

Diseño de Compiladores

Enero - Mayo 2016

PLearning

Documentación

Zyanya Valdés Esquivel

A01139628

Marcelo A. Cantú Quiroga

A00813174

a) DESCRIPCIÓN DEL PROYECTO

a.1) Objetivo y alcance del proyecto

Debido al crecimiento exponencial que ha tenido la tecnología en los últimos años, es fundamental que las personas se interesen en la programación desde pequeños. En este era digital, el mundo cada vez necesita más ingenieros de software. Y no solo eso, sino que también aprender a programar ayuda a desarrollar otras habilidades que son muy importantes para la vida diaria como lo son la solución de problemas y la lógica. El gran problema es que muchas veces los niños ven programar como algo tedioso, en donde se tiene que escribir mucho.

Por tal motivo, este proyecto tiene como propósito ayudar a niños o jóvenes que actualmente estén cursando la primaria o secundaria para que puedan aprender los principios básicos de programación de una forma interactiva. A parte de esto, se planea que el proyecto aumente su interés en el área de la programación gracias a la facilidad que tendrán ellos de crear programas rápidamente y de manera muy intuitiva.

Esto se llevará a cabo proveyendo un medio interactivo y gráfico para que sea más atractivo y fácil programar para ellos. Como propósito, por lo tanto, el lenguaje enseñará al usuario a desarrollar su lógica de programación, sin dejar de lado la enseñanza de los elementos que son comunes a la mayoría de los lenguajes modernos, como lo son las condiciones, los ciclos, la declaración de variables, el uso de funciones, escritura a pantalla, lectura, entre otros. Al mismo tiempo se espera que el lenguaje, mediante su interfaz intuitiva, cree un gusto en el usuario por programar.

a.2) Análisis de requerimientos y casos de uso

a.2.1 Análisis de requerimientos

*Requerimientos Funcionales*

*Requerimientos No-Funcionales*

a.2.2 Casos de uso



Ilustración 1 Casos de uso de PLearning

a.3) Descripción de los principales Test Cases

Todos los casos de prueba diseñados para el compilador prueban cosas comunes como la lectura del input, la escritura, la declaración de variables, la asignación, el retorno de una función. A continuación, se describen los principales casos de prueba y se describe que es lo que prueban de manera única.

|  |  |
| --- | --- |
| *Test Case* | *Descripción* |
| Factorial No Recursivo | Programa que calcula el factorial de un número de manera iterativa. Con esto se prueba el estatuto “for” y las llamadas a función. |
| Factorial Recursivo | Programa que calcula el factorial de un número de manera recursiva. Con esto se prueba el funcionamiento de la recursividad, así como el estatuto “if”. |
| Fibonacci Recursivo | Programa que calcula el numero n de la secuencia de Fibonacci de manera recursiva. Con esto se prueba la doble llamada recursiva en una sola expresión. |
| Fibonacci No Recursivo | Programa que calcula el numero n de la secuencia de Fibonacci de manera iterativa. Con esto se prueba |
| Merge Sort | Programa que ordena un arreglo de n cantidad de datos dados por el usuario utilizando el algoritmo MergeSort. Con esto se prueba la lectura y escritura en arreglos, el estatuto “while” y estatutos anidados (while con if). |
| Multiplicación de matrices | Programa que multiplica dos matrices dadas por el usuario. Con esto se prueba la lectura y escritura de matrices, las operaciones sobre algún elemento de la matriz, las variables dimensionadas como parámetros en funciones y la triple anidación del estatuto “For”. |
| Operaciones | Programa que obtiene el cuadrado, el incremento en uno y el decremento en uno de un número dado por el usuario. Con este programa se prueba los parámetros por referencia en las funciones. |

a.4) Descripción del proceso general seguido

a.4.1 Proceso general

Para el proyecto se utilizaron diversas herramientas de colaboración. GitHub se usó para controlar las versiones tanto del Front-End como del Back-End. Google Drive se utilizó para hacer colaboraciones en cuanto a los documentos requeridos para el proyecto. Por último, se utilizaron diversas redes sociales para poder establecer la comunicación entre los miembros del equipo.

Durante el tiempo que duró el proyecto se estableció una forma de trabajar ordenada basada en los avances ya provistos. Los miércoles de cada semana se iniciaba revisando cuales eran los requisitos para el siguiente avance, después, se discutía entre el equipo para la división de tareas y la implementación en sí. Posteriormente, cada quien trabajaba en su parte y todo se juntaba para que quedara listo para el siguiente miércoles. No siempre funcionó así ya que a veces el avance se tenía que recorrer para los siguientes días y todo el proceso también.

En los últimos avances se hizo una división entre la implementación de la máquina virtual y lo que hacía falta del compilador mientras todos trabajaban en la interfaz. Esto no era impedimento para que la persona que le tocaba cierta cosa, trabajara en otra cosa.

a.4.2 Bitácora

|  |  |  |
| --- | --- | --- |
| *Avance* | *Fecha* | *Comentarios* |
| Avance #1 | 02/03/2016 | Se terminó el análisis de léxico y el de sintaxis, marca los errores con una descripción del error y el número de línea en donde se encontró. |
| Avance #2 | 09/03/2016 | Se terminó la implementación del directorio de procedimientos y tablas de variables, se agregó el error de “variable previamente declarada”. |
| Avance #3 | 16/03/2016 | Se implementó el cubo semántico y la generación de cuádruplos para los estatutos aritméticos. Se agregó verificación se semántica para saber si dos tipos son compatibles y errores como “variable no declarada”. Se cambiaron los tipos de datos para que sean representados como enteros. |
| Avance #4 | 30/03/2016 | Se implementó la generación de cuádruplos para los estatutos “while” e “if”. Se empezó con el diseño para la generación de cuádruplos del estatuto “For”. |
| Avance #5 | 8/04/2016 | Se implementó la generación de cuádruplos para las llamadas a función, así como el estatuto “for”. También, se mejoró la documentación interna del código. |
| Avance #6 | 16/04/2016 | Se implementó la memoria para la máquina virtual junto con la ejecución de expresiones aritméticas, estatutos secuenciales y ciclos. Se empezó la interfaz del IDE implementando la funcionalidad de agregar estatutos, editarlos y eliminarlos. |
| Avance #7 | 24/04/2016 | Se terminó la máquina virtual, se implementó la generación de cuádruplos para arreglos y se continuó con la interfaz al implementar la funcionalidad de mover estatutos de renglón. |
| Avance #8 | 1/05/2016 | Se terminó la interfaz del IDE. Se implementó la generación de cuádruplos para los parámetros por referencia. Se modificó la máquina virtual para aceptar arreglos como parámetros. Se hicieron las pruebas correspondientes. |

a.4.2 Reflexión

*Marcelo Alberto Cantú Quiroga*

Este proyecto, sin lugar a dudas, fue el más retador que tuve en toda la carrera. No solo porque es mucho lo que se tiene que hacer, sino también porque que ese mucho conlleva emplear conocimientos que se adquirieron durante el transcurso de la carrera y no son cosas triviales que se puedan hacer rápido. Esto llegó a ser un desafío ya que hay que volver a repasar ciertos conceptos y eso hace que el proyecto se atrase.

