la serie de Fibonacci con recursión.  Sirve para validar las expresiones, entradas, salidas, ciclos, uso de funciones, parámetros y decisiones. |
| Sort | Ordenar un arreglo.  Valida que los arreglos se hayan implementado y que funcionen correctamente. |
| Find | Encontrar el índice de un valor dentro de un arreglo.  Valida que los arreglos se hayan implementado y que funcionen correctamente. |
| Multiplicación de Matrices | Multiplicar dos matrices.  Valida el uso correcto de matrices y que hayan sido implementadas. |
| Pruebas Especificas | Mostrar el funcionamiento del output grafico y tipos de resultados que se pueden obtener con el. |

**a.4) Bitácora y Reflexión**

**Bitácora:**

**AVANCE 1 – 8 de Octubre 2020**

Código de la sintaxis y gramática.

Me marca errores aun y no corre, pero ya tiene el cuerpo que necesita, solo es cuestión de checar esos errores y arreglarlos.

**AVANCE 2 – 14 de Octubre 2020**

Código de Semántica.

Arreglar los errores previos del avance 1, para que funcionara bien la sintaxis.

Se agregó el Cubo semántico funcional, la tabla de variables y la tabla de funciones.

**AVANCE 3 – 24 de Octubre 2020**

Se finalizaron de hacer las tablas de variables y funciones, y su construcción.

Falta investigar como acceder al valor de los TOKENS.

Clase para los stacks creada.

**AVANCE 4 – 30 de Octubre 2020**

Agregada la clase para los Cuádruplos.

Pendiente crear excepciones de PRINT, GOTO, etc.

Investigación de uso de diccionarios por medio de un curso para Python. Que me permitirá poder utilizar de mejor manera mis estructuras.

**AVANCE 5 – 6 de Noviembre 2020**

Diseño de la arquitectura.

Se creo la clase para los stacks.

Las tablas de funciones y variables ya se llenan con lo que lee.

**AVANCE 6 – 13 de Noviembre 2020**

Tabla Variables Funcional.

Tabla Funciones Funcional.

Cuádruplos Estatutos Funcionales.

Cuádruplos Funciones Especiales.

Cuádruplos Aritméticos Funcionales.

Los cuádruplos ya casi están listos. Faltan algunas modificaciones para verificar que funcionen bien.

Se eliminó el error en el que no se podían tener varios estatutos seguidos.

Los cuádruplos fueron arreglados.

No se pueden agregar valores de funciones a variables.

**AVANCE 7 – 21 de Noviembre 2020**

Maquina Virtual - Funcional Para Cuádruplos Simples.

Maquina Virtual - Funcional Para Cuádruplos De Funciones.

Maquina Virtual - Funcional Para Cuádruplos Condicionales.

Maquina Virtual - Ejecuta Código Grafico.

Maquina Virtual - Funciones Recursivas.

Estructura de Arreglos de Variables.

Cuádruplos de Arreglos de Variables.

Maquina Virtual – Arreglos.

La maquina virtual ya analiza los cuádruplos y toma decisiones para ejecutarlos.

Solo falta que ejecute funciones.

Se arreglo el problema donde no se podían hacer restas ni imprimir letreros.

La estructura de los arreglos ya es aceptada, tanto para declararlos como para llamar a un valor dentro de uno.

Las "tablas" de dimensiones de arreglos ya se crea con resultados (M1, M2, R).

Faltan los cuádruplos para los arreglos.

Cuádruplos de los arreglos listos y funcionales.

**25 de Noviembre 2020**

Documentación completada y entrega de proyecto.

**Reflexión:**

Desde que me platicaron en primer semestre sobre esta materia, que es la más difícil de la carrera, la idea de llevarla me daba mucho pánico. Estuve con esa mentalidad tres años hasta que llegó la hora de meterla en mi carga académica.

Definitivamente nunca había considerado terminar de esta manera este semestre. No solo disfruté mucho llevar la materia, que ya por si sola es bastante interesante, si no que el hecho de crear proyecto se fue convirtiendo poco a poco en mi parte favorita del semestre.

Me gustó mucho como este proyecto me retó y demostró lo mucho que mis habilidades han mejorado desde que era ese yo asustado de primer semestre apenas entrando en el mundo de las tecnologías computacionales. Crear algo tan complejo desde cero definitivamente será una de mis mejores experiencias lo largo de la carrera.

Me siento muy orgulloso de mi al entregar este proyecto y quiero agradecer mucho a mis profesores de esta clase por hacer este semestre tan interesante y una muy buena experiencia.

1. **Descripción del Lenguaje.**

**b.1)** El nombre de este lenguaje es ***Mayka***.

**b.2)** **Descripción Genérica del lenguaje.**

Mayka permite hacer cálculos y mostrar resultados con un output grafico. Los cálculos pueden ser bastantes variados, Mayka permite el uso de ciclos, funciones y cálculos con variables simples y de dimensiones (arreglos sencillos y matrices).

**b.3)** **Errores que pueden ocurrir dentro del programa:**

* ERROR: ARRAY INDEX OUT OF BOUNDS

Sucede cuando el valor para la dirección de un arreglo es mayor o igual que con el valor que fue declarado.

* ERROR: VARIABLE *NAME* NOT DECLARED

Sucede cuando una variable que se quiere utilizar no fue declarada al inicio del programa o dentro de alguna función.

