Tecnológico de Monterrey

**Grupo:**

TC3002B.201

**Materia:**

Desarrollo de aplicaciones avanzadas de ciencias computacionales

# Evidencia Final Compiladores: Desarrollo de herramienta de soporte al proceso de análisis de imágenes


**Integrantes:**

Diego Araque Fernandez - A01026037
Luis Fernando Valdeón - A01745186
Uriel Aguilar - A01781698


**Fecha de entrega:**

02/05/24

**Gramática implementada**

- Reglas

Regla 0 S' -> statement
Regla 1 statement -> VARIABLE SETTO expression
Regla 2 statement -> VARIABLE SETTO flow
Regla 3 flow -> VARIABLE CONNECT flow_functions
Regla 4 flow_functions -> flow_function_call CONNECT flow_functions
Regla 5 flow_functions -> flow_function_call
Regla 6 flow_function_call -> VARIABLE LPAREN params RPAREN
Regla 7 statement -> expression
Regla 8 expression -> expression PLUS term
Regla 9 expression -> expression MINUS term
Regla 10 expression -> term
Regla 11 expression -> string
Regla 12 string -> STRING
Regla 13 term -> term TIMES exponent
Regla 14 term -> term DIVIDE exponent
Regla 15 term -> exponent
Regla 16 exponent -> factor EXP factor
Regla 17 exponent -> factor
Regla 18 factor -> LPAREN expression RPAREN
Regla 19 factor -> NUMBER
Regla 20 factor -> VARIABLE
Regla 21 factor -> function_call
Regla 22 function_call -> VARIABLE LPAREN RPAREN
Regla 23 function_call -> VARIABLE LPAREN params RPAREN
Regla 24 params -> params COMMA expression
Regla 25 params -> expression

- Símbolos terminales

PLUS = ‘+’
MINUS = ‘-’
SETTO = ‘=’
TIMES = ‘*’
DIVIDE = ‘/’
EXP = ‘^’
LPAREN = ‘(‘
RPAREN = ‘)’
COMMA = ‘,’
CONNECT = ‘->’
NUMBER = ‘\d+\.?\d*’

VARIABLE = ‘[a-zA-Z_][a-zA-Z0-9_]*’
STRING = ‘\".*?\"’

- Funciones

load_image()
save_image()
gen_matrix()
gen_vector()
show_image()
multiplot_show()
histogram_visualization()
search_cv2()
numpy_sin()
numpy_cos()
numpy_tan()
numpy_arcsin()
numpy_arccos()
numpy_arctan()
numpy_sinh()
numpy_cosh()
numpy_tanh()
grabcut_segmentation()

- Símbolos reservados

pi = 3.14159265359
e = 2.71828182846
max = max
None = None

**Descripción de las funciones implementadas como herramientas y accesorios a la gramática**

1. serialize_graph(G): Esta función se le pasa el árbol creado en cualquier input que da el usuario y se convierte en texto, el cual luego se escribe en un archivo y se exporta.
2. load_image(image_path): Esta función implementa la función imread(img) de opencv. Nuestra implementación solo permite que la podamos llamar a través de otro nombre. Esta nos devuelve una matriz.
3. save_image(image, path): Esta función nos permite guardar una imagen usando la función imwrite() de opencv.
4. show_image(image): A esta función le pasamos una matriz que corresponde a una imagen, previamente pasada por la función load_image(). Esta función

implementa la funcionalidad imshow() de opencv y enseña la imagen que queremos en nuestro ordenador.