Para poder completar satisfactoriamente este proyecto fue fundamental seguir la agenda de avances que se dio al inicio del semestre. Nunca antes con ningún otro proyecto había seguido un calendario bien establecido y con entregas regulares. Esto hizo que en los últimos días no se tuviera que trabajar a altas horas de la noche para completarlo. También creo que fue factor para que el producto saliera de mejor calidad, ya que se cuenta con más tiempo para diseñar, codificar y probar cada elemento del programa. Lo que si pasó fue que al final, como no tomamos en cuenta los elementos extra del proyecto para los avances, se acumulará un poco el trabajo y le tuvimos que dedicar más tiempo que otros avances.

Lo más valioso del proyecto fue que es el primer proyecto que podría considerar grande. Esto me enseñó a que es fundamental seguir un plan si es que se quiere terminar a tiempo y con calidad. También, las características del proyecto propiciaron la modularidad, es decir, que cada parte necesita estar bien hecha y probada para que todo el proyecto funcione. Asimismo, aprendí que es importante diseñar antes de codificar, esto si bien se puede volver tedioso ya que muchas veces lo único que queremos es codificar, ayuda mucho a evitar el retrabajo, lo cual es esencial cuando se tienen proyectos tan grandes como este. Por último, aprendí lo vital que es la colaboración en equipo. Con este tipo de proyectos es imprescindible que haya una buena comunicación y una buena y equitativa repartición del trabajo, de lo contrario, es imposible terminar a tiempo.

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

Firma

*Zyanya Valdés Esquivel*

Me parece que este proyecto, al igual que la clase completa, ha sido vital para comprender en realidad lo que sucede detrás de todo lo que hacemos a diario con todos los compiladores que utilizamos. Desde el inicio de los avances aprendí que es muy importante tener total comprensión de lo que se está trabajando ya que posteriormente se va a tener que trabajar con esos cimientos y quizás, a menos que se cuente con una planeación muy completa, se van a tener que editar esos comienzos.

Encontré un gran reto con la abstracción que se tiene que asimilar cuando se trabaja con proyectos y temas de este tipo. Opino y creo que queda claro con las entregas de este proyecto que es de suma importancia saber administrar el tiempo que se trabaja y llevar los avances bien agendados para que no quede trabajo pendiente y se vaya acumulando para el final. Siempre ha sido importante notar eso, sin embargo, es durante trabajos y proyectos como este donde queda obvio que un trabajo hecho a última hora no se va a lograr, aún que se trabaje durante 48 horas seguidas sin dormir.

Finalmente, creo que una de las cosas que más consideré importantes durante este proyecto fue que es vital recurrir a la documentación que existe de alguna herramienta o lenguaje. Aunque existan foros de ayuda o blogs o tutoriales externos, la documentación oficial sigue siendo más útil para poder entender el cómo y por qué es importante utilizar alguna característica de dicha herramienta.

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

Firma

B) DESCRIPCIÓN DEL LENGUAJE

b.1) Nombre del lenguaje

PLearning

b.2) Principales características del lenguaje

b.2.1) Características del ambiente gráfico

El ambiente gráfico cuenta con 4 partes principales. La primera, que se encuentra en la columna izquierda y en la parte superior, se encuentran todos los estatutos que el usuario puede agregar a su código. La segunda, se encuentra en el centro y es en donde se coloca todo el código que el usuario arrastre y edite. La tercera se encuentra en la esquina superior derecha y es en donde el usuario puede teclear el input para los programas que desarrolle (esto para que no tenga la necesidad de lidiar con archivos). La última se encuentra en la esquina inferior derecha y es el output, en el cual se expone todo lo que se escriba a pantalla.

Básicamente lo que hace el usuario es arrastrar de cualquiera de las opciones de los estatutos hacia el centro, le aparecen los cuadros que tiene que llenar, los llena y oprime ENTER. También puede: eliminarlos simplemente al arrastrarlo hacia el basurero, editarlos dándole doble click al estatuto y moverlos arrastrándolos hacia cualquier parte. Por último, el usuario le da click en compilar, si no hay errores, se muestra en el output; si los hay, se muestran en la línea en donde sucedió el error o en caso que sean de ejecución, se muestran en el output. El programa también cuenta con las opciones de guardar y abrir cualquier archivo, los cuales se guardan con la extensión “.ple”.

