Descripción del Proyecto

# Instituto Tecnológico y de Estudios Superiores de Monterrey

# 

# Compiladores Enero-Mayo 2013

# Ing. Elda Quiroga

# Lenguaje: Simple

# Integrantes:

# Braulio Vladimir Chávez Núñez A00803220

# Teodoro Vargas Cortés A00808903

# 25 de Febrero del 2013

# VISIÓN

Durante el semestre de Enero-Mayo del año 2013, se creará el lenguaje de programación **Simple** (**Sim**ple **P**rogramming **L**anguage **E**loquent), el cual será una herramienta de programación de software visual enfocado a personas jóvenes (de entre 12 a 18 años) y con el paradigma de programación imperativo. Dicha herramienta estará disponible en la red para que cualquier persona, independientemente de su edad, tenga acceso a ella y pueda comprender fácilmente los fundamentos de la programación.

# OBJETIVOS DEL LENGUAJE

Dentro de los objetivos del lenguaje Simple, se encuentran:

* Facilidad para comprender los fundamentos de programación y fomentar el pensamiento crítico.
* Facilidad para crear programas sencillos por medios visuales.
* Soporte de funciones, strings, variables, operaciones matemáticas y ciclos.
* Interpretación de los programas desde el navegador del usuario, de tal forma que se evita el paso de descarga e instalación del compilador.
* Como posibilidad a futuro, permitir la internacionalización de los elementos del lenguaje para que sean más naturales para los usuarios de acuerdo con sus idiomas maternos.

# REQUERIMIENTOS

## Elementos léxicos

*Palabras reservadas y tokens:*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| int | le | + | ) | void |
| float | gt | - | [ | function |
| string | ge | \* | ] | return |
| boolean | eq | / | & | for |
| array | ne | = | : | if |
| true | not | { | , | else |
| false | and | } | ; | print |
| lt | or | ( | main | input |

