Tabla de Contenidos

Descripción del Proyecto 2

Descripción del Proyecto 2

Visión, Objetivos y Alcance del Proyecto 2

Análisis de Requerimientos y Casos de Uso Generales 2

Descripción de los Principales Casos de Pruebas 3

Descripción del Proceso General del Desarrollo del Proyecto 3

Descripción del Lenguaje 4

Nombre del Lenguaje 4

Descripción Genérica del Lenguaje 4

Descripción de Errores (compilación y ejecución) 4

Descripción del Compilador 5

Equipo de Cómputo, Lenguaje y Utilerías 5

Descripción del Análisis de Léxico 5

Descripción del Análisis de Sintaxis 6

Descripción de Generación de Código Intermedio y Análisis Semántico 8

Descripción de la Administración de Memoria en Compilación 14

Descripción de la Máquina Virtual 15

Equipo de Cómputo, Lenguaje y Utilerías 15

Descripción de la Administración de Memoria en Ejecución 15

Pruebas del Funcionamiento del Lenguaje 15

Codificación 15

Resultados 15

Listados Documentados 15

Comentarios de Documentación 15

Comentarios de Implementación 15

Manual de Usuario 16

# Descripción del Proyecto

## Descripción del Proyecto

### Visión, Objetivos y Alcance del Proyecto

#### Visión

Lograr crear un lenguaje para hispanohablantes fácil de entender, con el propósito de poner al alcance de otros sectores la posibilidad de aprender a programar sin toparse con barreras de idioma.

