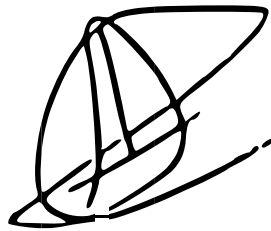




Tecnológico de Monterrey

Proyecto Final

Diseño de Compiladores



Iván Alejandro Anguiano Leal

A00817460

6 de Junio del 2022

Monterrey, Nuevo León

ÍNDICE

Descripción del proyecto	3
Visión	3
Objetivo	3
Alcance.....	3
Requerimientos	4
Requerimientos Funcionales.....	4
Requerimientos No Funcionales	4
Descripción de casos de uso	4
Proceso del desarrollo del proyecto	5
Commits.....	5
Reflexión	16
Descripción del Lenguaje	17
Nombre del Lenguaje.....	17
Descripción de las principales características del lenguaje.....	17
Errores	17
Descripción del Compilador	18
Equipo de Cómputo, Lenguaje y Utilerías especiales.....	18
Descripción del Análisis de Léxico	19
Descripción del Análisis de Sintáxis.....	20
Descripción de Generación de Código Intermedio y Análisis Semántico	24
Diagramas de Sintáxis	25
Acciones semánticas	28
Tablas de Consideraciones Semánticas.....	31
Descripción de Administración de Memoria en Compilación	32
Descripción de la Máquina Virtual.....	34
Equipo de cómputo, lenguaje y utilerías especiales usadas	34
Descripción del proceso de Administración de Memoria en ejecución	34
Pruebas de Funcionamiento del Lenguaje	35
Documentación de archivos	43
Documentación de Código del Proyecto.....	45

Descripción del proyecto

Visión

El propósito de este proyecto es hacer uso de los conocimientos del área de Computer Science a lo largo de la carrera como: Estructuras de Datos, Lenguajes de Programación, Matemáticas Computacionales; así como hacer uso de los conceptos básicos del proceso de compilación como: Análisis Léxico, Análisis Sintáctico, Análisis Semántico, Traducción y Generación de código intermedio para crear un compilador y una máquina virtual .

Objetivo

Diseñar y crear un lenguaje de programación que sea fácil de aprender y utilizar para programadores principiantes. El lenguaje contiene operaciones aritméticas básicas y operaciones booleanas así como operaciones con arreglos tanto unidimensionales como bidimensionales.

Alcance

El lenguaje contiene todos los elementos básicos de un lenguaje de programación, tales como:

- Declaración de Variables
- Declaración de Funciones/Módulos
- Llamadas de Funciones Void
- Expresiones de asignación
- Retorno de funciones
- Lectura de inputs
- Impresión de outputs
- Estatutos condicionales (if)
- Estatutos cíclicos (for, while)
- Expresiones booleanas
- Operaciones con matrices

Requerimientos

Requerimientos Funcionales

- Puede declarar funciones.
- Puede llamar funciones.
- Puede leer valores desde la línea de comandos.
- Puede imprimir valores a la consola.
- Recibe código inicializado con la palabra "program".
- Se pueden generar matrices y hacer operaciones con las mismas.
- Se puede transponer e invertir una matriz usando operadores especiales.
- Errores deben ser desplegados cuando sea necesario.

Requerimientos No Funcionales

- La sintáxis es fácil de entender para un principiante.
- El código se lee desde archivos .txt

Descripción de casos de uso

Nombre	Descripción
Factorial cíclico	Versión cíclica del cálculo de factorial.
Factorial recursivo	Versión recursiva del cálculo factorial utilizando módulos.
Fibonacci cíclico	Versión cíclica de calcular el numero N en una secuencia Fibonacci.
Fibonacci recursivo	Versión recursiva de calcular el numero N en una secuencia Fibonacci utilizando módulos.
Ordenamiento de burbuja (bubble sort)	Ordenamiento burbuja tradicional
Búsqueda en Arreglo	Encuentra un elemento en específico en un Arreglo.
Multiplicación de Matrices	Calcula la matriz resultante de la multiplicación entre 2 matrices.
Transpuesta de Matriz	Calcula la transpuesta de una matriz.
Inversa de Matriz	Genera la versión inversa de una matriz.
Determinante de Matriz	Calcula la determinante de una matriz.

Proceso del desarrollo del proyecto

Se utilizó **Github** para control de versiones. Se subían los avances semanales a **Canvas** con la descripción de lo que se había hecho hasta esa fecha.

Commits

commit 7bf5e7d3039f19a64726b08429caba7bb5aca90d (HEAD -> main, origin/main)

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Jun 3 19:55:48 2022 -0500

General bug fix

commit 5305a7b77cd2e71da4febbbbed126910ae57f23bc

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 31 21:44:06 2022 -0500

Added more array operations (inverse, transpose)

commit 51adb263777256d550274c188eeac9ae3ffd9f4a

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 31 17:58:29 2022 -0500

Added array operations + special operators

commit 90b4a46bbbf16c1b78a276c05324e413e6de738

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 31 01:28:32 2022 -0500

Started special operators, memory fix and adjustment

commit 94f30188e791edce76e4c66a3e66d8d035b7c20b

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 30 18:15:44 2022 -0500

Added missing assignments in VM + array verification

commit 36affc92390242519714bb3fb715996b1501b3cd

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 29 21:59:26 2022 -0500

Small error fix

commit 23bdb6f46ec20d04f1ef0a3e4682fddd6462beb9

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sat May 28 02:27:07 2022 -0500

Fixed warnings

commit 267bd497e6bdfa1d86f2456bdc8a792cf45a61ca

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sat May 28 02:23:34 2022 -0500

array and matrix fix

commit 8175139ac47903cff6eff230f11bca20f9c5c26e

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri May 27 17:49:49 2022 -0500

module and memory fix, updated test file

commit 7343913dcbaeacc542e854f312bbcf296dc28e58

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri May 27 17:06:14 2022 -0500

Fix

commit 9819bb4cc2066c64077530f0f1460d175726599d

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri May 27 16:34:30 2022 -0500

VM modules and arrays

commit edd78c37b3823ca96967a4625d31e1a510d8ce51

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri May 27 03:35:23 2022 -0500

Started virtual machine modules and arrays

commit a83d70fafc7f7c02aa29a66e8953fbd184bb2925

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 24 20:53:12 2022 -0500

Small fixes, added priorities and virtual machine execution for linear statements

commit 378432174879aa615af15d5d12ec074dacfc84f8

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 18 17:38:00 2022 -0500

array and matrix dimensions

commit 792868e6e3b34e618525a8bb269c22bd2cf082c7

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 17 22:44:12 2022 -0500

Added priority for parenthesis

commit 117e526086b46ffd40ae6cea5473df9418eb0cf8

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 17 16:23:07 2022 -0500

Added more coments

commit aeb6dc7a583861ad63aabb7375dd785dd656ebcb

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 16 22:55:44 2022 -0500

Updated README avance

commit b7c97749436ad5a73262d4df8310ca03a60861c7

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 16 22:51:07 2022 -0500

Updated comments

commit 50cdff62477317d34ade66b8e4713bf17328ff4a

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 16 22:39:02 2022 -0500

Fixed errors

commit e790b6bad7429f3a0da0f7fc95a39abdac457b7b

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 16 19:59:35 2022 -0500

Added new errors

commit 5dfcb86753492429e2a388d42ab2910a36aa972f

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 16 03:42:27 2022 -0500

Function calls

commit fed33f4cd8c3819b730a345d9ca8ec284b1d8565

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 15 17:44:07 2022 -0500

Fixed warnings

commit 5af3e5202c9f27c54eba37623219dbc2ef560683

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 15 17:20:24 2022 -0500

Added comments

commit d3e87933a5d90cd3ad4dea6ef3de130f7ffb2590

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 15 00:12:38 2022 -0500

More quadruples (functions)

commit 85ff9f8591b7908950eb9ca2090cdbc7805b2f70

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Thu May 12 00:38:01 2022 -0500

Some corrections

commit e108b0496f41bc09fcedc223487000f2023b0acc

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 11 23:12:47 2022 -0500

Removed some code

commit 505cecf6a01645aa173bc0d0a57eafc5e93d3b3

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 11 18:00:23 2022 -0500

Resolved conflicts and renaming

commit 257069c2943080ab6f9e2e693820360faf0c6f28

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 11 17:34:31 2022 -0500

Small correction

commit 85cd15d666d4afc1316cc81b089962f6632120bc

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 11 17:30:21 2022 -0500

While/For quadruples

commit 245c0ef74a0dfcc52163e986a8edeba8b800dc85

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 9 22:24:37 2022 -0500

if/else quadruples +readme avance #4

commit 95e6a02615b3db68ce9b0749836dea992f8b84cb

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 9 19:17:04 2022 -0500

Fix

commit 13c12f8f36d96db11e6bbbb5ab8fd7655c34fb96

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 9 18:14:13 2022 -0500

Added quadruple class

commit 38a786dd31b05ac88d75ab82c99cf76aa9f76770

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 9 17:28:06 2022 -0500

Linear statements quadruples

commit f1fc7f1cf72d3f18ebff774b2a71c0dec34b6c64

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 8 14:14:16 2022 -0500

Separated files + more quadruples

commit 0afce8be02ff11780adcad84897a8e9c77a7ea8c

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri May 6 16:20:05 2022 -0500

Semantic Cube and some quadruples

commit f7bc61034db2550cec4828fe0d74519d7073da6a

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 4 19:47:28 2022 -0500

Small fix on varTable

commit c0a7aab593b07f7fe1ee7c2a6578ec820df6767f

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Wed May 4 19:22:18 2022 -0500

Started DirFunc and varTable

commit f1cb5d1514443845ff4685f1c9f837201d00f8dc

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 3 23:31:30 2022 -0500

Small fix

commit 8c5c9f342e1f6cb0a65bad2498a509b3797d6ffb

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 3 22:17:36 2022 -0500

Separate parser and lexer into 2 different files

commit fae27dfcd7edb7b1ac6e0335390df92592aa537f

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Tue May 3 21:09:26 2022 -0500