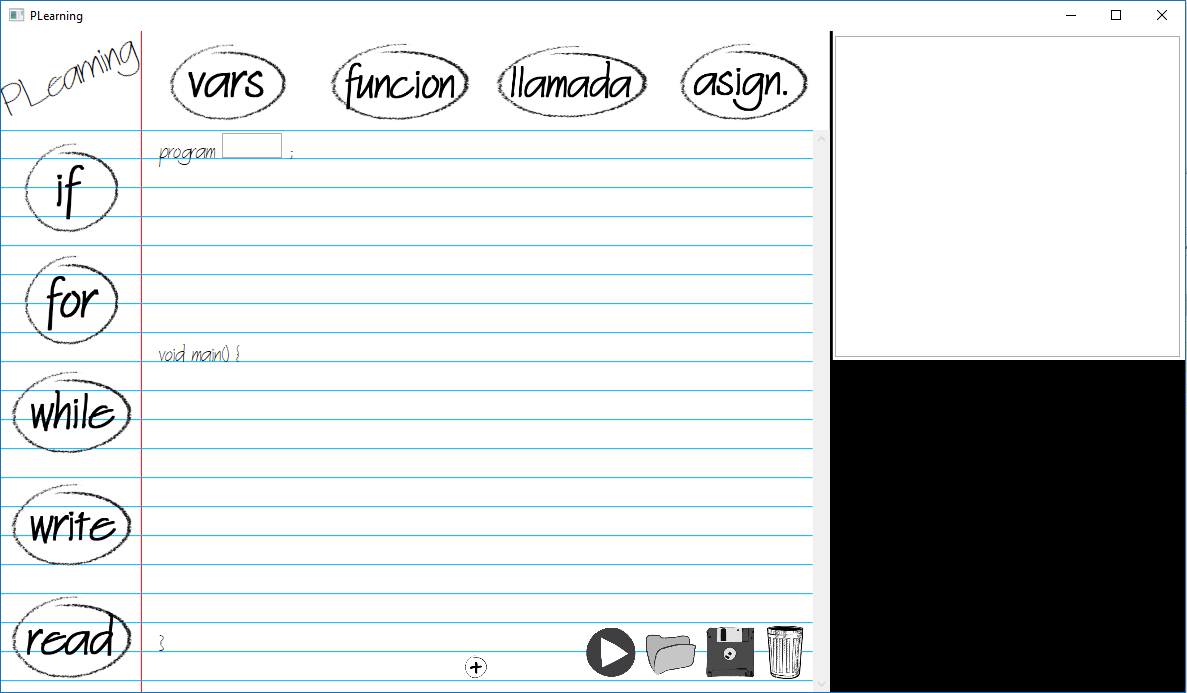


Ilustración 2 Pantalla principal del IDE

b.2.2) Características del lenguaje

El lenguaje cuenta con las mismas características principales de los lenguajes modernos como C++, Java y C# y se describen a continuación.

* Ciclos: El lenguaje cuenta con los estatutos “while” y “for”. El primero es para hacer repeticiones con una expresión booleana definida y el segundo es una ampliación del primero en el cual se incluye también la iniciación y modificación de la variable de control.
* Condición: Se cuenta con el estatuto “if-else”, el cual ejecuta cierto camino si la condición que se da es verdadera y ejecuta otro si es falsa.
* Escritura:Se utiliza el estatuto “print” para poder escribir expresiones en el output del programa, también mediante comas se pueden escribir varias expresiones en una sola línea.
* Lectura: Se utiliza el estatuto “write” para leer del input. La forma en que funciona es que lee lo siguiente que haya en la lista de inputs. Esta lista se puebla al obtener todo lo que el usuario haya escrito y separándolo con espacios y saltos de línea.
* Declaración de variables: No se utiliza ningún estatuto, solo se pone el tipo de variable, el nombre de la misma y, si así se desea, una asignación. Estas solo pueden ir al inicio de las funciones o del programa.
* Funciones: Las funciones se declaran al principio del programa especificando su tipo, nombre, parámetros y el cuerpo. Se llaman escribiendo el nombre y los parámetros entre paréntesis. Una llamada a una función void puede estar sola, mientras una que tenga retorno tiene que estar acompañada de una asignación a una variable.
* Variables dimensionadas: PLearning cuenta con la posibilidad de declarar y usar variables de una, dos y tres dimensiones. A parte de esto, también se pueden utilizar como parámetros de una función.
* Parámetros por referencia: A parte de los parámetros normales en donde solo se pasa el valor, también se pueden utilizar parámetros por referencia en donde la variable que se manda a la función cambia con la variable local de la misma.
* Expresiones: Tiene todas las operaciones aritméticas, uso de paréntesis, operadores lógicos y de comparación. Estos se pueden usar en muchas instancias del código, pero no por sí mismos.
* Tipos de datos: Los tipos de datos que se incluyen son entero, flotante, carácter, string y booleano.

b.3) Listado de errores

b.3.1 Errores de léxico y sintaxis

|  |  |
| --- | --- |
| *Mensaje* | *Ocurre cuando* |
| “Se esperaba <nombre de token>” | En la regla se esperaba un token y el analizador de sintaxis se encontró con otro. |
| “<nombre de la gramática> inválida” | Hay una palabra en la regla que no se encuentra en el diccionario o cuando el error de sintaxis sucede al final de la gramática. |

b.3.2 Errores de semántica

|  |  |
| --- | --- |
| *Mensaje* | *Ocurre cuando* |
| “No se indexo la variable dimensionada.” | Se utiliza una variable dimensionada sin especificar su índice. |
| “Tipos incompatibles.” | Se quieren hacer operaciones entre dos elementos con tipos que no son compatibles para realizar dicha operación. |
| “Variable no declarada.” | Se intenta utilizar un identificador que no se ha declarado anteriormente. |
| “Variable <nombre de variable> previamente declarada.” | Se intenta declarar una variable que ya ha sido declarado anteriormente. |
| “Los tipos de parámetros no coinciden con la función.” | Se intenta llamar a una función con parámetros que no coinciden en tipo con lo declarado. |
| “El parámetro que se declaró en la función no es por referencia.” | Se pone en la llamada a la función que un parámetro es por referencia cuando no se declaró así. |
| “El número de parámetros no coindice con la declaración de la función.” | En la llamada hay una mayor o menor cantidad de parámetros que lo que se declaró. |
| “El parámetro que se declaró en la función es por referencia.” | No se pone en la llamada que un parámetro es por referencia cuando en realidad si lo es. |
| “El número de dimensiones en el parámetro no coincide con lo declarado.” | Se intenta indexar una variable dimensionadas con una mayor o menos cantidad de dimensiones que lo declarado. |
| “El tamaño de la matriz dimensionada tiene que ser igual al declarado.” | Se pone como parámetro una variable que no coincide en dimensiones con lo que se declaró |
| “El número de parámetros no coindice con la declaración de la función.” | Se llama a una función con una cantidad de parámetros que es mayor o menor que lo que se declaró |
| “El indexamiento para variables dimensionadas necesita ser de tipo entero.” | La expresión que se evalúa para conocer el índice no es de tipo entero. |
| “Función <nombre de función> previamente declarada.” | Se intenta declarar una función que ya ha sido declarada con anticipación. |
| “La función que se intenta llamar no existe.” | Se hace una llamada a una función que no se ha declarado. |
| “La función tiene que regresar algo para usarse como expresión.” | Se intenta utilizar una función de tipo void como parte de una expresión. |
| “La llamada a la función tiene que ser a una función void.” | Se hace una llamada a una función sola que no es void. |
| “Tipos incompatibles en return.” | La expresión que se evalúa en el return no coincide con el tipo que debe de regresar la función. |
| “Una función que no es void tiene que tener return.” | No hay return en una función que no es void. |
| “No se pueden mandar variables dimensionadas como parámetros por referencia.” | Se intenta poner como parámetro por referencia una variable dimensionada. |
| “Se esperaba un booleano en el while.” | La expresión que se evalúa como condición en el while no es de tipo bool. |
| “Se esperaba un booleano en el for.” | La expresión que se evalúa como condición en el for no es de tipo bool. |
| “Se esperaba un booleano en el if” | La expresión que se evalúa como condición en el if no es de tipo bool. |
| “Tipos no compatibles en asignación.” | El lado derecho y el lado izquierdo de una asignación no son del mismo tipo. |
| “La variable que se intenta accesar no es dimensionada.” | Se intenta indexar una variable que no se declaró como dimensionada. |

