

**Proyecto Final**

**Diseño de Compiladores**

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

~~\_~~

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

Iván Alejandro Anguiano Leal

A00817460

22 de Noviembre del 2022

Monterrey, Nuevo León

ÍNDICE

[Descripción del proyecto 3](#_Toc105440975)

[Visión 3](#_Toc105440976)

[Objetivo 3](#_Toc105440977)

[Alcance 3](#_Toc105440978)

[Requerimientos 4](#_Toc105440979)

[Requerimientos Funcionales 4](#_Toc105440980)

[Requerimientos No Funcionales 4](#_Toc105440981)

[Descripción de casos de uso 4](#_Toc105440982)

[Proceso del desarrollo del proyecto 5](#_Toc105440983)

[Commits 5](#_Toc105440984)

[Reflexión 16](#_Toc105440985)

[Descripción del Lenguaje 17](#_Toc105440986)

[Nombre del Lenguaje 17](#_Toc105440987)

[Descripción de las principales características del lenguaje 17](#_Toc105440988)

[Errores 17](#_Toc105440989)

[Descripción del Compilador 18](#_Toc105440990)

[Equipo de Cómputo, Lenguaje y Utilerías especiales 18](#_Toc105440991)

[Descripción del Análisis de Léxico 19](#_Toc105440992)

[Descripción del Análisis de Sintáxis 20](#_Toc105440993)

[Descripción de Generación de Código Intermedio y Análisis Semántico 24](#_Toc105440994)

[Diagramas de Sintáxis 25](#_Toc105440995)

[Acciones semánticas 28](#_Toc105440998)

[Tablas de Consideraciones Semánticas 31](#_Toc105440999)

[Descripción de Administración de Memoria en Compilación 32](#_Toc105441000)

[Descripción de la Máquina Virtual 34](#_Toc105441001)

[Equipo de cómputo, lenguaje y utilerías especiales usadas 34](#_Toc105441002)

[Descripción del proceso de Administración de Memoria en ejecución 34](#_Toc105441003)

[Pruebas de Funcionamiento del Lenguaje 35](#_Toc105441004)

[Documentación de archivos 43](#_Toc105441005)

[Documentación de Código del Proyecto 45](#_Toc105441006)

# 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 en español que sea fácil de aprender y utilizar para programadores hispanohablantes principiantes. El lenguaje contiene operaciones aritméticas básicas, operaciones booleanas, algunas operaciones matemáticas así como operaciones con arreglos unidimensionales y 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 (si)
* Estatutos cíclicos (para, mientras)
* Expresiones booleanas
* Multiplicacion de matrices
* Operaciones y funciones matemáticas (raizcuadrada, fórmula general, pow, log, gamma, etc.)

# 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 “programa”.
* Se pueden generar arreglos/matrices y hacer operaciones (suma, resta, division, multiplicación) con las mismas.
* 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. |
| Otras Operaciones y Funciones matemáticas | Realiza operaciones y funciones matemáticas como raíz cuadrada, potencia de un número, fórmula general, función gamma, logaritmo, seno, coseno, etc.. |

# 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 5305a7b77cd2e71da4febbbed126910ae57f23bc  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 90b4a46bbbfe16c1b78a276c05324e413e6de738  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 505cecfd6a01645aa173bc0d0a57eafc5e93d3b3  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 361e9cfcddee31fb63a1253fe64c3e1595f7ebcd  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

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

~~\_~~

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

Iván Alejandro Anguiano Leal

# Descripción del Lenguaje

## Nombre del Lenguaje

N/A

## Descripción de las principales características del lenguaje

Es un lenguaje de programación que contiene operaciones aritméticas , operaciones booleanas simples, funciones matemáticas y también puede hacer uso de arreglos de una dimensión y bidimensionales para hacer operaciones aritméticas básicas con ellos.

Se puede usar para aprender sobre programación con usos básicos de almacenamiento de memoria temporal e input y output de resultados.