* ERROR: TYPE MISMATCH

Sucede cuando el tipo de los datos que se quieren usar no son compatibles a lo que el sistema necesita.

* ERROR: CANNOT ASSING VALUE *TYPE* TO *TYPE*

Al igual que el anterior, este error ocurre cuando hay un error de tipos en asignaciones.

* ERROR: FUNCTION *NAME* NOT DECLARED

Sucede cuando se quiere llamar a una función que no fue declarada.

* ERROR: WRONG NUMBER OF PARAMETERS. EXPECTED: *NUM*

Sucede cuando se esta llamando una función, pero la cantidad de parámetros dada es incorrecta a como se declaro la función.

* ERROR: PARAM TYPE. EXPECTED: *TYPE* GIVEN: *TYPE*

Sucede cuando alguno de los parámetros que se dan para llamar a una función son de tipos diferentes a lo que se declararon que necesita la función.

* ERROR: VAR *NAME* NOT DECLARED AS ARRAY

Sucede cuando una variable que si existe es llamada como arreglo pero que no fue declarada como una variable dimensionada.

* ERROR: *NAME* VARIABLE IS ALREADY DECLARED

Sucede cuando se quiere volver a declarar una variable que ya fue declarada previamente.

1. **Descripción del Compilador.**

**c.1)** **Equipo de Computo, lenguaje y herramientas.**

El compilador fue creado con VisualStudioCode para MacOS. Esta programado con el lenguaje de Python.

La librería especial utilizada para el Compilador fue:

* **Sly**: Es la librería que auxilia en el proceso de análisis de semántica y sintaxis (Lexer y Parser).

**c.2)** **Descripción del Análisis de Léxico.**

* Patrones de Construcción:

CTEFLOAT = r'([0-9]+)(\.)([0-9]+)?' CTEINT = r'[0-9]+' LETRERO = r'[\"]([a-zA-Z]\*)+[\"]' CTECHAR = r'\'[a-zA-Z\_]\'' ID = r'[a-zA-Z]([a-zA-Z]|[0-9\_])\*' MINUS = r'[\-]' PLUS = r'[\+]' TIMES = r'[\\*]' DIVIDE = r'[\/]' RELOP = r'(!=)|(==)|(<=)|(>=)|(<)|(>)' ASSIGN = r'=' LOGICOS = r'(\|)|(\&)'

* Tokens del lenguaje:

LITERALS = **;**  **:** **(** **)** **{** **}** **[** **]** **,"**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PROGRAM | VAR | MODULE | VOID | INT |
| ELSE | FOR | TO | THEN | DO |
| READ | MINUS | TIMES | DIVIDE | CTEINT |
| LINE | POINT | CIRCLE | ARC | PENUP |
| FLOAT | ASSIGN | CHAR | RELOP | IF |
| WHILE | RETURN | PLUS | MAIN | WRITE |
| CTEFLOAT | CTECHAR | LETRERO | ID | LOGICOS |
| PENDOWN | COLOR | SIZE | CLEAR | CALL |

**c.3) Descripción del Análisis de Sintaxis.**

PROGRAM → program id ; PROGRAM2 PROGRAM3 PROGRAM4 PROGRAM2 → VARS program2 | ε  
PROGRAM3 → FUNCS | ε PROGRAM4→ main ( ) { ESTATUTOS }

FUNCS → FUNCS2 | ε FUNCS2 → FUNCV FUNCS | FUNCR FUNCS

FUNCV → void module id ( PARAMETROS ) ; VARS { ESTATUTOS } | ε

FUNCR → TIPOV module id ( PARAMETROS ) ; VARS { ESTATUTOS RETURN0 } | ε

LECTURA → read ( id LECUTRA2 ) ; LECUTRA2 → , id LECUTRA2 | ε

ESCRITURA → write ( ESCRITURA2 ) ; ESCRITURA2 → PRINTE | ESCRITURA3 ESCRITURA3 → , ESCRITURA2 | ε

PRINTE → letrero PRINTE2 | EXP PRINTE2 PRINTE2 → , letrero PRINTE2 | , EXP PRINTE2 | ε

DECISION → if ( EXP ) then DECISION2 DECISION3 DECISION2 → { ESTATUTOS }  
DECISION3 → else { ESTATUTOS } | ε

REPETICIONDO → while ( EXPRESION ) do { ESTATUTOS }

REPETICIONFOR → for id = EXP to EXP do { ESTATUTOS }

ESTATUTOS → ESTATUTO ESTATUTOS | ε

ESTATUTO → FUNCIONC | ASIGNACION | LECTURA | ESCRITURA | DECISION | REPETICIONDO | REPETICIONFOR | EXP | ESPECIALES

FUNCIONC → call id ( FUNCIONC2 ) ; FUNCIONC2 → EXP FUNCION3 | ε FUNCIONC3 → , FUNCIONC2 | ε

ASIGNACION → ARRAY = EXP ; | ARRAY = FUNCIOND ; | id = EXP ; | id = FUNCIOND

FUNCIOND → id ( FUNCIOND2 ) ; FUNCIOND2→ EXP FUNCIOND3 | ε FUNCIOND3→ , FUNCIOND2 | ε

ESPECIALES → LINE | POINT | CIRCLE | ARC | PENUP | PENDOWN | COLOR | SIZE | CLEAR

LINE → line ( EXP , ) ;