Started semantics

commit 9b48df0f5b060adf3137a3e348b14277df95a8d4

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Mon May 2 22:55:45 2022 -0500

Started DataStructure

Started data structure to declare and initialize semantic cube, variable tables, function directory

commit 8b4b3462957e4ac06c53383d95a05760d7a092ad

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 1 17:46:43 2022 -0500

Actualizacion README Avance

commit 88e5adfa5f734ac0e38f78bf29d64d856903dc38

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 1 16:01:07 2022 -0500

More conflicts fixed + rules change

commit 361e9cfcdddee31fb63a1253fe64c3e1595f7ebcd

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 1 03:28:44 2022 -0500

Solved some conflicts

commit 326fd00999ee1f214f58110ab3fbe08b8ba7669e

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sun May 1 03:05:09 2022 -0500

tokens fix, added line number in errors

commit 994c9f9e21c6f79a6439484327947b58fd51de7c

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sat Apr 30 18:37:51 2022 -0500

Warnings fixed

commit c1d2a1d082f68d6f00cbfd40bd8f287029a6996a

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Sat Apr 30 15:13:37 2022 -0500

Typos + added FOR reserved word

commit b23d890a5703fff77c81f7677416fd7931811f7f

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Apr 29 23:23:20 2022 -0500

Module correction

commit 21863bb440d2f6906c0abd494497e54747459c4a

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Apr 29 20:14:46 2022 -0500

Function/Module rules

Agregue algunas reglas de funciones/modulos

commit 900f44072ce2a8e25ebbd61af30145091145b325

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Apr 29 18:39:45 2022 -0500

End of file new line

Updated end of file

commit 62633f5e9a9e6986044677c5bf6576735895e80b

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Apr 29 18:38:25 2022 -0500

README Update + Syntax rules

commit 1f8ee52a8e82cea072261be2082e07080dbf2fdd

Author: IAAL96 <ivan.anguiano17@hotmail.com>

Date: Fri Apr 29 16:30:19 2022 -0500

Initial commit

(END)

Reflexión

Todo este proceso fue muy interesante, todo un reto, si de por si este proyecto tiene la fama de ser difícil y estresante, tuve que también cargar con la presión de que probablemente es mi último semestre, por lo que sabía que tenía que aplicarme y que le tenía que dedicar mucho tiempo a este proyecto.

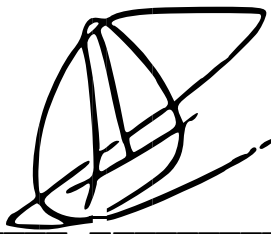
Sabiendo eso, aún así tomé la decisión de iniciar el proyecto con 2 semanas de retraso pues la fecha oficial de los primeros avances eran en fechas de CENEVAL, parciales, además que se me juntaba con otros pendientes, ya era mucho el estrés que cargaba y decidí “descansar” un poco sabiendo que me tenía que poner al corriente en las semanas posteriores.

Empecé con algo de incertidumbre pues en el primer examen me había ido pésimo y no me tenía mucha confianza pues leyendo el material de la clase de semestres anteriores no entendía mucho, pero gracias a las clases presenciales y a que se nos proporcionaron las grabaciones de clases de semestres anteriores me fue más fácil entender todos los conceptos y lo que tenía que hacer para desarrollar el compilador.

Me puse al corriente para la cuarta semana dedicándole varias horas diarias al proyecto, y mediante más iba avanzando más me hacía sentido lo que estaba desarrollando con lo aprendido en clase por lo que batallaba menos a la hora de tener que corregir los errores y bugs.

Al final terminé el proyecto unos 3 días antes de la fecha de entrega, pero cometí el pequeño error de descuidar la documentación por lo que tuve que hacerla completa en esos 3 días, si bien no es difícil, si es algo tedioso ya que toma mucho tiempo el hacer todos los diagramas.

Como conclusión, fue difícil, estresante por momentos, pero todo eso se convierte en aprendizaje, creo que este tipo de proyectos te ayudan a crecer como programador, al final se logró terminarlo y pues como dicen en el fútbol... “sufriendo sabe mejor”.



Iván Alejandro Anguiano Leal

Descripción del Lenguaje

Nombre del Lenguaje

Pug++

Descripción de las principales características del lenguaje

Es un lenguaje de programación que contiene operaciones aritméticas y booleanas simples. Se puede usar para aprender sobre programación con usos básicos de almacenamiento de memoria temporal e input y output de resultados. También se puede hacer uso de arreglos simples y arreglos bidimensionales con operaciones aritméticas básicas.

Errores

Compilación	
Error sintáctico.	Token inesperado en alguna línea en específico
Type mismatch.	Type mismatch en la asignación de una variable.
Type mismatch en operación.	Operandos en una operación aritmética no son compatibles.
Type mismatch en condición.	Operandos en una operación condicional no son del mismo tipo.
Variable indefinida.	Se usó una variable indefinida.
Redefinición de variable.	Se define una Variable con un ID que ya fue usado y ya no puede volver a usarse.
Número de argumentos inesperados.	Argumentos en el uso del Módulo no coinciden con los que se usaron en la declaración del Módulo.
Type mismatch Módulo.	Tipo del Módulo y la variable asignada no son tipos compatibles.
Return en Función Void.	Hay un Return en una Función tipo Void.
No hay Return en tipo Función.	Un tipo Función no tiene valor de Return.
Matriz accesada como Arreglo.	Una variable de tipo Matriz es llamada con sólo 1 índice.
Type mismatch en Index.	Índice usado en llamada de Arreglo no es Int.
Variable no subindicada como Matriz.	Una variable que no es matriz es llamada con 2 o más índices.
Variable no subindicada como Arreglo.	Una variable simple es llamada con un índice.
Parametro de Array en llamada a Módulo.	Se llama a un Módulo con un Arreglo como parámetro.

Print inválido en variable de Arreglo.	Un Arreglo es enviado a un operador print como parámetro.
Operador inválido en Arreglos.	Un Arreglo es usado como operando por un operador que no acepta arreglos como operandos.
Operación inválida.	Cualquier tipo de operación inválida.
Dimensiones no coinciden.	Se llama a una operación entre variables dimensionadas pero las dimensiones de ambas no coinciden.
Asignación inválida a variable de Arreglo.	Se le asigna una variable inválida a la variable de un Arreglo.
Tamaño del Arreglo debe ser positivo.	En la declaración del Arreglo, el tamaño es negativo.
Cálculo de determinante inválido,	Dimensiones del arreglo son inválidas para el cálculo de la Determinante.

Ejecución	
Índice fuera de los límites (out of bounds)	El acceso al índice del Arreglo o Matriz está fuera del rango de memoria de la variable.

Descripción del Compilador

Equipo de Cómputo, Lenguaje y Utilerías especiales

Marca: HP

Modelo: 16-c0011dx

Sistema Operativo: Windows 11 Home

Lenguaje utilizado: Python 3.10

Analizador Léxico y Sintáctico: PLY

Descripción del Análisis de Léxico

#Palabras Reservadas

```
reserved = {  
    'if': 'IF',  
    'then': 'THEN',  
    'else': 'ELSE',  
    'while': 'WHILE',  
    'to': 'TO',  
    'for': 'FOR',  
    'program': 'PROGRAM',  
    'main': 'MAIN',  
    'var': 'VAR',  
    'void': 'VOID',  
    'int': 'INT',  
    'float': 'FLOAT',  
    'char': 'CHAR',  
    'return': 'RETURN',  
    'read': 'READ',  
    'print': 'PRINT',  
    'function': 'FUNCTION'  
}
```

#Tokens

```
tokens = [  
    'PLUS',  
    'MINUS',  
    'DIVIDE',  
    'MULTIPLY',  
    'NOTEQUAL',  
    'ISEQUAL',  
    'GT',  
    'LT',  
    'AND',  
    'OR',  
    'LEFTPAR',  
    'RIGHTPAR',  
    'COMA',  
    'EQUAL',  
    'SEMICOLON',  
    'ID',  
    'EXCLAMATION',  
    'QUESTION',  
    'DOLLARSIGN',  
    'LEFTBRACK',  
    'RIGHTBRACK',  
    'LEFTBRACE',  
    'RIGHTBRACE',  
    'CST_INT',  
    'CST_FLOAT',  
    'CST_STRING',  
    'CST_CHAR',  
    'COMMENT_TEXT'  
] + list(reserved.values())
```

```

t_PLUS = r'\+'
t_MINUS = r'\-'
t_DIVIDE = r'\/'
t_MULTIPLY = r'\*'
t_NOTEQUAL = r'<>'
t_ISEQUAL = r'=='
t_GT = r'>'
t_LT = r'<'
t_AND = r'&'
t_OR = r'\|'
t_LEFTPAR = r'\('
t_RIGHTPAR = r'\)'
t_COMA = r','
t_EQUAL = r'='
t_SEMICOLON = r';'
t_LEFTBRACK = r'\['
t_EXCLAMATION = r'!'
t_QUESTION = r'\?'
t_DOLLARSIGN = r'\$'
t_RIGHTBRACK = r'\]'
t_LEFTBRACE = r'\{'
t_RIGHTBRACE = r'\}'
t_CST_INT = r'[0-9]+'
t_CST_FLOAT = r'[0-9]+\.[0-9]'
t_CST_CHAR = r'"(\\"|^[^"])"'
t_CST_STRING = r'"(\\"|^[^"])"'
t_COMMENT_TEXT = r'%%.*\n'

```

```

#Ignorados
t_ignore = " \t\r"

#ID
def t_ID(t):
    r'[a-zA-Z_][a-zA-Z0-9_]*'
    if t.value in reserved:
        t.type = reserved[t.value]
    return t

#Salto de linea
def t_newline(t):
    r'\n+'
    t.lexer.lineno += t.value.count("\n")

#Error
def t_error(t):
    print("Caracter ilegal '%s' en la linea %d"
          % t.value, t.lexer.lineno)
    t.lexer.skip(1)
    exit(0)