## 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

|  |  |
| --- | --- |
|  |  |
|  |  |

## Descripción del Análisis de Sintáxis

|  |
| --- |
| 'program : PROGRAMA ID globalTable PUNTOYCOMA declaration programFunc main'  'globalTable : '  'error : '  'main : mainTable PRINCIPAL LEFTPAR RIGHTPAR LEFTBRACE declaration statement RIGHTBRACE'  'mainTable : '  '''programFunc : function programFunc  | '''  '''assignment : ID dimArray IGUAL hyperExpression PUNTOYCOMA’’’  '''declaration : VAR declarationPrim  | '''  '''declarationPrim : primitive vars PUNTOYCOMA declarationPrim  | '''  '''primitive : INT  | FLOAT  | CHAR '''  'return : REGRESA LEFTPAR hyperExpression RIGHTPAR PUNTOYCOMA'  'if : SI LEFTPAR hyperExpression RIGHTPAR createJQif ENTONCES LEFTBRACE statement RIGHTBRACE ifElse updateJQ'  'createJQif : '  'updateJQ : '  '''ifElse : SINO createJQelse LEFTBRACE statement RIGHTBRACE  | '''  'createJQelse : '  'for : PARA forAssignment HASTA pushJumpFor hyperExpression createQuadFor LEFTBRACE statement RIGHTBRACE updateQuadFor'  'pushJumpFor : '  'createQuadFor : '  'updateQuadFor : '  'forAssignment : ID IGUAL CST\_INT addTypeInt'  'pushLoop : '  'startLoop : '  'endLoop : '  'comment : COMMENT\_TEXT'  'while : MIENTRAS pushLoop LEFTPAR hyperExpression RIGHTPAR startLoop LEFTBRACE statement RIGHTBRACE endLoop'  'vars : ID addVarsToTable varsArray varsComa'  'addVarsToTable : '  '''varsComa : COMA vars  | '''  '''varsMatrix : LEFTBRACK CST\_INT addTypeInt RIGHTBRACK setCols  | '''  '''varsArray : LEFTBRACK CST\_INT addTypeInt RIGHTBRACK setRows varsMatrix  | '''  'setRows : '  'setCols : '  'function : functionType ID addFuncToDir LEFTPAR param RIGHTPAR setParamLength LEFTBRACE declaration statement RIGHTBRACE'  '''param : primitive ID addFuncParams functionParam  | '''  '''functionParam : COMA param  | '''  'addFuncParams : '  'setParamLength : '  '''functionType : FUNCION primitive  | FUNCION VOID setVoidType'''  '''cst\_primitive : CST\_INT addTypeInt  | CST\_FLOAT addTypeFloat  | CST\_CHAR addTypeChar'''  'addTypeInt : '  'addTypeFloat : '  'addTypeChar : '  'addFuncToDir : '  '''hyperExpression : superExpression evaluateHyperExp opHyperExpression  | superExpression opMatrix evaluateOpMatrix  | superExpression evaluateHyperExp'''  'evaluateOpMatrix : '  'evaluateHyperExp : '  '''opHyperExpression : AND addOperator  | OR addOperator'''  '''superExpression : exp evaluateSuperExp opSuperExpression exp evaluateSuperExp  | exp evaluateSuperExp'''  '''opSuperExpression : MAYOR\_QUE addOperator  | MENOR\_QUE addOperator  | MAYOR\_IGUAL addOperator  | MENOR\_IGUAL addOperator  | DIFERENTE\_A addOperator  | IGUAL\_A addOperator'''  'evaluateSuperExp : '  '''opMatrix : addOperator'''  '''exp : term evaluateTerm expFunction  | term evaluateTerm '''  'evaluateTerm : '  '''expFunction : MAS addOperator exp  | MENOS addOperator exp '''  'setVoidType : '  '''term : factor evaluateFactor termFunction  | factor evaluateFactor'''  'evaluateFactor : '  '''termFunction : MULTIPLICA addOperator term  | DIVIDE addOperator term '''  'addOperator : '  '''factor : LEFTPAR addFF hyperExpression RIGHTPAR removeFF  | cst\_primitive  | module  | ID dimArray'''  'addFF : '  'removeFF : '  'read : LEE LEFTPAR id\_list RIGHTPAR PUNTOYCOMA'  '''id\_list : ID dimArray addRead id\_listFunction'''  '''id\_listFunction : COMA id\_list  | '''  'addRead : '  '''print : IMPRIME LEFTPAR printFunction RIGHTPAR PUNTOYCOMA  '''printFunction : print\_param COMA printFunction2  | print\_param '''  'printFunction2 : printFunction'  '''print\_param : hyperExpression addPrint  | CST\_STRING addPrintString '''  'addPrintString : '  'addPrint : '  'module : ID checkFunctionExists generateERASize LEFTPAR moduleFunction nullParam RIGHTPAR generateGosub'  'checkFunctionExists : '  'generateERASize : '  'nullParam : '  'generateGosub : '  'generateParam : '  'nextParam : '  '''dimArray : addOperandId addTypeId LEFTBRACK readIDType hyperExpression verifyRows RIGHTBRACK dimMatrix  | addOperandId addTypeId '''  'addOperandId : '  'addTypeId : '  'readIDType : '  'verifyRows : '  '''dimMatrix : LEFTBRACK hyperExpression verifyCols RIGHTBRACK  | checkMatAsArray '''  'verifyCols : '  'checkMatAsArray : '  '''statement : return checkVoidType  | if statement  | comment statement  | read statement  | print statement  | assignment statement  | module PUNTOYCOMA statement  | for statement  | checkNonVoidType'''  'checkVoidType : '  'checkNonVoidType : ' |

