**a). DESCRIPCION DEL PROYECTO**

**a.1) Visión, Objetivos y Alcance del Proyecto.**

**1. Misión**

Incentivar a que los jóvenes se involucren al área de desarrollo de software mientras aprenden del cuidado animal por medio de un lenguaje de programación sencillo creado específicamente para ellos.

**2. Visión**

Convertirse en el primer contacto para la educación en los lenguajes de programación.

**3. Objetivo**

Satisfacer las necesidades, cuidados y actividades de una mascota por medio de órdenes programadas. Con un área de aplicación educativa y formativa, en la cual alumnos de secundaria aprendan fundamentos básicos de la lógica de programación de manera divertida mientras se sensibilizan del cuidado de los animales.

**a.2) Análisis de Requerimientos y Casos de Uso generales.**

**1. Requerimientos de alto nivel**

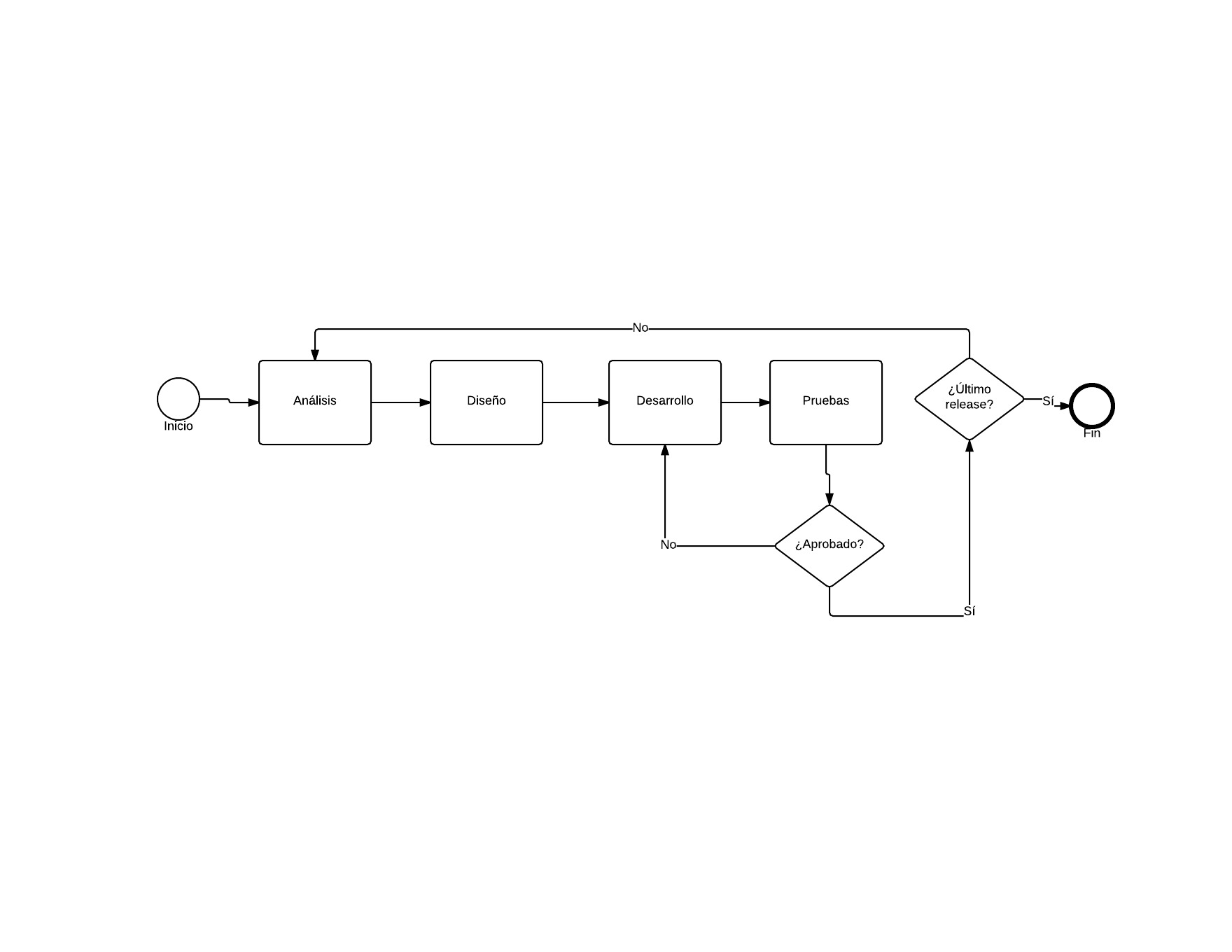
**a) Requerimientos Funcionales**

a.3) Descripción de los principales Test Cases.

|  |  |  |  |
| --- | --- | --- | --- |
| Número | Nombre | Flujo esperado | Salida |
| 1 | Asignación de variables | Se logra asignar una variable local, una variable global, una lista y una variable temporal a las direcciones en memoria correspondientes. |  |
| 2 | Operaciones aritméticas | Una operación aritmética con diferentes operadores y operando realiza las operaciones correctas |  |
| 3 | Expresiones | Las expresiones >, <, == si hacen las comparaciones de forma correcta. |  |
| 4 | Condición | Se ejecuta o no se ejecuta de forma correcta el código dentro de una condición. |  |
| 5 | Ciclo | Se puede manejar entrar y salir a un ciclo. |  |
| 6 | Manejo de funciones | Se pueden llamar diferentes funciones creadas en el código y se ejecutan en el orden correcto. |  |
| 7 | Recursividad | Funcionan correctamente algoritmos recursivos como Fibonacci y/o el factorial. |  |
| 8 | Agregar/eliminar elementos de una lista encadenada | Funciona correctamente el agregar/eliminar elementos de una lista. |  |
| 9 | Ordenar elementos de una lista encadenada. | Se pueden ordenar los elementos de una lista. |  |
