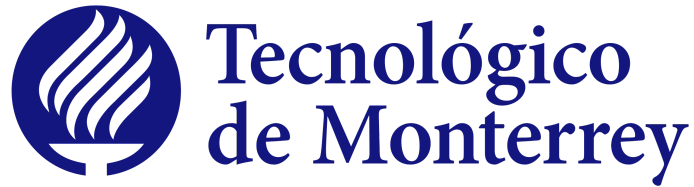


**Instituto Tecnológico y de Estudios
Superiores de Monterrey**
Campus Monterrey



Diseño de compiladores
R--

Eloy Hernández Lúa
A01066325

Elda Guadalupe Quiroga González
Héctor Gibrán Ceballos Cancino

Noviembre 2021

Descripción del proyecto	3
Descripción del lenguaje	5
Descripción del compilador	6
Tokens	6
Patrones de Construcción (regex)	7
Diagramas de Sintaxis	10
Consideraciones semánticas	14
Descripción de la máquina virtual	19
Estructura y justificación	19
Asociación hecha con las direcciones virtuales	20
Pruebas del funcionamiento del lenguaje	20

Descripción y documentación técnica

Descripción del proyecto

Propósito y Alcance del Proyecto

El propósito principal de este proyecto es tener un primer acercamiento a lo que es el diseño e implementación de un compilador, abarcando cuestiones como su construcción léxica, semántica, fabricación de código intermedio e implementación de una máquina virtual que interprete y ejecute este código.

En particular, este proyecto busca cubrir los elementos básicos de un lenguaje de programación de uso general con el valor agregado de tener funcionalidad de un lenguaje especializado en estadística.

Análisis de Requerimientos y Test Cases

Dentro de los requerimientos del proyecto se encuentran los siguientes:

- Manejo de declaración de variables en un scope local y global
- Manejo de estatuto de asignación
- Manejo de estatuto de llamada a funciones void y con valores de retorno
- Manejo de estatuto de retorno
- Manejo de estatuto de lectura y escritura
- Manejo de estatutos de decisión condicionales (if, if-else)
- Manejo de estatutos de repetición condicionales y no condicionales (while, for)
- Manejo de expresiones aritméticas, lógicas y relacionales
- Implementación de funciones especiales propias del lenguaje
- Manejo de variables unidimensionales (arreglos)
- Manejo de llamadas recursivas en funciones

Dentro de los casos de prueba (anexos al final del documento y en el repositorio) se evalúa la funcionalidad de los requerimientos ya establecidos.

Descripción del proceso general

Para llevar a cabo el proyecto, se trabajó de manera exhaustiva por un plazo de aproximadamente 8 semanas, se llevó el control de versiones a través de [GitHub](#) donde se tuvo 50+ commits con descripciones detalladas del avance. Dentro del mismo repositorio se desglosa en el archivo [Readme.md](#) el contenido de los avances clave del proyecto.

github.com/eloyhernandezlua/miRLike/commits/master

Issues Pull requests Actions Projects 1 Wiki Security Insights Settings

master

Commits on Nov 23, 2021

- [FINAL] - documentacion de codigo
eloyhernandezlua committed 20 hours ago
5d93368
- [FINAL]
eloyhernandezlua committed 22 hours ago
c42f604
- FINAL tests and special funcs
eloyhernandezlua committed 2 days ago
4c36cf8

Commits on Nov 21, 2021

- [7th ADVANCE]
eloyhernandezlua committed 3 days ago
38c4022
- Merge branch 'master' of https://github.com/eloyhernandezlua/miRLike
eloyhernandezlua committed 3 days ago
a414b04
- [7th ADVANCE]
eloyhernandezlua committed 3 days ago
3d15ee2
- diagramas cool en limpio
eloyhernandezlua committed 3 days ago
Verified 01b39c4
- FIBONACCI Y FACTORIAL JALANDO ITERATIVO Y RECURSIVO
eloyhernandezlua committed 3 days ago
241dc80
- FIBONACCI
eloyhernandezlua committed 3 days ago
0c3498d
- Manejo de arreglos, algunos bugs
eloyhernandezlua committed 3 days ago
d414ff6

Resto de los commits disponibles [aquí](#)

Reflexión

Considero de manera personal que es un proyecto altamente demandante y definitivamente en primeras instancias muy abstracto, al menos yo tardé más de lo que hubiese querido en poder aterrizar los conceptos en líneas de código que no solo hicieran sentido para ese nuevo concepto, si no que tuvieran una propia integración con lo pasado y que propiciarán lo mismo para lo siguiente.

Como recomendación, para personas al menos con las mismas características y aptitudes que yo, sería mucho más sencillo poder llevar en paralelo el avance del parser y la máquina virtual, esto para poder tener una retroalimentación más pronta de que significan los cuádruplos o manera de código intermedio que se haya escogido y que vaya haciendo un poco más de sentido el porqué de las cosas, creo que yo lo entendí mucho más y terminé de comprender cosas que ya había hecho hasta que las vi funcionar en la máquina virtual, esto obviamente llevó a muchas correcciones y mejoras que no tenía presentes por esta misma falta de aterrizaje del concepto.

Descripción del lenguaje

Nombre del lenguaje

R -- : Esto por falta de originalidad del desarrollador y por el intento de homenaje que se tuvo hacia el lenguaje con características estadísticas: R.

Características del lenguaje

El lenguaje cuenta con las funcionalidades básicas de cualquier lenguaje de programación, en este caso, el lenguaje es fuertemente tipado, esto ya que requiere de especificar de manera explícita el tipo de dato a manejar en cada momento, con declaraciones inmutables en el código. El lenguaje también tendrá incluido en sus estatutos base, varias funcionalidades para cálculos estadísticos.

Listado de los errores

- "FUNCION EXISTENTE REPETIDA"
- "ID DE PROGRAMA O VARIABLE REPETIDO"
- "INVALID OPERATION, TYPE MISMATCH"
- "VARIABLE PREVIAMENTE DECLARADA"
- "TIPO DE DATO NO ACEPTADO"
- "TYPE MISMATCH"
- "NO EXISTE"
- "VARIABLE SIN VALOR"
- "VARIABLE SIN TIPO"
- "ERROR EN TIPO DE DATO"
- "INVALID OPERATION"
- "FUNCTION EXPECTED NO PARAMETERS"
- "MISSING OR TOO MUCH PARAMETERS"
- "VAR HAS NO DIMENSIONS"
- "ONLY NUMBERS ARE ALLOWED"

- "Unexisting value"
- "Trying to use none values"
- "INDEX OUT OF BOUNDS"
- "NOT A SINGLE CHAR"

Descripción del compilador

Equipo de cómputo, lenguaje y utilerías especiales usadas en el desarrollo del proyecto

Para este proyecto se utilizó una Mac Pro (2013), Python como lenguaje de programación y librerías: *PLY*, *os*, *re* y *sys*.

Descripción del Análisis de Léxico

Tokens

- ID
- PTCOMA
- APAR
- CPAR
- ALLA
- CLLA
- COMA
- ACOR
- CCOR
- IGUAL
- DOSPNTS
- STRING
- MAS
- MENOS
- DIV
- POR
- CTEI
- CTEF
- CTEC
- MAYQ
- MENQ
- MAYI
- MENI
- IGUALIGUAL
- DIF
- NOT
- 'Program' : 'PROGRAM',
- 'VARS' : 'VARS',
- 'main' : 'MAIN',
- 'function' : 'FUNCTION',
- 'void' : 'VOID',
- 'int' : 'INT',
- 'float' : 'FLOAT',
- 'char' : 'CHAR',
- 'str' : 'STR',

- 'bool' : 'BOOL',
- 'return' : 'RETURN',
- 'read' : 'READ',
- 'write' : 'WRITE',
- 'if' : 'IF',
- 'true' : 'TRUE',
- 'false' : 'FALSE',
- 'then' : 'THEN',
- 'else' : 'ELSE',
- 'while' : 'WHILE',
- 'do' : 'DO',
- 'for' : 'FOR',
- 'to' : 'TO',
- 'and' : 'AND',
- 'or' : 'OR',
- 'media' : 'MEDIA'

Patrones de Construcción (regex)

PTCOMA = r'\;'

APAR = r'\('

CPAR = r'\)'

ALLA = r'\{'

CLLA = r'\}'

COMA = r'\,'

ACOR = r'\['

CCOR = r'\]'

IGUAL = r'\='

DOSPNTS = r'\.'

