A picture containing text

Description automatically generated

Lenguaje airo - Avance 0

TC4038 Diseño de Compiladores

Pablo Vera Terán  
A01730223

Table of Contents

[Consideraciones semánticas 3](#_Toc133880375)

[Declaraciones 3](#_Toc133880376)

[Programas 3](#_Toc133880377)

[Variables 3](#_Toc133880378)

[Funciones 4](#_Toc133880379)

[Estatutos 4](#_Toc133880380)

[Asignación 4](#_Toc133880381)

[Llamada a función sin retorno 5](#_Toc133880382)

[Lectura 5](#_Toc133880383)

[Escritura 5](#_Toc133880384)

[Cargar datos 5](#_Toc133880385)

[Estatuto de decisión 5](#_Toc133880386)

[Estatuto de repetición 5](#_Toc133880387)

[Expresiones 6](#_Toc133880388)

[Característica especial 6](#_Toc133880389)

[Tokens 6](#_Toc133880390)

[Palabras reservadas 6](#_Toc133880391)

[Expresiones regulares 6](#_Toc133880392)

[Diagramas de sintaxis 7](#_Toc133880393)

[Gramática 10](#_Toc133880394)

Lenguaje airo

Proyecto de Diseño de Compiladores: Individual

A continuación, se describen las características generales del lenguaje imperativo que se desarrolla como parte del curso Diseño de Compiladores en el programa de Ingeniería de Tecnologías Computacionales.

# Consideraciones semánticas

Para las secciones de código en el documento:

* Las secciones en itálicas son opcionales.
* Las palabras y símbolos en negritas son reservadas.
* El numeral (#) indica comentario.
* La separación entre líneas de código se da por el salto de línea.
* Una sección inicia con una llave curva de apertura ({) y termina con una llave curva de cierre (}).

En airo existen:

* <VAR\_ID>: Identificadores para variables.
* <FUNC\_ID>: Identificadores para funciones.
* Palabras reservadas: selección de palabras provenientes del inglés que no pueden usarse como identificadores.

## Declaraciones

### Programas

La estructura general de un programa escrito en airo es:

**Program** Nombre\_prog

*<Declaración de variables Globales>*

*<Definición de Funciones>*

*# esto es un comentario*

**main ()**

**{**

*<Estatutos>*

**}**

### Variables

Las variables globales y locales en airo se declaran de la siguiente manera:

**var** <VAR\_ID> **:** <tipo>*<DIMS>*

donde

<tipo> puede ser **int**, para datos numéricos enteros; **float** para datos numéricos decimales; **char** para unidades de texto; **bool** para datos booleanos (TRUE, FALSE); o **frame** para un tipo de estructura de datos usada para estadística descriptiva, la cual se detalla más adelante.

*<DIMS>* es una sección opcional que de mencionarse determina si la variable declarada es un vector —de una dimensión— o una matriz —de dos dimensiones— del tipo declarado. Esta sección tiene la notación **[**N**]** o **[**N**][**M**]** donde cada dimensión contiene N o M elementos y estos están numerados de 1 a N y de 1 a M. N representa a la primera dimensión (filas), y M a la segunda (columnas). Todos los elementos en un vector y en una matriz deben ser del mismo tipo, es decir, no puede haber vectores o matrices cuyos elementos sean de tipos variados.

Los letreros o *strings* son almacenados como vectores de tipo **char**.

Algunos ejemplos válidos son:

**var** nombre **:** **char[15]**

**var** coordenada **:** **float[**2**]**

**var** data : **frame[**5**][**2**]**

**var** unaLetra **:** **char**

**var** isOn **: bool**

### Funciones

Dentro del programa existe la sección *<Definición de Funciones>*, donde puede haber cero o más funciones definidas. Cada una se declara de la siguiente manera:

**func** <FUNC\_ID> **(***<PARAM\_LIST>***) :** <tipo-retorno> **{**

<Declaración de Variables locales y Estatutos>

***return*** *<VAR\_ID>*

**}**

donde

<tipo-retorno> representa el tipo de dato que regresa la función y puede ser **void**, **int, float**, **char**, **bool**, o **frame**.

***rete*** *<VAR\_ID>* es un estatuto opcional que se debe declarar si <tipo-retorno> es **int, float**, **char**, **bool**, o **frame.**

Para funciones cuyo retorno sea un vector o una matriz, no es necesario indicar las dimensiones, solo se debe indicar la palabra reservada que corresponda al tipo de dato del vector o matriz.