POINT → point ( ) ;

CIRCLE → circle ( EXP ) ;

ARC → arc ( EXP ) ;

PENUP → penup ( ) ;

PENDOWN → pendown ( ) ;

COLOR → color ( EXP, EXP, EXP ) ;

SIZE → size ( EXP ) ;

CLEAR → clear ( ) ;

EXPRESION → COMPARAR logicos COMPARAR | COMPARAR

COMPARAR → EXP relop EXP

RETURN → return ( EXP ) ;

EXP → TERMINO EXP2 EXP2 → plus EXP | minus EXP | ε

TERMINO → FACTOR TERMINO2 TERMINO2 → times TERMINO | divide TERMINO | ε

FACT → ( EXP ) | FACT2 FACT2 → FACT3 | FACT4 FACT3 → + FACT4 | - FACT4 FACT4 → VARCTE

PARAMETROS → TIPOV id PARAMETROS2 | ε PARAMETROS2 → , TIPOV id PARAMETROS2 | ε

FACTOR → ( EXP ) | plus VARCTE | minus VARCTE | VARCTE

VARCTE → FUNCIOND | ARRAY | id | ctei | ctef | ctechar | letrero

ARRAY → id [ EXP ] ARRAY2 → , EXP | ε

VARS → VARSN | VARSD

VARSN → var TIPOV : id VARSN2 ; | ε VARSN2 → , id VARSN2 | ε

VARSD → var TIPOV : id [ cteint VARSD2 ] ; VARSD2 → , cteint | ε

TIPOV → int | float | char

**c.4) Descripción de Generación de Código Intermedio y Análisis Semántico.**

* **Direcciones Virtuales**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo | Base | Dirección Global | Dirección Local | Total |
| Enteros | 10000 | 10000 - 10999 | 11000 - 14999 | 4999 |
| Flotantes | 15000 | 15000 - 15999 | 16000 - 17999 | 2999 |
| Caracteres | 18000 | 18000 - 18999 | 19000 - 19999 | 1999 |
| Temporales | 20000 | 20000 - 20999 | 21000 - 21999 | 1999 |
| Constantes | 22000 | 22000 | 23999 | 1999 |

* **Diagramas de Sintaxis y Puntos Neurálgicos**

**Diagram

Description automatically generated**

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Crea el directorio de funciones, agregando el ID del programa. Crea cuádruplo GOTO y hace push a la Pila de Saltos. |
| 2 | Pop a la pila de Saltos. Actualiza el GOTO con el contador para ir al main. |

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Luego de verificar que la variable exista, crea el cuadruplo READ. |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Push PTypes void. |
| 2 | Agrega modulo al Directorio de funciones. |
| 3 | Agrega cuádruplo de end y reinicia los contadores de variables |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega modulo al Directorio de funciones. |
| 2 | Agrega la variable constante que tendra el valor de retorno del modulo |
| 3 | Agrega cuádruplo de end y reinicia los contadores de variables |

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Crea el cuádruplo Write con expresión. |
| 2 | Agrega constante letrero a la tabla. |
| 3 | Hace push a los stacks con letreros. |
| 4 | Crea el cuádruplo Write con letrero. |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Con base si el tipo es correcto, agrega el cuádruplo con gotof y hace un push a la pila de saltos. Comienza el if. |
| 2 | Actualiza el cuádruplo de gotof. Final del if. |
| 3 | Hace el cuádruplo goto. Comienza el else. |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega el id a la pila de ID. |
| 2 | Crea cuádruplo para asignarle el valor al ID. |
| 3 | Crea cuádruplo para verificar que ID sea menor que el valor cada vez. Crea el gotof y hace Push a la pila de saltos. |
| 4 | Crea cuádruplo para sumarle 1 al ID. Actualiza el cuádruplo de gotof y hace un cuádruplo goto con pop de para volver al do. |

Diagram

Description automatically generatedDiagram, schematic

Description automatically generatedDiagram

Description automatically generated

A close up of a clock

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Verifica que la funcion haya sido declarada previamente. |
| 2 | Crea cuádruplo ERA. |
| 3 | Verifica que el tipo de parametro dado concuerde con el que se declaro y crea el cuádruplo PARAM. |
| 4 | Aumenta el counter de parametros y regresa error si se excede el numero de parametros. |
| 5 | Regresa error si no se esperaban parámetros. |
| 6 | Crea cuádruplo GOSUB. |

Diagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedA drawing of a person

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Crea el cuádruplo con el token inicial |

*Aplica para todas las funciones especiales.*

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega & o | a la Pila de Operandos |
| 2 | Crea cuádruplo con & o | |
| 3 | Agrega Relops a la pila de operandos (!=)|(==)|(<=)|(>=)|(<)|(>) |
| 4 | Crea el cuádruplo con Relops (!=)|(==)|(<=)|(>=)|(<)|(>) |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Crea el cuádruplo si hay un + o – en la pila de operandos |
| 2 | Agrega + o – a la pila de operandos |
| 3 | Crea el cuádruplo si hay un \* o / en la pila de operandos |
| 4 | Agrega \* o / a la pila de operandos |

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega fondo falso a la Pila de operandos |
| 2 | Saca el fondo falso de la Pila de Operandos |
| 3 | Crea cuádruplo para asignar el valor de EXP a la variable global del return de la función |

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega variable a la tabla de la función |
| 2 | Agrega parámetros al modulo |