STRING = r'\".*\"'

MAS = r'\+'

MENOS = r'\-'

DIV = r'\V'

POR = r'*'

CTEC = r'\".\\"'

MAYQ = r'\>'

MENQ = r'\<'

MAYI = r'\>|='

MENI = r'\<|='

IGUALIGUAL = r'\|=|'

DIF = r'\!|='

NOT = r'\!'

CTEF(t) = r'-?\d+\.\d+'

CTEI(t) = r'-?\d+'

ID(t) = r'[A-Za-z]([A-Za-z][0-9])*

Descripción del Análisis de Sintaxis

program \rightarrow PROGRAM varss funcns MAIN APAR CPAR ALLA estatutos CLLA

varss \rightarrow VARS vars

vars \rightarrow tipo DOSPNTS ID varspvp varspv vars

vars \rightarrow empty

varspvp \rightarrow ACOR CTEI CCOR

varspvp \rightarrow empty

varspv \rightarrow PTCOMA

varspv \rightarrow COMA ID varspvp varspv

funcns \rightarrow FUNCTION funcsp ID funcspv

funcns \rightarrow empty

funcspv \rightarrow APAR params CPAR PTCOMA varss ALLA estatutos CLLA funcns

funcsp \rightarrow VOID

funcsp \rightarrow tipo

estatutos \rightarrow asig estatutop

estatutos \rightarrow return estatutop

estatutos \rightarrow lectura estatutop

estatutos \rightarrow escritura estatutop

estatutos \rightarrow cond estatutop

estatutos \rightarrow while estatutop

estatutos \rightarrow for estatutop

estatutos \rightarrow exp estatutop

tipo \rightarrow INT

tipo \rightarrow FLOAT

tipo \rightarrow CHAR

params \rightarrow tipo DOSPNTS ID ididx paramsp

paramsp \rightarrow COMA params

paramsp \rightarrow empty

asig \rightarrow ID ididx IGUAL asigvp PTCOMA

asigv \rightarrow exp asigvpv

asigv \rightarrow empty

asigppp \rightarrow COMA asigp

asigppp \rightarrow empty

asigpp \rightarrow exp

ididx \rightarrow ACOR exp CCOR

return \rightarrow RETURN APAR exp CPAR PTCOMA

lectura \rightarrow READ APAR ID ididx lecturapp CPAR PTCOMA

escritura \rightarrow WRITE APAR escriturap escriturapp CPAR PTCOMA

escriturap \rightarrow pushEsc

escriturap \rightarrow exp

pushEsc \rightarrow STRING

pushESc \rightarrow CTEC

escriturapp \rightarrow COMA escriturap escriturapp

escriturapp \rightarrow empty

cond \rightarrow IF APAR exp CPAR THEN ALLA estatutos CLLA condpp

condpp \rightarrow ELSE ALLA estatutos CLLA

condpp \rightarrow empty

while \rightarrow WHILE APAR exp CPAR DO ALLA estatutos CLLA

for \rightarrow FOR ID ididx IGUAL exp TO exp DO ALLA estatutos CLLA

exp \rightarrow texp expp

expp \rightarrow OR exp

expp \rightarrow empty

texp \rightarrow gexp texpp

texpp \rightarrow AND texp

texpp \rightarrow empty

gexp \rightarrow mexp gexpp

gexpp \rightarrow addPO mexp

gexpp \rightarrow empty

addPO \rightarrow MAYQ

addPO \rightarrow MENQ

addPO \rightarrow MAYI
addPO \rightarrow MENI
addPO \rightarrow IGUALIGUAL
addPO \rightarrow DIF

mexp \rightarrow termino mexpp

mexpp \rightarrow operSR mexp
mexpp \rightarrow empty

operSR \rightarrow MENOS | MAS

termino \rightarrow factor terminop

terminop \rightarrow oper termino

oper \rightarrow DIV | POR

factor \rightarrow APAR exp CPAR
factor \rightarrow ctes

factorp \rightarrow APAR factorParams CPAR
factorp \rightarrow ididx

factorParams \rightarrow exp factorpp

factorpp \rightarrow COMA exp factorpp
factorpp \rightarrow empty

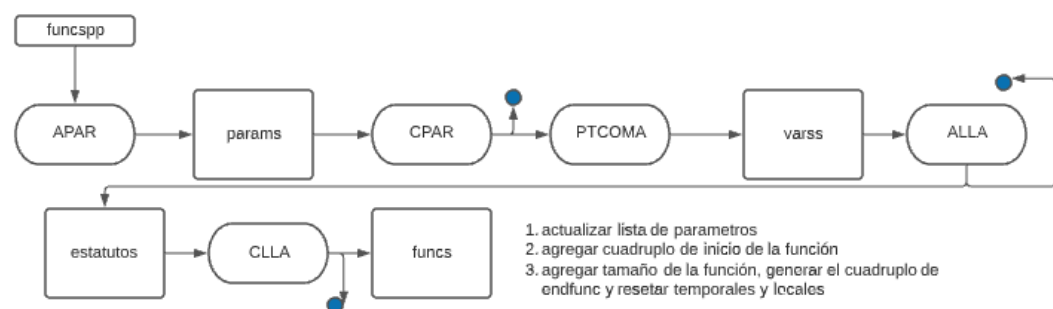
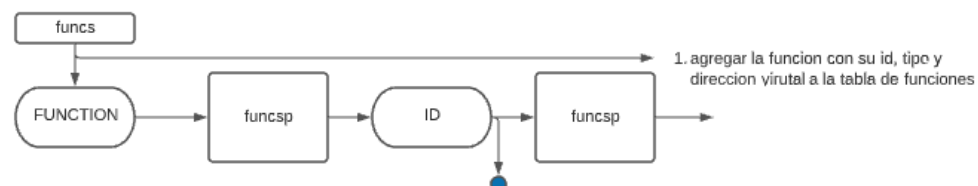
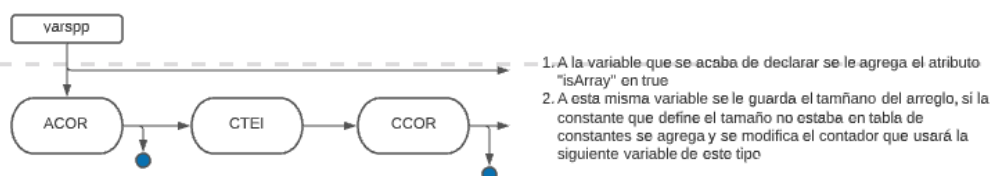
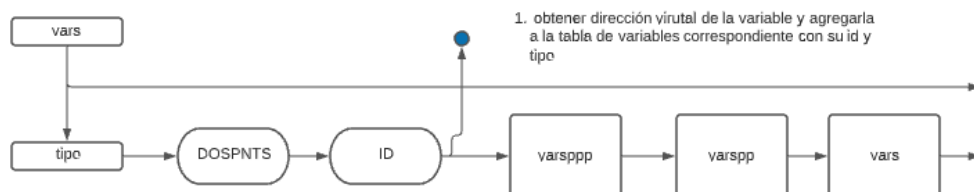
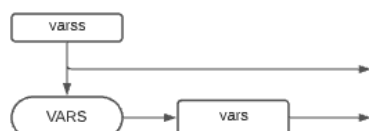
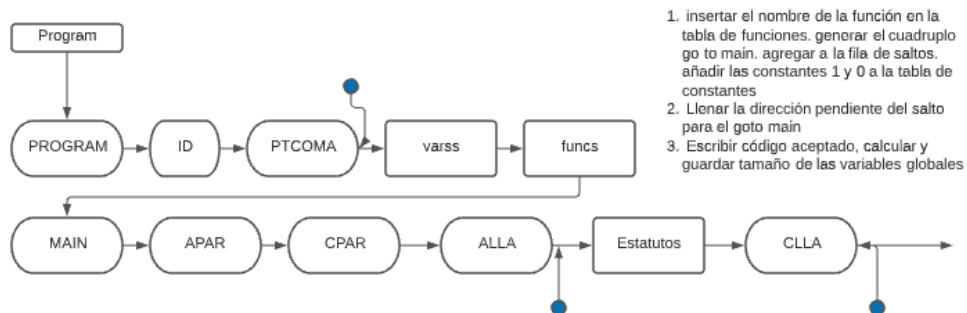
ctes \rightarrow CTEC | CTEI | CTEF
ctes \rightarrow ID factorp