## Descripción de Generación de Código Intermedio y Análisis Semántico

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

Cuadruplo = Qn (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, y el VERIFICA para variables dimensionadas.

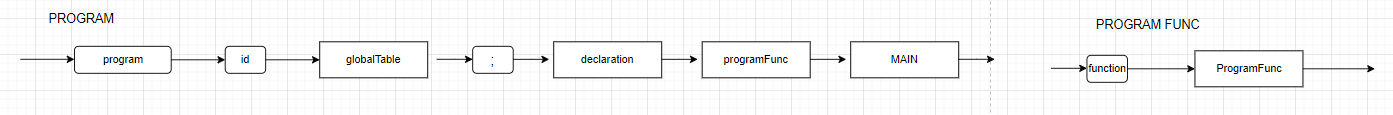
Los operandos y el resultado son “direcciones” de memoria dependiendo su rango de valores de dirección. Los valores de las direcciones para cada tipo de variable o constante se establecieron de la siguiente manera:

* 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
* Temporal pointer: 12000-12999
* Void: 13000-13999

## Diagramas de Sintáxis

Imagen que contiene Diagrama

Descripción generada automáticamente

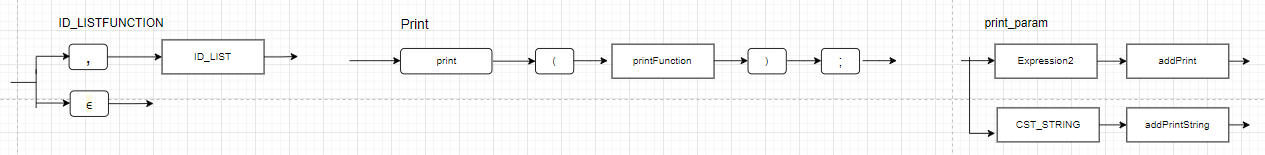


Diagrama

Descripción generada automáticamente

## Gráfico de cajas y bigotes Descripción generada automáticamenteImagen que contiene biombo, interior, ventana, tabla Descripción generada automáticamenteGráfico, Gráfico de cajas y bigotes Descripción generada automáticamente

Diagrama

Descripción generada automáticamente

## Diagrama Descripción generada automáticamente

Gráfico

Descripción generada automáticamenteDiagrama

Descripción generada automáticamente

## 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. |
| primitive | 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 GOTOF 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 GOTOF a cuádruplos. |
| updateQuadFor | Actualizar el cuádruplo GOTOF 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. |
| evaluateHyperExp | Evalúa operador y operandos de expresiones booleanas del tipo AND y OR. |
| evaluateSuperexp | 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. |
| addFF | Hace push a un paréntesis al stack de operadores como fondo falso. |
| removeFF | 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 accesar 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, Mayor o Igual, Menor o Igual

|  |  |  |  |
| --- | --- | --- | --- |
| **< , >, <=, >=** | **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.