Diagram

Description automatically generatedDiagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega el ID a la Pila de ID y su tipo a la pila tipos. |
| 2 | Agrega ID y DIM a la pila de Dimensiones. Agrega fondo falso a la pila de operandos. LIM1 = True. LIM2 = False |
| 3 | Crea el cuádruplo para verificar que el valor de la expresión este dentro del rango del arreglo. Con LIM1 y LIM2 sabe de cual rango. |
| 4 | LIM1 = False. LIM2 = True |
| 5 | Crea cuádruplo para sumar el valor de F a la Dirección del arreglo. |
| 6 | Agregar tipo a la Pila de Tipos |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega variable a la tabla de la función |
| 2 | Agrega variable a la tabla de la función sin hacer pop de la pila de tipos |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Hace push a la pila de ID con la variable y a la Pila de tipos el tipo de la var. |
| 2 | Agrega Variable entera a la tabla |
| 3 | Agrega a la pila de ID con el entero y a la pila de tipos int |
| 4 | Agrega Variable flotante a la tabla |
| 5 | Agrega a la pila de ID con el flotante y a la pila de tipos float |
| 6 | Agrega Variable de Carácter a la tabla |
| 7 | Agrega a la pila de ID con el carácter y a la pila de tipos char |
| 8 | Agrega Variable Letrero a la tabla |
| 9 | Agrega a la pila de ID con el Letrero y a la pila de tipos letrero |

Diagram

Description automatically generated

|  |  |
| --- | --- |
| Punto Neurálgico | Descripción |
| 1 | Agrega el tipo a la pila de tipos |
| 2 | Consigue el valor de LIM1 de cteint. R = LIM1 + 1 |
| 3 | Consigue el valor de LIM2 de cteint. R = R\*(LIM2+1) |
| 4 | M1 = R/(LIM1 +1). Si M1 es diferente de 1: M2 = M1/(LIM2+1). Size = LIM1\*LIM2. Agrega la variable a la tabla, adicionalmente con sus valores de M1, M2, LIM1, LIM2 y Size. |

* **Tabla de consideraciones Semánticas**

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo Uno** | **Tipo Dos** | **Operador** | **Tipo de Resultado** |
| Int | Int | + | **Int** |
| Float | Float | + | **Float** |
| Int | Float | + | **Float** |
| Float | Int | + | **Float** |
| Int | Int | - | **Int** |
| Float | Float | - | **Float** |
| Int | Float | - | **Float** |
| Float | Int | - | **Float** |
| Int | Int | \* | **Int** |
| Float | Float | \* | **Float** |
| Int | Float | \* | **Float** |
| Float | Int | \* | **Float** |
| Int | Int | / | **Int** |
| Float | Float | / | **Float** |
| Int | Float | / | **Float** |
| Float | Int | / | **Float** |
| Int | Int | \* | **Int** |
| Float | Float | \* | **Float** |
| Int | Float | \* | **Float** |
| Float | Int | \* | **Float** |
| Int | Int | != | **Int** |
| Float | Float | != | **Int** |
| Int | Float | != | **Int** |
| Float | Int | != | **Int** |
| Int | Int | == | **Int** |
| Float | Float | == | **Int** |
| Int | Float | == | **Int** |
| Float | Int | == | **Int** |
| Int | Int | <= | **Int** |
| Float | Float | <= | **Int** |
| Int | Float | <= | **Int** |
| Float | Int | <= | **Int** |
| Int | Int | >= | **Int** |
| Float | Float | >= | **Int** |
| Int | Float | >= | **Int** |
| Float | Int | >= | **Int** |
| Int | Int | < | **Int** |
| Float | Float | < | **Int** |
| Int | Float | < | **Int** |
| Float | Int | < | **Int** |
| Int | Int | > | **Int** |
| Float | Float | > | **Int** |
| Int | Float | > | **Int** |
| Float | Int | > | **Int** |
| Int | Int | = | **Int** |
| Float | Float | = | **Int** |
| Char | Char | = | **Int** |
| Int | Int | & | **Int** |
| Int | Int | | | **Int** |
| Int | Int | LINE | **Int** |
| Int | \_ | CIRCLE | **Int** |
| Float | \_ | CIRCLE | **Float** |
| Int | \_ | ARC | **Int** |
| Float | \_ | ARC | **Float** |
| Int | \_ | SIZE | **Int** |
| Float | \_ | SIZE | **Int** |

Cualquier combinación que no este en la tabla genera como resultado ‘error’.

**c.5) Descripción detallada del proceso de Administración de Memoria usado en la compilación.**

* **Directorio de Funciones**

El directorio de funciones es una estructura de datos de diccionario. Al momento de dar de alta una nueva función, se crean dentro otros dos diccionarios, uno que es la tabla de variables y otra la tabla de parámetros. La *cantidad* es un arreglo que contiene cuantas variables de tipo enteras, flotantes, caracteres y temporales utiliza, esto para el manejo de memoria posterior.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Función ID | Tipo | Scope | Cantidad de Cuádruplos | Variables | Parámetros | Cantidad |
|  |  |  |  |  |  |  |
| Fibonacci | Int | global | 20 | Tabla de Variables | Tabla de Parámetros | [2,0,0,0] |

* **Tabla de Variables**

La estructura de la tabla de variables es de diccionario, pero en este caso, se guardan con diferente parametrización, ya que para las variables dimensionadas se requieren guardar datos extras.