5. search_cv2(function_name): Esta función nos permite buscar cualquier función de la librería de opencv y devolvérsela en caso de que exista o devolvernos None en el caso contrario.
6. gen_matrix(a, b, *args): Esta función nos genera una matriz de numpy. Nos presentan a y b como las dimensiones y los argumentos que queremos en la matriz. Lo que hace esta función es crear un arreglo de numpy con esas características.
7. gen_vector(*args): Esta función genera un arreglo de numpy con todos los argumentos que se le pasa a la función.
8. multiplot_show(nrows, ncols, *args): Esta función nos crea un multiplot con la librería matplotlib. Le pasamos la cantidad de filas y columnas que queremos que existan en nuestro plot, y en cada una graficamos un gráfico diferente. Algo muy importante que se hace en esta función es que se intercambian los canales de color rojo y azul de rgb. Por lo que la imagen que se muestra está un poco distorsionada.
9. histogram_visualization(image): Esta función nos permite pasar una imagen y enseñamos un histograma que muestra como varia cada uno de los tres canales (rgb) en la imagen.
10. grabcut_segmentation(image): En esta función se implementa el algoritmo grabcut. Al cual le pasamos una imagen que posee dos propiedades. Una es la imagen completa y la otra es una imagen que tiene líneas que delimitan lo que queremos cortar. Al pasar esta imagen por este algoritmo de opencv somos capaces de hacer la correcta segmentación de la imagen. Usamos unas transformaciones y lo pasamos por la función grabCut() de opencv, al final devolvemos una imagen que hace la segmentación correcta.
11. numpy_sin(x): En esta función pasamos un número e implementamos la función del seno de numpy y devolvemos el valor.
12. numpy_cos(x): En esta función pasamos un número e implementamos la función del coseno de numpy y devolvemos el valor.
13. numpy_tan(x): En esta función pasamos un número e implementamos la función del tangente de numpy y devolvemos el valor.
14. numpy_arcsin(x): En esta función pasamos un número e implementamos la función del arcoseno de numpy y devolvemos el valor.
15. numpy_arccos(x): En esta función pasamos un número e implementamos la función del arcocoseno de numpy y devolvemos el valor.
16. numpy_arctan(x): En esta función pasamos un número e implementamos la función del arcotangente de numpy y devolvemos el valor.
17. numpy_sinh(x): En esta función pasamos un número e implementamos la función del seno hiperbólico de numpy y devolvemos el valor.
18. numpy_cosh(x): En esta función pasamos un número e implementamos la función del coseno hiperbólico de numpy y devolvemos el valor.

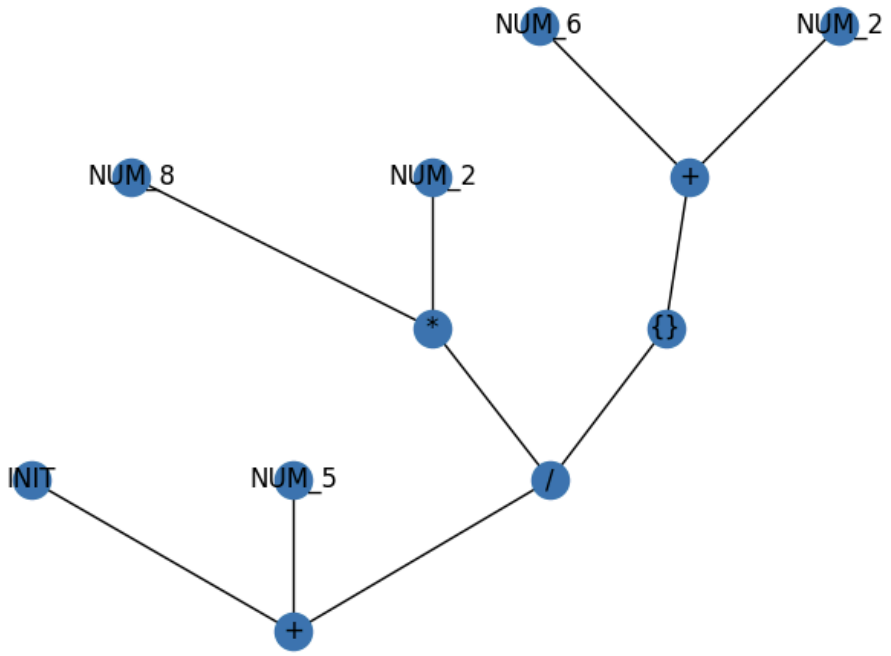
19. numpy_tanh(x): En esta función pasamos un número e implementamos la función del tangente hiperbólico de numpy y devolvemos el valor.

**Demostración de una o varias expresiones y el árbol de sintaxis abstracto demostrando**

- Precedencia de operadores

Para la precedencia de operadores usamos el siguiente input para probar nuestra gramática. 5+8*2/(6+2), el resultado de esta operación es 7. A continuación presentamos el árbol generado y el resultado de nuestro programa.