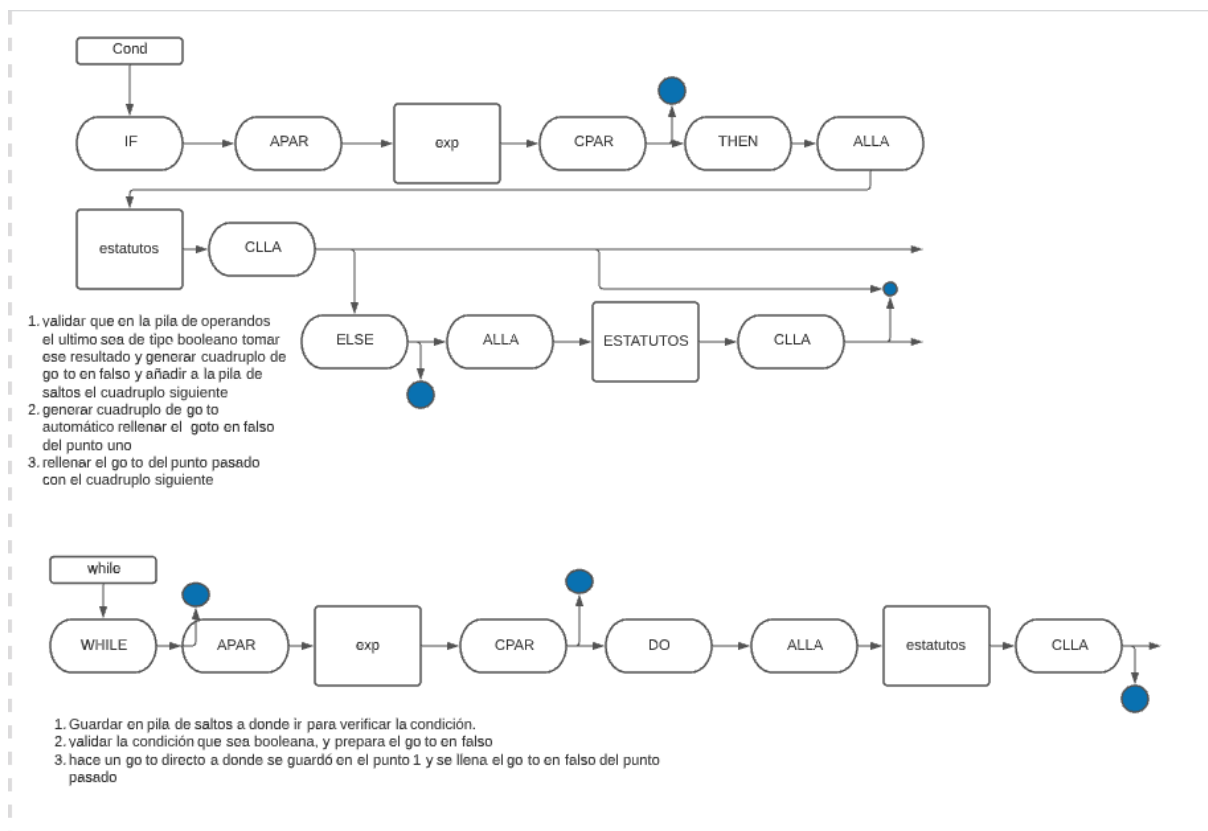
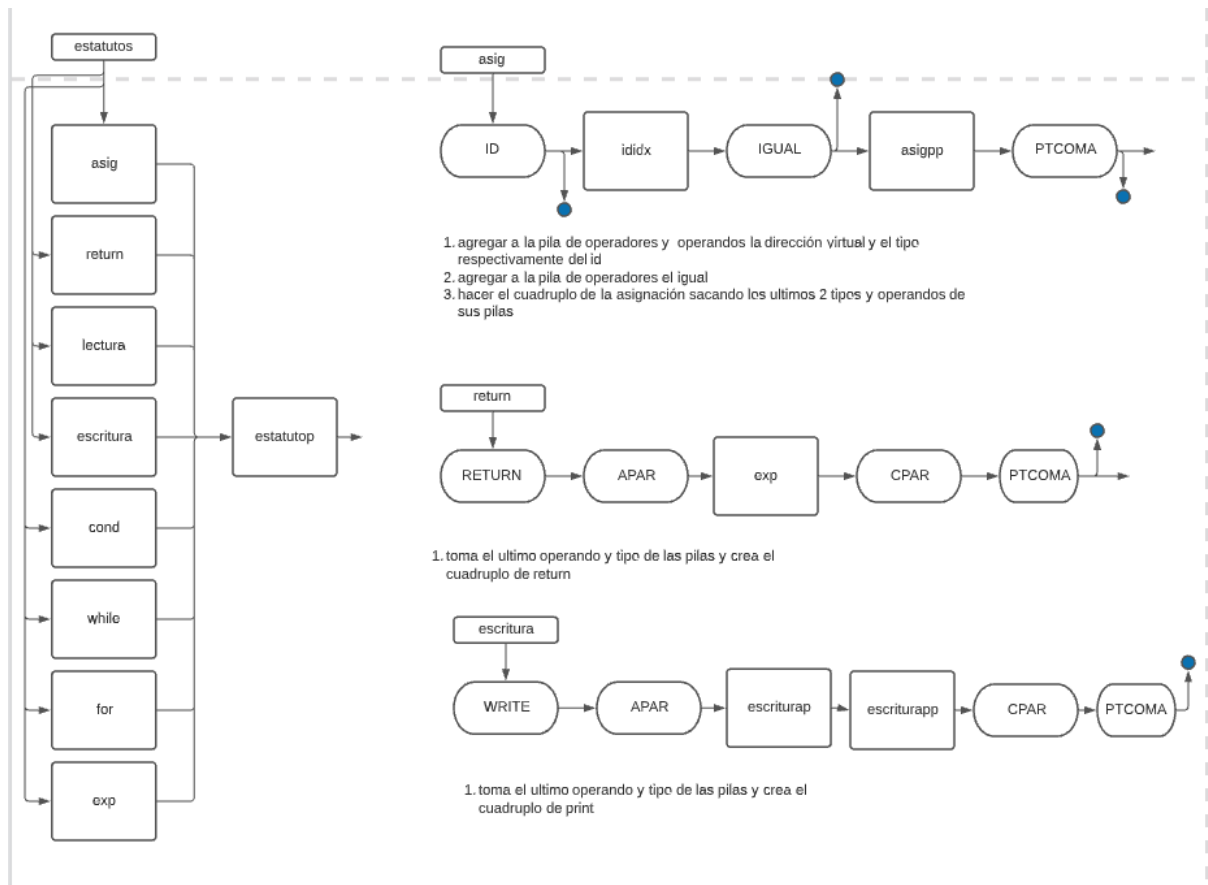
Generación de Código Intermedio y Análisis Semántico