```

Descripción del Análisis de Sintaxis

```

'program : PROGRAM ID globalTable SEMICOLON declaration programFunc main'

'main : mainTable MAIN LEFTPAR RIGHTPAR LEFTBRACE declaration statement
RIGHTBRACE'

'''programFunc : function programFunc
                | '''

'assignment : ID dimArray EQUAL Expression2 SEMICOLON'

'''declaration : VAR declarationPDT
                | '''

```

```

'''declarationPDT : PDT vars SEMICOLON declarationPDT
                | '''

'''PDT : INT
        | FLOAT
        | CHAR '''

'return : RETURN LEFTPAR Expression2 RIGHTPAR SEMICOLON'

'if : IF LEFTPAR Expression2 RIGHTPAR createJQif THEN LEFTBRACE statement
RIGHTBRACE ifElse updateJQ'

'''ifElse : ELSE createJQelse LEFTBRACE statement RIGHTBRACE
          | '''

'for : FOR forAssignment TO pushJumpFor Expression2 createQuadFor LEFTBRACE
statement RIGHTBRACE updateQuadFor'

'forAssignment : ID EQUAL CST_INT addTypeInt'

'comment : COMMENT_TEXT'

'while : WHILE pushLoop LEFTPAR Expression2 RIGHTPAR startLoop LEFTBRACE
statement RIGHTBRACE endLoop'

'vars : ID addVarsToTable varsArray varsComa'

'''varsComa : COMA vars
            | '''

'''varsMatrix : LEFTBRACK CST_INT addTypeInt RIGHTBRACK setCols
              | '''

'''varsArray : LEFTBRACK CST_INT addTypeInt RIGHTBRACK setRows varsMatrix
             | '''

'function : functionType ID addFuncToDir LEFTPAR param RIGHTPAR setParamLength
LEFTBRACE declaration statement RIGHTBRACE'

'''param : PDT ID addFuncParams functionParam
          | '''

'''functionParam : COMA param
                  | '''

```

```

'''functionType : FUNCTION PDT
                | FUNCTION VOID setVoidType'''

'''cst_PDT : CST_INT addTypeInt
          | CST_FLOAT addTypeFloat
          | CST_CHAR addTypeChar'''

'''Expression2 : supExpression evaluateExp2 OperatorExpression2
               | supExpression opMatrix evaluateOpMatrix
               | supExpression evaluateExp2'''

'''OperatorExpression2 : AND addOperator
                      | OR addOperator'''

'''supExpression : exp evaluatesupExp OperatorsupExpression exp evaluatesupExp
                 | exp evaluatesupExp'''

'''OperatorsupExpression : GT addOperator
                        | LT addOperator
                        | NOTEQUAL addOperator
                        | ISEQUAL addOperator'''

'''opMatrix : EXCLAMATION addOperator
           | QUESTION addOperator
           | DOLLARSIGN addOperator '''

'''exp : term evaluateTerm expFunction
      | term evaluateTerm '''

'''expFunction : PLUS addOperator exp
              | MINUS addOperator exp '''

'''term : factor evaluateFactor termFunction
       | factor evaluateFactor'''

'''termFunction : MULTIPLY addOperator term
               | DIVIDE addOperator term '''

'''factor : LEFTPAR addPriorityFF Expression2 RIGHTPAR removePriorityFF
         | cst_PDT
         | module
         | ID dimArray'''

'read : READ LEFTPAR id_list RIGHTPAR SEMICOLON'

```

```

'id_list : ID dimArray addRead id_listFunction'
'''id_listFunction : COMA id_list
    | '''

'print : PRINT LEFTPAR printFunction RIGHTPAR SEMICOLON'
'''printFunction : print_param COMA printFunction2
    | print_param '''

'printFunction2 : printFunction'

'''print_param : Expression2 addPrint
    | CST_STRING addPrintString '''

'module : ID checkFunctionExists generateERASize LEFTPAR moduleFunction
nullParam RIGHTPAR generateGosub'

'''dimArray : addOperandId addTypeId LEFTBRACK readIDType Expression2
verifyRows RIGHTBRACK dimMatrix
    | addOperandId addTypeId '''

'''dimMatrix : LEFTBRACK Expression2 verifyCols RIGHTBRACK
    | checkMatAsArray '''

'''statement : return checkVoidType
    | if statement
    | comment statement
    | read statement
    | print statement
    | assignment statement
    | module SEMICOLON statement
    | for statement
    | while statement
    | checkNonVoidType'''

'''moduleFunction : Expression2 generateParam nextParam COMA moduleFunction
    | Expression2 generateParam
    | '''

```

Descripción de Generación de Código Intermedio y Análisis Semántico

Las operaciones se realizan mediante cuádruplos generados usando el formato que vimos en clase:

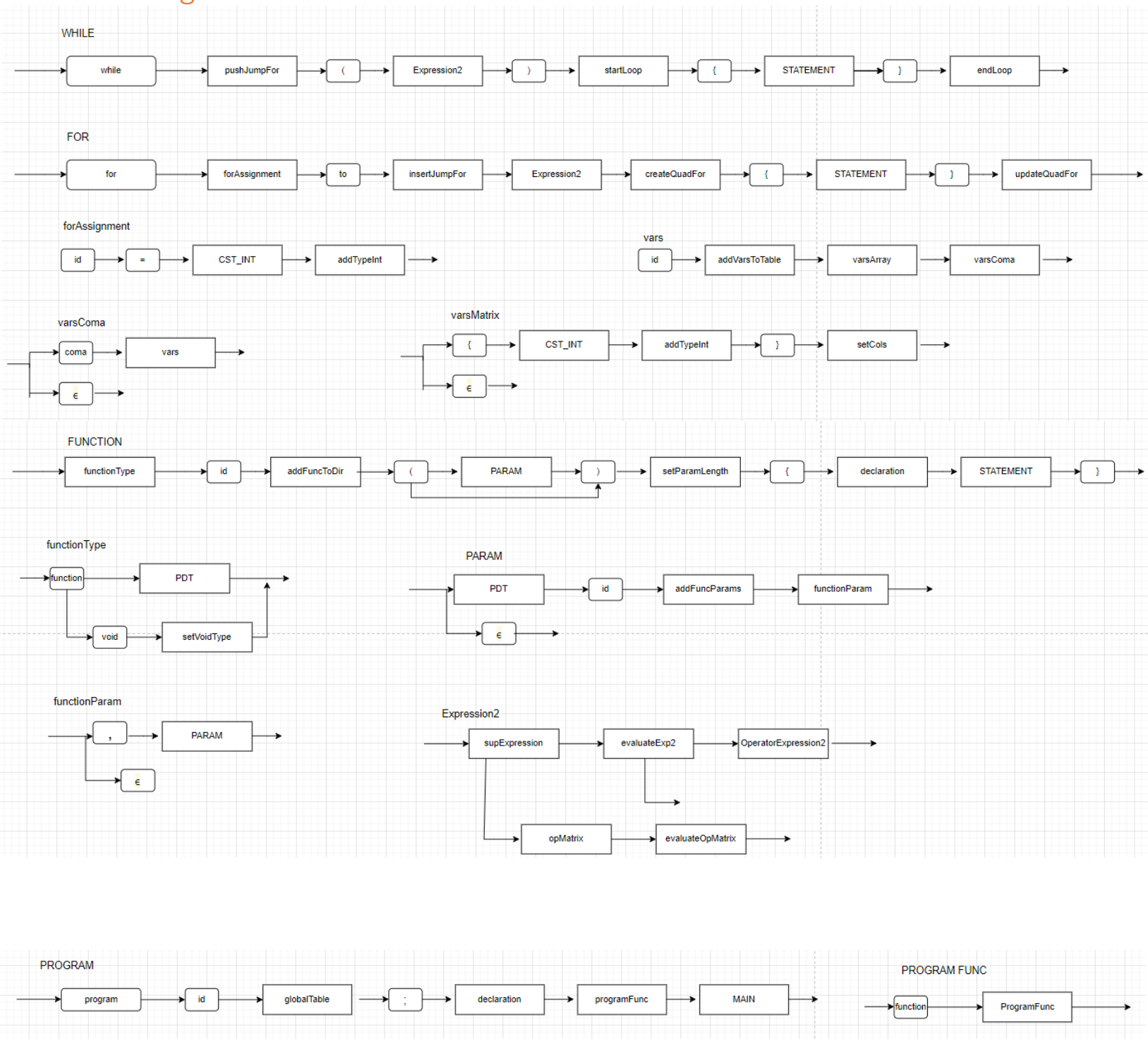
(operador, operando_izquierdo, operando_derecho, resultado)