- Árbol



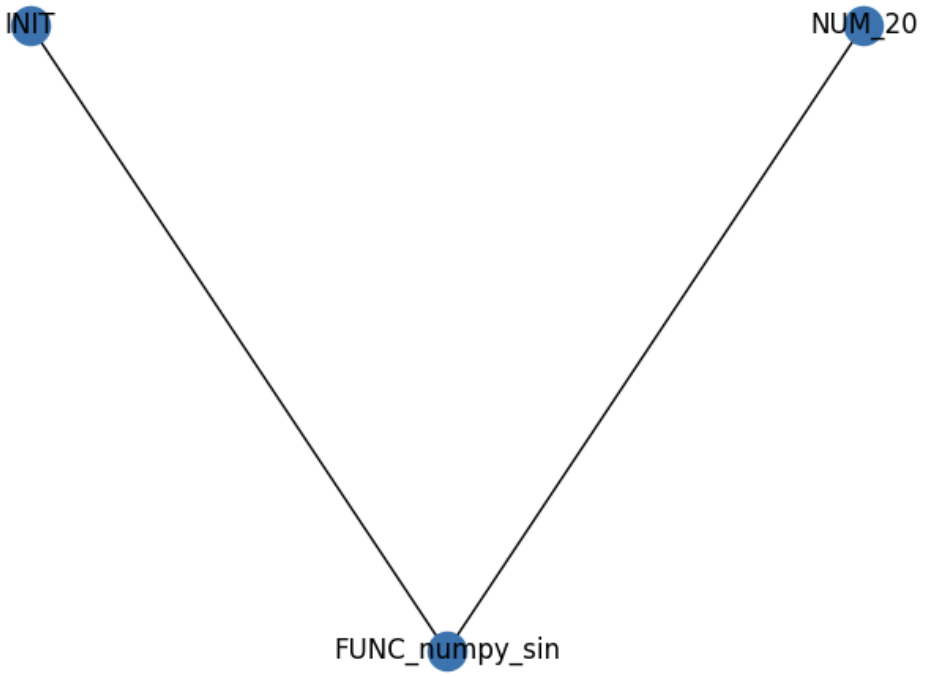

- Resultado

```
input>> 5+8*2/(6+2)
From Node 1 []
Current Node:  NUMBER
From Node 2 []
Current Node:  NUMBER
From Node 3 []
Current Node:  NUMBER
From Node 4 [8, 2]
Current Node:  TIMES
multiplying:  [8, 2]
From Node 5 []
Current Node:  NUMBER
From Node 6 []
Current Node:  NUMBER
From Node 7 [6, 2]
Current Node:  PLUS
adding:  [6, 2]
From Node 8 [8]
Current Node:  GROUP
From Node 9 [16, 8]
Current Node:  DIVIDE
dividing:  [16, 8]
From Node 10 [5, 2.0]
Current Node:  PLUS
adding:  [5, 2.0]
From Node 0 [7.0]
Current Node:  INITIAL
RESULT:  7.0
```

- Llamadas a funciones

En el caso de llamadas a funciones, presentaremos una de las hechas con numpy. En este caso numpy_sin(20), cuyo resultado es 0.9129, a continuación adjuntamos pruebas del funcionamiento.

- Árbol



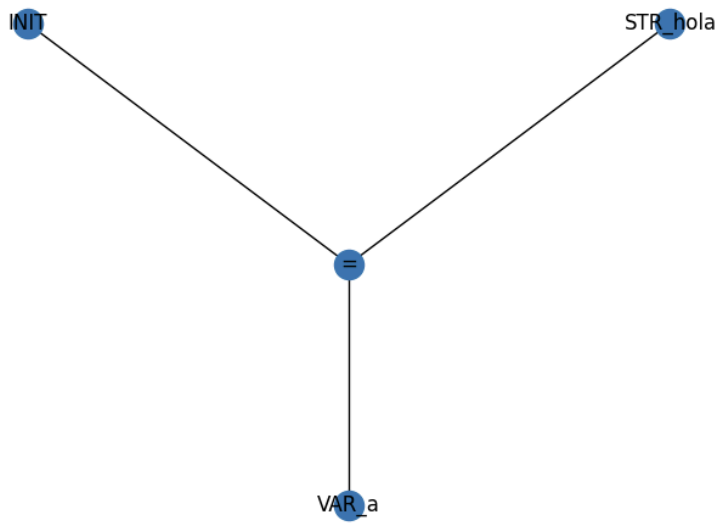

- Resultado

```
input>> numpy_sin(20)
From Node 1 []
Current Node:  NUMBER
From Node 2 [20]
Current Node:  FUNCTION_CALL
From Node 0 [0.9129452507276277]
Current Node:  INITIAL
```

- Asignación de variables

Para la asignación de variables hacemos una simple asignación para probar el funcionamiento. En este caso probamos a = “hola”. A continuación presentamos el árbol y el resultado de nuestro sistema.