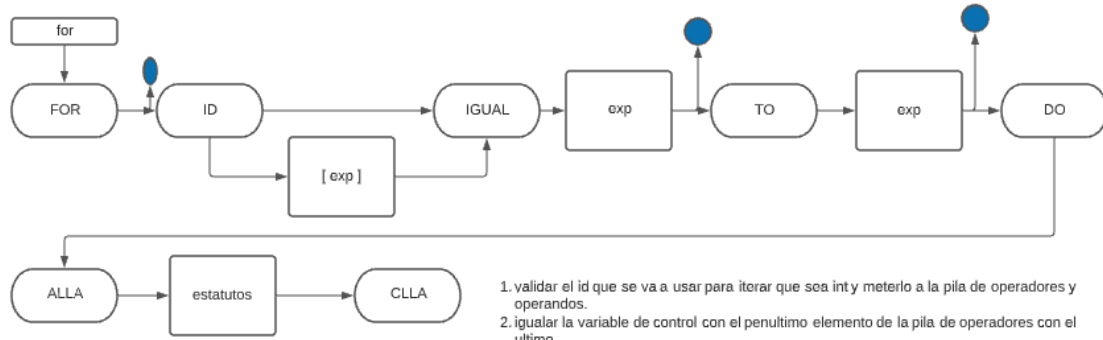
Diagramas de Sintaxis

Para la visualización de los diagramas y que no perdieran calidad se pueden ver en la siguiente liga o existe un .pdf dentro del repositorio con el código

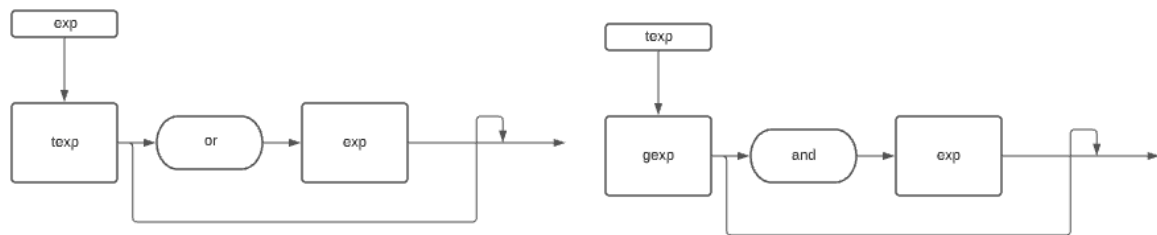
https://lucid.app/lucidchart/391b6a25-c00c-4a1b-9eb2-295fd09d37e8/edit?viewport_loc=-2840%2C-486%2C11556%2C5084%2C0_0&invitationId=inv_1513ee42-5805-44d3-a37c-c8d6705430c4

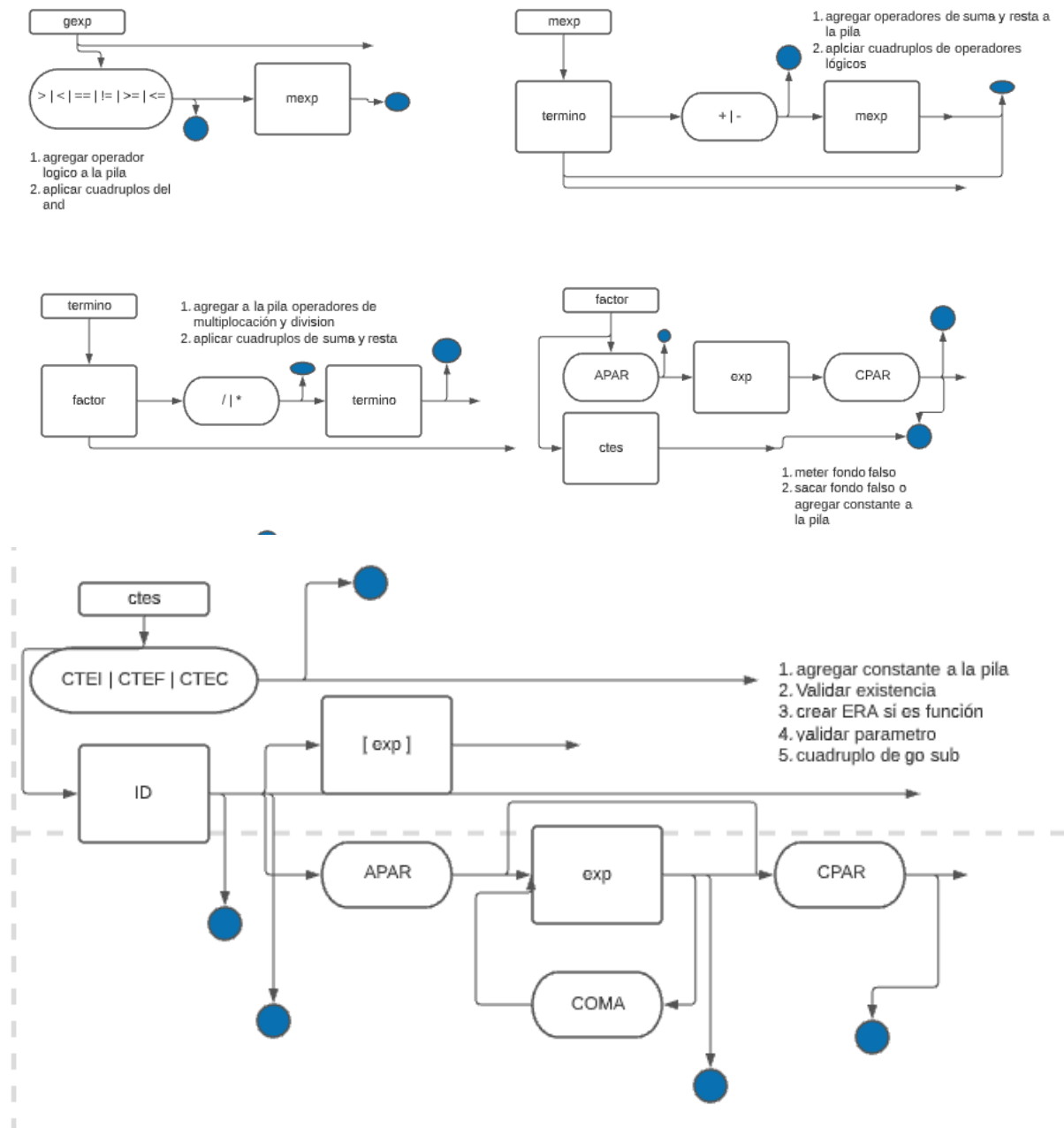






1. validar el id que se va a usar para iterar que sea int y meterlo a la pila de operadores y operandos.
2. igualar la variable de control con el penultimo elemento de la pila de operadores con el ultimo
3. hacer que la variable de control final valga lo mismo que la ultima expresion calculada, generar cuadruplo que de un bool si es menor o mayor comparado con la variable de control agregar la direccion a la pila de saltos y el cuadruplo de go to en falso





Consideraciones semánticas

Para el proyecto se realizaron las siguientes verificaciones para realizar operaciones entre dos datos, cualquier caso que no se encuentra en la siguiente lista, se considera un error de tipo `type mismatch`. De la manera en la que se construye la verificación es la concatenación de los tipos de los operandos y el operador:

```
'intint=',
'floatfloat=',
'charchar=',
'boolbool=',
'intint+',
'intint-',
```