#### Objetivo

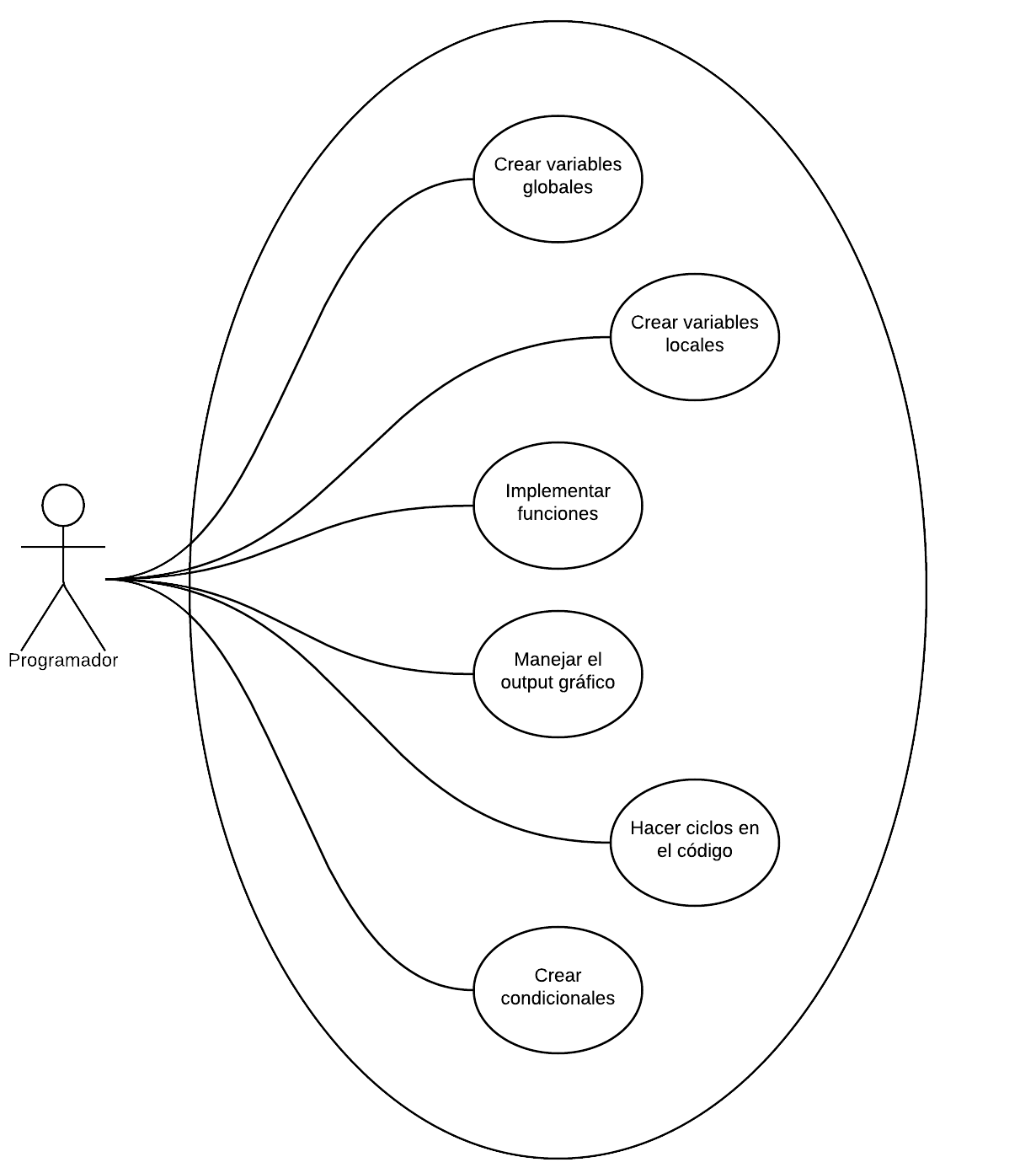
Nuestro objetivo radica principalmente en llegar a las comunidades mexicanas que no tienen una educación bilingüe. Creemos que hay mucho talento mexicano que podría desarrollarse en el área de la programación y que muchas personas podrían crear una carrera a partir de esto. Sin embargo existen muchas barreras para que ellos lo logren, entre ellas se encuentra la barrera del idioma. Por lo tanto queremos crear un lenguaje de programación fácil de usar y entender para hispanohablantes. De esta manera podremos interesar a personas de cortas edades entre los 10 y 15 años, para que después puedan seguir introduciéndose dentro del ámbito de la computación y salir adelante con su vida profesional. Lo ideal sería que después de crear el lenguaje, se cree un servicio comunitario que se encargue de enseñar este lenguaje a adolescentes de comunidades rurales sin acceso a una educación de esta índole. (Esto último queda fuera del alcance del proyecto de este semestre pero se propondrá a alguna asociación interesada.)

#### Alcance

El lenguaje que se va a desarrollar debe ser fácil de entender y de utilizar por hispanohablantes. Las instrucciones serán simples. Permitirá el uso de ciclos, recursión y expresiones aritméticas. Así mismo se implementará una interfaz de salida gráfica con la que el programador podrá ver el resultado de las operaciones gráficas que realice. Esta interfaz también será bastante sencilla, su objetivo será amenizar el proceso de aprendizaje de la programación.

### Análisis de Requerimientos y Casos de Uso Generales

* Variables globales y locales
* Uso de ciclos 'mientras’ y ‘hacer-mientras’
* Condicionales ‘si - si no’ para la toma de decisiones
* Uso de variables tanto locales como globales
* Llamadas a funciones
* Operaciones aritméticas
* Implementación de recursividad
* Output gráfico con movimientos básicos



### Descripción de los Principales Casos de Pruebas

1. Recursividad

* Factorial
* Fibonacci

1. Ciclos

* Factorial
* Fibonacci
* Búsqueda en vectores
* Ordenamiento de vectores
* Multiplicación de matrices

1. Gráfico

* Movimiento del componente gráfico

### Descripción del Proceso General del Desarrollo del Proyecto

El desarrollo general del proyecto fue siguiendo las entregas especificadas al inicio del semestre. Comenzando por el léxico y la sintaxis del nuevo lenguaje y las tablas de variables. Luego se definió el cubo semántico y por consiguiente los cuádruplos. Así mismo se comenzó a trabajar en paralelo con la máquina virtual que primeramente reconocería cuádruplos básicos. Después ya aceptaría instrucciones que modificarían la interfaz gráfica de salida.