- Árbol



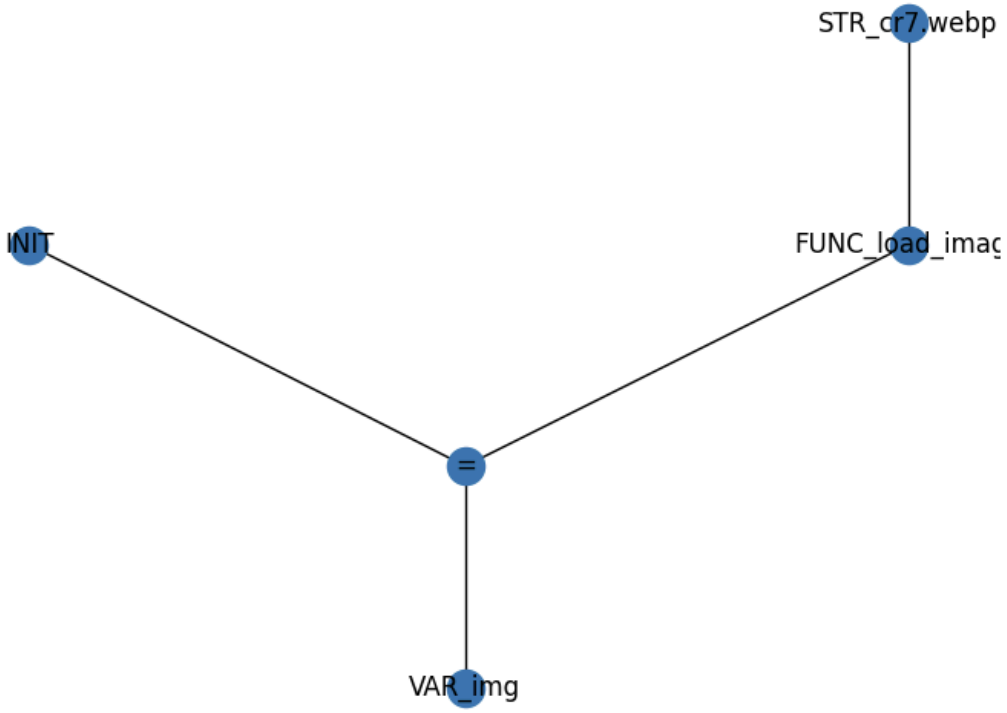

- Resultado

```
input>> a= "hola"
From Node 3 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  STRING
From Node 2 ['a', 'hola']
Current Node:  ASSIGN
From Node 0 ['hola']
Current Node:  INITIAL
input>> a
From Node 1 []
Current Node:  VARIABLE
From Node 0 ['hola']
Current Node:  INITIAL
```