| 10 | Buscar un elemento en una lista encadenada. | Se pueden buscar los elementos de una lista. |  |
| 11 | Unir dos listas encadenadas en una | Se pueden manipular las listas para encadenar listas y mantenerlas ordenadas. |  |
| 12 | No permite sumar elementos de diferente tipo | Marca un error al querer sumar un int y un float |  |
| 13 | Generar la respuesta en el tipo correcto para una operación. | Por ejemplo si se multiplican dos ints genera un int. |  |
| 14 | Si se durante la ejecución se cumple la condición de salida, sale del while | Se sale del while al cumplir condición de salida |  |
| 15 | Los if anidados funcionan correctamente | Al generar un if dentro de otro if funciona correctamente. |  |
| 16 |  |  |  |
| 17 |  |  |  |
| 18 |  |  |  |
| 19 |  |  |  |
| 20 |  |  |  |
| 21 |  |  |  |
| 22 |  |  |  |
| 23 |  |  |  |
| 24 |  |  |  |
| 25 |  |  |  |
| 26 |  |  |  |
| 27 |  |  |  |
| 28 |  |  |  |

a.4) Descripción del PROCESO general seguido para el desarrollo del proyecto, incluyendo Bitácoras generales y un pequeño párrafo de reflexión de cada alumno, en relación a los principales aprendizajes logrados (firmarlo).

El proceso de desarrollo de software utilizado fue incremental en el cual se realizaron las etapas de desarrollo de software por “releases” planteados para la herramienta:



**\***Las bitácoras se adjuntan al presente documento en un anexo

Reflexión Guillermo Elizondo: En este proyecto más allá del aprendizaje obtenido de compiladores aprendí mucho sobre cómo organizarme, cómo administrar un proyecto correctamente para llevarlo a su éxito. Lo anterior se logró con equipo increíble, mi compañero Eduardo. Este proyecto es muy grande y me recordó muchos momentos en lo que me dedicaba a programar lo cual fueron muy buenos recuerdos. También el hecho de entender una lógica de un lenguaje de programación abre la mente a poder hacer más y diferentes desarrollos fuera de lo tradicional y así cumplir con nuestra misión de impactar al mundo.