'intint*',
'intint/',
'intint>',
'intint<',
'intint>=',
'intint<=',
'intint==',
'intint!=',
'floatint+',
'floatint-',
'floatint*',
'floatint/',
'floatint>',
'floatint<',
'floatint>=',
'floatint<=',
'floatint==',
'floatint!=',
'intfloat+',
'intfloat-',
'intfloat*',
'intfloat/',
'intfloat>',
'intfloat<',
'intfloat>=',
'intfloat<=',
'intfloat==',
'intfloat!=',
'floatfloat+',
'floatfloat-',
'floatfloat*',
'floatfloat/',
'floatfloat>',
'floatfloat<',
'floatfloat>=',
'floatfloat<=',
'floatfloat==',
'floatfloat!=',
'charchar==',
'charchar!=',
'stringstring==',
'stringstring!=',
'boolbool==',
'boolbool!=',
'boolbooland',
'boolboolor',
'boolbool>',
'boolbool<',

'boolbool>=',
'boolbool<='

Agregado a las validaciones, dentro de la misma, se retorna un valor del tipo resultante dado por lo siguiente:

Si el operador pertenecer a los operadores aritméticos:

- De ser 2 enteros, el resultado es entero
- Flotante y flotante a flotante
- Entero y flotante o flotante y entero a flotante

Si el operador es relacional o relacional por defecto se retorna un tipo booleano

La generación de cuádruplos se hace con la siguiente estructura:

```
class Cuad:
    def __init__(self, op, dir1, dir2, recep):
        global Cuadruplos
        self.count = len(Cuadruplos) + 1
        self.op = op
        self.dir1 = dir1
        self.dir2 = dir2
        self.recep = recep
```

Donde sus elementos son 5 valores **numéricos**:

- count: Lleva la cuenta de que número de cuádruplo es, eso para su mejor manejo en máquina virtual y manejo de saltos
- op: El número del operador a que hace referencia (mapa de relación número operador aquí abajo)
- dir1: una dirección de memoria virtual
- dir2: una segunda dirección de memoria virtual
- recep: la dirección virtual que recibe la acción, o en caso de cuádruplos sin interacción como return, goto's, etc. solo la dirección virtual que requieren

Mapa relacional de operadores para su uso en cuádruplos:

" : -1,
'+' : 1,
'-' : 2,
'*' : 3,
'/' : 4,
'==' : 5,
'<' : 6,
'>' : 7,
'<=' : 8,
'>=' : 9,
'!=' : 10,

'=' : 11,
'print' : 12,
'read' : 13,
'goto' : 14,
'gotoF' : 15,
'gotoT' : 16,
'return' : 17,
'and' : 18,
'or' : 19,
'endFunc' : 20,
'era' : 21,
'parameter' : 22,
'gosub' : 23,
'ver' : 24,
'media' : 25,
'mediahar' : 26,
'mediana' : 27,
'medianagroup' : 28,
'moda' : 29,
'pstdev' : 30,
'stdev' : 31,
'pvariance' : 32,
'variance' : 33,
'plotx' : 34

Administración de Memoria usado en la compilación

Dentro del proceso de compilación la memoria de direcciones virtuales es dividida en varias secciones dadas por el siguiente mapa:

#	-----	
#	globales int	2000
#	globlaes float	5000
#	globales char	9000
#	locales int	12000
#	locales float	20000
#	locales char	28000
#	temp int	30000
#	temp float	32000
#	temp char	34000
#	temp bool	36000
#	const int	38000
#	const float	40000
#	const char	43000
#	fun vars	46000
#	pointers	50000
#	-----	

Dónde *func vars* hace referencia a las variables globales que almacenan los valores de retorno de funciones y *pointers* a las variables que no almacenan datos, si no que almacenan una dirección.

Además de esto se hace uso de lo siguiente para dar estructura a los datos:

- *mainFuncTable*: Dónde se almacenan todas las funciones con su información correspondiente (nombre, tamaño, dirección de inicio, variables, etc.)
- *globalVariables*: El mapeo de las variables usadas en un scope global con su id, dirección virtual, tipo, etc.
- *localVariables*: Igual que las variables globales pero en un scope local que se reinicia en cada cambio de contexto de función
- *currentScope*: Indica en qué scope se está trabajando actualmente en compilación
- *currentType*: Almacena el tipo de datos que se están trabajando en ese momento
- *usedNamesGlobal*: Almacena los nombre que fueron ya utilizados ya sea por variables o funciones en el scope global para no ser repetidos
- *usedNamesLocal*: Almacena los nombres ya utilizados por variables locales para evitar repetidos
- *PilaO*: Lleva el control de los operandos que se están manejando en las expresiones guardando sus direcciones virtuales
- *POoper*: Lleva el control de los operadores que tienen que ser ejecutados
- *Ptipos*: Se relacionan con la pila de operandos, guardando el tipo que les corresponde
- *PJumps*: Almacena la dirección de un segmento en donde será necesario incluir un salto dentro de las instrucciones del código
- *Cuadрупlos*: Guarda la secuencia de cuadрупlos elaborados por el compilador
- *ParameterTable*: lleva un control de los parámetros que se envían en una llamada a una función

- ParameterTableList: Almacena los tipos de datos de una función para generar su firma
- tablaConstantes: Almacena las constantes numéricas y chars que se usan directamente en el código con su dirección virtual.
- VControl: Usada para los estatutos de que requieren este tipo de variables para su funcionamiento, por ejemplo el for
- VFinal: Mismo caso que la variable de control
- contTemps: Permite guardar el número de los temporales usados para posteriormente calcular el tamaño de una función
- contParams: Almacena en una pila los parámetros que se utilizan en una función para después compararlo con la firma

Descripción de la máquina virtual

Utilerias

Además de las utilizadas en el compilador se agregó: Numpy, matplotlib y statistics

Administración de Memoria en ejecución

Estructura y justificación

Finalmente se optó en ejecución por tomar una variante distinta a la vista en clase, de manera que en la memoria de ejecución también se manejara la estructura de los mapas (diccionarios) esto por lo siguiente: Al ya tener desde la parte de compilación prácticamente un identificador único para cada elemento y como desarrollador saber que significaba cada rango, no había problema para que en cada instancia de memoria se preservarán eso identificadores. Python por la manera que funciona, nos otorga una ventaja sobre el uso por ejemplo de vectores o estructuras más formales como una serie de listas encadenadas que llevaran un control más granular de los espacios de memoria. Esto porque en Python tanto las listas como los diccionarios tienen el mismo grado de complejidad $O(n)$, con la diferencia que la velocidad de respuesta y por lo tanto eficiencia del programa en general es mucho mayor en los diccionarios, mientras más grande se vaya haciendo. más se obtiene ventaja, llegando a ser incluso un millón de veces más rápido que una lista.

Entonces se generó una clase memoria la cual cuenta con un atributo *mem* donde se generará su memoria. Esta clase será instanciada inicialmente y por defecto para la memoria global, quedando siempre activa, y cuando se generan llamadas a funciones cada una de estas llamadas genera una nueva instancia de memoria, cada uno de los estatutos manejados por la máquina entonces solo deben dar prioridad a lo que haya en memoria local. Si hay múltiples llamadas por ejemplo en un caso recursivo se irán “durmiendo” en un stack de ejecución el cual almacena estas memorias locales, dejando únicamente viva la última instancia de memoria. La manera de regresar a las memorias pasadas que quedaron dormidas solo puede ser de 2 formas, un return, el cual si hay memorias en el stack, regresa a la última y que llegue a un cuádruplo de endFunc, esto cambiará a la última

memoria en el stack o a la global en caso de no haber en el stack y destruirá la memoria que acaba de ejecutar el endfunc.