#### Bitácora General:

* Octubre 12 - 16: Léxico, sintaxis, tabla de variables y cubo semántico
* Octubre 26 - 30: Cuádruplos y directorio de funciones
* Noviembre 2 - 6: Máquina virtual y generación de código de operaciones aritméticas
* Noviembre 9 - 13: Salida gráfica y generación de código de estatutos secuenciales
* Noviembre 16 - 20: Generación de código de estatutos condicionales
* Noviembre 23 - 25: Generación de código de arreglos y documentación

#### Reflexiones:

//TODO

## Descripción del Lenguaje

### Nombre del Lenguaje

B3

### Descripción Genérica del Lenguaje

El lenguaje B3 es un lenguaje bastante sencillo con instrucciones en español que permite que el programador declare variables tanto globales como locales de tipo Entero, Decimal, Cadena y Booleanas. Así mismo se podrán declarar arreglos para que el usuario los manipule. El programador podrá también cambiar el flujo de ejecución con estatutos condicionales y cíclicos y también con llamadas a diferentes funciones. La salida de interfaz gráfica que maneja este lenguaje de programación ayudará a que el programador visualice algunas de las operaciones que está realizando en una consola gráfica y con colores. Las funciones que este podrá realizar en la consola son bastante básicas, ya que engloba los movimientos en el eje X y Y del plano cartesiano y al irse moviendo el actor principal podrá o no ir pintando una línea de su camino realizado. Así mismo el programador a parte de poder mover y dibujar en el plano podrá cambiar el color del “lápiz” con el que se va trazando el camino programado.

### Descripción de Errores (compilación y ejecución)

//TODO

## Descripción del Compilador

### Equipo de Cómputo, Lenguaje y Utilerías

Equipo de Cómputo: Windows y Mac OS X

Lenguaje: Java

Utilerías: ANTLR para Java

### Descripción del Análisis de Léxico

#### Patrones de Construcción

* CTEC: '"'[a-zA-Z0-9]'"'; // Constante de Carácter
* CTES: '"'[a-zA-Z0-9]\*'"'; // Constante Cadena
* CTEF: [0-9]\*'.'[0-9]+; // Constante Decimal
* CTEI: [0-9]+; // Constante Entera
* ID: [a-zA-Z0-9]+; // Identificador
* WS: [ \n\t\r]+ -> skip; // Salto de línea

#### Enumeración de Tokens

* OP: '(';
* CP: ')';
* OK: '{';
* CK: '}';
* OB: '[';
* CB: ']';
* COM: ',';
* INT: 'ent';
* FLT: 'dec';
* BOL: 'bool';
* CHA: 'car';
* STR: 'cadena';
* VOID: 'nada';
* INCL: 'incluir';
* MAIN: 'main';
* LWHILE: 'mientras';
* LDO: 'hacer';
* LFOR: 'para';
* CIF: 'si';
* CELSE: 'sino';
* BRK: 'romper';
* RTRN: ‘regresar';
* PINU: 'pintar';
* PIND: 'flotar';
* COL: 'color';
* PRNT: 'imprime';
* MOV: 'mover';
* OAND: '&&';
* OOR: '||';
* ONOT: '!';
* HT: '#';
* IGU: '=';
* DEL: ';';
* MLT: '\*';
* DIV: '/';
* MOD: '%';
* PLS: '+';
* SUB: '-';
* MAI: '>=';
* MEI: '<=';
* OIG: '==';
* DIF: '!=';
* MAY: '>';
* MEN: '<';
* CTBT: 'verdadero';
* CTBF: 'falso';