Reflexión Eduardo Bañuelos:

**b). DESCRIPCION DEL LENGUAJE:**

**b.1) Nombre del lenguaje**

Cat

b.2) Descripción genérica de las principales características del lenguaje (en forma narrativa).

El lenguaje cat es un lenguaje procedural que permite generar estatus lógicos que llevan a una salida gráfica. Tiene un tipo de dato estructurado que son las listas. La salida visual permite mover a un pequeño gato en la pantalla y realizar diferentes acciones con él como crecerlo de tamaño o replicarlo. El lenguaje maneja funciones y debe tener un “main” para inicializar la ejecución. Además cat maneja variables de tipo integral, string o char, de punto flotante y booleanas. Permite tener un tipo de ciclo “while” así como manejar condiciones.

b.3) Descripción de los errores que pueden ocurrir, tanto en compilación como en ejecución.

Durante la compilación se pueden tener tres tipos de errores:

Errores léxicos: No se escribió correctamente algún estatuto predefinido. Los errores léxicos que se manejan en el lenguaje son:

* Errores de creación de identificadores
* Errores en palabras reservadas
* Errores en operadores

Errores sintácticos: No se maneja el orden o la lógica correcta de las instrucciones. Los errores sintácticos que se manejan son:

* Error de orden de operaciones
* Errores de gramática incorrecta

Errores semánticos: Los errores semánticos que se manejan son:

* Error de asignación de tipos de datos

Los errores de ejecución que se manejan son:

* Errores de precisión en punto flotante
* Errores de tendencia a infinito
* Errores de sobrepaso de memoria temporal

**c). DESCRIPCION DEL COMPILADOR:**

**c.1) Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto.**

Se utilizaron dos computadoras: una PC con procesador Intel i5 y una computadora Mac Pro. Como lenguaje base se utilizó Python con la utilería py que maneja lex y yacc para el léxico y sintaxis. Para el “output” gráfico se utiliza la librería Tkinter de python en el que toma lo generado por la máquina virtual y se ejecuta visualmente.

c.2) Descripción del Análisis de Léxico. Debe incluir:

**Enumeración de los "tokens" del lenguaje y su código asociado.**

**Constantes**

Constante int: cte.i

Constante float: cte.f

Constante String: cte.String

**Tipos de datos**

int: [0-9]\*

float: [0-9]\*.[0-9]\*

string: [a-zA-Z0-9\_]\*

**Palabras reservadas**

var: variable

int: Integer

float: Float

if: if

else: else

print: print

while: while

read: El Sistema lee datos.

Eat: hace que el gato coma.

fish: comida del gato.

Clean: hace que el gato limpie.

poop: desecho de gato.

pPlay: hacer que el gato juegue.

ball: pelota del gato.

bag: lista de objetos del gato.

Add: agregar a la lista.

move: hace que el gato se mueva 1 posición en la dirección que esta viendo.

turnLeft: gira la dirección del gato a la izquierda.

turnRight: gira la dirección del gato a la derecha.

**Caracteres de los Delimitadores**

:

.

=

;

(

)

{

}

<| > | >=| >= | == | =!

**Caracteres de Operadores**

+

-

\*

/

**Código asociado**

t\_COLON = r':'

t\_POINT = r'\.'

t\_EQUAL = r'='

t\_SEMICOLON = r':'

t\_COMA = r','

t\_LPAR = r'\('

t\_RPAR = r'\)'

t\_LCBRACKET = r'\{'

t\_RCBRACKET = r'\}'

t\_LBRACKET = r'\['

t\_RBRACKET = r'\]'

t\_COMPARISON = r'<=|>=|<|>|==|!=|&&|\|\|'

t\_MULTIPLY = r'\\*'

t\_DIVIDE = r'/'

t\_PLUS = r'\+'

t\_MINUS = r'-'

t\_ignore = " \t"

def t\_NUMFLOAT(t):

r'-?\d+\.\d\*(e-?\d+)?'