# 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

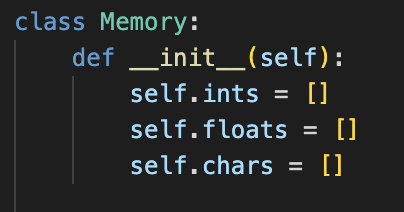
**Sistema Operativo:** Windows 11 Home

**Lenguaje utilizado:** Python 3.10

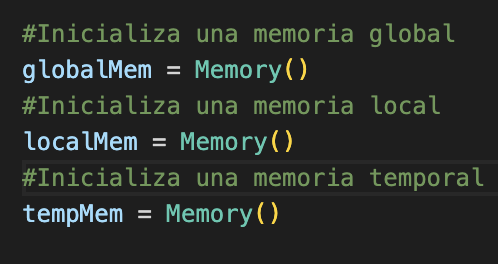
**Analizador Léxico y Sintáctico:** PLY

## Descripción del proceso de Administración de Memoria en ejecución

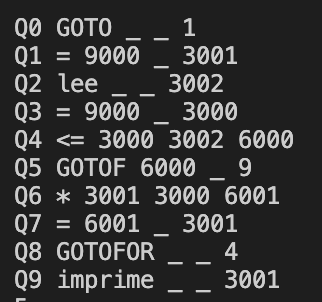
Durante la ejecución, se depende de una clase “Memory” que tiene una lista de variables de tipo int, float y char.



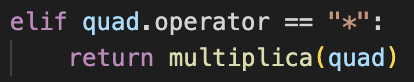
La máquina virtual inicializa una memoria global, una local y otra temporal usando el constructor de la clase “Memory”.

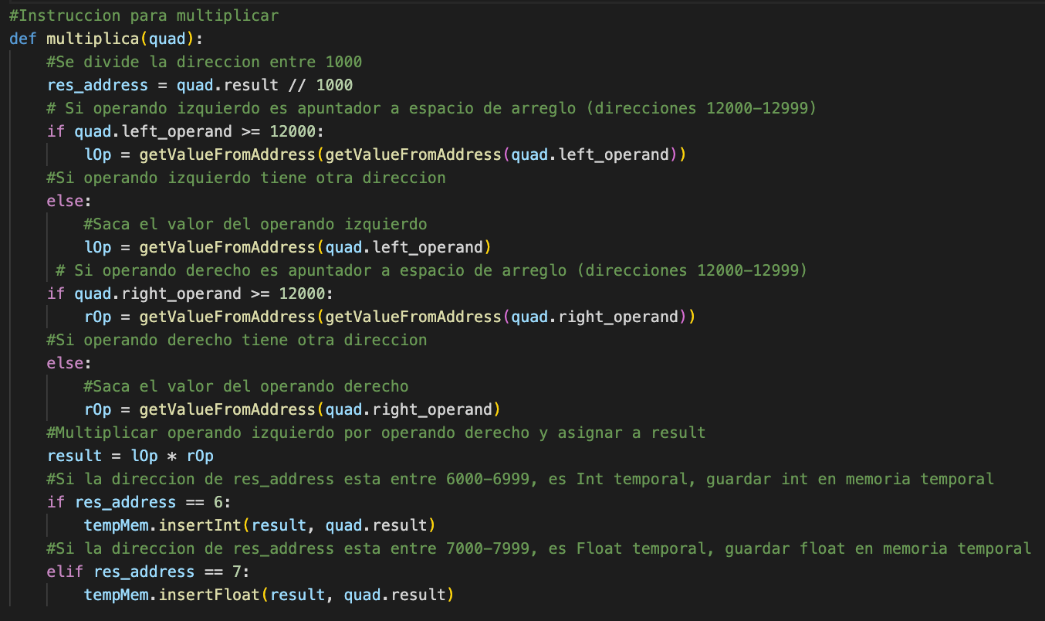


El resultado de esto es un objeto de memoria global, memoria local y memoria temporal, cada uno de estos objetos tendrá una lista de int, float y char. También hacemos uso de la tabla de constantes obtenida durante la compilación, aunque invertimos las claves con las direcciones dentro de ellas para 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, no los valores como tal:



La máquina virtual va leyendo los cuádruplos y dependiendo del operador es la instrucción que ejecuta, por ejemplo, en la imágen de arriba en el cuádruplo 6 se tiene un \* como operador por lo que al llegar a ese cuádruplo pasará a ejecutar la instrucción de multiplica.





Lo que hace este fragmento de código es sacar los valores de los operandos izquierdo y derecho utilizando las direcciones de “memoria” que el cuádruplo contiene, los multiplica y “guarda” el resultado en la dirección de “memoria” temporal que viene en el cuádruplo como resultado.