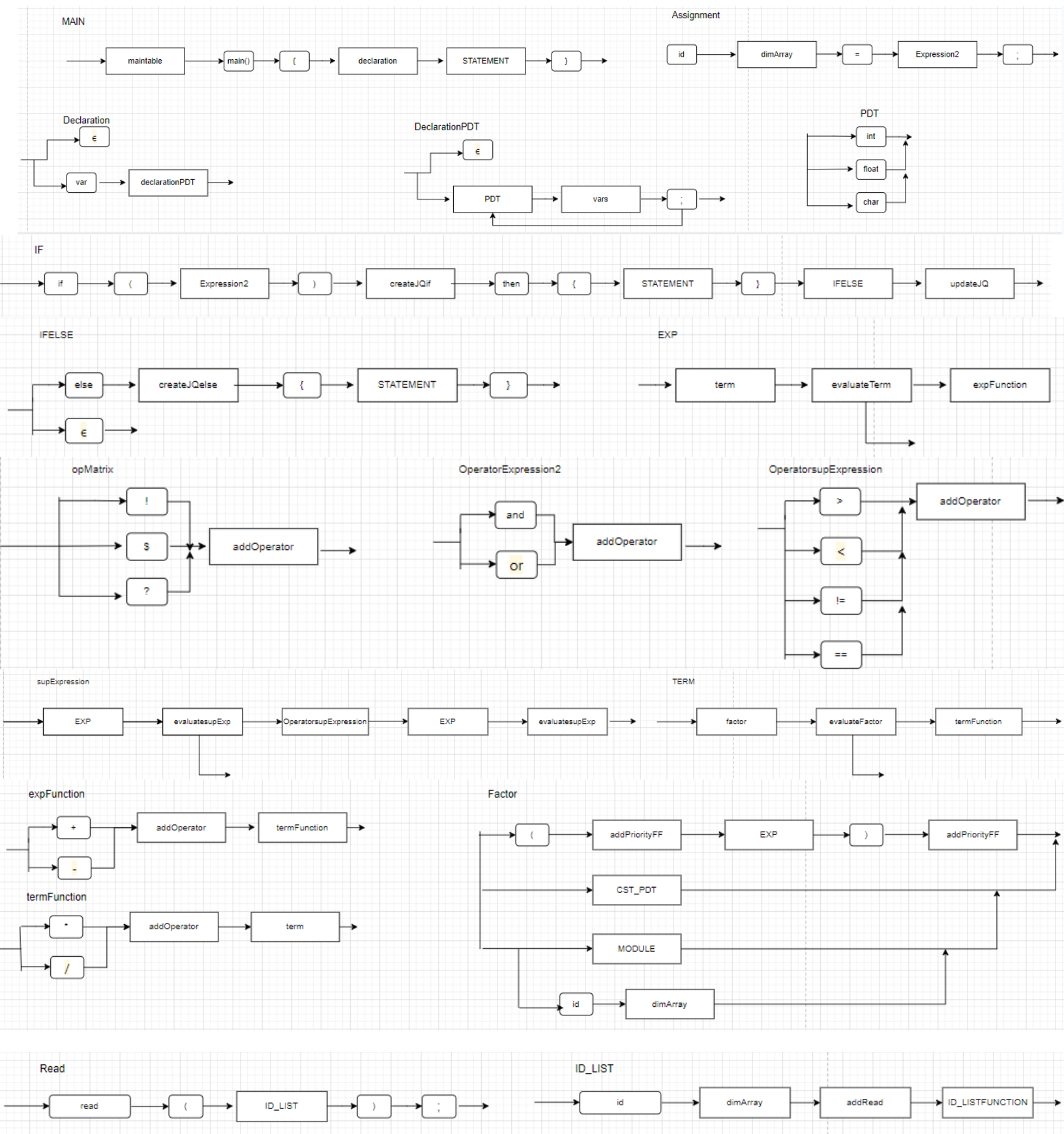
En este formato, el operador puede ser cualquier operador que se encuentre dentro del rango de operadores del lenguaje, sean lógicos o matemáticos, también algunos operadores especiales como el GOTO y GOTOF que se usan en ciclos y condiciones, así como operadores que se usan para el manejo de llamadas de módulos y cambio de contexto como GOSUB y ERA.

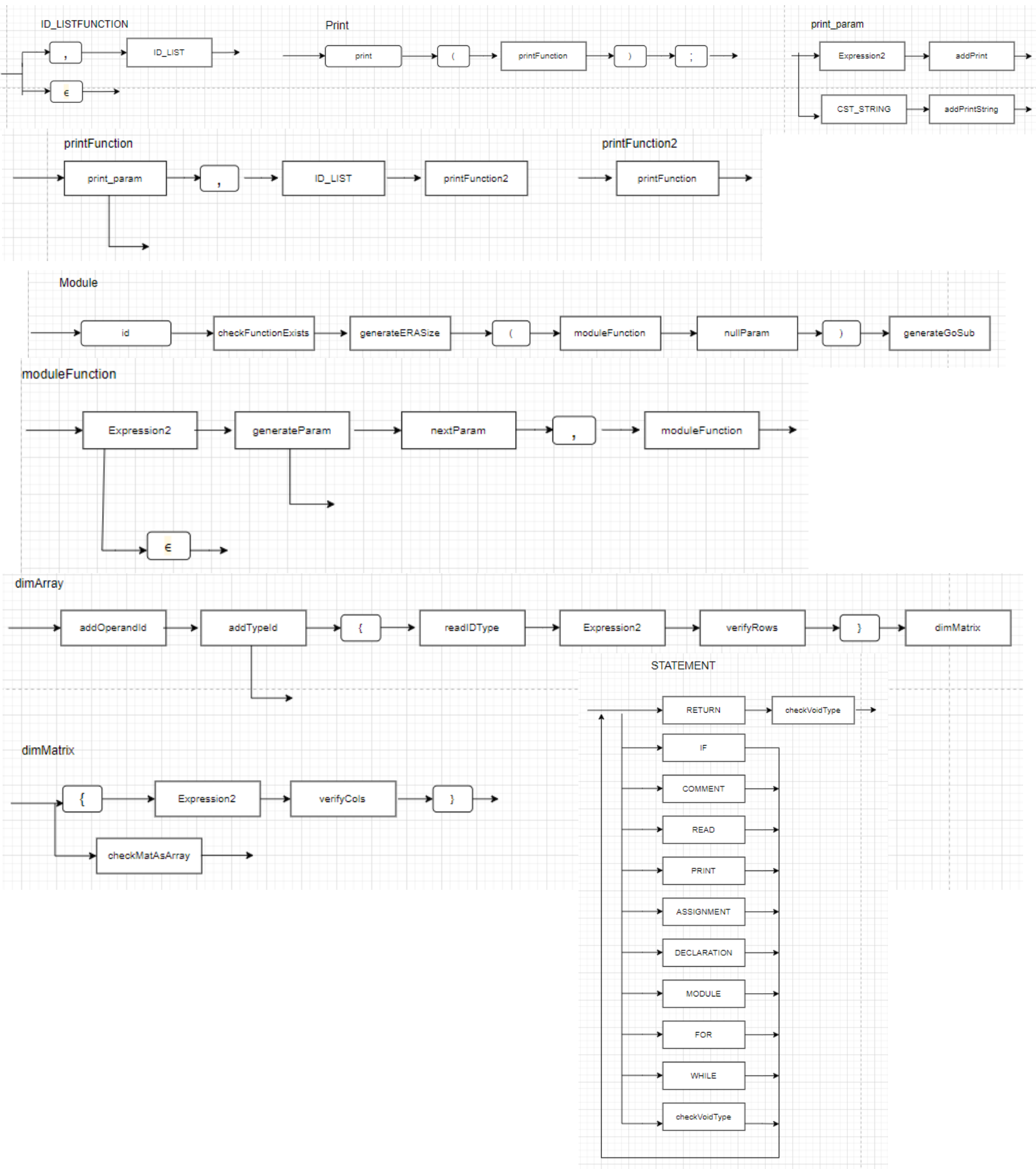
Los operandos y el resultado son direcciones de memoria dependiendo su rango de valores de dirección. Las direcciones para cada tipo de variable o constante se establecieron de la siguiente manera:

- Int global: 0-999
- Float global: 1000-1999
- Char global: 2000-2999
- Int local: 3000-3999
- Float local: 4000-4999
- Char local: 5000-5999
- Int temporal: 6000-6999
- Float temporal: 7000-7999
- Char temporal: 8000-8999
- Int constante: 9000-9999
- Float constante: 10000-10999
- Char constante: 11000-11999
- Apuntador temporal: 12000-12999
- Void: 13000-13999

Diagramas de Sintáxis







Acciones semánticas

Nombre	Acción
globalTable	Inicializar programa y crear tabla de variables.
mainTable	Agregar main a varTable e inicializar propiedades de la función main. Actualizar el cuádruplo main para saltar al inicio del programa.
assignment	Generar cuádruplo en la varTable correspondiente.
declaration	Definir cuádruplo start para una función.
PDT	Cambiar el tipo actual por una declaración.
createJQif	Checar tipo y valor de expresión y generar cuádruplo de salto.
updateJQ	Actualizar cuádruplo de salto con el id del cuádruplo al que se va a saltar.
createJQelse	Crear cuádruplo de salto para estatuto else.
pushLoop	Hacer push al id del cuádruplo a la pila de saltos.
startLoop	Checar el tipo del resultado de la expresión, generar cuádruplo y hacer push al id a la pila de saltos.
endLoop	Generar cuádruplo una vez que el while termine y actualizar GOTO con el id al final del cuádruplo del loop.
pushJumpFor	Push al id del cuádruplo al cual saltar saltar a la pila de saltos.
createQuadFor	Agregar GOTO a cuádruplos.
updateQuadFor	Actualizar el cuádruplo GOTO con el ID del cuádruplo al cual saltar para FOR.
forAssignment	Agregar iterador a la tabla de constantes y crear variable iterativa.
addVarsToTable	Agrega ID actual y su tipo a la tabla de variables.
varsArray	Sólo para declaración de arreglos, guarda la dirección base en las constantes de la tabla de variables.
setRows	Definir la cantidad de filas de una variable dimensionada.
setCols	Definir la cantidad de columnas de una variable dimensionada.