### Descripción del Análisis de Sintaxis

#### Gramática Formal

|  |  |
| --- | --- |
| **Init** | includes body maindecl |
| **Body** | functions  | declarations  | asignations |
| **Includes** | | include includes |
| **Include** | HT INCL include2 |
| **Include2** | CTES  | MEN CTES MAY |
| **Functions** | | functiondecl functions |
| **Functiondecl** | type ID OP params CP OK estatutos CK |
| **Maindecl** | INT MAIN OP CP OK estatutos CK |
| **Type** | INT  | FLT  | BOL  | STR  | VOID |
| **Params** | | params2 |
| **Params2** | declaration params3 |
| **Params3** | | COM params2 |
| **Declarations** | | declaration declarations |
| **Declaration** | type multivar DEL |
| **Multivar** | ID dasign multivar2 |
| **Multivar2** | | COM multivar |
| **Dasign** | | IGU exp  | multidim |
| **Multidim** | | OB dimd CB |
| **Dimd** | exp dimd2 |
| **Dimd2** | | COM dimd |
| **Asignations** | | asignation asignations |
| **Asignation** | ID asignation2 DEL |
| **Asignation2** | IGU exp  | OB dima CB IGU exp |
| **Dima** | exp dima2 |
| **Dima2** | | COM dima |
| **Estatutos** | | estatuto estatutos |
| **Estatuto** | BRK  | RTRN  | functioncall  | asignation  | declaration  | print  | instructions  | loops  | conditional |
| **Loops** | llwhile  | lldowhile  | llfor |
| **Llwhile** | LWHILE OP exp CP OK estatutos CK |
| **Llfor** | LFOR OP asignation DEL exp DEL asignations CP OK estatutos CK |
| **Lldowhile** | LDO OK estatutos CK LWHILE OP exp CP |
| **Functioncall** | ID OP params CP DEL |
| **Conditional** | CIF OP exp CP OK estatutos CK conditional2 |
| **Conditional2** | | CELSE OK estatutos CK |
| **Print** | PRNT exp DEL |
| **Instructions** | PINU DEL  | PIND DEL  | COL OP CTEI COM CTEI COM CTEI CP  | MOV OP CTEI COM CTEI CP |
| **Exp** | exo exp2 |
| **Exp2** | | OAND exp |
| **Exo** | exn exo2 |
| **Exo2** | | OOR exo |
| **Exn** | exi exn2 |
| **Exn2** | | ONOT exn |
| **Exi** | exm exi2 |
| **Exi2** | | MAI exm  | MEI exm  | OIG exm  | DIF exm  | MAY exm  | MEN exm |
| **Exm** | ex exm2 |
| **Exm2** | | MOD exm |
| **Ex** | termino ex2 |
| **Ex2** | | PLS ex  | SUB ex |
| **Termino** | factor termino2 |
| **Termino2** | | MLT termino  | DIV termino |
| **factor** | OP exp CP  | PLS var  | SUB var  | var |
| **Var** | ID  | cte |
| **Cte** | CTEF  | CTEI  | CTES  | CTBT  | CTBF |

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

#### Código de Operación y Direcciones Virtuales

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operación** | **Asociación** | **Operación** | **Asociación** | **Operación** | **Asociación** | |
| + | DATA.ADD (4) | - | DATA.SUB (5) | \* | DATA.MUL (6) |
| / | DATA.DIV (7) | % | DATA.MOD (8) | && | DATA.AND (9) |
| || | DATA.OOR (10) | < | DATA.OLT (11) | > | DATA.OGT (12) |
| == | DATA.EQT (13) | <> | DATA.DIF (14) | >= | DATA.GOE (15) |
| <= | DATA.LOE (16) | = | DATA.EQS (17) | GOTO | DATA.GOT (18) |
| GOTO T | DATA.GTT (19) | GOTO F | DATA.GTF (20) | Constante | DATA.ACTE (21) |
| EOF | DATA.EOF (22) | Prepara Cambio | DATA.PC (23) | Cambio de Contexto | DATA.CC (24) |
| Termina Cambio | DATA.TC (25) | Retorno | DATA.RET (26) | Parámetro | DATA.PARAM (27) |
| Variables | DATA.DECL (28) | Cadena 1 | DATA.STR1 (29) | Cadena 2 | DATA.STR2 (30) |
| Imprimir | DATA.PRNT (31) | Imprimir Línea | DATA.PNTL (32) | Memoria | DATA.GAP (33) |
| Lápiz | DATA.PIN (34) | Mover | DATA.MVE (35) | Rojo | DATA.RED (36) |
| VERDE | DATA.GRN (37) | AZUL | DATA.BLU (38) | Color | DATA.COL (39) |
| Booleana? | DATA.VBL (40) | Rango? | DATA.VRG (41) |  |  |