|  |  |  |
| --- | --- | --- |
| ID | Tipo | Memoria |
|  |  |  |
| X | float | 15000 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ID | Tipo | Memoria | Tamaño | M1 | M2 | L1 | L2 |
|  |  |  |  |  |  |  |  |
| Arr | Int | 10000 | 9 | 9 | 1 | 0 | 9 |

* **Tabla de Parámetros**

La tabla de Parámetros es también un diccionario, donde se guardan en base a su numero y al tipo.

|  |  |
| --- | --- |
| Numero de Parámetro | Tipo |
|  |  |
| 1 | INT |

* **Cuádruplos**

Los cuádruplos son creados con un diccionario donde se van guardando en base a la iteración.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Iteración | Operador | Termino 1 | Termino 2 | Asignar |
|  |  |  |  |  |
| 1 | + | A | B | T1 |

* **Pilas**

Para las pilas, hay una clase Stack que se inicializa con un arreglo vacío, la clase contiene definida la funciones para las acciones que se requieren hacer con los arreglos.

|  |  |
| --- | --- |
| Pila ID | Descripción |
| POper | Pila de Operaciones. Aquí van los +, -, /, etc. |
| PID | Pila de Variables. Aquí se guardan las variables que se van llamando. |
| PPar | Pila de Parámetros. Stack para hacer push cuando se lean parámetros en la llamada de una función. |
| PQPar | Pila de tipos para los cuádruplos de los Parámetros. |
| PSaltos | Pila de Saltos para ir guardando los cuádruplos donde se necesita editar el GOTO |
| PTypes | Pila de tipos. Se guarda el tipo de variable cuando se declara. |
| PQTypes | Pila de tipos para los cuádruplos. Se guardan los tipos para ir haciendo cuádruplos. |
| PDim | Pila de dimensiones, se guarda que dimensión es la que se esta usando en el arreglo. |

1. **Descripción de la Maquina Virtual.**

**d.1)** **Equipo de Computo, lenguaje y herramientas.**

El compilador fue creado con VisualStudioCode para MacOS. Esta programado con el lenguaje de Python.

La librería especial utilizada para la Maquina Virtual fue:

* **Turtle**: Librería que se utiliza para el output grafico de las funciones especiales.

**d.2)****Descripción detallada del proceso de Administración de Memoria en ejecución.**

* **Diccionario de funciones.**

Cuando lee un ERA, se manda a llamar a la función para pasar las cantidades de datos de la función al diccionario de funciones. Por cada tipo de dato, se tiene un arreglo que se va llenando con un push para tener la cantidad de espacios que se necesitan.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Función | Enteros | Flotantes | Caracteres | Temporales |
|  |  |  |  |  |
| 1 | [0,0,0,0] | [0,0] | [] | [0,0,0,0,0] |

* **Direcciones Virtuales**

|  |  |
| --- | --- |
| Enteros | 10000  10999 |
| Enteros Funciones | 11000  14999 |
| Flotantes | 15000  15999 |
| Flotantes Funciones | 16000  17999 |
| Caracteres | 18000  18999 |
| Caracteres Funciones | 19000  19999 |
| Temporales | 20000  20999 |
| Temporales Funciones | 21000  21999 |
| Constantes | 22000  23999 |

1. **Pruebas del Funcionamiento del Lenguaje.**

* **Factorial de 10**

**Text

Description automatically generated**

|  |  |  |
| --- | --- | --- |
| 1 ['GOTO', 0, 0, 15] | 10 ['=', 21003, 0, 11001] | 9 ['WRITE', 0, 0, 22004] |
| 2 ['=', 22001, 0, 11001] | 11 ['+', 10002, 22001, 10002] | 20 ['GOTO', 0, 0, 27] |
| 3 ['+', 11000, 22001, 21000] | 12 ['GOTO', 0, 0, 6] | 21 ['ERA', 0, 0, 'fact'] |
| 4 ['=', 21000, 0, 11000] | 13 ['=', 11001, 0, 22000] | 22 ['PARAM', 10000, 1, 20001] |
| 5 ['=', 22001, 0, 10002] | 14 ['END', 0, 0, 0] | 23 ['GOSUB', 'fact', 0, 2] |
| 6 ['=', 11000, 0, 21001] | 15 ['WRITE', 0, 0, 22002] | 24 ['=', 22000, 0, 10001] |
| 7 ['<', 10002, 21001, 21002] | 16 ['READ', 0, 0, 10000] | 25 ['WRITE', 0, 0, 22005] |
| 8 ['GOTOF', 21002, 0, 13] | 17 ['<', 10000, 22003, 20000] | 26 ['WRITE', 0, 0, 10001] |
| 9 ['\*', 11001, 10002, 21003] | 18 ['GOTOF', 20000, 0, 21] |