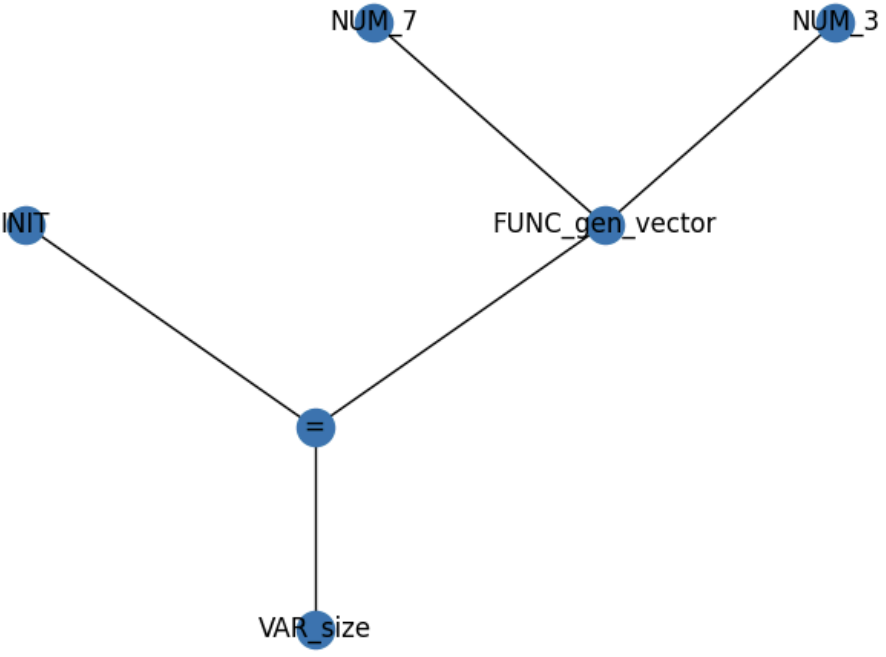
- Implementación de flujos de imágenes y aplicación de filtros de Open CV

Para los flujos haremos una demostración con las funciones load_image(), gen_vector(), blur() y show_image(). En esta prueba se observa como en vez de tener una función muy grande que recibe muchos parametros, el resultado de una pasa a la siguiente hasta llegar a un resultado. A continuación presentamos los árboles generados y la imágen del principio y del final.

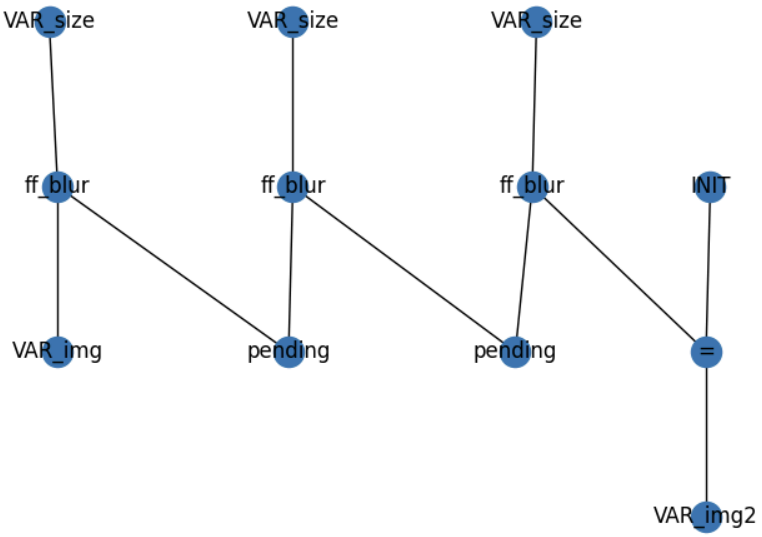
- Árboles
  - img = load_image()




  - size = gen_vector()



- img2 = img -> blur(size) -> blur(size) -> blur(size)




- Resultado
  - Imágen Inicial

  - Imágen Final

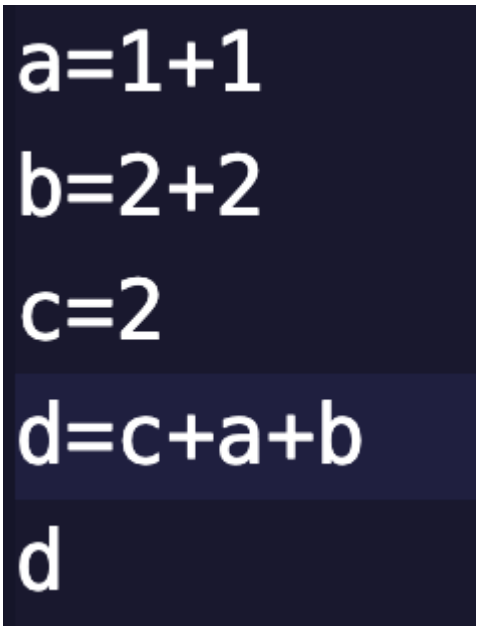
- Cada una de las nuevas características implementadas

El profesor nos dio la opción de realizar diferentes actividades a nuestro código. A continuación presentamos las que hicimos, y adjuntamos pruebas de que estas funcionan.