Function	Crea cuádruplo ENDFUNC y define tabla de variable local.
addFuncToDir	Verificar tipo de función e insertar función en el directorio de funciones con tipo, tabla de variables y parámetros.
setVoidType	Definir tipo actual de función como Void.
addFuncParams	Agregar una lista de tipos de parámetros al scope de la función.
setParamLength	Definir la cantidad de parámetros en la función.
addTypeInt	Guardar int en tabla de constantes y hacer push al operando a la pila de operandos.
addTypeFloat	Guardar Float en tabla de constantes y hacer push al operando a la pila de operandos.
addTypeChar	Guardar Char en tabla de constantes y hacer push al operando a la pila de operandos.
evaluateOpMatrix	Evalúa operador y operandos de la operación de una variable dimensionada.
evaluateExp2	Evalúa operador y operandos de expresiones booleanas del tipo AND y OR.
evaluatesupExp	Evalúa operador y operandos de expresiones booleanas del tipo >, <, == y <>.
evaluateTerm	Evalúa operador y operandos del tipo + y – para variables y variables dimensionadas.
evaluateFactor	Evalúa operadores y operandos del tipo * y / para variables y variables dimensionadas (en dimensionadas solo *)
addOperator	Hace push a un operador read a la pila de operadores.
addPriorityFF	Hace push a un paréntesis al stack de operadores como fondo falso.
removePriorityFF	Hace pop a al paréntesis del stack de operadores.
addRead	Genera un cuádruplo READ y le hace push a la lista de cuádruplos.
addPrint	Genera un cuádruplo PRINT y le hace push a la lista de cuádruplos.

addPrintString	Lee un string y lo guarda en la tabla de constantes para después ser impreso por el operador PRINT.
checkVoidType	Lanza un error si hay un RETURN en una función Void.
checkNonVoidType	Lanza un error si no hay RETURN en una función que NO es Void.
checkFunctionExists	Verifica que una función existe en el directorio de funciones y le hace push al operador del módulo a la pila de operadores.
generateERASize	Crea el cuádruplo ERA con la dirección de la función que será llamada.
nullParam	Lanza un error si falta un parámetro en la llamada de una función.
generateGoSub	Genera el cuádruplo GoSub con la dirección de la función a llamar y guarda el resultado en una dirección temporal si NO es void.
generateParam	Genera el cuádruplo PARAM con el operando que está siendo leído.
nextParam	Suma 1 al iterador de param.
dimArray	Hace pop al id y scope de la matriz o arreglo a usar.
addOpperandId	Hace push al id del arreglo a la pila de ID de arreglos y el scope a la pila de scope.
AddTypeID	Hace push a los tipos de la matriz a la pila de tipos.
readIDType	Checa tipos y operandos y lanza error si hay un mismatch.
verifyRows	Genera el cuádruplo Verify del índice que está siendo usado para ver si está dentro del rango correcto de número de filas.
dimMatrix	Genera el cuádruplo para sumar la dirección base y la constante del índice que está siendo usada para acceder al espacio de memoria correcto.
verifyCols	Genera el cuádruplo verify del segundo índice que está siendo usado para ver si está dentro del rango correcto del número de filas.
checkMatAsArray	Lanza un error si sólo se está usando un índice en una Matriz.

Tablas de Consideraciones Semánticas

Suma, Resta y Multiplicación

+, -, *	int	float	char
int	int	float	ERROR
float	float	float	ERROR
char	ERROR	ERROR	ERROR

Menor que, Igual que

<, >	int	float	char
int	int	int	ERROR
float	int	int	ERROR
char	ERROR	ERROR	ERROR

División

/	int	float	char
int	float	float	ERROR
float	float	float	ERROR
char	ERROR	ERROR	ERROR

Diferente a, igual a

<>, ==	int	float	char
int	int	int	ERROR
float	int	int	ERROR
char	ERROR	ERROR	int

And, Or

Los operadores & y | funcionan igual que en Python, donde el valor del operando izquierdo se toma en una operación OR, y el valor del operador derecho se toma en una operación AND (por ejemplo, la operación "x" | 1 da "x", mientras que la operación "x" & 1 da 1, sin embargo, si hay un 0, que es falso, en el lado izquierdo de una operación OR se obtendrá el lado derecho, y si tiene un 0 en el lado derecho de una operación AND, se obtendrá un 0.

Descripción de Administración de Memoria en Compilación

En compilación, se depende en gran parte de la estructura de datos “Dictionary” de Python para guardar toda la información de las variables y funciones. Esta estructura se usa para almacenar grupos de objetos. Consiste en un mapeo de pares clave-valor, donde cada clave está asociada a un valor. Puede contener datos con tipos de datos iguales o diferentes, no está ordenado y es mutable.

Esta estructura de datos tiene un muy eficiente tiempo de búsqueda de $O(1)$.

```
# Ejemplo de dirFunc

"global":
  "type": "void"
  "vars": variableTable["global"] -> "i":
                                          "type": "int"
                                          "value": 1
                                          "address": 0

"main":
  "type": "void",
  "vars": variableTable["main"] -> "c":
                                          "type": "char"
                                          "value": "y"
                                          "address": 2000

"funcion_uno":
  "type": "int",
  "params": Queue[int, int, float]
  "paramsLength": len(params)
  "vars": variableTable["funcion_uno"] -> "x":
                                          "type": "int"
                                          "value": 1
                                          "address": 5000
```


Se usan nombres de funciones o scopes como claves en la hashtable del directorio de funciones.

Por ejemplo, "funcion_uno", functionDir["funcion_uno"] nos indica que es una función de tipo int y que cuenta con 3 parámetros de tipo int, int y float.

Si accedemos a functionDir["funcion_uno"]["vars"], obtendríamos la tabla de variables de esta función, que es una referencia a la tabla de variables de la función y contiene todas las variables con sus respectivas direcciones y tipos, las cuales se asignan durante el proceso de compilación.

En variableTable["constants"], se guardan las constantes identificadas en el proceso de parseo. Las claves son los valores en sí y almacenan sus direcciones. Por ejemplo, variableTable["constants"]["x"] tendría la dirección 11000, que es para chars constantes.

Como ya se había mencionado, la estructura para los cuádruplos es la siguiente:

(operador, operando_izquierdo, operando_derecho, resultado)

Estos se construyen utilizando el constructor de clases "Quadruple" y después se almacenan en una clase "Quadruples" que almacena todos estos objetos "Quadruple".

Descripción de la Máquina Virtual

Equipo de cómputo, lenguaje y utilerías especiales usadas

Marca: HP

Modelo: 16-c0011dx

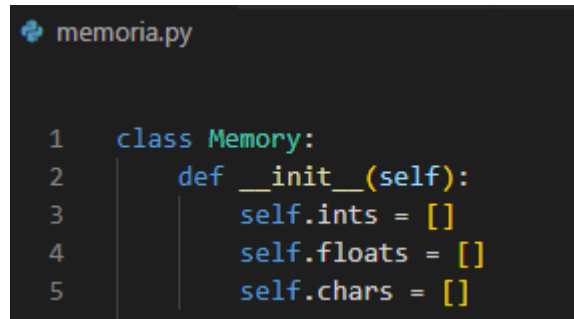
Sistema Operativo: Windows 11 Home

Lenguaje utilizado: Python 3.10

Analizador Léxico y Sintáctico: PLY

Descripción del proceso de Administración de Memoria en ejecución

Durante la ejecución, se depende de una clase “Memory” que tiene una lista de variables de tipo int, float y char.

A screenshot of a code editor with a dark background. The file name 'memoria.py' is visible in the top left corner. The code defines a class 'Memory' with an '__init__' method. Inside the method, three lists are initialized: 'self.ints = []', 'self.floats = []', and 'self.chars = []'. The lines are numbered 1 through 5 on the left margin.

```
1 class Memory:
2     def __init__(self):
3         self.ints = []
4         self.floats = []
5         self.chars = []
```

La máquina virtual inicializa una memoria global, una local y otra temporal usando el constructor de la clase “Memory”. El resultado de esto es un objeto de memoria global, memoria local y memoria temporal, cada uno de estos objetos tendrá una lista de int, float y char. También hacemos uso de la tabla de constantes obtenida durante la compilación, aunque invertimos las claves con las direcciones dentro de ellas para tener en su lugar las direcciones como clave y usar esas direcciones para obtener el valor real de la constante, ya que lo que recibimos con los cuádruplos son las direcciones.

Pruebas de Funcionamiento del Lenguaje

Factorial Cíclico	Cuádruplos	Output
<pre> program factorialCiclico; main() { var int x, result; result = 1; for x = 1 to x < 5 { result = result * x; } print(result); } </pre>	<pre> Q0 GOTO _ _ 1 Q1 = 9000 _ 3001 Q2 = 9000 _ 3000 Q3 < 3000 9001 6000 Q4 GOTO 6000 _ 8 Q5 * 3001 3000 6001 Q6 = 6001 _ 3001 Q7 GOTOFOR _ _ 3 Q8 print _ _ 3001 </pre>	24

Factorial Recursivo	Cuádruplos	Output
<pre> program factorialRecursivo; function int factorial(int x) { if (x > 1) then { return(x * factorial(x - 1)); } return(1); } main() { var int y; y = factorial(5); print(y); } </pre>	<pre> Q0 GOTO _ _ 12 Q1 > 3000 9000 6000 Q2 GOTO 6000 _ 10 Q3 ERA 0 _ _ Q4 - 3000 9000 6001 Q5 PARAM 6001 _ 3000 Q6 GOSUB 0 _ 1 Q7 = 0 _ 6002 Q8 * 3000 6002 6003 Q9 RETURN _ _ 6003 Q10 RETURN _ _ 9000 Q11 ENDFUNC _ _ _ Q12 ERA 0 _ _ Q13 PARAM 9001 _ 3000 Q14 GOSUB 0 _ 1 Q15 = 0 _ 6003 Q16 = 6003 _ 3000 Q17 print _ _ 3000 </pre>	120

Fibonacci Cíclico	Cuádruplos	Output
<pre> program fibonacciCiclico; main() { var int nthTerm, first, second, result, i; first = 0; second = 1; %% nthTerm = termino de la serie fibonacci nthTerm = 9; %% ajustar termino para el ciclo nthTerm = nthTerm + 1; for i = 2 to i < nthTerm { result = first + second; first = second; second = result; } print(result); } </pre>	<pre> Q0 GOTO _ _ 1 Q1 = 9000 _ 3001 Q2 = 9001 _ 3002 Q3 = 9002 _ 3000 Q4 + 3000 9001 6000 Q5 = 6000 _ 3000 Q6 = 9003 _ 3004 Q7 < 3004 3000 6001 Q8 GOTOF 6001 _ 14 Q9 + 3001 3002 6002 Q10 = 6002 _ 3003 Q11 = 3002 _ 3001 Q12 = 3003 _ 3002 Q13 GOTOFOR _ _ 7 Q14 print _ _ 3003 </pre>	34