El lenguaje soporta llamadas a funciones dentro de otras funciones o la misma función, es decir, soporta llamadas recursivas.

### Estatutos

#### Asignación

Cuando se asigna una cadena de caracteres a una variable, ésta debe ser de tipo **char[]** y la notación es la siguiente:

<VAR\_ID>*<DIMS>* **=** **“***<TEXTO>***”**

Cuando se asigna el resultado de una expresión a una variable, éste debe ser del mismo tipo que el de aquella, y la notación es la siguiente:

<VAR\_ID>*<DIMS>* **=** <EXPR>

Cuando se asigna el valor retorno de una función a una variable, éste debe ser del mismo tipo que el de aquella, y la notación es la siguiente:

<VAR\_ID>*<DIMS>* **=** <FUNC\_ID>**(***<ARGS>***)**

donde

<FUNC\_ID> hace referencia a una función cuyo tipo no es **void**

*<ARGS>* es una sección opcional donde se enlistan los argumentos que se pasan a la función.

Por ejemplo:

<VAR\_ID>*<DIMS>* **=** **“**Hola mundo**”**

<VAR\_ID>*<DIMS>* **=** <FUNC\_ID>**(***<ARGS>***)** **+** 42

<VAR\_ID>*<DIMS>* **=** <FUNC\_ID>**(***<ARGS>***)** **OR** <FUNC\_ID>**(***<ARGS>***)**

#### Llamada a función sin retorno

Cuando se hace una llamada a una función que no tiene retorno (de tipo **void**), la notación es la siguiente.

<FUNC\_ID>**(***<ARGS>***)**

#### Lectura

En un programa escrito en airo, cuando se quiere leer —como input— un valor para asignarse en un identificador se lee uno a la vez y se usa la siguiente notación.

<VAR\_ID>*<DIMS>* **= read()**

#### Escritura

En un programa escrito en aire, cuando se quiere escribir —como output— el valor de una variable, se lee uno a la vez y se usa cualquiera de las siguientes notaciones.

**write(**<VAR\_ID>*<DIMS>***)**

**write(“**<TEXTO>**”)**

#### Cargar datos

airo soporta la carga de datos numéricos a través de un archivo de texto plano de tipo *csv*. La función de carga contabiliza la cantidad de registros (N), así como la cantidad de variables (M) a través del contenido del archivo de texto plano; obtiene los primeros M elementos (nombres de columnas) que se encuentran antes del primer salto de línea, y los guarda en una matriz de tipo **char[**15**][**M**]**; después obtiene el resto de los elementos (los datos) y los guarda en una matriz de tipo **float[**N**][**M**]**, donde N es el número de registros y M es el número de columnas. El vector y la matriz conforman el tipo de dato **frame** del cual existe una variable previamente declarada a través de la cual se puede acceder con notación de punto. El uso es a través de cualquiera de las siguientes notaciones.

<VAR\_ID(tipo frame)> **= load(**<VAR\_ID(tipo char[])>**)**

<VAR\_ID(tipo frame)> **= load(“**<TEXTO>**”)**

por ejemplo,

**var** data **: frame[**432**][**10**]**

**var** filename **: char[**15**]**

filename **=** “data.txt”

data **= load(**filename**)**

#### Estatuto de decisión

La notación para estatutos de decisión según una condición es la siguiente.

**when (**<Expresión lógica-relacional>**) then {**

*<Estatutos>*

**}**

#### Estatuto de repetición

La notación para estatutos de repetición según una condición es la siguiente.

**while (**<Expresión lógica-relacional>**) then {**

*<Estatutos>*

**}**

### Expresiones

Las expresiones en airo son las tradicionales (como en C y en Java), existen tres tipos, descritos en la siguiente tabla.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tipo de expresión | Operadores | Tipo de operandos | Tipo de retorno | Ejemplo |
| Aritmética | **+**, **-** , **\***, **/** | numéricos (**int** y **float**) | numéricos (**int** y **float**) | 3 **+** 4 **/** 2 |
| Lógica booleana | **AND**, **OR, NOT** | Expresiones lógicas, Expresiones relacionales, TRUE, FALSE | TRUE, FALSE | **NOT(**FALSE **AND** TRUE**)** |
| Relacional | **<** , **>**, **==**, **>=**, **<=**, **!=** | numéricos (**int** y **float**) | TRUE, FALSE | 4 **==** 3 |

Se manejan las prioridades tradicionales, se pueden emplear paréntesis para alterarla.