b.3.3 Errores de ejecución

|  |  |
| --- | --- |
| *Mensaje* | *Ocurre cuando* |
| “No se reconoce ningún tipo en el input.” | Se lee un valor del input y no tiene el formato de alguno de los tipos de datos del lenguaje. |
| “El tipo de variable y el input no coinciden.” | Se lee del input a una variable que no es igual en tipo a lo leído. |
| “No hay suficientes datos en el input.” | Se quieren leer cosas del input pero no hay más cosas por leer. |
| “El índice de la variable está fuera del rango.” | Se pone algún índice en una variable dimensionada y no está en el rango que se declaró para dicha variable. |

C) DESCRIPCIÓN DEL COMPILADOR

c.1) Equipo de cómputo, lenguaje y utilerías

El equipo de cómputo utilizado para realizar el compilador fueron computadoras que corrían Windows. Para el desarrollo del compilador, se utilizó el generador Coco/R, en su versión para C#. No se utilizaron librerías especiales más que las contenidas en .NET.

c.2) Descripción del Análisis Léxico

c.2.1 Palabras reservadas

|  |  |  |  |
| --- | --- | --- | --- |
| program | void | int | float |
| string | bool | char | if |
| else | and | main | or |
| for | while | print | read |
| print | function | true | false |
| ref | return |  |  |

c.2.2 Elementos de instancia única

|  |  |  |  |
| --- | --- | --- | --- |
| *Token* | *Expresión Regular* | *Token* | *Expresión Regular* |
| PYC | “;” | MENOS | “-“ |
| COMA | “,” | MULT | “\*” |
| IGUAL | “=” | DIV | “/” |
| PARAB | “(“ | MAQUE | “>” |
| PARCI | “)” | MEQUE | “<” |
| LLAVEAB | “{“ | MAQUEEQ | “>=” |
| LLAVECI | “}” | MEQUEEQ | “<=” |
| CORCHAB | “[“ | DIFERENTE | “!=” |
| CORCHCI | “]” | COMPARACION | “==” |
| MAS | “+” |  |  |

c.2.2 Elementos de instancia múltiple

|  |  |
| --- | --- |
| *Token* | *Expresión Regular* |
| ID | [a-zA-Z][a-zA-Z0-9]\* |
| CTEENTERA | [0-9]+ |
| CTEFLOAT | [0-9]+\.[0-9]+ |
| CTESTRING | ' " ' . ' " ' |
| CTECHAR | ' \' ' . '\' ' |

\* El punto quiere decir “Todos los caracteres a excepción del salto de línea”.

\* El \ se usa para escapar caracteres.

c.3) Descripción del Análisis de Sintaxis

PLearning = PROGRAM ID PYC { vars } { funcion } main.

vars = tipo ( atomicos | dimensionadas ) PYC.

atomicos = ID { COMA ID } [ IGUAL ( ctelet | ctenum | ctebool ) ] .

dimensionadas = CORCHAB CTEENTERA [ COMA CTEENTERA [ COMA CTEENTERA ] ] CORCHCI ID { COMA ID }.

funcion = FUNCTION regresa ID PARAB [ parametro { COMA parametro } ] PARCI LLAVEAB { vars } {estatuto} [ RETURN expresion PYC ] LLAVECI .

parametro =[ REF ] tipo ( ID | CORCHAB CTEENTERA [ COMA CTEENTERA [ COMA CTEENTERA ] ] CORCHCI ID ) .

main = VOID MAIN PARAB PARCI LLAVEAB { vars } { estatuto } LLAVECI.

regresa = ( tipo | VOID ).

tipo = ( INT | FLOAT | STRING | BOOL | CHAR ).

bloque = LLAVEAB { estatuto } LLAVECI .

ciclowhile = WHILE PARAB expresion PARCI bloque.

ciclofor = FOR PARAB ID IGUAL expresion PYC expresion PYC expresion PARCI bloque .

ciclos = ( ciclowhile | ciclofor ).

condicion = IFF PARAB expresion PARCI bloque [ ELSE bloque ].

estatuto = ( condicion | escritura | ciclos | asignacion | llamada ).

lectura = READLINE PARAB PARCI.

escritura = PRINT PARAB expresion { COMA expresion } PARCI PYC.

asignacion = ID [ cuantificador] IGUAL ( expresion | lectura ) PYC.

llamada = ID PARAB [ ( REF ID | expresion ) { COMA ( REF ID | expresion ) } ] PARCI PYC.

cuantificador = CORCHAB expresion [ COMA expresion [ COMA expresion ] ] CORCHCI.

expresion = comparacion { ( AND | OR ) comparacion }.

comparacion = exp [ ( MAQUE | MEQUE | MAQUEEQ | MEQUEEQ | DIFERENTE | COMPARACION ) exp ].

exp = term { (MAS | MENOS) term }.

term = factor { ( MULT | DIV) factor }.

factor = PARAB expresion PARCI | ctevar.

ctevar = (ctelet | ctebool | ctenum | identificador).

identificador = ID (PARAB [ ( REF ID | expresion ) { COMA ( REF ID | expresion ) } ] PARCI | cuantificador | ).

ctelet = (CTESTRING | CTECHAR ).

ctebool = (TRUE | FALSE).

ctenum = [MAS | MENOS ] ( CTEENTERA | CTEFLOAT ) .