# Pruebas de Funcionamiento del Lenguaje

|  |  |  |
| --- | --- | --- |
| **Factorial Cíclico** | **Cuádruplos** | **Output** |
| programa factorialCic;  principal() {  var int c, resultado, num;  resultado = 1;  lee(num);  para c = 1 hasta c <= num {  resultado = resultado \* c;  }  imprime(resultado);  }  Input: 7 | Q0 GOTO \_ \_ 1  Q1 = 9000 \_ 3001  Q2 lee \_ \_ 3002  Q3 = 9000 \_ 3000  Q4 <= 3000 3002 6000  Q5 GOTOF 6000 \_ 9  Q6 \* 3001 3000 6001  Q7 = 6001 \_ 3001  Q8 GOTOFOR \_ \_ 4  Q9 imprime \_ \_ 3001 | **5040** |

|  |  |  |
| --- | --- | --- |
| **Factorial Recursivo** | **Cuádruplos** | **Output** |
| programa factorialRec;  funcion int factorial(int x) {  si (x > 1) entonces {  regresa(x \* factorial(x - 1));  }  regresa(1);  }  principal() {  var int x, y;  lee(y);  x = factorial(y);  imprime(x);  }  Input: 6 | Q1 > 3000 9000 6000  Q2 GOTOF 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 REGRESA \_ \_ 6003  Q10 REGRESA \_ \_ 9000  Q11 ENDFUNC \_ \_ \_  Q12 lee \_ \_ 3001  Q13 ERA 0 \_ \_  Q14 PARAM 3001 \_ 3000  Q15 GOSUB 0 \_ 1  Q16 = 0 \_ 6003  Q17 = 6003 \_ 3000  Q18 imprime \_ \_ 3000 | **720** |

|  |  |  |
| --- | --- | --- |
| **Fibonacci Cíclico** | **Cuádruplos** | **Output** |
| programa fibonacciCic;    principal() {  var int nTermino, primero, segundo, resultado, x;  primero= 0;  segundo= 1;  imprime("Inserta numero: ");  lee(nTermino);  para x = 2 hasta x <= nTermino {  resultado = primero + segundo;  primero = segundo;  segundo = resultado;  }  imprime(resultado);  }  Input: 8 | Q0 GOTO \_ \_ 1  Q1 = 9000 \_ 3001  Q2 = 9001 \_ 3002  Q3 imprime \_ \_ 11000  Q4 lee \_ \_ 3000  Q5 = 9002 \_ 3004  Q6 <= 3004 3000 6000  Q7 GOTOF 6000 \_ 13  Q8 + 3001 3002 6001  Q9 = 6001 \_ 3003  Q10 = 3002 \_ 3001  Q11 = 3003 \_ 3002  Q12 GOTOFOR \_ \_ 6  Q13 imprime \_ \_ 3003 | **21** |

|  |  |  |
| --- | --- | --- |
| **Fibonacci Recursivo** | **Cuádruplos** | **Output** |
| programa fibonacciRecursivo;  funcion int fibonacci(int n) {  var int a,b;  si(n <= 1) entonces {  regresa(n);  }  a = fibonacci(n-1);  b = fibonacci(n-2);  regresa(a+b);  }  principal() {  var int x;  imprime("Valor de x: ");  lee(x);  imprime(fibonacci(fibonacci(x)));  }  Input: 4 | Q0 GOTO \_ \_ 19  Q1 <= 3000 9000 6000  Q2 GOTOF 6000 \_ 4  Q3 REGRESA \_ \_ 3000  Q4 ERA 0 \_ \_  Q5 - 3000 9000 6001  Q6 PARAM 6001 \_ 3000  Q7 GOSUB 0 \_ 1  Q8 = 0 \_ 6002  Q9 = 6002 \_ 3001  Q10 ERA 0 \_ \_  Q11 - 3000 9001 6003  Q12 PARAM 6003 \_ 3000  Q13 GOSUB 0 \_ 1  Q14 = 0 \_ 6004  Q15 = 6004 \_ 3002  Q16 + 3001 3002 6005  Q17 REGRESA \_ \_ 6005  Q18 ENDFUNC \_ \_ \_  Q19 imprime \_ \_ 11000  Q20 lee \_ \_ 3000  Q21 ERA 0 \_ \_  Q22 ERA 0 \_ \_  Q23 PARAM 3000 \_ 3000  Q24 GOSUB 0 \_ 1  Q25 = 0 \_ 6006  Q26 PARAM 6006 \_ 3000  Q27 GOSUB 0 \_ 1  Q28 = 0 \_ 6007  Q29 imprime \_ \_ 6007 | **2**  **(el ejemplo saca fibonacci del fibonacci)** |