Asociación hecha con las direcciones virtuales

Como ya fue mencionado, la asociación es directa ya que la máquina virtual es capaz de interpretar los mismos rangos de memorias virtuales que se manejaron en compilación para su ejecución en su instancia actual de memoria sin hacer desperdicios de memoria.

Pruebas del funcionamiento del lenguaje

A continuación se muestran algunas pruebas hechas, para fines demostrativos se agregará el código que se utilizó, los cuádruplos que generó y los resultados de ejecución.

Pruebas de operaciones aritméticas:

```
1  Program Estatutos;
2  √ VARS
3      int: a, b, c;
4      float: d, e, f;
5      char: g, h, i, j;
6
7  √ main(){
8      a = 12;
9      b = -21;
10     c = 43;
11     write("enteros iniciados ", a , b , c);
12     a = b * b;
13     b = b - a;
14     c = b / a;
15     write("operaciones enteros ", a, b, c);
16
17     d = 3.14;
18     e = 12.453;
19     f = -67.324232341;
20     write("floats iniciados: ", d, e, f);
21     d = d*d*d*d;
22     e = 10.0/3;
23     f = f - (f*2);
24     write("floats con operaciones: ", d, e, f);
25
26     g = 'e';
27     h = 'l';
28     i = 'o';
29     j = 'y';
30     write("chars iniciados", g, h, i, j);
31 }
```

```

['1', '14', '-1', '-1', '2']
['2', '11', '38002', '-1', '2001']
['3', '11', '38003', '-1', '2002']
['4', '11', '38004', '-1', '2003']
['5', '12', '-1', '-1', '"enteros iniciados "']
['6', '12', '-1', '-1', '2001']
['7', '12', '-1', '-1', '2002']
['8', '12', '-1', '-1', '2003']
['9', '3', '2002', '2002', '30000']
['10', '11', '30000', '-1', '2001']
['11', '2', '2002', '2001', '30001']
['12', '11', '30001', '-1', '2002']
['13', '4', '2002', '2001', '30002']
['14', '11', '30002', '-1', '2003']
['15', '12', '-1', '-1', '"operaciones enteros "']
['16', '12', '-1', '-1', '2001']
['17', '12', '-1', '-1', '2002']
['18', '12', '-1', '-1', '2003']
['19', '11', '40000', '-1', '5001']
['20', '11', '40001', '-1', '5002']
['21', '11', '40002', '-1', '5003']
['22', '12', '-1', '-1', '"floats iniciados: "']
['23', '12', '-1', '-1', '5001']
['24', '12', '-1', '-1', '5002']
['25', '12', '-1', '-1', '5003']
['26', '3', '5001', '5001', '32000']
['27', '3', '32000', '5001', '32001']
['28', '3', '32001', '5001', '32002']
['29', '11', '32002', '-1', '5001']
['30', '4', '40003', '38005', '32003']
['31', '11', '32003', '-1', '5002']
['32', '3', '5003', '38006', '32004']
['33', '2', '5003', '32004', '32005']
['34', '11', '32005', '-1', '5003']
['35', '12', '-1', '-1', '"floats con operaciones: "']
['36', '12', '-1', '-1', '5001']
['37', '12', '-1', '-1', '5002']
['38', '12', '-1', '-1', '5003']
['39', '11', '43000', '-1', '9001']
['40', '11', '43001', '-1', '9002']
['41', '11', '43002', '-1', '9003']
['42', '11', '43003', '-1', '9004']
['43', '12', '-1', '-1', '"chars iniciados"']
['44', '12', '-1', '-1', '9001']
['45', '12', '-1', '-1', '9002']
['46', '12', '-1', '-1', '9003']
['47', '12', '-1', '-1', '9004']

```

```
enteros iniciados
```

```
12
```

```
-21
```

```
43
```

```
operaciones enteros
```

```
441
```

```
-462
```

```
-2
```

```
floats iniciados:
```

```
3.14
```

```
12.453
```

```
-67.324232341
```

```
floats con operaciones:
```

```
97.21171216
```

```
3.3333333333333335
```

```
67.324232341
```

```
chars iniciados
```

```
'e'
```

```
'l'
```

```
'o'
```

```
'y'
```

Pruebas de estatutos condicionales

≡ conds

```
1  Program condicionales;
2  VARS
3  |   int: i, j, k;
4
5  main(){
6  |   i = 10;
7  |   j = 5;
8  |   k = 0;
9
10 |   if(i > j) then {
11 |       k = k + 5;
12 |       write("nueva k: ", k);
13 |   }
14
15 |   if(k != 0) then {
16 |       k = 0;
17 |       write("k volvió a 0");
18 |   }
19
20 |   if(k == 0 and j >= i) then {
21 |       write("esto nunca apareció en pantalla");
22 |   }else{
23 |       write("si funciona el else");
24 |   }
25 }
```



```

~/Proyecto_Compiladores python3 ./mv.py
Ingresa el nombre del archivo a compilar: conds
Código aceptado

['1', '14', '-1', '-1', '2']
['2', '11', '38002', '-1', '2001']
['3', '11', '38003', '-1', '2002']
['4', '11', '38000', '-1', '2003']
['5', '7', '2001', '2002', '36000']
['6', '15', '36000', '-1', '11']
['7', '1', '2003', '38003', '30000']
['8', '11', '30000', '-1', '2003']
['9', '12', '-1', '-1', 'nueva k: ']
['10', '12', '-1', '-1', '2003']
['11', '10', '2003', '38000', '36001']
['12', '15', '36001', '-1', '15']
['13', '11', '38000', '-1', '2003']
['14', '12', '-1', '-1', 'k volvió a 0']
['15', '5', '2003', '38000', '36002']
['16', '9', '2002', '2001', '36003']
['17', '18', '36002', '36003', '36004']
['18', '15', '36004', '-1', '21']
['19', '12', '-1', '-1', 'esto nunca apareció en pantalla']
['20', '14', '-1', '-1', '22']
['21', '12', '-1', '-1', 'si funciona el else']
nueva k:
5
k volvió a 0
si funciona el else

```

```

≡ for
1   Program MyRlike;
2   VARS
3   |   int: i, j;
4
5
6   main(){
7   |   j = 1;
8   |   for i = 0 to 10 do {
9   |   |   j = j + (j*i);
10  |   |   write(j);
11  |   |   }
12  |   write("Terminó el ciclo");
13  |   }

```

~/Proyecto_Compiladores  python3 ./mv.py
 Ingresa el nombre del archivo a compilar: for
 Código aceptado

```

['1', '14', '-1', '-1', '2']
['2', '11', '38001', '-1', '2002']
['3', '11', '38000', '-1', '2001']
['4', '11', '38002', '-1', '30000']
['5', '6', '2001', '30000', '36000']
['6', '15', '36000', '-1', '15']
['7', '3', '2002', '2001', '30001']
['8', '1', '2002', '30001', '30002']
['9', '11', '30002', '-1', '2002']
['10', '12', '-1', '-1', '2002']
['11', '1', '2001', '38001', '30003']
['12', '11', '30003', '-1', '2001']
['13', '11', '30003', '-1', '2001']
['14', '14', '-1', '-1', '5']
['15', '12', '-1', '-1', '"Terminó el ciclo"']
1
2
6
24
120
720
5040
40320
362880
3628800
Terminó el ciclo