Fibonacci Recursivo	Cuádruplos	Output
<pre> program fibonacciRecursivo; function int fibonacci(int n) { var int a, b; if (n < 2) then { return(n); } a = fibonacci(n - 1); b = fibonacci(n - 2); return(a + b); } main() { print(fibonacci(12)); } </pre>	<pre> Q1 < 3000 9000 6000 Q2 GOTOF 6000 _ 4 Q3 RETURN _ _ 3000 Q4 ERA 0 _ _ Q5 - 3000 9001 6001 Q6 PARAM 6001 _ 3000 Q7 GOSUB 0 _ 1 Q8 = 0 _ 6002 Q9 = 6002 _ 3001 Q10 ERA 0 _ _ Q11 - 3000 9000 6003 Q12 PARAM 6003 _ 3000 Q13 GOSUB 0 _ 1 Q14 = 0 _ 6004 Q15 = 6004 _ 3002 Q16 + 3001 3002 6005 Q17 RETURN _ _ 6005 Q18 ENDFUNC _ _ _ Q19 ERA 0 _ _ Q20 PARAM 9002 _ 3000 Q21 GOSUB 0 _ 1 Q22 = 0 _ 6006 Q23 print _ _ 6006 </pre>	144

Ordenamiento Burbuja (bubble sort)	Cuádruplos	Output
<pre> program bubbleSort; var int array[5]; main() { var int sorted, i, changed, aux; sorted = 0; i = 0; changed = 0; %% assign array array[0] = 2; array[1] = 9; array[2] = 7; array[3] = 20; array[4] = 15; while (sorted == 0) { if (array[i] > array[i + 1]) then { aux = array[i]; array[i] = array[i + 1]; array[i + 1] = aux; changed = 1; } if (i == 3) then { if (changed == 1) then { i = 0; changed = 0; } else { sorted = 1; } } i = i + 1; } for i = 0 to i < 5 { print(array[i]); } } </pre>	<pre> Q0 GOTO _ _ 1 Q1 = 9002 _ 3000 Q2 = 9002 _ 3001 Q3 = 9002 _ 3002 Q4 VERIFY 9002 0 4 Q5 + 9001 9002 12000 Q6 = 9003 _ 12000 Q7 VERIFY 9004 0 4 Q8 + 9001 9004 12001 Q9 = 9005 _ 12001 Q10 VERIFY 9003 0 4 Q11 + 9001 9003 12002 Q12 = 9006 _ 12002 Q13 VERIFY 9007 0 4 Q14 + 9001 9007 12003 Q15 = 9008 _ 12003 Q16 VERIFY 9009 0 4 Q17 + 9001 9009 12004 Q18 = 9010 _ 12004 Q19 == 3000 9002 6000 Q20 GOTOF 6000 _ 53 Q21 VERIFY 3001 0 4 Q22 + 9001 3001 12005 Q23 + 3001 9004 6001 Q24 VERIFY 6001 0 4 Q25 + 9001 6001 12006 Q26 > 12005 12006 6002 Q27 GOTOF 6002 _ 42 Q28 VERIFY 3001 0 4 Q29 + 9001 3001 12007 Q30 = 12007 _ 3003 Q31 VERIFY 3001 0 4 Q32 + 9001 3001 12008 Q33 + 3001 9004 6003 Q34 VERIFY 6003 0 4 Q35 + 9001 6003 12009 Q36 = 12009 _ 12008 Q37 + 3001 9004 6004 Q38 VERIFY 6004 0 4 Q39 + 9001 6004 12010 Q40 = 3003 _ 12010 Q41 = 9004 _ 3002 Q42 == 3001 9007 6005 Q43 GOTOF 6005 _ 50 Q44 == 3002 9004 6006 Q45 GOTOF 6006 _ 49 Q46 = 9002 _ 3001 Q47 = 9002 _ 3002 Q48 GOTO _ _ 50 Q49 = 9004 _ 3000 Q50 + 3001 9004 6007 Q51 = 6007 _ 3001 Q52 GOTO _ _ 19 Q53 = 9002 _ 3001 Q54 < 3001 9000 6008 Q55 GOTOF 6008 _ 60 Q56 VERIFY 3001 0 4 Q57 + 9001 3001 12011 Q58 print _ _ 12011 Q59 GOTOFOR _ _ 54 </pre>	<pre> 2 7 9 15 20 </pre>

Búsqueda en Arreglo	Cuádruplos	Output
<pre> program busquedaArreglo; var int array[6]; function int find(int i, int j) { if (j < 0) then { return(0 - 1); } if (array[j] == i) then { return(j); } return(find(i, j - 1)); } main() { var int result; %% assign array array[0] = 4; array[1] = 9; array[2] = 10; array[3] = 3; array[4] = 8; array[5] = 6; result = find(3, 5); print(result); } </pre>	<pre> Q0 GOTO __ 18 Q1 < 3001 9002 6000 Q2 GOTO 6000 _ 5 Q3 - 9002 9003 6001 Q4 RETURN __ 6001 Q5 VERIFY 3001 0 5 Q6 + 9001 3001 12000 Q7 == 12000 3000 6002 Q8 GOTO 6002 _ 10 Q9 RETURN __ 3001 Q10 ERA 6 _ __ Q11 PARAM 3000 _ 3000 Q12 - 3001 9003 6003 Q13 PARAM 6003 _ 3001 Q14 GOSUB 6 _ 1 Q15 = 6 _ 6004 Q16 RETURN __ 6004 Q17 ENDFUNC _ _ _ Q18 VERIFY 9002 0 5 Q19 + 9001 9002 12001 Q20 = 9004 _ 12001 Q21 VERIFY 9003 0 5 Q22 + 9001 9003 12002 Q23 = 9005 _ 12002 Q24 VERIFY 9006 0 5 Q25 + 9001 9006 12003 Q26 = 9007 _ 12003 Q27 VERIFY 9008 0 5 Q28 + 9001 9008 12004 Q29 = 9008 _ 12004 Q30 VERIFY 9004 0 5 Q31 + 9001 9004 12005 Q32 = 9009 _ 12005 Q33 VERIFY 9010 0 5 Q34 + 9001 9010 12006 Q35 = 9000 _ 12006 Q36 ERA 6 _ __ Q37 PARAM 9008 _ 3000 Q38 PARAM 9010 _ 3001 Q39 GOSUB 6 _ 1 Q40 = 6 _ 6005 Q41 = 6005 _ 3000 Q42 print __ 3000 </pre>	<p>3</p>

Multiplicación de Matrices	Cuádruplos	Output
<pre> program multiplicacionMatrices; main() { var int matrix1[3][2], matrix2[2][3], result[3][3], i, j; matrix1[0][0] = 1; matrix1[0][1] = 2; matrix1[1][0] = 3; matrix1[1][1] = 4; matrix1[2][0] = 5; matrix1[2][1] = 6; matrix2[0][0] = 1; matrix2[0][1] = 2; matrix2[0][2] = 3; matrix2[1][0] = 3; matrix2[1][1] = 4; matrix2[1][2] = 5; result = matrix1 * matrix2; for j = 0 to j < 3 { for i = 0 to i < 3 { print(result[i][j]); } } } </pre>	<pre> Q0 GOTO _ 1 Q1 VERIFY 9005 3000 3002 Q2 * 9005 9000 6000 Q3 + 6000 9005 6001 Q4 VERIFY 6001 3000 3005 Q5 + 9002 6001 12000 Q6 = 9006 _ 12000 Q7 VERIFY 9005 3000 3002 Q8 * 9006 9000 6002 Q9 + 6002 9005 6003 Q10 VERIFY 6003 3000 3005 Q11 + 9002 6003 12001 Q12 = 9001 _ 12001 Q13 VERIFY 9006 3000 3002 Q14 * 9005 9000 6004 Q15 + 6004 9006 6005 Q16 VERIFY 6005 3000 3005 Q17 + 9002 6005 12002 Q18 = 9000 _ 12002 Q19 VERIFY 9006 3000 3002 Q20 * 9006 9000 6006 Q21 + 6006 9006 6007 Q22 VERIFY 6007 3000 3005 Q23 + 9002 6007 12003 Q24 = 9007 _ 12003 Q25 VERIFY 9001 3000 3002 Q26 * 9005 9000 6008 Q27 + 6008 9001 6009 Q28 VERIFY 6009 3000 3005 Q29 + 9002 6009 12004 Q30 = 9008 _ 12004 Q31 VERIFY 9001 3000 3002 Q32 * 9006 9000 6010 Q33 + 6010 9001 6011 Q34 VERIFY 6011 3000 3005 Q35 + 9002 6011 12005 Q36 = 9009 _ 12005 Q37 VERIFY 9005 3006 3007 Q38 * 9005 9001 6012 Q39 + 6012 9005 6013 Q40 VERIFY 6013 3006 3011 Q41 + 9003 6013 12006 Q42 = 9006 _ 12006 Q43 VERIFY 9005 3006 3007 Q44 * 9006 9001 6014 Q45 + 6014 9005 6015 Q46 VERIFY 6015 3006 3011 Q47 + 9003 6015 12007 Q48 = 9001 _ 12007 Q49 VERIFY 9005 3006 3007 Q50 * 9001 9001 6016 Q51 + 6016 9005 6017 Q52 VERIFY 6017 3006 3011 Q53 + 9003 6017 12008 Q54 = 9000 _ 12008 Q55 VERIFY 9006 3006 3007 Q56 * 9005 9001 6018 Q57 + 6018 9006 6019 Q58 VERIFY 6019 3006 3011 Q59 + 9003 6019 12009 Q60 = 9000 _ 12009 Q61 VERIFY 9006 3006 3007 Q62 * 9006 9001 6020 Q63 + 6020 9006 6021 Q64 VERIFY 6021 3006 3011 Q65 + 9003 6021 12010 Q66 = 9007 _ 12010 Q67 VERIFY 9006 3006 3007 Q68 * 9001 9001 6022 Q69 + 6022 9006 6023 Q70 VERIFY 6023 3006 3011 Q71 + 9003 6023 12011 Q72 = 9008 _ 12011 Q73 ARR* {'address': 3000, 'rows': 3, 'cols': 2} {'address': 3006, 'rows': 2, 'cols': 3} 6024 Q74 ARR= {'address': 6024, 'rows': 3, 'cols': 3, 'type': 'int'} _ {'type': 'int', 'address': 3012, 'rows': 3, 'cols': 3} Q75 = 9005 _ 3022 Q76 < 3022 9000 6033 Q77 GOTOF 6033 _ 89 Q78 = 9005 _ 3021 Q79 < 3021 9000 6034 Q80 GOTOF 6034 _ 88 Q81 VERIFY 3021 3012 3014 Q82 * 3022 9000 6035 Q83 + 6035 3021 6036 Q84 VERIFY 6036 3012 3020 Q85 + 9004 6036 12012 Q86 print _ _ 12012 Q87 GOTOFOR _ _ 79 Q88 GOTOFOR _ _ 76 </pre>	<pre> 7 15 23 10 22 34 13 29 45 </pre>