A close up of a logo

Description automatically generated

* **Fibonacci de 14 (Con output grafico) Recursión Cíclica**

Text

Description automatically generated

|  |  |  |
| --- | --- | --- |
| 1 ['GOTO', 0, 0, 2] | 8 ['GOTOF', 20001, 0, 21] | 15 ['=', 10002, 0, 10001] |
| 2 ['=', 22000, 0, 10001] | 9 ['<=', 10004, 22001, 20002] | 16 ['=', 10003, 0, 10002] |
| 3 ['=', 22001, 0, 10002] | 10 ['GOTOF', 20002, 0, 13] | 17 ['SIZE', 0, 0, 22003] |
| 4 ['=', 22002, 0, 10000] | 11 ['=', 10004, 0, 10003] | 18 ['ARC', 0, 0, 10003] |
| 5 ['=', 22000, 0, 10004] | 12 ['GOTO', 0, 0, 17] | 19 ['+', 10004, 22001, 10004] |
| 6 ['=', 10000, 0, 20000] | 13 ['+', 10001, 10002, 20003] | 20 ['GOTO', 0, 0, 6] |
| 7 ['<', 10004, 20000, 20001] | 14 ['=', 20003, 0, 10003] |

A picture containing graphical user interface

Description automatically generatedDiagram, venn diagram

Description automatically generated

* **Find**

**Text

Description automatically generated**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 ['GOTO', 0, 0, 2] | 0 ['VERIFY', 22003, 0, 7] | 19 ['\*T', 22007, 1, 20012] | 28 ['+T', 20018, 10002, 20019] | 37 ['+T', 20023, 10002, 20024] |
| 2 ['VERIFY', 22000, 0, 7] | 11 ['\*T', 22003, 1, 20006] | 20 ['+T', 20012, 10002, 20013] | 29 ['=', 22012, 0, 20019] | 38 ['=', 20024, 0, 10001] |
| 3 ['\*T', 22000, 1, 20000] | 12 ['+T', 20006, 10002, 20007] | 21 ['=', 22008, 0, 20013] | 30 ['WRITE', 0, 0, 22013] | 39 ['==', 10001, 22004, 20026] |
| 4 ['+T', 20000, 10002, 20001] | 13 ['=', 22004, 0, 20007] | 22 ['VERIFY', 22009, 0, 7] | 31 ['=', 22000, 0, 10000] | 40 ['GOTOF', 20026, 0, 43] |
| 5 ['=', 22001, 0, 20001] | 14 ['VERIFY', 22005, 0, 7] | 23 ['\*T', 22009, 1, 20015] | 32 ['=', 22001, 0, 20021] | 41 ['WRITE', 0, 0, 22014] |
| 6 ['VERIFY', 22002, 0, 7] | 15 ['\*T', 22005, 1, 20009] | 24 ['+T', 20015, 10002, 20016] | 33 ['<', 10000, 20021, 20022] | 42 ['WRITE', 0, 0, 10000] |
| 7 ['\*T', 22002, 1, 20003] | 16 ['+T', 20009, 10002, 20010] | 25 ['=', 22010, 0, 20016] | 34 ['GOTOF', 20022, 0, 45] | 43 ['+', 10000, 22002, 10000] |
| 8 ['+T', 20003, 10002, 20004] | 17 ['=', 22006, 0, 20010] | 26 ['VERIFY', 22011, 0, 7] | 35 ['VERIFY', 10000, 0, 7] | 44 ['GOTO', 0, 0, 32] |
| 9 ['=', 22003, 0, 20004] | 18 ['VERIFY', 22007, 0, 7] | 27 ['\*T', 22011, 1, 20018] | 36 ['\*T', 10000, 1, 20023] |

**Graphical user interface, text

Description automatically generated**

* **Multiplicación de Matrices (Resultado Incorrecto, pero si genera cuádruplos)**