t.value = float(t.value)

return t

def t\_NUMINT(t):

r'\d+'

t.value = int(t.value)

return t

def t\_BOOLEANTYPE(t):

r'true|false'

t.value = bool(t.value)

return t

def t\_ID(t):

r'[a-zA-Z\_][a-zA-Z0-9\_]\*'

if t.value in reserved:

t.type = reserved.get(t.value,'ID')

#print t

return t

def t\_newline(t):

r'\n+'

t.lexer.lineno += t.value.count("\n")

def t\_error(t):

print("Illegal character '%s'" % t.value[0])

t.lexer.skip(1)

c.3) Descripción del Análisis de Sintaxis. Debe incluir:

o Gramática Formal empleada para representar las estructuras sintácticas (Sin “codificar”).

<VarCte>->id|cte i|cte f

<Exp>-> <Termino>

<Exp>-> <Termino>+

<Exp>-> <Termino>-

<Termino>-> <Factor>

<Termino> -> <Factor>\*

<Termino>-> <Factor>/

<Factor>-> {<Expresion>}

<Factor>-> {+<varcte>}

<Factor>-> {-<varcte>}

<Factor>-> {<varcte>}

<Expresion>-> <Exp>

<Expresion>-> <Exp> >

<Expresion>-> <Exp> <

<Expresion>-> <Exp> >=

<Expresion>-> <Exp> <=

<Expresion>-> <Exp> ==

<Tipo>-> int

<Tipo>-> float

<Asing>->id=<expresion>

<Dec> <tipo> id

<Estatuto> -> <Asign>

<Estatuto> -> <Impresion>

<Estatuto> -> <Condicion>

<S>-> <Estatuto><S>

<S>-> <Estatuto>

<S>-> Vacío

<Bloque>-> { <S>}

<func>->func id(){<block>}

<func>->func id (dec){<block>}

<condicion>->if(<expression>){<block>}

<condicion>->if(<expression>){<block>}else{<block>}

<cycle>-> while(<expression>){<block>}

<impresion>-> print (<exp>)

<list>->id=[<w>]

<w>->id<w>

<w>->cte.int<w>

<w>->id

<w>->cte.int

<llamada>->id (dec)