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

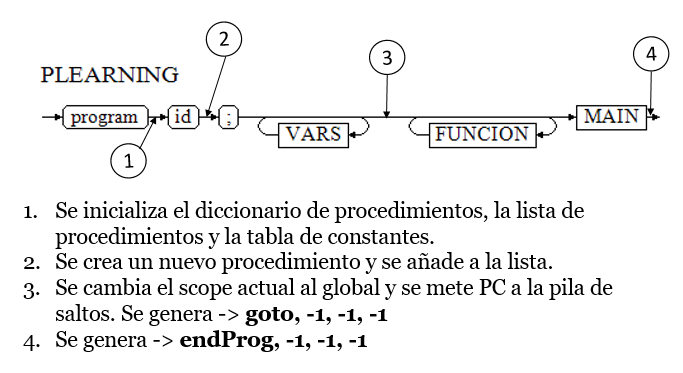
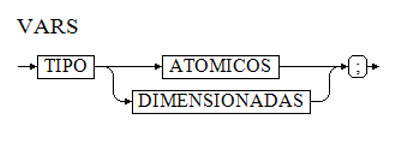
c.4.1 Códigos de operación

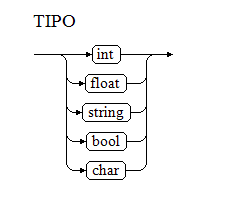
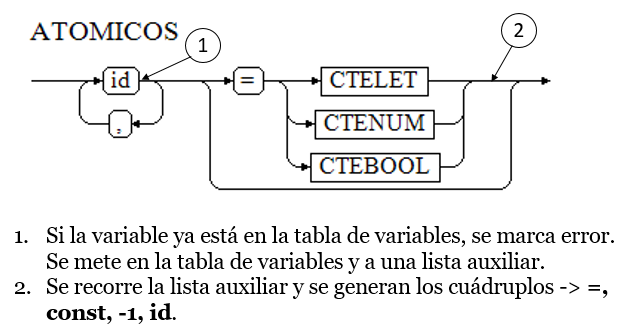
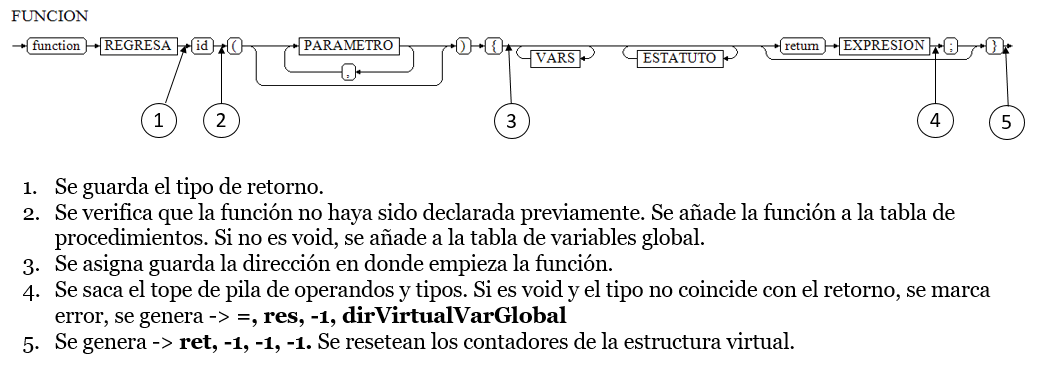
|  |  |
| --- | --- |
| Codigo | Descripción |
| And | Operación lógica and. |
| Or | Operación lógica or. |
| MoreThan | Operación de comparación mayor que. |
| LessThan | Operación de comparación menor que. |
| Different | Operación de comparación diferente. |
| EqualComparison | Operación de comparación igual. |
| Sum | Operación aritmética suma. |
| Substraction | Operación aritmética resta. |
| Multiplication | Operación aritmética multiplicación. |
| Division | Operación aritmética división. |
| MoreThanEq | Operación comparación mayor o igual a que. |
| LessThanEq | Operación de comparación menor o igual a que. |
| Assignment | Operación aritmética de asignación. |
| Goto | Mueve el program counter a la posición indicada. |
| GotoF | Si la condición que se da es falsa, mueve el program counter a la posición indicada. |
| Print | Sirve para imprimir en pantalla. |
| Read | Sirve para leer del input. |
| Era | Sirve para crear el registro de activación de la función. |
| Param | Sirve para asignar el valor de la llamada al valor de la variable de la función. |
| Ret | Sirve para regresar el program counter a donde se había quedado antes de la llamada. |
| Gosub | Sirve para mandar a dormir a la memoria actual. |
| EndProg | Indica que ya se acabó el programa. |
| Verify | Verifica que el índice de una variable dimensionada esté dentro del rango. |
| EndFunc | Indica que ya se acabó la función y no vienen más parámetros por referencia que resolver. |
| Ref | Asigna de la memoria de la función a la última memoria dormida los parámetros por referencia. |

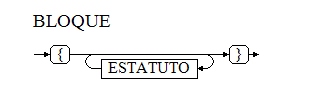
c.4.2 Direcciones virtuales asociadas

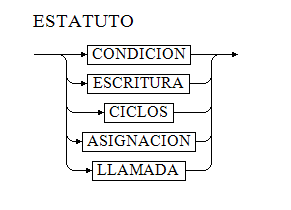
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Tipo de dato / Tipo de variable* | *Int* | *Float* | *String* | *Bool* | *Char* |
| *Global* | 10,000 a 11, 999 | 12,000 a 13,999 | 14,000 a 15,999 | 16,000 a 17,999 | 18,000 a 19,999 |
| *Local* | 20,000 a 21, 999 | 22,000 a 23,999 | 24,000 a 25,999 | 26,000 a 27,999 | 28,000 a 29,999 |
| *Tempora* | 30,000 a 30, 999 | 31,000 a 31,999 | 32,000 a 32,999 | 33,000 a 33,999 | 34,000 a 34,999 |
| *Constante* | 35,000 a 36, 999 | 37,000 a 38,999 | 39,000 a 40,999 | 41,000 a 42,999 | 43,000 a 44,999 |