**Enteros Globales** [0 – 2999]

**Decimales Globales** [3000 – 5999]

**Cadenas Globales** [6000 – 7999]

**Booleanas Globales** [8000 – 8999]

**Enteros Locales** [9000 – 11999]

**Decimales Locales** [12000 – 14999]

**Cadenas Locales** [15000 – 16999]

**Booleanas Locales** [17000 – 17999]

**Enteras Temporales** [18000 – 20999]

**Decimales Temporales** [21000 – 23999]

**Cadenas Temporales** [24000 – 25999]

**Booleanas Temporales** [26000 – 26999]

**Enteras Constantes** [27000 – 29999]

**Decimales Constantes** [30000 – 32999]

**Cadenas Constantes** [33000 – 34999]

**Booleanas Constantes** [35000 – 35999]

#### Diagramas de Sintaxis

//TODO

#### Descripción de Acciones Semánticas y de Código

#### Tabla de Consideraciones Semánticas

|  |  |  |  |
| --- | --- | --- | --- |
| **Operando 1** | **Operando 2** | **Operador** | **Resultado** |
| Entero | Entero | DATA.ADD | Entero |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.SUB | Entero |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MUL | Entero |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIV | Entero |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MOD | Entero |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.AND | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OOR | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OLT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OGT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIF | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.GOE | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.LOE | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQS | Entero |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operando 1** | **Operando 2** | **Operador** | **Resultado** |
| Decimal | Entero | DATA.ADD | Decimal |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.SUB | Decimal |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MUL | Decimal |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIV | Decimal |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MOD | Decimal |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.AND | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OOR | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OLT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OGT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQT | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIF | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.GOE | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.LOE | Booleano |
| Decimal | Booleano |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQS | ERROR |
| Decimal | Decimal |
| Cadena | ERROR |
| Booleano | ERROR |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operando 1** | **Operando 2** | **Operador** | **Resultado** |
| Cadena | Entero | DATA.ADD | ERROR |
| Decimal | ERROR |
| Cadena | Cadena |
| Booleano | ERROR |
| Entero | DATA.SUB | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MUL | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIV | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MOD | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.AND | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OOR | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OLT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OGT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIF | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.GOE | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.LOE | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQS | ERROR |
| Decimal | ERROR |
| Cadena | Cadena |
| Booleano | ERROR |

|  |  |  |  |
| --- | --- | --- | --- |
| **Operando 1** | **Operando 2** | **Operador** | **Resultado** |
| Booleano | Entero | DATA.ADD | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.SUB | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MUL | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.DIV | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.MOD | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.AND | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | Booleano |
| Entero | DATA.OOR | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | Booleano |
| Entero | DATA.OLT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.OGT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQT | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | Booleano |
| Entero | DATA.DIF | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | Booleano |
| Entero | DATA.GOE | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.LOE | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | ERROR |
| Entero | DATA.EQS | ERROR |
| Decimal | ERROR |
| Cadena | ERROR |
| Booleano | Booleano |