|  |  |  |
| --- | --- | --- |
| **Ordenamiento Burbuja (bubble sort)** | **Cuádruplos** | **Output** |
| programa bubbleSort;  var int arreglo[5];  principal() {  var int i, x, aux ;  x = 0;  i = 0;  aux = 0;  //asignar arreglo  arreglo[0] = 5;  arreglo[1] = 9;  arreglo[2] = 14;  arreglo[3] = 8;  arreglo[4] = 0;  para i = 0 hasta i <= 4 {  para x = 0 hasta x <= 3 {  si (arreglo[i] < arreglo[x]) entonces {  aux = arreglo[i];  arreglo[i] = arreglo[x];  arreglo[x] = aux;  }  }  }  para i = 0 hasta i <= 4 {  imprime(arreglo[i]);  }  } | Q0 GOTO \_ \_ 1  Q1 = 9002 \_ 3001  Q2 = 9002 \_ 3000  Q3 = 9002 \_ 3002  Q4 VERIFICA 9002 0 4  Q5 + 9001 9002 12000  Q6 = 9000 \_ 12000  Q7 VERIFICA 9003 0 4  Q8 + 9001 9003 12001  Q9 = 9004 \_ 12001  Q10 VERIFICA 9005 0 4  Q11 + 9001 9005 12002  Q12 = 9006 \_ 12002  Q13 VERIFICA 9007 0 4  Q14 + 9001 9007 12003  Q15 = 9008 \_ 12003  Q16 VERIFICA 9009 0 4  Q17 + 9001 9009 12004  Q18 = 9002 \_ 12004  Q19 = 9002 \_ 3000  Q20 <= 3000 9009 6000  Q21 GOTOF 6000 \_ 44  Q22 = 9002 \_ 3001  Q23 <= 3001 9007 6001  Q24 GOTOF 6001 \_ 43  Q25 VERIFICA 3000 0 4  Q26 + 9001 3000 12005  Q27 VERIFICA 3001 0 4  Q28 + 9001 3001 12006  Q29 < 12005 12006 6002  Q30 GOTOF 6002 \_ 42  Q31 VERIFICA 3000 0 4  Q32 + 9001 3000 12007  Q33 = 12007 \_ 3002  Q34 VERIFICA 3000 0 4  Q35 + 9001 3000 12008  Q36 VERIFICA 3001 0 4  Q37 + 9001 3001 12009  Q38 = 12009 \_ 12008  Q39 VERIFICA 3001 0 4  Q40 + 9001 3001 12010  Q41 = 3002 \_ 12010  Q42 GOTOFOR \_ \_ 23  Q43 GOTOFOR \_ \_ 20  Q44 = 9002 \_ 3000  Q45 <= 3000 9009 6003  Q46 GOTOF 6003 \_ 51  Q47 VERIFICA 3000 0 4  Q48 + 9001 3000 12011  Q49 imprime \_ \_ 12011  Q50 GOTOFOR \_ \_ 45 | **0**  **5**  **8**  **9**  **14** |