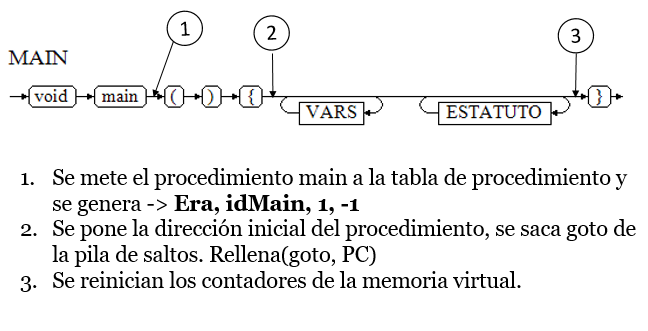
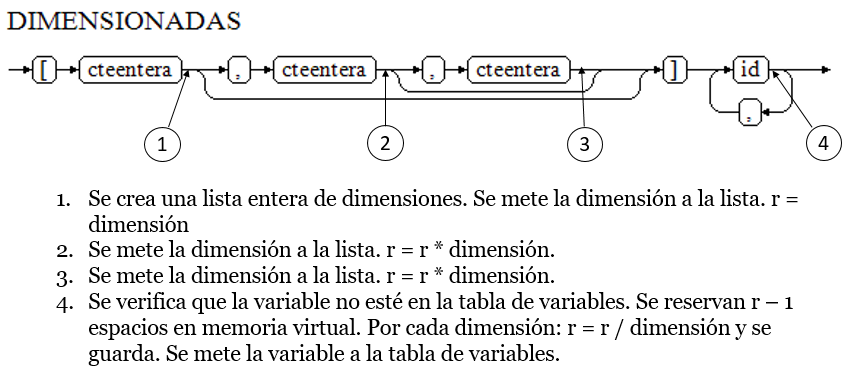
c.4.3 Diagramas de sintaxis

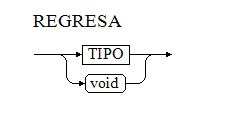
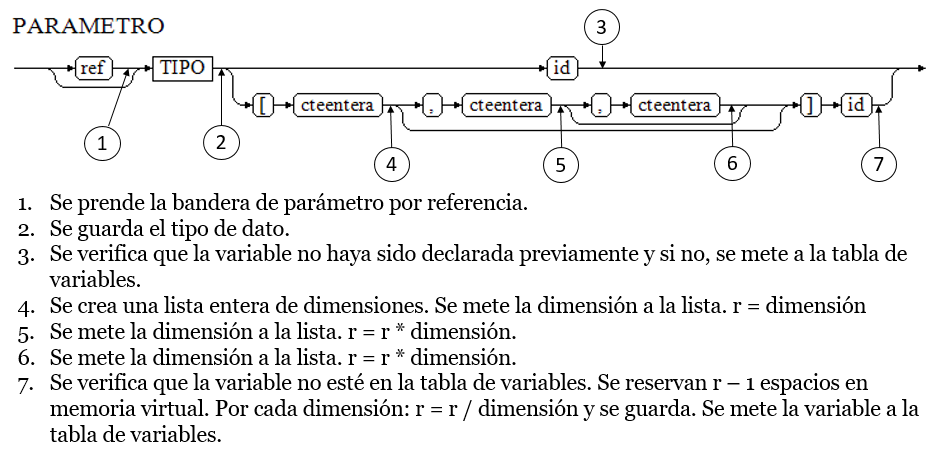


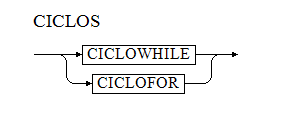


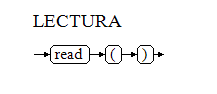


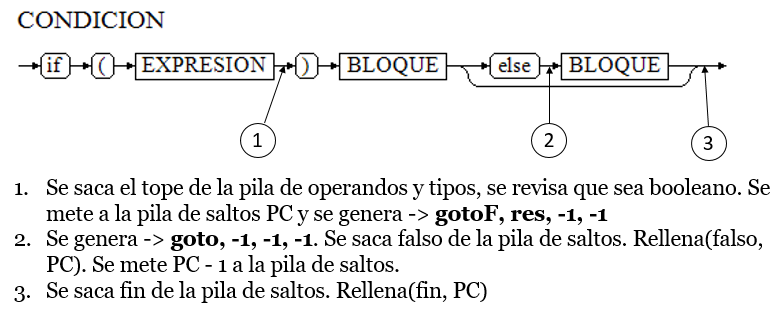
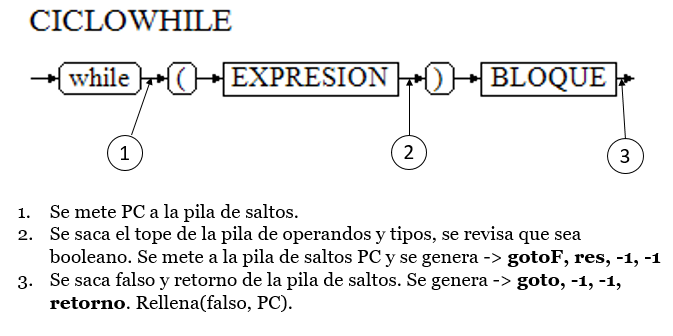


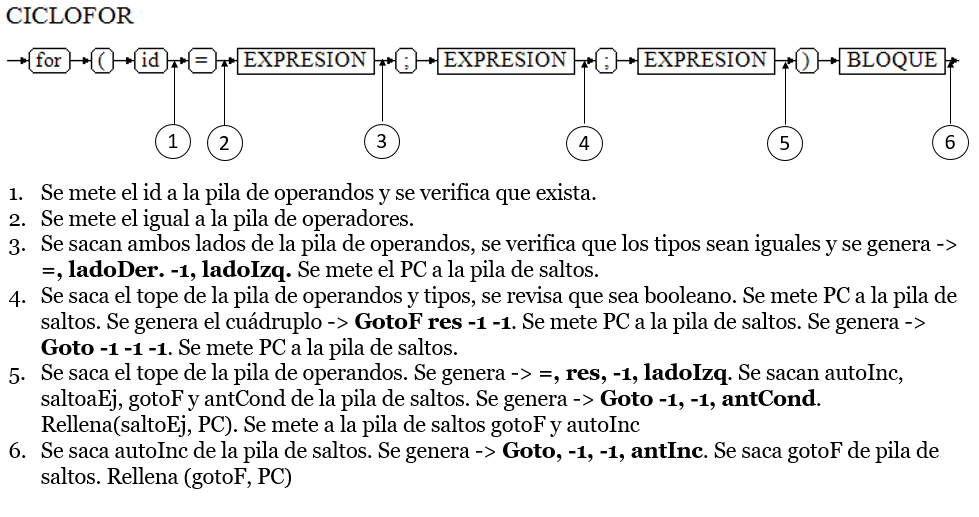
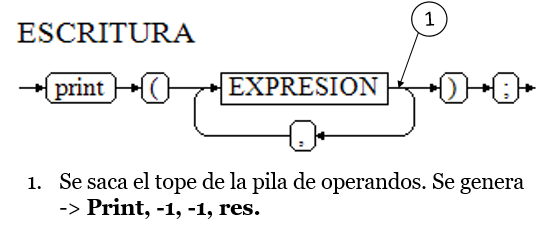