Transpuesta de Matriz	Cuádruplos	Output
program transpuestaMatriz;	Q0 GOTO __ 1	1
main() {	Q1 VERIFY 9003 3002 3004	2
var int i, j, matrix[3][3],	Q2 * 9003 9000 6000	3
result[3][3];	Q3 + 6000 9003 6001	4
	Q4 VERIFY 6001 3002 3010	5
	Q5 + 9001 6001 12000	6
	Q6 = 9004 _ 12000	7
	Q7 VERIFY 9004 3002 3004	8
	Q8 * 9003 9000 6002	9
	Q9 + 6002 9004 6003	
%% assign matrix	Q10 VERIFY 6003 3002 3010	
matrix[0][0] = 1;	Q11 + 9001 6003 12001	
matrix[1][0] = 4;	Q12 = 9005 _ 12001	
matrix[2][0] = 7;	Q13 VERIFY 9006 3002 3004	
matrix[0][1] = 2;	Q14 * 9003 9000 6004	
matrix[1][1] = 5;	Q15 + 6004 9006 6005	
matrix[2][1] = 8;	Q16 VERIFY 6005 3002 3010	
matrix[0][2] = 3;	Q17 + 9001 6005 12002	
matrix[1][2] = 6;	Q18 = 9007 _ 12002	
matrix[2][2] = 9;	Q19 VERIFY 9003 3002 3004	
	Q20 * 9004 9000 6006	
	Q21 + 6006 9003 6007	
	Q22 VERIFY 6007 3002 3010	
	Q23 + 9001 6007 12003	
	Q24 = 9006 _ 12003	
	Q25 VERIFY 9004 3002 3004	
	Q26 * 9004 9000 6008	
	Q27 + 6008 9004 6009	
	Q28 VERIFY 6009 3002 3010	
	Q29 + 9001 6009 12004	
	Q30 = 9008 _ 12004	
	Q31 VERIFY 9006 3002 3004	
	Q32 * 9004 9000 6010	
	Q33 + 6010 9006 6011	
	Q34 VERIFY 6011 3002 3010	
	Q35 + 9001 6011 12005	
	Q36 = 9009 _ 12005	
	Q37 VERIFY 9003 3002 3004	
	Q38 * 9006 9000 6012	
	Q39 + 6012 9003 6013	
	Q40 VERIFY 6013 3002 3010	
	Q41 + 9001 6013 12006	
	Q42 = 9000 _ 12006	
	Q43 VERIFY 9004 3002 3004	
	Q44 * 9006 9000 6014	
	Q45 + 6014 9004 6015	
	Q46 VERIFY 6015 3002 3010	
	Q47 + 9001 6015 12007	
	Q48 = 9010 _ 12007	
	Q49 VERIFY 9006 3002 3004	
	Q50 * 9006 9000 6016	
	Q51 + 6016 9006 6017	
	Q52 VERIFY 6017 3002 3010	
	Q53 + 9001 6017 12008	
	Q54 = 9011 _ 12008	
	Q55 ARR! {'type': 'int', 'address': 3002, 'rows': 3, 'cols': 3} _ 6018	
	Q56 ARR= {'address': 6018, 'rows': 3, 'cols': 3, 'type': 'int'} _ {'type': 'int', 'address': 3011, 'rows': 3, 'cols': 3}	
	Q57 = 9003 _ 3001	
	Q58 < 3001 9000 6027	
	Q59 GOTO 6027 _ 71	
	Q60 = 9003 _ 3000	
	Q61 < 3000 9000 6028	
	Q62 GOTO 6028 _ 70	
	Q63 VERIFY 3000 3011 3013	
	Q64 * 3001 9000 6029	
	Q65 + 6029 3000 6030	
	Q66 VERIFY 6030 3011 3019	
	Q67 + 9002 6030 12009	
	Q68 print __ 12009	
	Q69 GOTOFOR __ 61	
	Q70 GOTOFOR __ 58	

Inversa de Matriz	Cuádruplos	Output
<pre> program inversaMatriz; main() { var int i, j, matrix[3][3]; float result[3][3]; %% assign matrix matrix[0][0] = 1; matrix[1][0] = 2; matrix[2][0] = 0 - 1; matrix[0][1] = 2; matrix[1][1] = 0 - 3; matrix[2][1] = 1; matrix[0][2] = 0 - 1; matrix[1][2] = 0; matrix[2][2] = 3; result = matrix?; for j = 0 to j < 3 { for i = 0 to i < 3 { print(result[i][j]); } } } </pre>	<pre> Q0 GOTO _ _ 1 Q1 VERIFY 9002 3002 3004 Q2 * 9002 9000 6000 Q3 + 6000 9002 6001 Q4 VERIFY 6001 3002 3010 Q5 + 9001 6001 12000 Q6 = 9003 _ 12000 Q7 VERIFY 9003 3002 3004 Q8 * 9002 9000 6002 Q9 + 6002 9003 6003 Q10 VERIFY 6003 3002 3010 Q11 + 9001 6003 12001 Q12 = 9004 _ 12001 Q13 VERIFY 9004 3002 3004 Q14 * 9002 9000 6004 Q15 + 6004 9004 6005 Q16 VERIFY 6005 3002 3010 Q17 + 9001 6005 12002 Q18 - 9002 9003 6006 Q19 = 6006 _ 12002 Q20 VERIFY 9002 3002 3004 Q21 * 9003 9000 6007 Q22 + 6007 9002 6008 Q23 VERIFY 6008 3002 3010 Q24 + 9001 6008 12003 Q25 = 9004 _ 12003 Q26 VERIFY 9003 3002 3004 Q27 * 9003 9000 6009 Q28 + 6009 9003 6010 Q29 VERIFY 6010 3002 3010 Q30 + 9001 6010 12004 Q31 - 9002 9000 6011 Q32 = 6011 _ 12004 Q33 VERIFY 9004 3002 3004 Q34 * 9003 9000 6012 Q35 + 6012 9004 6013 Q36 VERIFY 6013 3002 3010 Q37 + 9001 6013 12005 Q38 = 9003 _ 12005 Q39 VERIFY 9002 3002 3004 Q40 * 9004 9000 6014 Q41 + 6014 9002 6015 Q42 VERIFY 6015 3002 3010 Q43 + 9001 6015 12006 Q44 - 9002 9003 6016 Q45 = 6016 _ 12006 Q46 VERIFY 9003 3002 3004 Q47 * 9004 9000 6017 Q48 + 6017 9003 6018 Q49 VERIFY 6018 3002 3010 Q50 + 9001 6018 12007 Q51 = 9002 _ 12007 Q52 VERIFY 9004 3002 3004 Q53 * 9004 9000 6019 Q54 + 6019 9004 6020 Q55 VERIFY 6020 3002 3010 Q56 + 9001 6020 12008 Q57 = 9000 _ 12008 Q58 ARR? {'type': 'int', 'address': 3002, 'rows': 3, 'cols': 3} _ 7000 Q59 ARR= {'address': 7000, 'rows': 3, 'cols': 3, 'type': 'float'} _ {'type': 'float', 'address': 4000, 'rows': 3, 'cols': 3} Q60 print _ _ 11000 Q61 = 9002 _ 3001 Q62 < 3001 9000 6021 Q63 GOTOF 6021 _ 75 Q64 = 9002 _ 3000 Q65 < 3000 9000 6022 Q66 GOTOF 6022 _ 74 Q67 VERIFY 3000 4000 4002 Q68 * 3001 9000 6023 Q69 + 6023 3000 6024 Q70 VERIFY 6024 4000 4008 Q71 + 10000 6024 12009 Q72 print _ _ 12009 Q73 GOTOFOR _ _ 65 Q74 GOTOFOR _ _ 62 </pre>	<pre> 0.44999999999999996 0.3 0.04999999999999996 0.35 -0.09999999999999999 0.15 0.15 0.09999999999999999 0.35 </pre>