|  |  |  |
| --- | --- | --- |
| **Búsqueda en Arreglo** | **Cuádruplos** | **Output** |
| programa busquedaArreglo;  var int arreglo[6];  funcion int find(int i, int j) {  si (j < 0) entonces {  regresa(0-1);  }  si (arreglo[j] == i) entonces {  regresa(j);  }  regresa(find(i, j - 1));  }  principal() {  var int resultado,x;    arreglo[0] = 4;  arreglo[1] = 9;  arreglo[2] = 10;  arreglo[3] = 3;  arreglo[4] = 8;  arreglo[5] = 6;  imprime("Teclea el valor a buscar: ");  lee(x);  resultado = find(x, 5);  si(resultado < 0) entonces{  imprime("Valor no encontrado");  }  sino{  imprime(resultado);  }  }  Input: 8 | Q0 GOTO \_ \_ 18  Q1 < 3001 9002 6000  Q2 GOTOF 6000 \_ 5  Q3 - 9002 9003 6001  Q4 REGRESA \_ \_ 6001  Q5 VERIFICA 3001 0 5  Q6 + 9001 3001 12000  Q7 == 12000 3000 6002  Q8 GOTOF 6002 \_ 10  Q9 REGRESA \_ \_ 3001  Q10 ERA 6 \_ \_  Q11 PARAM 3000 \_ 3000  Q12 - 3001 9003 6003  Q13 PARAM 6003 \_ 3001  Q14 GOSUB 6 \_ 1  Q15 = 6 \_ 6004  Q16 REGRESA \_ \_ 6004  Q17 ENDFUNC \_ \_ \_  Q18 VERIFICA 9002 0 5  Q19 + 9001 9002 12001  Q20 = 9004 \_ 12001  Q21 VERIFICA 9003 0 5  Q22 + 9001 9003 12002  Q23 = 9005 \_ 12002  Q24 VERIFICA 9006 0 5  Q25 + 9001 9006 12003  Q26 = 9007 \_ 12003  Q27 VERIFICA 9008 0 5  Q28 + 9001 9008 12004  Q29 = 9008 \_ 12004  Q30 VERIFICA 9004 0 5  Q31 + 9001 9004 12005  Q32 = 9009 \_ 12005  Q33 VERIFICA 9010 0 5  Q34 + 9001 9010 12006  Q35 = 9000 \_ 12006  Q36 imprime \_ \_ 11000  Q37 lee \_ \_ 3001  Q38 ERA 6 \_ \_  Q39 PARAM 3001 \_ 3000  Q40 PARAM 9010 \_ 3001  Q41 GOSUB 6 \_ 1  Q42 = 6 \_ 6005  Q43 = 6005 \_ 3000  Q44 < 3000 9002 6006  Q45 GOTOF 6006 \_ 48  Q46 imprime \_ \_ 11001  Q47 GOTO \_ \_ 49  Q48 imprime \_ \_ 3000 | **4** |

|  |  |  |
| --- | --- | --- |
| **Multiplicación de Matrices** | **Cuádruplos** | **Output** |
| programa multiplicacionMatrices;  principal() {  var int matriz1[3][2], matriz2[2][3], resultado[3][3], i, j;  matriz1[0][0] = 1;  matriz1[0][1] = 2;  matriz1[1][0] = 3;  matriz1[1][1] = 4;  matriz1[2][0] = 5;  matriz1[2][1] = 6;  matriz2[0][0] = 1;  matriz2[0][1] = 2;  matriz2[0][2] = 3;  matriz2[1][0] = 3;  matriz2[1][1] = 4;  matriz2[1][2] = 5;  resultado = matriz1 \* matriz2;  para j = 0 hasta j < 3 {  para i = 0 hasta i < 3 {  imprime(resultado[i][j]);  }  }  } |  | **7**  **15**  **23**  **10**  **22**  **34**  **13**  **29**  **45** |

|  |  |  |
| --- | --- | --- |
| **Funciones Matemáticas** | **Cuádruplos** | **Output** |
| programa matematicas;  principal() {  imprime(raizcuadrada(pow(raizcuadrada(16),3)));  imprime(exponencial(residuo(5,2)));  imprime(redondear.arriba(raizcuadrada(5)));  imprime(redondear.abajo(grados(4.50)));  imprime(gamma(pow(2,5)));  imprime(residuo(224,125));  imprime(radianes(100.03));  imprime(grados(8.90));  imprime(seno(coseno(3.141528)));  imprime(coseno(tangente(90)));  imprime(tangente(90));  imprime(logaritmo(raizcuadrada(pow(5,2))));  imprime(logaritmo.gamma(7));  cuadratica(2,9,10);  } | Q0 GOTO \_ \_ 1  Q1 raizcuadrada 9000 \_ 6000  Q2 pow 6000 9001 6001  Q3 raizcuadrada 6001 \_ 6002  Q4 imprime \_ \_ 6002  Q5 residuo 9002 9003 6003  Q6 exponencial 6003 \_ 6004  Q7 imprime \_ \_ 6004  Q8 raizcuadrada 9002 \_ 6005  Q9 redondearArriba 6005 \_ 6006  Q10 imprime \_ \_ 6006  Q11 grados 10000 \_ 7000  Q12 redondearAbajo 7000 \_ 7001  Q13 imprime \_ \_ 7001  Q14 pow 9003 9002 6007  Q15 gamma 6007 \_ 6008  Q16 imprime \_ \_ 6008  Q17 residuo 9004 9005 6009  Q18 imprime \_ \_ 6009  Q19 radianes 10001 \_ 7002  Q20 imprime \_ \_ 7002  Q21 grados 10002 \_ 7003  Q22 imprime \_ \_ 7003  Q23 coseno 10003 \_ 7004  Q24 seno 7004 \_ 7005  Q25 imprime \_ \_ 7005  Q26 tangente 9006 \_ 6010  Q27 coseno 6010 \_ 6011  Q28 imprime \_ \_ 6011  Q29 tangente 9006 \_ 6012  Q30 imprime \_ \_ 6012  Q31 pow 9002 9003 6013  Q32 raizcuadrada 6013 \_ 6014  Q33 logaritmo 6014 \_ 6015  Q34 imprime \_ \_ 6015  Q35 logaritmoGamma 9007 \_ 6016  Q36 imprime \_ \_ 6016  Q37 cuadratica 9003 9008 9009 | **8.0**  **2.718281828459045**  **3**  **257**  **8.222838654177925e+33**  **-26.0**  **1.7458528507699278**  **509.9324376664327**  **-0.8414709836786413**  **-0.41177780729510316**  **-1.995200412208242**  **1.6094379124341003**  **6.579251212010102**  **-2.0**  **-2.5** |