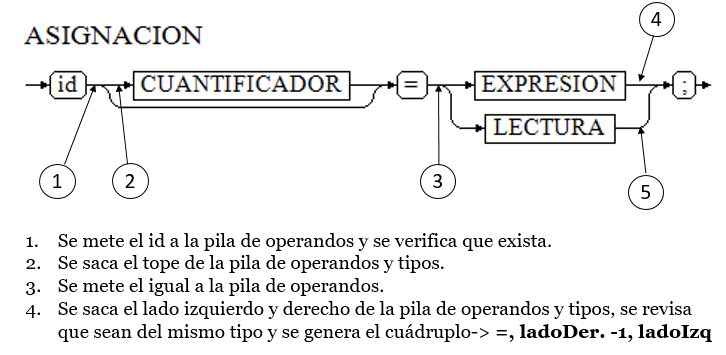


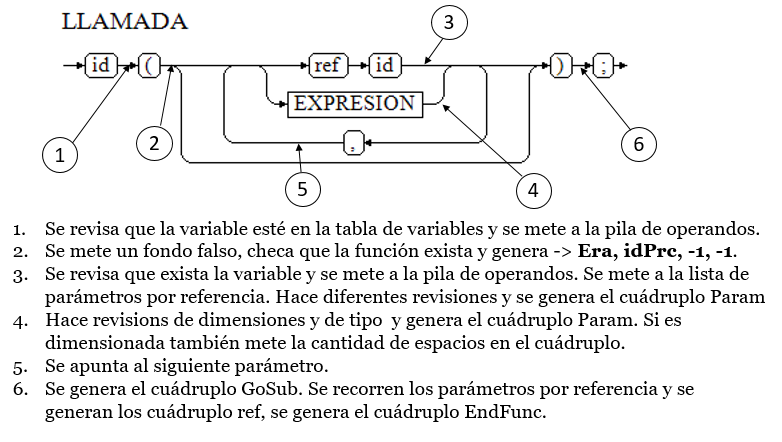
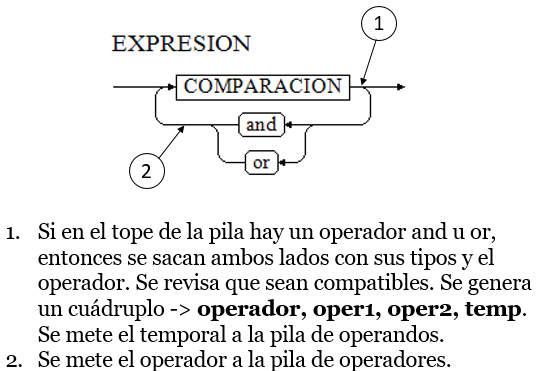


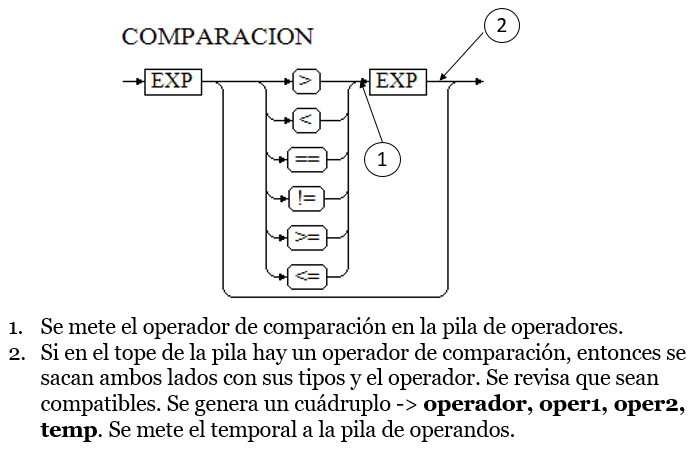


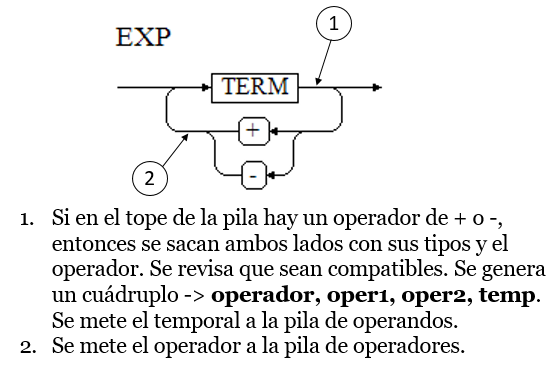


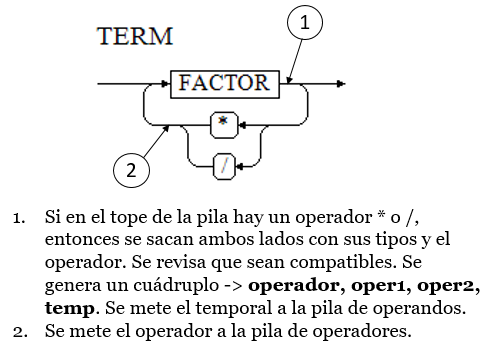
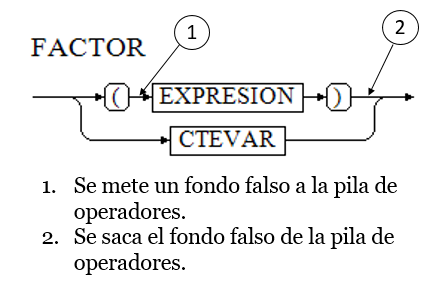
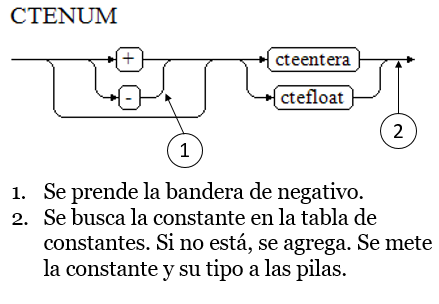