Determinante de Matriz	Cuádruplos	Output
<pre> program determinanteMatriz; main() { var float result; int i, j, matrix[3][3]; %% assign matrix matrix[0][0] = 1; matrix[1][0] = 0; matrix[2][0] = 1; matrix[0][1] = 8; matrix[1][1] = 0 - 1; matrix[2][1] = 2; matrix[0][2] = 0 - 2; matrix[1][2] = 8; matrix[2][2] = 2; result = matrix\$; print(result); } </pre>	<pre> Q0 GOTO __ 1 Q1 VERIFY 9002 3002 3004 Q2 * 9002 9000 6000 Q3 + 6000 9002 6001 Q4 VERIFY 6001 3002 3010 Q5 + 9001 6001 12000 Q6 = 9003 _ 12000 Q7 VERIFY 9003 3002 3004 Q8 * 9002 9000 6002 Q9 + 6002 9003 6003 Q10 VERIFY 6003 3002 3010 Q11 + 9001 6003 12001 Q12 = 9002 _ 12001 Q13 VERIFY 9004 3002 3004 Q14 * 9002 9000 6004 Q15 + 6004 9004 6005 Q16 VERIFY 6005 3002 3010 Q17 + 9001 6005 12002 Q18 = 9003 _ 12002 Q19 VERIFY 9002 3002 3004 Q20 * 9003 9000 6006 Q21 + 6006 9002 6007 Q22 VERIFY 6007 3002 3010 Q23 + 9001 6007 12003 Q24 = 9005 _ 12003 Q25 VERIFY 9003 3002 3004 Q26 * 9003 9000 6008 Q27 + 6008 9003 6009 Q28 VERIFY 6009 3002 3010 Q29 + 9001 6009 12004 Q30 - 9002 9003 6010 Q31 = 6010 _ 12004 Q32 VERIFY 9004 3002 3004 Q33 * 9003 9000 6011 Q34 + 6011 9004 6012 Q35 VERIFY 6012 3002 3010 Q36 + 9001 6012 12005 Q37 = 9004 _ 12005 Q38 VERIFY 9002 3002 3004 Q39 * 9004 9000 6013 Q40 + 6013 9002 6014 Q41 VERIFY 6014 3002 3010 Q42 + 9001 6014 12006 Q43 - 9002 9004 6015 Q44 = 6015 _ 12006 Q45 VERIFY 9003 3002 3004 Q46 * 9004 9000 6016 Q47 + 6016 9003 6017 Q48 VERIFY 6017 3002 3010 Q49 + 9001 6017 12007 Q50 = 9005 _ 12007 Q51 VERIFY 9004 3002 3004 Q52 * 9004 9000 6018 Q53 + 6018 9004 6019 Q54 VERIFY 6019 3002 3010 Q55 + 9001 6019 12008 Q56 = 9004 _ 12008 Q57 ARR\$ ('type': 'int', 'address': 3002, 'rows': 3, 'cols': 3) _ 7000 Q58 = 7000 _ 4000 Q59 print __ 4000 </pre>	<p>43.999999999999999</p>

Documentación de archivos

Nombre	Detalles
EstructuraDatos.py	<p>Declara e inicializa las estructuras de datos que se usan en el proyecto tales como:</p> <ul style="list-style-type: none">- Directorio de Funciones (DirFunc)- Tabla de Variables (varTable)- Cubo Semántico- Pila de Operadores- Pila de Operandos- Pila de Tipos- Pila de Operandos de Arreglos y Matrices- Diccionario de Asignación de Direcciones a los Tipos- Lista de Operadores- Diccionario de IDs para tipos <p>Crea los objetos Stack() y Queue()</p> <p>Se usa en:</p> <ul style="list-style-type: none">- parser.py: importa objetos inicializados.- cuadрупlos.py: importa la estructura de datos Stack.- maquinavirtual.py: importa la tabla de variables.
errores.py	<p>Declara y exporta una clase Error() que centraliza los displays de errores. A todos los errores en tiempo de compilación se les pasa como argumento el número de línea donde ocurre el error para mostrar dónde ocurrió el error, mientras que los errores en tiempo de ejecución sólo muestran el tipo de error.</p> <p>Se usa en:</p> <ul style="list-style-type: none">- parser.py: importa la clase Error() y las usa en las funciones de sintáxis y gramática.

lexer.py	<p>Hace uso del módulo Lex de PLY. Declara las palabras reservadas del lenguaje en un diccionario.</p> <p>Enlista todos los tokens para simbolizar todos los operadores del lenguaje.</p> <p>Declara las expresiones regulares para cada token. Este lexer luego se pasa al parser, que utiliza este análisis léxico para realizar su análisis de sintaxis.</p> <p>Se usa en:</p> <p>parser.py: Importa el lexer para hacer uso de los tokens y números de línea para pasárselos a la clase de Error.</p>
cuadрупlos.py	<p>Declara las clases Quadruple() y Quadruples().</p> <p>La clase Quadruple es capaz de crear un objeto con operador, operando izquierdo, operando derecho y resultado.</p> <p>La clase Quadruples guarda la lista de cuádruplos, la pila de saltos y es capaz de manipularlas para luego pasársela a la máquina virtual.</p> <p>Usado en:</p> <p>parser.py: importa cuádruplos para crearlos en sus respectivas acciones maquinavirtual.py: importa la lista de cuádruplos para iterarla y ejecutar el código.</p>
maquinavirtual.py	<p>Declara el método executeQuads() que está a cargo de iterar a través de toda la lista de cuádruplos.</p> <p>Con cada cuádruplo que lee, ejecuta la instrucción relacionada con su operador.</p> <p>La máquina virtual también es responsable de la creación de las instancias de la clase Memory(), la pila de</p>

	<p>memoria local, la pila de apuntadores y el mapa de memoria de constantes.</p> <p>Usado en:</p> <p>parser.py: Importa el método <code>executeQuads()</code> y lo corre después de que todo el código ha sido analizado y parseado para ejecutarlo.</p>
numpy	<p>Librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.</p> <p>Incorpora una nueva clase de objetos llamados arrays que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación.</p> <p>Usado en:</p> <p>maquinavirtual.py: Una vez que los arreglos/matrices son procesados se usan las funciones de numpy para hacer las operaciones entre las matrices.</p>

Documentación de Código del Proyecto

```
#startLoop: Checar tipo del resultado de la expresion, generar cuádruplo y
hacer push al id de salto al stack de salto.
def p_startLoop(t):
    'startLoop : '
    result_type = types.pop()
    #Checar tipo y valor de expresion y agregar cuádruplo al stack
    if result_type == "int":
        res = operands.pop()
        operator = "GOTOF"
        # Generar Cuádruplo y hacerle push a la lista
        tmp_quad = Quadruple(operator, res, "_", "_")
        Quadruples.push_quad(tmp_quad)
        # Push al stack de saltos
        Quadruples.push_jump(-1)
```

```
else :  
    Error.condition_type_mismatch(t.lexer.lineno)
```

```
#function: Crea cuádruplo ENDFUNC y define tabla de variables locales.  
def p_function(t):  
    'function : functionType ID addFuncToDir LEFTPAR param RIGHTPAR  
    setParamLength LEFTBRACE declaration statement RIGHTBRACE'  
    #Resetear scope a global cuando se salga del scope de la función, eliminar  
    varTable y referenciar en functionDir  
    global currentScope  
    # Crear cuádruplo endfunc para terminar función  
    temp_quad = Quadruple("ENDFUNC", "_", "_", "_")  
    Quadruples.push_quad(temp_quad)  
    # Variables temporales = longitud del cuádruplo de función al máximo y  
    resetear func_quads  
    functionDir[currentScope]["varLength"] =  
    len(functionDir[currentScope]["vars"])  
    Quadruples.function_quads = 0  
    currentScope = "global"  
    # Resetear direcciones locales  
    addresses["lInt"] -= addresses["lInt"] % 1000  
    addresses["lFloat"] -= addresses["lFloat"] % 1000  
    addresses["lChar"] -= addresses["lChar"] % 1000  
    global returnMade  
    returnMade = False
```

```
#addFuncToDir: Verifica tipo de función e inserta la función al directorio de  
funciones con tipo, varTable y parámetros.  
def p_addFuncToDir(t):  
    'addFuncToDir : '  
    # Si la función existe en global scope, dar error  
    if t[-1] in variableTable["global"]:  
        Error.redefinition_of_variable(t[-1], t.lexer.lineno)  
    else:  
        global currentScope  
        global currentType  
        # Agregar función a variableTable de currentScope  
        variableTable["global"][t[-1]] = {"type": currentType}  
        if currentType == "int":  
            address = addresses["gInt"]  
            addresses["gInt"] += 1
```

```

elif currentType == "float":
    address = addresses["gFloat"]
    addresses["gFloat"] += 1
elif currentType == "char":
    address = addresses["gChar"]
    addresses["gChar"] += 1
else:
    address = addresses["void"]
variableTable["global"][t[-1]]["address"] = address
# Cambiar scope al nuevo id de la funcion
currentScope = t[-1]
# Inicializar variableTable y functionDir por nuevo id de la funcion
variableTable[currentScope] = {}
functionDir[currentScope] = {}
# Definir nuevo tipo de funcion y vars como referencia a
variableTable[currentScope]
functionDir[currentScope]["type"] = currentType
functionDir[currentScope]["vars"] = variableTable[currentScope]
functionDir[currentScope]["params"] = Queue()

```

```

#addFuncParams: Agrega una lista de tipos de parametros al scope de la funcion.
def p_addFuncParams(t):
    'addFuncParams : '
    # Si parametro de la funcion existe en el scope, dar error
    if t[-1] in variableTable[currentScope]:
        Error.redefinition_of_variable(t[-1], t.lexer.lineno)
    else:
        # Agregar parametro de la funcion a variableTable de currentScope
        variableTable[currentScope][t[-1]] = {"type": currentType}
        if currentType == "int":
            variableTable[currentScope][t[-1]]["address"] = addresses["lInt"]
            addresses["lInt"] += 1
        elif currentType == "float":
            variableTable[currentScope][t[-1]]["address"] = addresses["lFloat"]
            addresses["lFloat"] += 1
        else:
            variableTable[currentScope][t[-1]]["address"] = addresses["lChar"]
            addresses["lChar"] += 1
        if "params" not in functionDir[currentScope]:
            functionDir[currentScope]["params"] = Queue()
        # Insertar currentTypes en params Queue
        functionDir[currentScope]["params"].enqueue(currentType)

```