### Descripción de la Administración de Memoria en Compilación

#### Descripción de las Estructuras

Se crean cuatro clases de tipo MemoryIndex que contiene los rangos de cada tipo de dato (Entero, Decimal, Cadena y Booleano). Las cuatro clases englobarán los diferentes tipo de memoria, Global, Local, Temporal y Constantes.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memoria Global (memG) | 0 – 2999  (Enteros) | 3000 – 5999  (Decimal) | 6000 – 7999  (Cadena) | 8000 – 8999  (Booleano) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memoria Local (memL) | 9000 – 11999 (Enteros) | 12000 – 14999 (Decimal) | 15000 – 16999 (Cadena) | 17000 – 17999 (Booleano) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memoria Temporal (memT) | 18000 – 20999 (Enteros) | 21000 – 23999 (Decimal) | 24000 – 25999 (Cadena) | 26000 – 26999 (Booleano) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Memoria Constantes (memC) | 27000 – 29999 (Enteros) | 30000 – 32999 (Decimal) | 33000 – 34999 (Cadena) | 35000 – 35999 (Booleano) |

Ahora se describirán las tablas de variables, para esto se usó un HashMap, que contiene una llave y un valor. Se crea un HashMap para las variables Globales y así mismo uno diferente para las variables Locales. Cada vez que se entra a un cambio de contexto nuevo el HashMap de variables locales se reinicia. Dentro del valor de la llave se guardan varios atributos de la variable que se guarda. Los atributos de cada llave son los siguientes:

* **Nombre**: Nombre de la variable (por ejemplo: “X”).
* **Tipo**: [0,1,2,3] (Entero, Decimal, Cadena, Booleano).
* **Dirección**: Dirección dónde se guardará la variable en memoria virtual (por ejemplo: 27000).
* **M0**: Cantidad de casillas (por ejemplo: 14).
* **Tiene Dimensión?:** [T,F] (True si la variable tiene más de una dimensión).
* **Dimensión**: ArrayList<dirección> (La clase dirección tiene como elementos tamaño y M).

A continuación mostraremos un ejemplo de cómo se vería la tabla de variables si guardamos en ella la variable X[2,7] en memoria Global.

|  |  |
| --- | --- |
| Llave | Valor |
| “X” | **Nombre**: “X”  **Tipo**: 0  **Dirección**: 0  **M0**: 14  **Tiene** **Dimensión?:** T  **Dimensión**:   |  |  | | --- | --- | | Tamaño | M | | 2 | 7 | | Tamaño | M | | 7 | 1 | |

## Descripción de la Máquina Virtual

### Equipo de Cómputo, Lenguaje y Utilerías

Equipo de Cómputo: Windows y Mac OS X

Lenguaje: Java

Utilerías: Librerías de Java, OpenGL para Java (JOGL)

### Descripción de la Administración de Memoria en Ejecución

#### Descripción de las Estructuras

Para la memoria en ejecución se utiliza un HashMap para cada tipo de memoria (Global, Local, Temporal, Constante). Cada uno de estos HashMaps tendrá dentro de ellos los rangos que se recibirán de la memoria de compilación para cada tipo de dato (Entero, Decimal, Cadena, Booleano).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **memG** |  | **memL** |  | **memT** |  | **memC** |
| Entero  (0 – 2999) | Entero  (9000 – 11999) | Entero  (18000 – 20999) | Entero  (27000 – 29999) |
| Decimal  (3000 – 5999) | Decimal  (12000 – 14999) | Decimal  (21000 – 23999) | Decimal  (30000 – 32999) |
| Cadena  (6000 – 7999) | Cadena  (15000 – 16999) | Cadena  (24000 – 25999) | Cadena  (33000 – 34999) |
| Booleano  (8000 – 8999) | Booleano  (17000 – 17999) | Booleano  (26000 – 26999) | Booleano  (35000 – 35999) |

## Pruebas del Funcionamiento del Lenguaje

### Codificación

### Resultados

## Listados Documentados

### Comentarios de Documentación

### Comentarios de Implementación

# Manual de Usuario