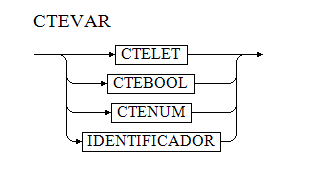
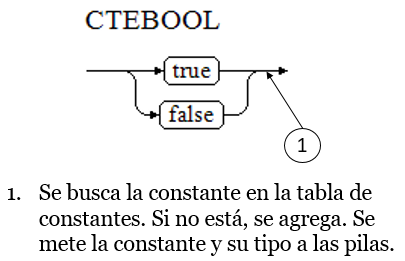
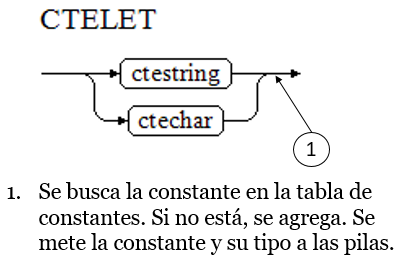


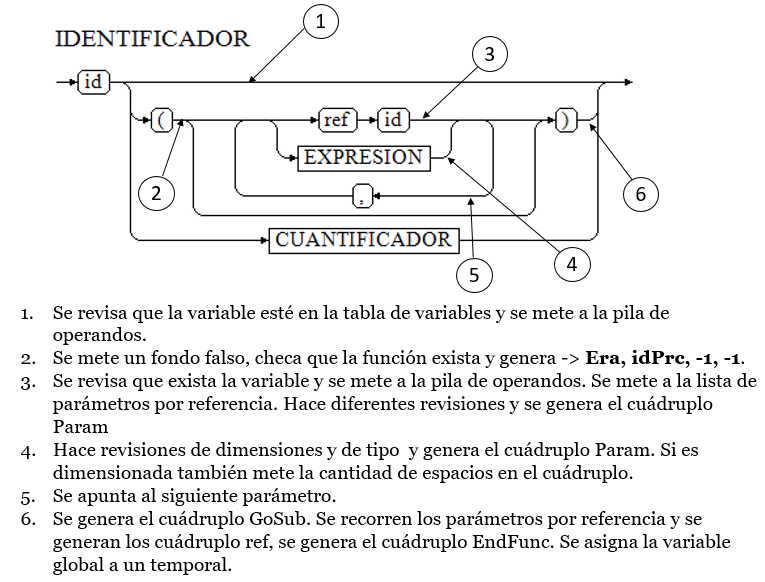


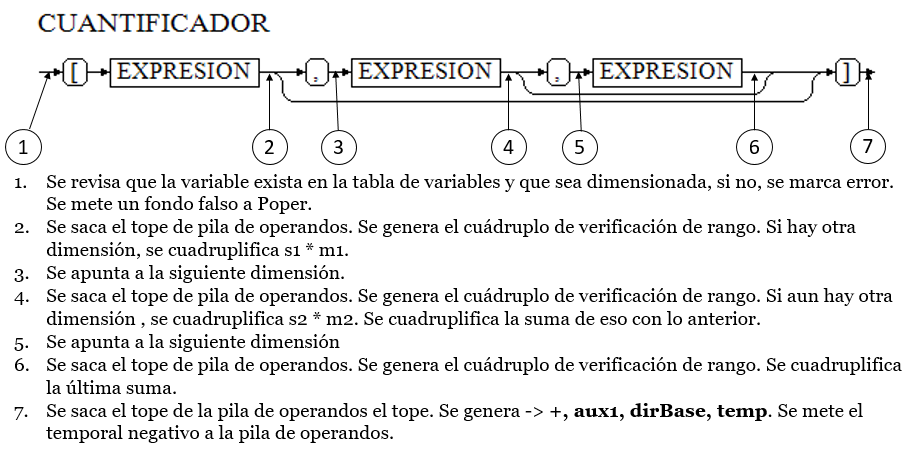












c.4.4 Tabla de consideraciones semánticas

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Operandos* | | *Operadores* | | | | | | | | | | | |
| *Operando Izquierdo* | *Operando derecho* | And | Or | > | < | != | == | + | - | \* | / | >= | <= |
| Int | Int |  |  |  |  |  |  |  |  |  |  |  |  |
| Int | Float |  |  |  |  |  |  |  |  |  |  |  |  |
| Int | String |  |  |  |  |  |  |  |  |  |  |  |  |
| Int | Bool |  |  |  |  |  |  |  |  |  |  |  |  |
| Int | Char |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | Int |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | Float |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | String |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | Bool |  |  |  |  |  |  |  |  |  |  |  |  |
| Float | Char |  |  |  |  |  |  |  |  |  |  |  |  |
| String | Int |  |  |  |  |  |  |  |  |  |  |  |  |
| String | Float |  |  |  |  |  |  |  |  |  |  |  |  |
| String | String |  |  |  |  |  |  |  |  |  |  |  |  |
| String | Bool |  |  |  |  |  |  |  |  |  |  |  |  |
| String | Char |  |  |  |  |  |  |  |  |  |  |  |  |
| Bool | Int |  |  |  |  |  |  |  |  |  |  |  |  |
| Bool | Float |  |  |  |  |  |  |  |  |  |  |  |  |
| Bool | String |  |  |  |  |  |  |  |  |  |  |  |  |
| Bool | Bool |  |  |  |  |  |  |  |  |  |  |  |  |
| Bool | Char |  |  |  |  |  |  |  |  |  |  |  |  |
| Char | Int |  |  |  |  |  |  |  |  |  |  |  |  |
| Char | Float |  |  |  |  |  |  |  |  |  |  |  |  |
| Char | String |  |  |  |  |  |  |  |  |  |  |  |  |
| Char | Bool |  |  |  |  |  |  |  |  |  |  |  |  |
| Char | Char |  |  |  |  |  |  |  |  |  |  |  |  |

c.5) Descripción del proceso de Administración de Memoria

D) DESCRIPCIÓN DE LA MÁQUINA VIRTUAL

d.1) Equipo de cómputo, lenguaje y utilerías

d.2) Descripción del proceso de Administración de Memoria

d.2.1 Especificación gráfica de estructuras de datos

d.2.2 Asociación entre direcciones virtuales y reales

E) PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE

F) LISTADOS DEL PROYECTO