```

Fibonacci iterativo y cíclico:

```
≡ fibo
1  Program playground;
2  VARS
3      int: i, j, siguiente, actual, temporal;
4      float: e;
5  function int fibo (int: j);
6  {
7      if (j < 2) then {
8          return(j);
9      } else {
10         return(fibo(j - 1) + fibo(j - 2));
11     }
12 }
13 main(){
14     i = fibo(10);
15     write(i);
16
17     siguiente = 1;
18     actual = 0;
19     temporal = 0;
20     for j = 1 to 11 do {
21         temporal = actual;
22         actual = siguiente;
23         siguiente = siguiente + temporal;
24     }
25
26     write(" ^ recursivo ---- iterativo v ");
27     write(actual);
28 }
```

```

~/Proyecto_Compiladores python3 ./mv.py
Ingresa el nombre del archivo a compilar: fibo
Código aceptado

['1', '14', '-1', '-1', '19']
['2', '6', '12000', '38002', '36000']
['3', '15', '36000', '-1', '6']
['4', '17', '12000', '-1', '46000']
['5', '14', '-1', '-1', '18']
['6', '21', '-1', '-1', 'fibo']
['7', '2', '12000', '38001', '30000']
['8', '22', '30000', '-1', '12000']
['9', '23', 'fibo', '-1', '2']
['10', '11', '46000', '-1', '30001']
['11', '21', '-1', '-1', 'fibo']
['12', '2', '12000', '38002', '30002']
['13', '22', '30002', '-1', '12000']
['14', '23', 'fibo', '-1', '2']
['15', '11', '46000', '-1', '30003']
['16', '1', '30001', '30003', '30004']
['17', '17', '30004', '-1', '46000']
['18', '20', '-1', '-1', '-1']
['19', '21', '-1', '-1', 'fibo']
['20', '22', '38003', '-1', '12000']
['21', '23', 'fibo', '-1', '2']
['22', '11', '46000', '-1', '30000']
['23', '11', '30000', '-1', '2001']
['24', '12', '-1', '-1', '2001']
['25', '11', '38001', '-1', '2003']
['26', '11', '38000', '-1', '2004']
['27', '11', '38000', '-1', '2005']
['28', '11', '38001', '-1', '2002']
['29', '11', '38004', '-1', '30001']
['30', '6', '2002', '30001', '36000']
['31', '15', '36000', '-1', '40']
['32', '11', '2004', '-1', '2005']
['33', '11', '2003', '-1', '2004']
['34', '1', '2003', '2005', '30002']
['35', '11', '30002', '-1', '2003']
['36', '1', '2002', '38001', '30003']
['37', '11', '30003', '-1', '2002']
['38', '11', '30003', '-1', '2002']
['39', '14', '-1', '-1', '30']
['40', '12', '-1', '-1', '" ^ recursivo ---- iterativo v "']
['41', '12', '-1', '-1', '2004']
55
^ recursivo ---- iterativo v
55
~/Proyecto_Compiladores

```

Factorial recursivo e iterativo:

```
≡ basicOps.txt    ≡ for    ≡ factorial  ×  ≡
≡ factorial
1  Program playground;
2  VARS
3      int: i, j, num;
4      float: e;
5  function int fact (int: j);
6  VARS int: i;
7  {
8      if (j == 1) then {
9          return(j);
10     } else {
11         return(j * (fact(j - 1)) );
12     }
13 }
14 main(){
15     i = fact(4);
16     write(i);
17
18     j = 1;
19     num = 4;
20     while(num > 1) do {
21         j = j * num;
22         num = num - 1;
23     }
24     write(" ^ recursivo ---- iterativo v ");
25     write(j);
26
27 }
```

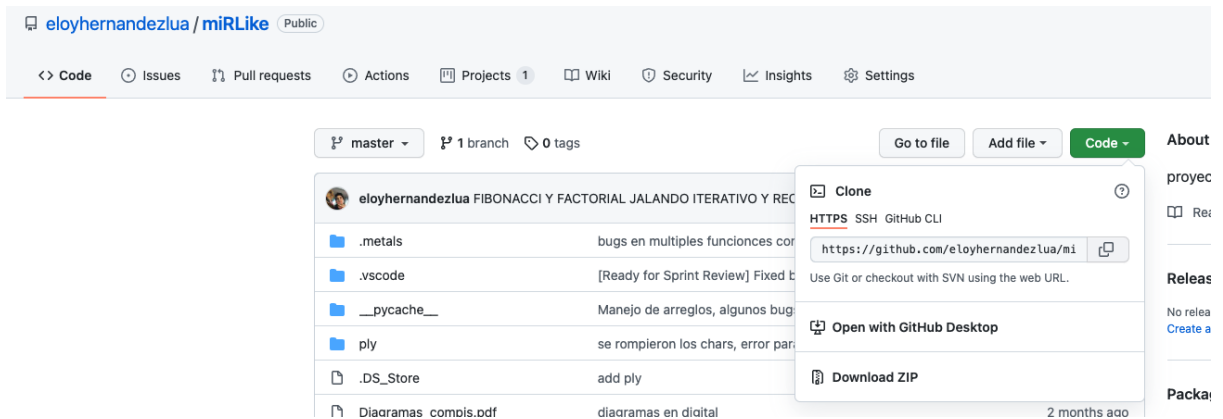
```
~/Proyecto_Compiladores python3 ./mv.py
Ingresa el nombre del archivo a compilar: factorial
Código aceptado

['1', '14', '-1', '-1', '14']
['2', '5', '12000', '38001', '36000']
['3', '15', '36000', '-1', '6']
['4', '17', '12000', '-1', '46000']
['5', '14', '-1', '-1', '13']
['6', '21', '-1', '-1', 'fact']
['7', '2', '12000', '38001', '30000']
['8', '22', '30000', '-1', '12000']
['9', '23', 'fact', '-1', '2']
['10', '11', '46000', '-1', '30001']
['11', '3', '12000', '30001', '30002']
['12', '17', '30002', '-1', '46000']
['13', '20', '-1', '-1', '-1']
['14', '21', '-1', '-1', 'fact']
['15', '22', '38002', '-1', '12000']
['16', '23', 'fact', '-1', '2']
['17', '11', '46000', '-1', '30000']
['18', '11', '30000', '-1', '2001']
['19', '12', '-1', '-1', '2001']
['20', '11', '38001', '-1', '2002']
['21', '11', '38002', '-1', '2003']
['22', '7', '2003', '38001', '36000']
['23', '15', '36000', '-1', '29']
['24', '3', '2002', '2003', '30001']
['25', '11', '30001', '-1', '2002']
['26', '2', '2003', '38001', '30002']
['27', '11', '30002', '-1', '2003']
['28', '14', '-1', '-1', '22']
['29', '12', '-1', '-1', '" ^ recursivo ---- iterativo v "']
['30', '12', '-1', '-1', '2002']
24
^ recursivo ---- iterativo v
24
~/Proyecto_Compiladores
```