Text

Description automatically generatedA picture containing text

Description automatically generatedText

Description automatically generatedText

Description automatically generatedText

Description automatically generated

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 ['GOTO', 0, 0, 2] | 25 ['<', 10027, 22002, 20008] | 49 ['GOTOF', 20017, 0, 59] | 73 ['+T2', 20026, 10028, 20027] | 97 ['+T2', 20035, 10029, 20036] | 121 ['=', 20050, 0, 10027] |
| 2 ['=', 22000, 0, 10027] | 26 ['GOTOF', 20008, 0, 43] | 50 ['VERIFY', 10027, 0, 3] | 74 ['+T', 20027, 10009, 20028] | 98 ['+T', 20036, 10000, 20037] | 122 ['GOTO', 0, 0, 88] |
| 3 ['=', 22000, 0, 10028] | 27 ['<', 10028, 22002, 20009] | 51 ['\*T', 10027, 4, 20018] | 75 ['=', 20028, 0, 10029] | 99 ['=', 20037, 0, 10030] | 123 ['=', 22000, 0, 10027] |
| 4 ['=', 22000, 0, 10029] | 28 ['GOTOF', 20009, 0, 39] | 52 ['VERIFY', 10028, 0, 3] | 76 ['WRITE', 0, 0, 10029] | 100 ['VERIFY', 10029, 0, 3] | 124 ['=', 22000, 0, 10028] |
| 5 ['WRITE', 0, 0, 22001] | 29 ['VERIFY', 10027, 0, 3] | 53 ['+T2', 20018, 10028, 20019] | 77 ['+', 10028, 22003, 20030] | 101 ['\*T', 10029, 4, 20039] | 125 ['WRITE', 0, 0, 22007] |
| 6 ['<', 10027, 22002, 20000] | 30 ['\*T', 10027, 4, 20010] | 54 ['+T', 20019, 10009, 20020] | 78 ['=', 20030, 0, 10028] | 102 ['VERIFY', 10028, 0, 3] | 126 ['<', 10027, 22002, 20051] |
| 7 ['GOTOF', 20000, 0, 23] | 31 ['VERIFY', 10028, 0, 3] | 55 ['=', 22005, 0, 20020] | 79 ['GOTO', 0, 0, 68] | 103 ['+T2', 20039, 10028, 20040] | 127 ['GOTOF', 20051, 0, 144] |
| 8 ['<', 10028, 22002, 20001] | 32 ['+T2', 20010, 10028, 20011] | 56 ['+', 10028, 22003, 20022] | 80 ['=', 22000, 0, 10028] | 104 ['+T', 20040, 10009, 20041] | 128 ['<', 10028, 22002, 20052] |
| 9 ['GOTOF', 20001, 0, 19] | 33 ['+T', 20011, 10000, 20012] | 57 ['=', 20022, 0, 10028] | 81 ['+', 10027, 22003, 20031] | 105 ['=', 20041, 0, 10031] | 129 ['GOTOF', 20052, 0, 140] |
| 10 ['VERIFY', 10027, 0, 3] | 34 ['=', 20012, 0, 10029] | 58 ['GOTO', 0, 0, 48] | 82 ['=', 20031, 0, 10027] | 106 ['VERIFY', 10027, 0, 3] | 130 ['VERIFY', 10027, 0, 3] |
| 11 ['\*T', 10027, 4, 20002] | 35 ['WRITE', 0, 0, 10029] | 59 ['=', 22000, 0, 10028] | 83 ['GOTO', 0, 0, 66] | 107 ['\*T', 10027, 4, 20043] | 131 ['\*T', 10027, 4, 20053] |
| 12 ['VERIFY', 10028, 0, 3] | 36 ['+', 10028, 22003, 20014] | 60 ['+', 10027, 22003, 20023] | 84 ['=', 22000, 0, 10027] | 108 ['VERIFY', 10028, 0, 3] | 132 ['VERIFY', 10028, 0, 3] |
| 13 ['+T2', 20002, 10028, 20003] | 37 ['=', 20014, 0, 10028] | 61 ['=', 20023, 0, 10027] | 85 ['=', 22000, 0, 10028] | 109 ['+T2', 20043, 10028, 20044] | 133 ['+T2', 20053, 10028, 20054] |
| 14 ['+T', 20003, 10000, 20004] | 38 ['GOTO', 0, 0, 27] | 62 ['GOTO', 0, 0, 46] | 86 ['=', 22000, 0, 10029] | 110 ['+T', 20044, 10018, 20045] | 134 ['+T', 20054, 10018, 20055] |
| 15 ['=', 22002, 0, 20004] | 39 ['=', 22000, 0, 10028] | 63 ['=', 22000, 0, 10027] | 87 ['WRITE', 0, 0, 22006] | 111 ['\*', 10000, 10009, 20047] | 135 ['=', 20055, 0, 10029] |
| 16 ['+', 10028, 22003, 20006] | 40 ['+', 10027, 22003, 20015] | 64 ['=', 22000, 0, 10028] | 88 ['<', 10027, 22002, 20032] | 112 ['=', 20047, 0, 20045] | 136 ['WRITE', 0, 0, 10029] |
| 17 ['=', 20006, 0, 10028] | 41 ['=', 20015, 0, 10027] | 65 ['=', 22000, 0, 10029] | 89 ['GOTOF', 20032, 0, 123] | 113 ['+', 10029, 22003, 20048] | 137 ['+', 10028, 22003, 20057] |
| 18 ['GOTO', 0, 0, 8] | 42 ['GOTO', 0, 0, 25] | 66 ['<', 10027, 22002, 20024] | 90 ['<', 10028, 22002, 20033] | 114 ['=', 20048, 0, 10029] | 138 ['=', 20057, 0, 10028] |
| 19 ['=', 22000, 0, 10028] | 43 ['=', 22000, 0, 10027] | 67 ['GOTOF', 20024, 0, 84] | 91 ['GOTOF', 20033, 0, 119] | 115 ['GOTO', 0, 0, 92] | 139 ['GOTO', 0, 0, 128] |
| 20 ['+', 10027, 22003, 20007] | 44 ['=', 22000, 0, 10028] | 68 ['<', 10028, 22002, 20025] | 92 ['<', 10029, 22002, 20034] | 116 ['+', 10028, 22003, 20049] | 140 ['=', 22000, 0, 10028] |
| 21 ['=', 20007, 0, 10027] | 45 ['WRITE', 0, 0, 22004] | 69 ['GOTOF', 20025, 0, 80] | 93 ['GOTOF', 20034, 0, 116] | 117 ['=', 20049, 0, 10028] | 141 ['+', 10027, 22003, 20058] |
| 22 ['GOTO', 0, 0, 6] | 46 ['<', 10027, 22002, 20016] | 70 ['VERIFY', 10027, 0, 3] | 94 ['VERIFY', 10027, 0, 3] | 118 ['GOTO', 0, 0, 90] | 142 ['=', 20058, 0, 10027] |
| 23 ['=', 22000, 0, 10027] | 47 ['GOTOF', 20016, 0, 63] | 71 ['\*T', 10027, 4, 20026] | 95 ['\*T', 10027, 4, 20035] | 119 ['=', 22000, 0, 10028] | 143 ['GOTO', 0, 0, 126] |
| 24 ['=', 22000, 0, 10028] | 48 ['<', 10028, 22002, 20017] | 72 ['VERIFY', 10028, 0, 3] | 96 ['VERIFY', 10029, 0, 3] | 120 ['+', 10027, 22003, 20050] |