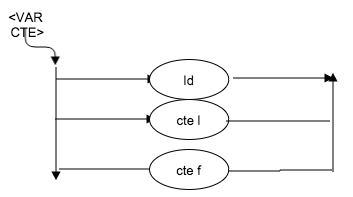
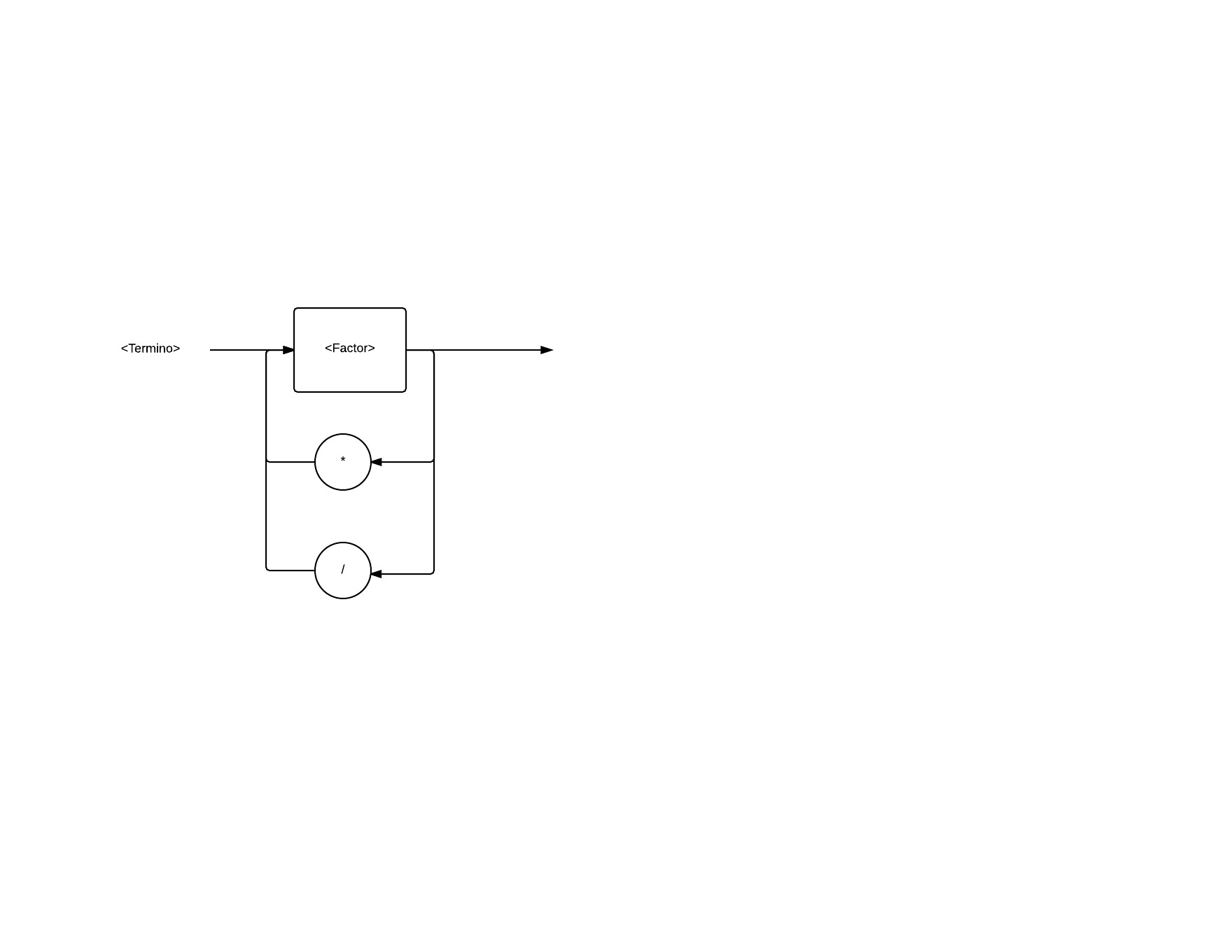
<llamada>-> id()