1. Aceptar archivos y Ejecutar el contenido.

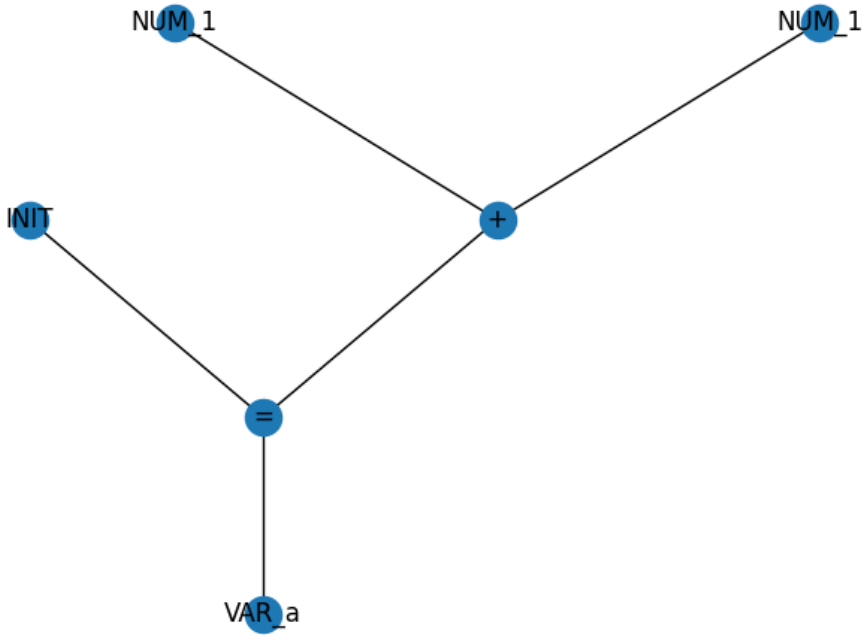
Para esta funcionalidad al correr nuestro programa se le pide al usuario el nombre del archivo que quiere correr. Si el usuario esto desea ingresa el nombre y se corre todo el contenido. A continuación presentamos una prueba de como funciona. El resultado de nuestro archivo buscamos que sea 8.