# Documentación de archivos

|  |  |
| --- | --- |
| **Nombre** | **Detalles** |
| **EstructuraDatos.py** | Declara e inicializa las estructuras de datos que se usan en el proyecto tales como:   * 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   Crea los objetos Stack() y Queue()  Se usa en:   * **parser.py**: importa objetos inicializados. * **cuadruplos.py**: importa la estructura de datos Stack. * **maquinavirtual.py**: importa la tabla de variables. |
| **errores.py** | 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.  Se usa en:   * **parser.py:** importa la clase Error() y las usa en las funciones de sintáxis y gramática. |
| **lexer.py** | Hace uso del módulo Lex de PLY. Declara las palabras reservadas del lenguaje en un diccionario.  Enlista todos los tokens para simbolizar todos los operadores del lenguaje.  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.  Se usa en:  **parser.py**: Importa el lexer para hacer uso de los tokens y números de línea para pasárselos a la clase de Error. |
| **cuadruplos.py** | Declara las clases Quadruple() y Quadruples().  La clase Quadruple es capaz de crear un objeto con operador, operando izquierdo, operando derecho y resultado.  La clase Quadruples guarda la lista de cuadruplos, la pila de saltos y es capaz de manipularlas para luego pasársela a la máquina virtual.  Usado en:  **parser.py**: importa cuadruplos para crearlos en sus respectivas acciones  **maquinavirtual.py**: importa la lista de cuádruplos para iterarla y ejecutar el código. |
| **maquinavirtual.py** | Declara el método executeQuads() que está a cargo de iterar a través de toda la lista de cuádruplos.  Con cada cuádruplo que lee, ejecuta la instrucción relacionada con su operador.  La máquina virtual también es responsable de la creación de las instancias de la clase Memory(), la pila de memoria local, la pila de apuntadores y el mapa de memoria de constantes.  Usado en:  **parser.py**: Importa el método executeQuads() y lo corre después de que todo el código ha sido analizado y parseado para ejecutarlo. |
| **numpy** | Librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para un gran volumen de datos.  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.  Usado en:  **maquinavirtual.py**: Una vez que los arreglos/matrices son procesados se usan las funciones de numpy para hacer las operaciones entre las matrices. |

# Documentación de Código del Proyecto

|  |
| --- |
| #startLoop: Checar tipo del resultado de la expresion, generar cuadruplo 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 cuadruplo al stack      if result\_type == "int":          res = operands.pop()          operator = "GOTOF"          # Generar Cuadruplo 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 cuadruplo 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 funcion, eliminar varTable y referenciar en functionDir      global currentScope      # Crear cuadruplo endfuc para terminar funcion      temp\_quad = Quadruple("ENDFUNC", "\_", "\_", "\_")      Quadruples.push\_quad(temp\_quad)      # Variables temporales = longitud del cuadruplo de funcion al maximo 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 funcion e inserta la funcion al directorio de funciones con tipo, varTable y parametros.  def p\_addFuncToDir(t):      'addFuncToDir : '      # Si la funcion 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 funcion 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) |