### Característica especial

Esta versión del proyecto permitirá trabajar y manipular (estadísticamente) los datos contenidos en un archivo de texto plano. La carga de dicho archivo a través de la función **load()** permitirá reconocer el nombre de las variables al inicio del contenido. Se incluye una función **state()** que proporciona un resumen del resultado de un conjunto de funciones de estadística descriptiva básica que se pudieran aplicar sobre ese conjunto de datos (ejemplo: calcular medias, medianas, modas, desviaciones estándar, cuartiles).

# Tokens

## Palabras reservadas

AND, BOOL, CHAR, ELSE, FALSE, FLOAT, FRAME, FUNC, INT, LOAD, MAIN, NOT, OR, PROGRAM, READ, RETURN, THEN, TRUE, VAR, VOID, WHEN, WHILE, WRITE.

## Expresiones regulares

<ASGNMNT> = r'=',

<CLBRACE> = r'\}',

<CLBRACKET> = r'\]',

<CLPARENTH> = r'\)',

<COLON> = r':',

<COMMA> = r',',

<DIVIDE> = r'/',

<EQUAL> = r'==',

<FLOAT> = r'\d+\.\d+'

<GREATER> = r'>',

<GREATEREQ> = r'>=',

<ID> = r'[a-zA-Z\_][a-zA-Z\_0-9]\*',

<INT> = r'\d+'

<LESS> = r'<',

<LESSEQ> = r'<=',

<MINUS> = r'-',

<OPBRACE> = r'\{',

<OPBRACKET> = r'\[',

<OPPARENTH> = r'\(',

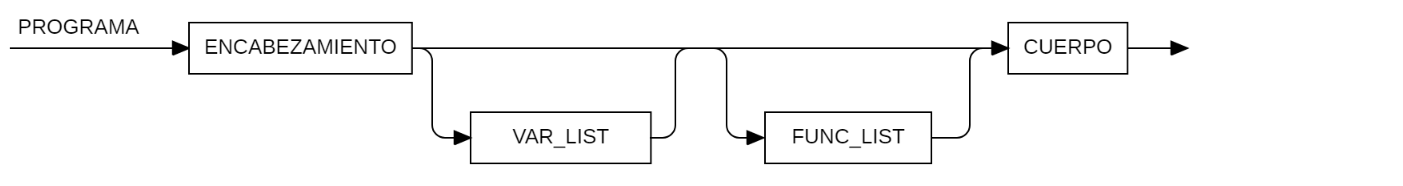
<PLUS> = r'\+',

<STRING> = r'"([^"\\]|\\.)\*"',

<TIMES> = r'\\*',

<UNEQUAL> = r'!='

# Diagramas de sintaxis

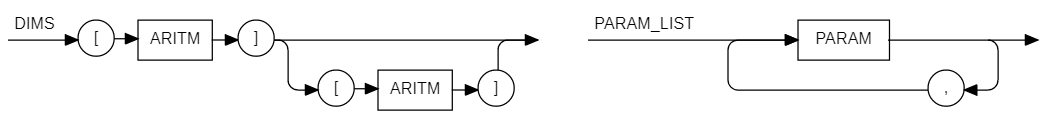


Diagram

Description automatically generated

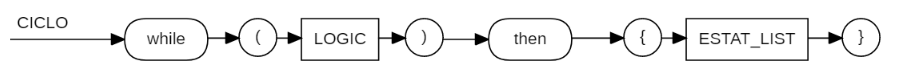
A picture containing graphical user interface

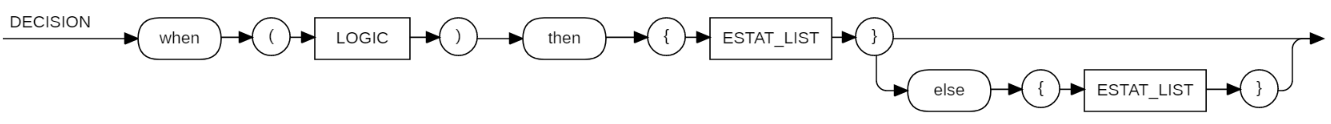
Description automatically generated

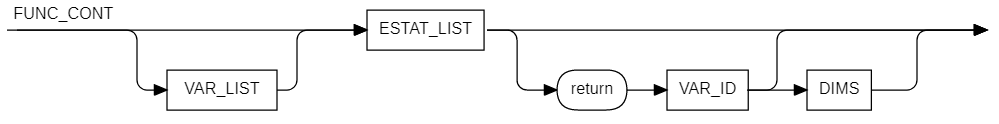


Diagram

Description automatically generated







Diagram

Description automatically generated

A picture containing text, device

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generatedDiagram

Description automatically generated|

Diagram

Description automatically generatedDiagram

Description automatically generated

Diagram

Description automatically generated Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram

Description automatically generated

Diagram, schematic

Description automatically generated

# Gramática

program : encabezamiento var\_list func\_list cuerpo

        | encabezamiento var\_list cuerpo

        | encabezamiento func\_list cuerpo

        | encabezamiento cuerpo

encabezamiento : PROGRAM ID

cuerpo : MAIN OPPARENTH CLPARENTH OPBRACE estat\_list CLBRACE

variable : VAR ID COLON var\_typ

        | VAR ID COLON var\_typ dims

var\_list : variable var\_list

        | variable

func\_list : func func\_list

        | func

estat\_list : estat estat\_list

        | estat

param\_list : param COMMA param\_list

        | param

dims : OPBRACKET aritm CLBRACKET

    | OPBRACKET aritm CLBRACKET OPBRACKET aritm CLBRACKET

func : FUNC ID OPPARENTH CLPARENTH func\_typ OPBRACE func\_cont CLBRACE

    | FUNC ID OPPARENTH param\_list CLPARENTH func\_typ OPBRACE func\_cont CLBRACE

ciclo : WHILE OPPARENTH logic CLPARENTH THEN OPBRACE estat\_list CLBRACE

decision : WHEN OPPARENTH logic CLPARENTH THEN OPBRACE estat\_list CLBRACE

        | WHEN OPPARENTH logic CLPARENTH THEN OPBRACE estat\_list CLBRACE ELSE OPBRACE estat\_list CLBRACE

func\_cont : var\_list estat\_list RETURN ID dims

        | var\_list estat\_list RETURN ID

        | estat\_list RETURN ID dims

        | var\_list RETURN ID dims

        | estat\_list RETURN ID

        | var\_list RETURN ID

        | RETURN ID dims

        | RETURN ID

estat : asign

    | llam\_void

    | lectura

    | escritura

    | carga\_dt

    | decision

    | ciclo

carga\_dt : ID ASGNMNT LOAD OPPARENTH ID CLPARENTH

        | ID ASGNMNT LOAD OPPARENTH CONST\_STRING CLPARENTH

param : ID COLON var\_typ

    | ID COLON var\_typ dims

var\_typ : INT

        | FLOAT

        | CHAR

        | BOOL

        | FRAME

func\_typ : INT

        | FLOAT

        | CHAR

        | BOOL

        | FRAME

        | VOID

aritm : term PLUS aritm

    | term MINUS aritm

    | term

term : factor TIMES term

    | factor DIVIDE term

    | factor

factor : OPPARENTH aritm CLPARENTH

    | ID

    | ID dims

    | CONST\_INT

    | CONST\_FLOAT

logic : oprnd AND logic

    | oprnd OR logic

    | NOT oprnd

    | oprnd

expr : aritm

    | logic

oprnd : FALSE

    | TRUE

    | relac

    | OPPARENTH logic CLPARENTH

relac : aritm EQUAL aritm

    | aritm UNEQUAL aritm

    | aritm LESS aritm

    | aritm LESSEQ aritm

    | aritm GREATER aritm

    | aritm GREATEREQ aritm

args : ID dims COMMA args

    | ID COMMA args

    | ID dims

    | ID

lectura : READ OPPARENTH CLPARENTH

escritura : WRITE OPPARENTH ID CLPARENTH

        | WRITE OPPARENTH ID dims CLPARENTH

        | WRITE OPPARENTH CONST\_STRING CLPARENTH

llam\_void : OPPARENTH CLPARENTH

        | OPPARENTH args CLPARENTH

asign : ID ASGNMNT expr

    | ID ASGNMNT CONST\_STRING

    | ID ASGNMNT ID OPPARENTH CLPARENTH

    | ID ASGNMNT ID OPPARENTH args CLPARENTH

    | ID ASGNMNT lectura

    | ID dims ASGNMNT expr

    | ID dims ASGNMNT CONST\_STRING

    | ID dims ASGNMNT ID OPPARENTH CLPARENTH

    | ID dims ASGNMNT ID OPPARENTH args CLPARENTH

    | ID dims ASGNMNT lectura