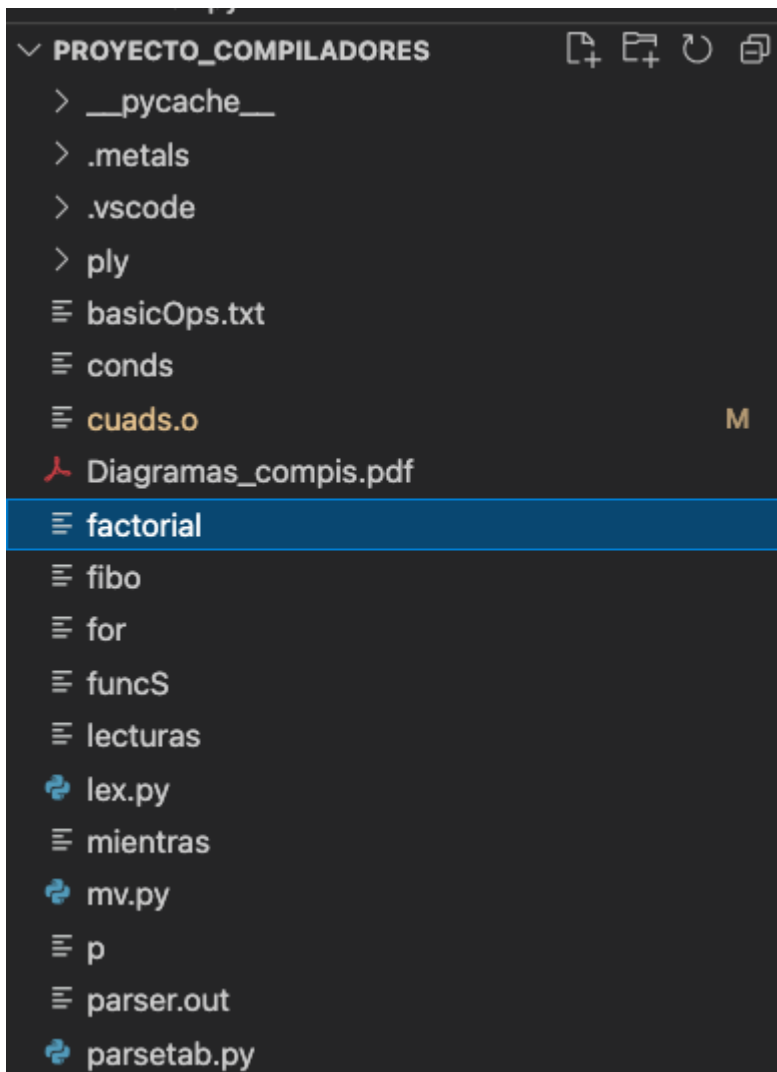
Manual de usuario

Para poder usar en su totalidad el proyecto es necesario lo siguiente:

1. Clonar el repositorio desde github: <https://github.com/eloyhernandezlua/miRLike>



2. Instalar python3 en caso de no tenerlo
3. Instalar las dependencias corriendo en terminal dentro del directorio del proyecto
`pip3 install requirements.txt`
4. Se escribe el código en cualquier archivo nuevo dentro del directorio principal con la sintaxis ya establecida



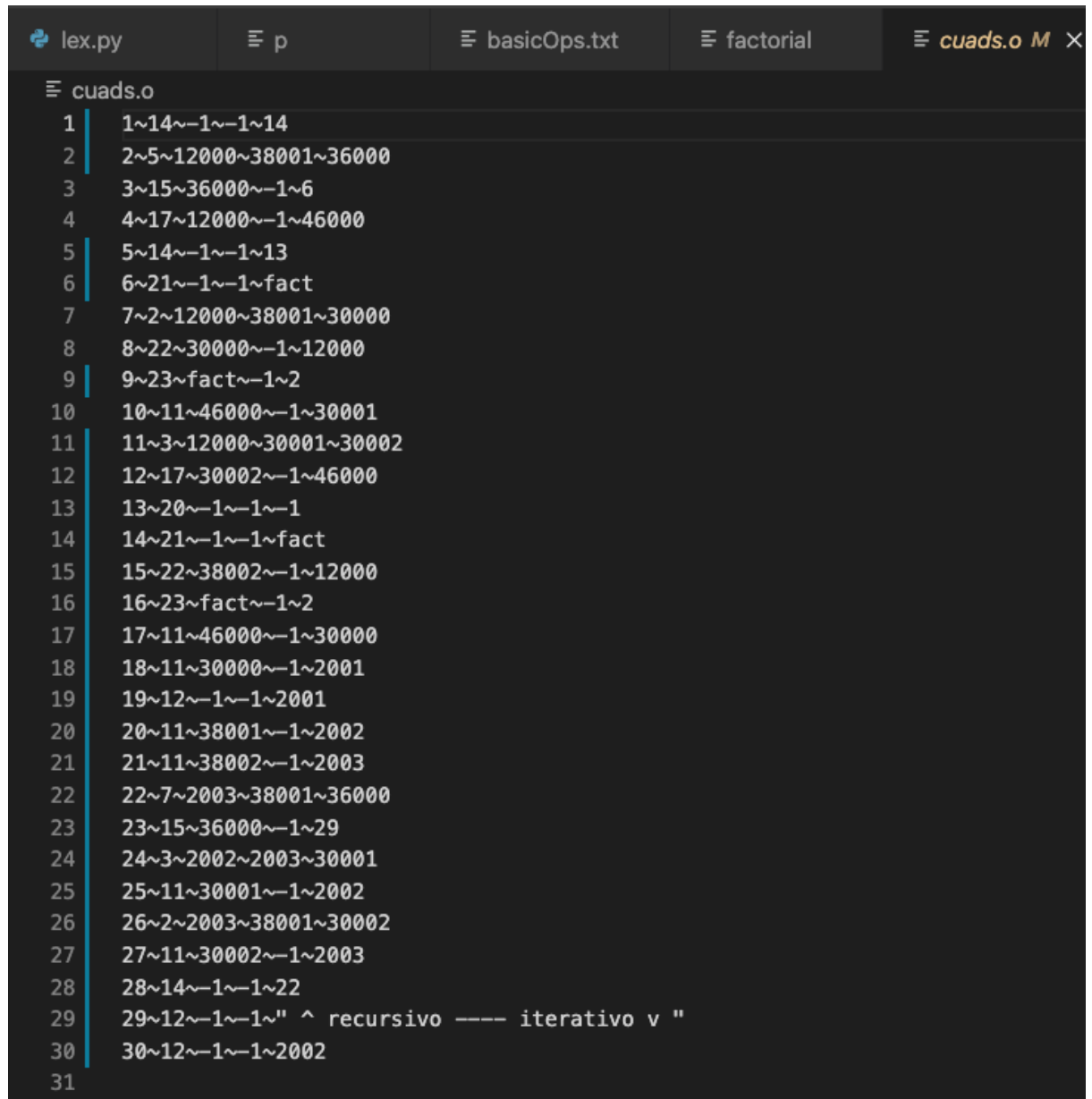
5. Desde terminal se corre el comando: `python3 ./mv.py`



- Una vez que se haya corrido, estará esperando como input el nombre del archivo donde se escribió el código que se quiere correr

```
~/Proyecto_Compiladores > python3 ./mv.py  
Ingresa el nombre del archivo a compilar: facotrial
```

- Listo, una vez dando enter se debería ejecutar el código, si el usuario quiere conocer el código intermedio que se generó (cuadрупlos) podrá consultar el archivo **cuads.o** que se creó de manera automática



```
lex.py p basicOps.txt factorial cuads.o M X  
cuads.o  
1 1~14~-1~-1~14  
2 2~5~12000~38001~36000  
3 3~15~36000~-1~6  
4 4~17~12000~-1~46000  
5 5~14~-1~-1~13  
6 6~21~-1~-1~fact  
7 7~2~12000~38001~30000  
8 8~22~30000~-1~12000  
9 9~23~fact~-1~2  
10 10~11~46000~-1~30001  
11 11~3~12000~30001~30002  
12 12~17~30002~-1~46000  
13 13~20~-1~-1~-1  
14 14~21~-1~-1~fact  
15 15~22~38002~-1~12000  
16 16~23~fact~-1~2  
17 17~11~46000~-1~30000  
18 18~11~30000~-1~2001  
19 19~12~-1~-1~2001  
20 20~11~38001~-1~2002  
21 21~11~38002~-1~2003  
22 22~7~2003~38001~36000  
23 23~15~36000~-1~29  
24 24~3~2002~2003~30001  
25 25~11~30001~-1~2002  
26 26~2~2003~38001~30002  
27 27~11~30002~-1~2003  
28 28~14~-1~-1~22  
29 29~12~-1~-1~" ^ recursivo ---- iterativo v "  
30 30~12~-1~-1~2002  
31
```

VIDEO

Video de como usar y demostrativo disponible en la siguiente liga:

<https://youtu.be/zJmJkNRG1KU>