- Archivo

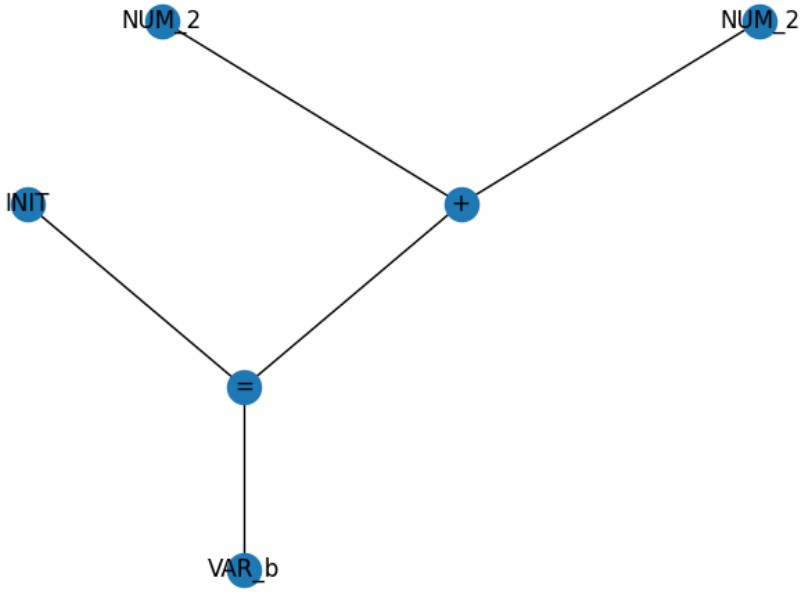



- Árboles

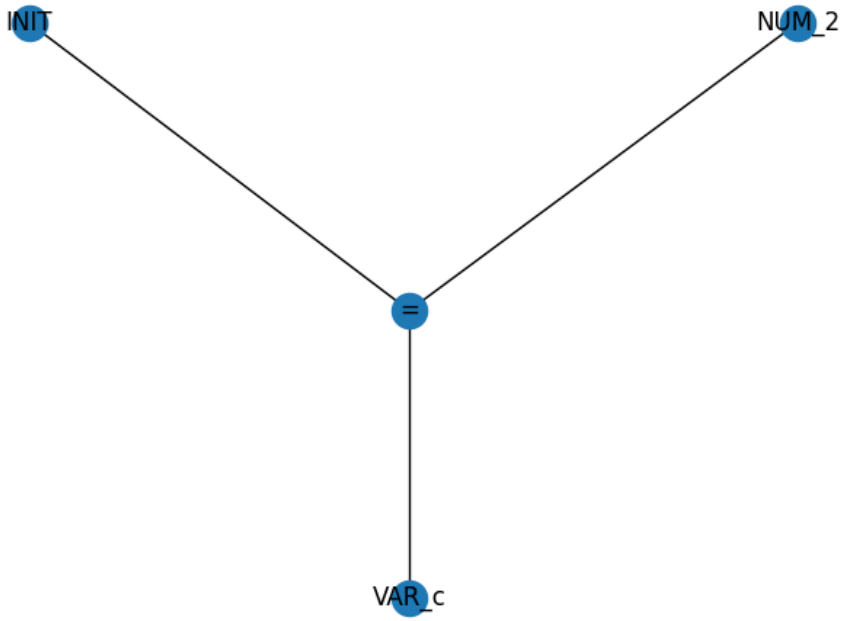
- a = 1+1



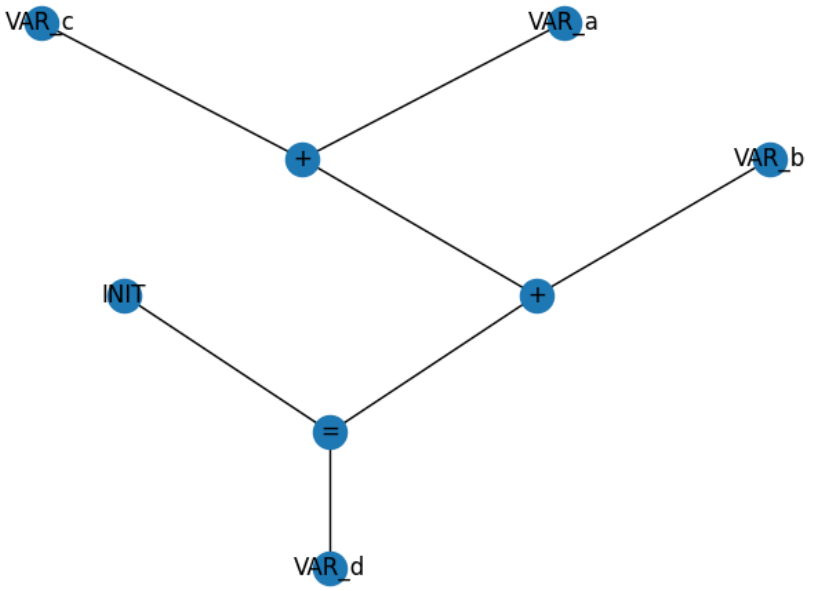

- b = 2+2



- c = 2




- d = c+a+b



- d


- Resultados
  - a = 1+1

```
Enter file you want to read or press enter if you don't have a file: file.txt
From Node 5 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  NUMBER
From Node 2 []
Current Node:  NUMBER
From Node 3 [1, 1]
Current Node:  PLUS
adding:  [1, 1]
From Node 4 ['a', 2]
Current Node:  ASSIGN
From Node 0 [2]
Current Node:  INITIAL
RESULT:  2
```

- b = 2+2

```
From Node 5 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  NUMBER
From Node 2 []
Current Node:  NUMBER
From Node 3 [2, 2]
Current Node:  PLUS
adding:  [2, 2]
From Node 4 ['b', 4]
Current Node:  ASSIGN
From Node 0 [4]
Current Node:  INITIAL
RESULT:  4
```

- c = 2

```
From Node 3 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  NUMBER
From Node 2 ['c', 2]
Current Node:  ASSIGN
From Node 0 [2]
Current Node:  INITIAL
RESULT:  2
```

- d = c+a+B

```
From Node 7 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  VARIABLE
From Node 2 []
Current Node:  VARIABLE
From Node 3 [2, 2]
Current Node:  PLUS
adding:  [2, 2]
From Node 4 []
Current Node:  VARIABLE
From Node 5 [4, 4]
Current Node:  PLUS
adding:  [4, 4]
From Node 6 ['d', 8]
Current Node:  ASSIGN
From Node 0 [8]
Current Node:  INITIAL
RESULT:  8
```