# Expresiones regulares:

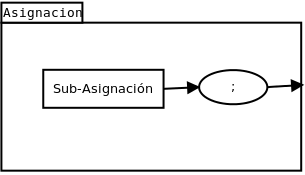
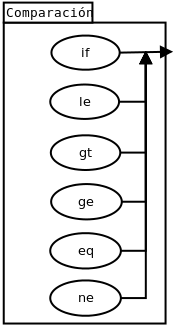
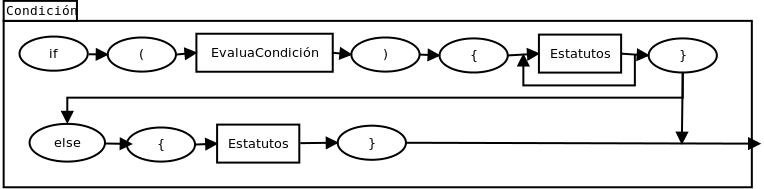
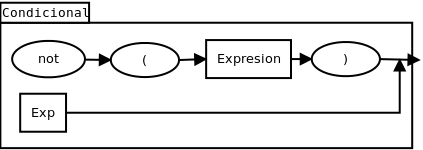
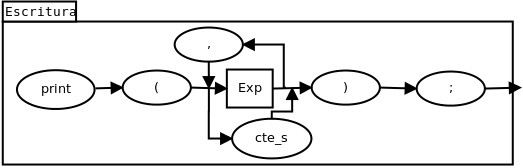
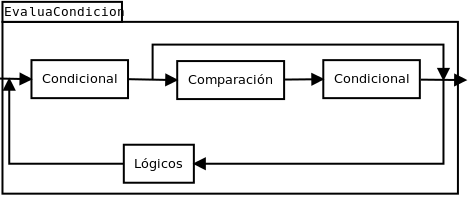
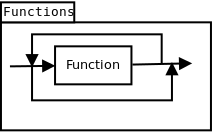
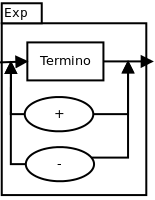
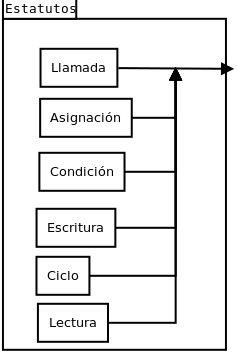
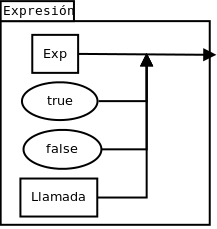
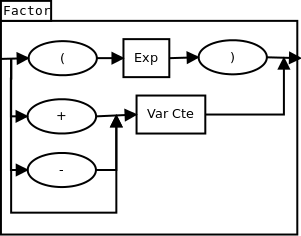
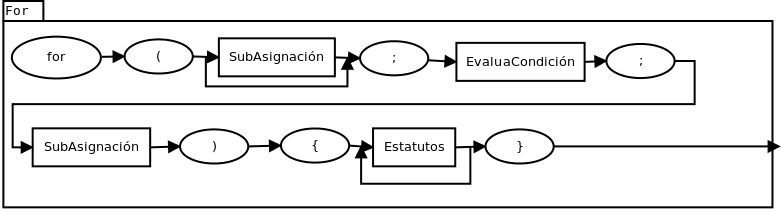
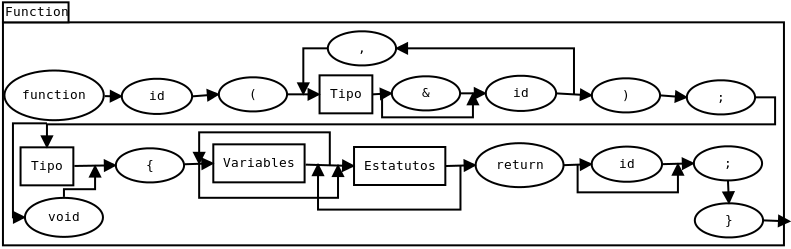
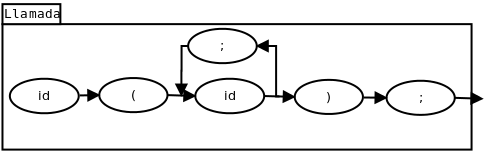
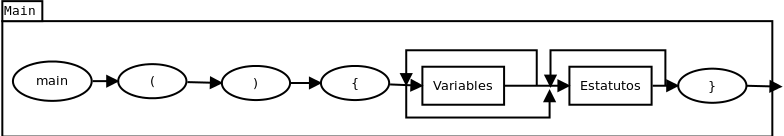
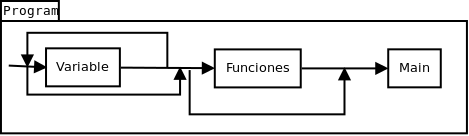
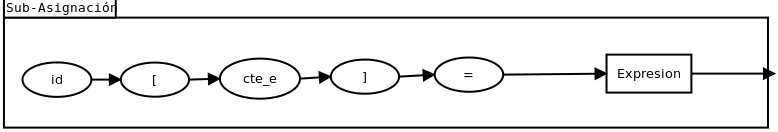
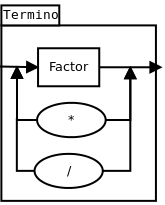
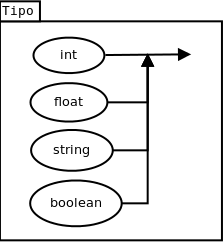
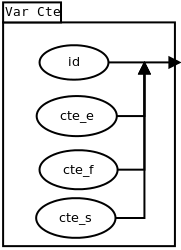
**Enteros**: [0-9]+

**Flotantes**: [0-9]+"."[0-9]+

**String**: \"(\\.|[^\\"])\*\"

**Identificadores**: [a-zA-Z][a-zA-Z0-9\_]{29}

# Diagramas de sintaxis



# 

# Principales características semánticas

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **OP. 1** | **OP. 2** | **+** | **-** | **\*** | **/** | **lt, gt,**  **le, ge, eq, ne** | **and** | **or** |
| **int** | **int** | int | int | int | int | bool | bool | bool |
| **int** | **float** | float | float | float | float | bool | bool | bool |
| **int** | **bool** | x | x | x | x | x | bool | bool |
| **int** | **string** | x | x | x | x | x | x | x |
| **float** | **float** | float | float | float | float | bool | bool | bool |
| **float** | **bool** | x | x | x | x | x | bool | bool |
| **float** | **string** | x | x | x | x | x | x | x |
| **bool** | **bool** | x | x | x | x | x | bool | bool |
| **bool** | **string** | x | x | x | x | x | x | x |
| **string** | **string** | x | x | x | x | x | x | x |

|  |  |
| --- | --- |
| **Op** | **not** |
| **int** | bool |
| **flot** | bool |
| **bool** | bool |
| **string** | x |

# Descripción de funciones especiales del lenguaje

Dado que el alcance del lenguaje es básico, se proporcionarán dos funciones para entrada y salida de datos, las cuales son:

**input**: Esta función sirve para obtener el valor de entrada para una sola variable. Su forma de uso será:

input(*tipo*, *variable*);

Donde *tipo* corresponde a los tipos básicos del lenguaje: int (entero), float (flotante), boolean (booleano, tomando 0 como falso y cualquier otro valor como verdadero y convirtiéndolo a 1) y string (cadenas de texto, utilizando espacio, tabulador o cambio de línea como delimitadores).