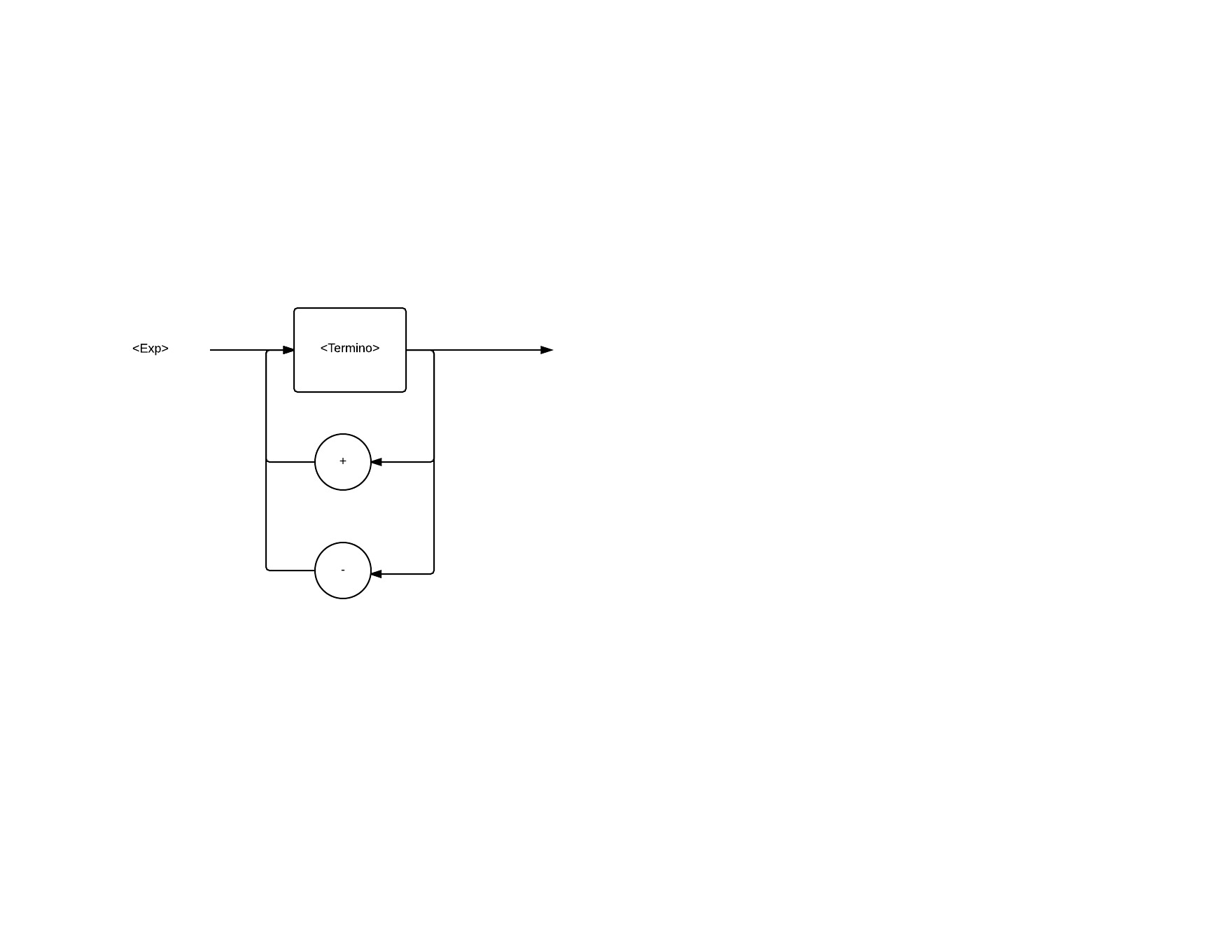
<add>->id.add(id)

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

o Código de operación y direcciones virtuales asociadas a los elementos del código.

o Diagramas de Sintaxis con las acciones correspondientes.



o Breve descripción de cada una de las acciones semánticas y de código (no más de 2 líneas).

o Tabla de consideraciones semánticas.

c.5) Descripción detallada del proceso de Administración de Memoria usado en la compilación.

o Especificación gráfica de CADA estructura de datos usada (Dir.Proc, Var's, Cuádruplos, etc...)

d). DESCRIPCION DE LA MÁQUINA VIRTUAL:

d.1) Equipo de cómputo, lenguaje y utilerías especiales usadas (en caso de estar en diferente que el compilador).

Se utilizó exactamente lo mismo que para el compilador. Sólo se utilizó la librería Tkinter de python en el que toma lo generado por la máquina virtual y se ejecuta visualmente.

d.2) Descripción detallada del proceso de Administración de Memoria en ejecución (Arquitectura). Incluir:

o Especificación gráfica de CADA estructura de datos usada (Memoria Local, global, etc..)

o Asociación hecha entre las direcciones virtuales (compliación) y las reales (ejecución).

e). PRUEBAS DEL FUNCIONAMIENTO DEL LENGUAJE :

e.1) Incluir pruebas que "comprueben” el funcionamiento del proyecto:

o Codificación de la prueba (en su lenguaje).

o Resultados arrojados por la generación de código intermedio y por la ejecución.

f). LISTADOS PERFECTAMENTE DOCUMENTADOS DEL PROYECTO:

f.1) Incluir comentarios de Documentación, es decir: para cada módulo, una pequeña explicación de qué hace, qué

parámetros recibe, qué genera como salida y cuáles son los módulos más importantes que hacen uso de él.

f.2) Dentro de los módulos principales se esperan comentarios de Implementación, es decir: pequeña descripción de

cuál es la función de algún estatuto que sea importante de ese módulo.