Text

Description automatically generated

* **Sort**

Text

Description automatically generatedA picture containing diagram

Description automatically generatedA picture containing diagram

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| 1 ['GOTO', 0, 0, 2] | 16 ['=', 22000, 0, 10011] | 31 ['>', 10014, 10015, 20013] | 46 ['+', 10011, 22003, 20021] |
| 2 ['=', 22000, 0, 10011] | 17 ['=', 22000, 0, 10012] | 32 ['GOTOF', 20013, 0, 42] | 47 ['=', 20021, 0, 10011] |
| 3 ['=', 22000, 0, 10012] | 18 ['WRITE', 0, 0, 22004] | 33 ['=', 10014, 0, 10013] | 48 ['GOTO', 0, 0, 19] |
| 4 ['=', 22000, 0, 10013] | 19 ['<', 10011, 22002, 20005] | 34 ['VERIFY', 10011, 0, 10] | 49 ['=', 22000, 0, 10011] |
| 5 ['WRITE', 0, 0, 22001] | 20 ['GOTOF', 20005, 0, 49] | 35 ['\*T', 10011, 1, 20014] | 50 ['=', 22000, 0, 10012] |
| 6 ['<', 10011, 22002, 20000] | 21 ['<', 10012, 22002, 20006] | 36 ['+T', 20014, 10000, 20015] | 51 ['WRITE', 0, 0, 22005] |
| 7 ['GOTOF', 20000, 0, 16] | 22 ['GOTOF', 20006, 0, 45] | 37 ['=', 10015, 0, 20015] | 52 ['<', 10011, 22002, 20022] |
| 8 ['READ', 0, 0, 10012] | 23 ['VERIFY', 10011, 0, 10] | 38 ['VERIFY', 10012, 0, 10] | 53 ['GOTOF', 20022, 0, 62] |
| 9 ['VERIFY', 10011, 0, 10] | 24 ['\*T', 10011, 1, 20007] | 39 ['\*T', 10012, 1, 20017] | 54 ['VERIFY', 10011, 0, 10] |
| 10 ['\*T', 10011, 1, 20001] | 25 ['+T', 20007, 10000, 20008] | 40 ['+T', 20017, 10000, 20018] | 55 ['\*T', 10011, 1, 20023] |
| 11 ['+T', 20001, 10000, 20002] | 26 ['=', 20008, 0, 10014] | 41 ['=', 10013, 0, 20018] | 56 ['+T', 20023, 10000, 20024] |
| 12 ['=', 10012, 0, 20002] | 27 ['VERIFY', 10012, 0, 10] | 42 ['+', 10012, 22003, 20020] | 57 ['=', 20024, 0, 10012] |
| 13 ['+', 10011, 22003, 20004] | 28 ['\*T', 10012, 1, 20010] | 43 ['=', 20020, 0, 10012] | 58 ['WRITE', 0, 0, 10012] |
| 14 ['=', 20004, 0, 10011] | 29 ['+T', 20010, 10000, 20011] | 44 ['GOTO', 0, 0, 21] | 59 ['+', 10011, 22003, 20026] |
| 15 ['GOTO', 0, 0, 6] | 30 ['=', 20011, 0, 10015] | 45 ['=', 22000, 0, 10012] | 60 ['=', 20026, 0, 10011] |
|  |  |  | 61 ['GOTO', 0, 0, 52] |

A close up of a logo

Description automatically generatedA picture containing text

Description automatically generated

* Dibujar Círculos

Text

Description automatically generatedText

Description automatically generated

|  |  |  |  |
| --- | --- | --- | --- |
| 1 ['GOTO', 0, 0, 12] | 9 ['GOTO', 0, 0, 4] | 17 ['GOTOF', 20001, 0, 31] | 25 ['=', 20003, 0, 10001] |
| 2 ['WRITE', 0, 0, 22000] | 10 ['=', 10001, 0, 22001] | 18 ['WRITE', 0, 0, 10002] | 26 ['+', 10002, 22005, 20004] |
| 3 ['END', 0, 0, 0] | 11 ['END', 0, 0, 0] | 19 ['WRITE', 0, 0, 10001] | 27 ['=', 20004, 0, 10002] |
| 4 ['<', 10001, 10003, 21000] | 12 ['=', 22002, 0, 10002] | 20 ['CIRCLE', 0, 0, 10002] | 28 ['+', 10001, 22005, 20005] |
| 5 ['GOTOF', 21000, 0, 10] | 13 ['=', 22002, 0, 10001] | 21 ['\*', 10001, 22004, 20002] | 29 ['=', 20005, 0, 10001] |
| 6 ['ARC', 0, 0, 10001] | 14 ['<=', 10002, 22003, 20000] | 22 ['=', 20002, 0, 10001] | 30 ['GOTO', 0, 0, 16] |
| 7 ['\*', 10001, 10001, 21001] | 15 ['GOTOF', 20000, 0, 32] | 23 ['CIRCLE', 0, 0, 10001] | 31 ['GOTO', 0, 0, 14] |
| 8 ['=', 21001, 0, 10001] | 16 ['<=', 10001, 22003, 20001] | 24 ['\*', 10001, 22004, 20003] |