**print**: Esta función sirve para imprimir el valor de un string o una variable, ya sea entero, flotante o booleano (en caso de ser verdadero, imprimirá 1; de lo contrario 0). Su forma de uso será:

print(*variable*[*,variable*]);

Donde *variable* es un identificador o una cadena de texto. En caso de imprimir más variables o cadenas de texto, estas se separarán con una coma (,).

## Limitantes:

* Se podrán declarar variables globales y locales. En caso de que haya conflicto de identificadores, se escogerá siempre la variable más interna (local).
* No se puede declarar dos veces el mismo identificador de variable en el mismo nivel (global o local).
* El lenguaje podrá ser ejecutado sobre navegadores.
* El usuario utilizará elementos gráficos para agregar los elementos del lenguaje al área de trabajo (esto es, declaraciones, ciclos, funciones, condicionales, etc). De ser necesario, se podrá completar la información restante por medio del teclado.
* En caso de requerir entrada de datos, el usuario debe proporcionarlos en una sección de la página web dedicada para ello.
* La salida de la ejecución se mostrará en una sección de la página web dedicada para ello. Se imprimirán números (enteros o flotantes), booleanos (0 o 1) y cadenas de texto.

**CÓDIGO EJEMPLO**

int rows = 5;

int contador = 0;

array boolean flags : rows;

string mensaje = "Hola mundo";

function inicializa\_array() : void

{

int iterador;

for (iterador = 0; iterador < rows; iterador = iterador + 1) {

flags[iterador] = 1;

}

}

function numero\_impresiones() : int

{

contador = contador + 1;

print(mensaje);

return contador;

}

main()

{

int j;

int num;

print("Inicializando array.\n");

inicializa\_array();

for (j = 0; j < rows; j = j + 1) {

num = numero\_impresiones();

if (num eq 2) {

j = rows;

} else {

flags[j] = 0;

}

}

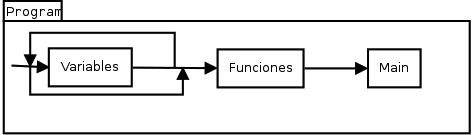
print("Fin de programa.\n");

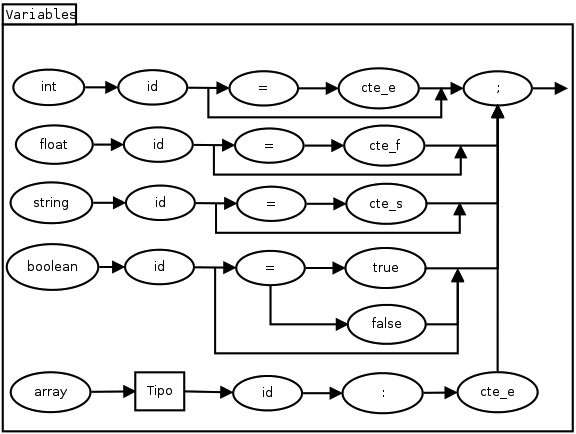
}

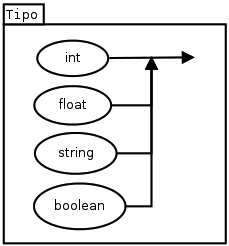
# PLATAFORMA DE DESARROLLO

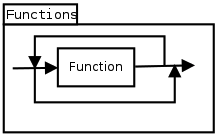
Se utilizarán plataformas Unix para el desarrollo del compilador. Éste se realizará con los lenguajes de programación C y C++, apoyados con las herramientas de Flex y Yacc. En cuanto a la codificación de los programas, se utilizarán Html, CSS, Javascript y jQuery para los bloques gráficos y el área de trabajo; y Javascript para el proceso de interpretación.

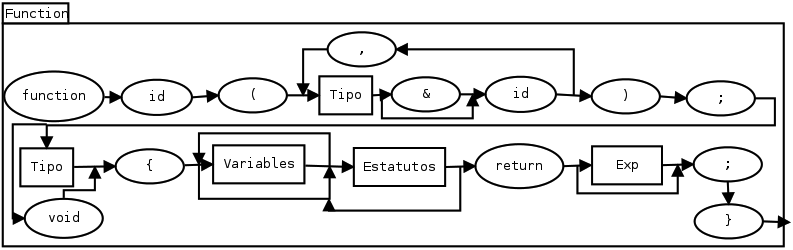
**Corrección de los diagramas de sintaxis**

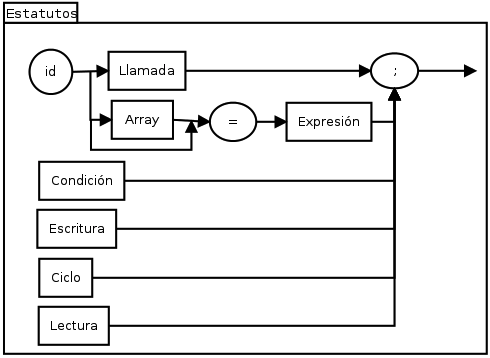


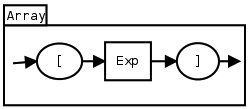


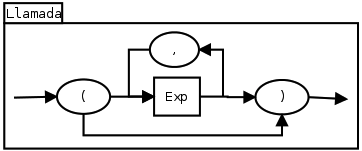


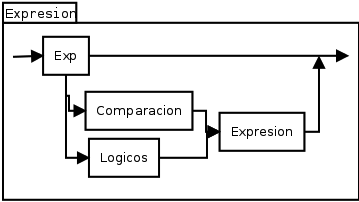


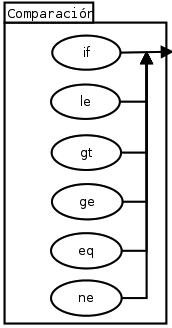


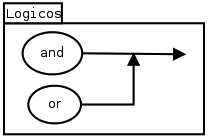


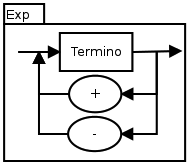


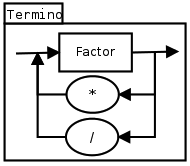


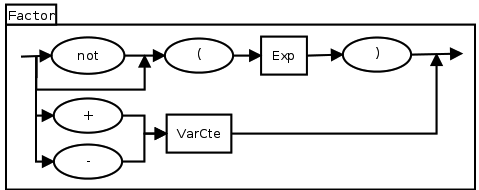


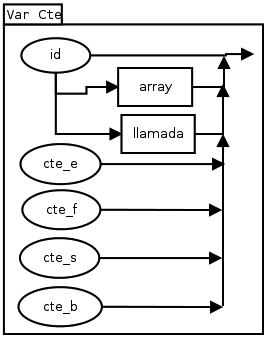


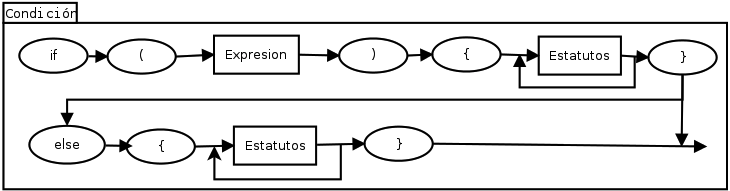


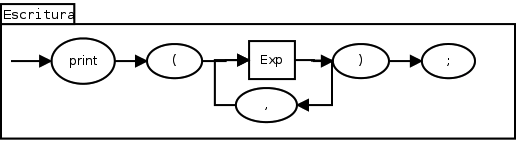


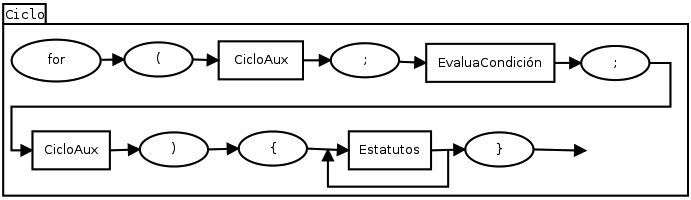


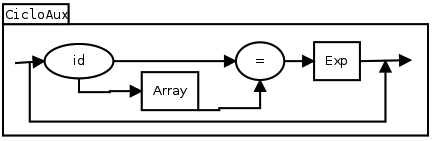


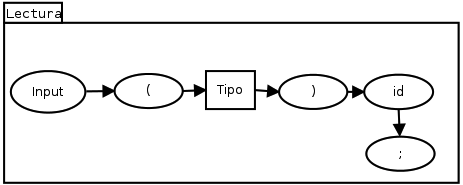


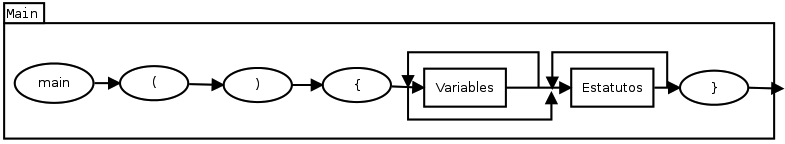












**Plataforma de desarrollo**

Dado que la intención del proyecto es la de ejecutarse en navegadores web, se optó por cambiar el lenguaje de programación para el compilador de C++ a Ruby, específicamente utilizando para ello la herramienta *Antlr4ruby* (Antlr es un generador de parsers escrito en Java, pero cuenta con el soporte para varios lenguajes de programación).