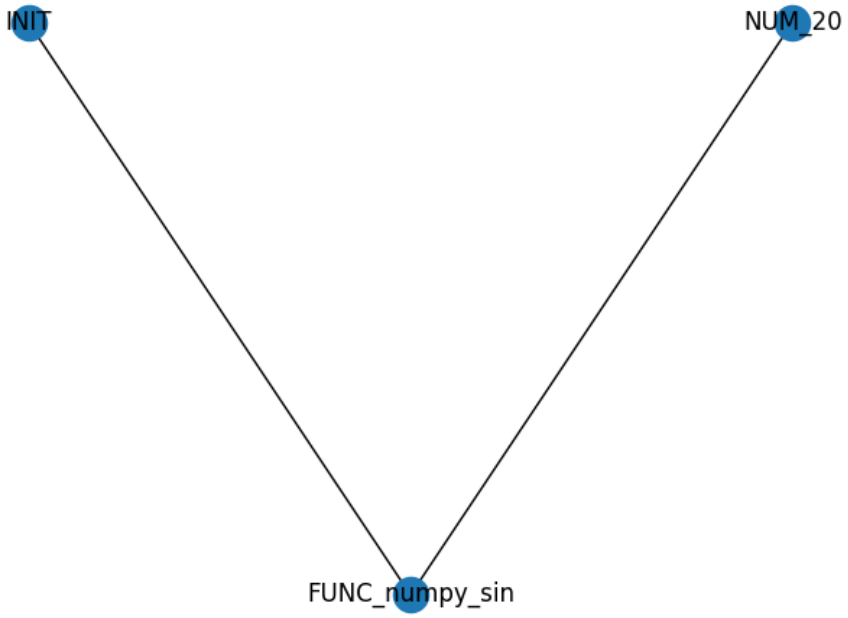
- d

```
From Node 1 []
Current Node:  VARIABLE
From Node 0 [8]
Current Node:  INITIAL
RESULT:  8
```

2. Aceptar 9 funciones de numpy.

Implementamos 9 funciones de numpy para nuestra grámatica. Todas caen en el ámbito trigonométrico. A continuación presentamos todas con pruebas de su funcionamiento.

- numpy_sin(20): El seno de 20 es 0.9129, a continuación adjuntamos pruebas del funcionamiento.
    - Árbol

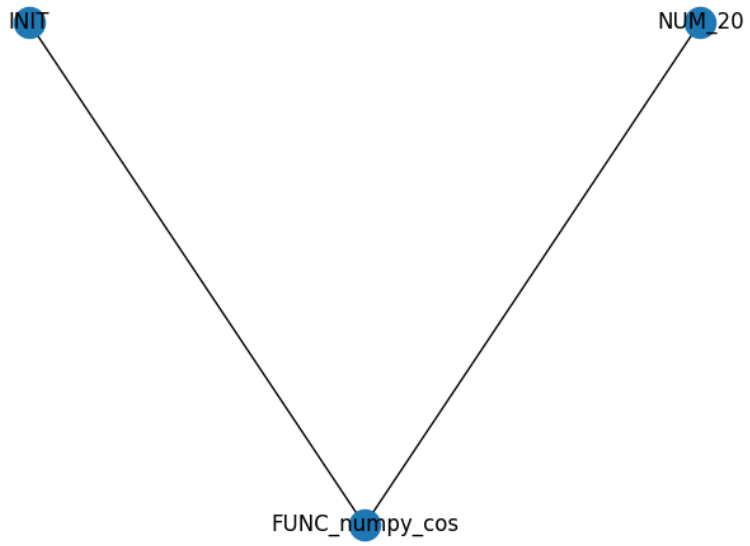



    - Resultado

```
input>> numpy_sin(20)
From Node 1 []
Current Node:  NUMBER
From Node 2 [20]
Current Node:  FUNCTION_CALL
From Node 0 [0.9129452507276277]
Current Node:  INITIAL
```

- numpy_cos(20): El coseno de 20 es 0.4080, a continuación adjuntamos pruebas del funcionamiento.

- Árbol

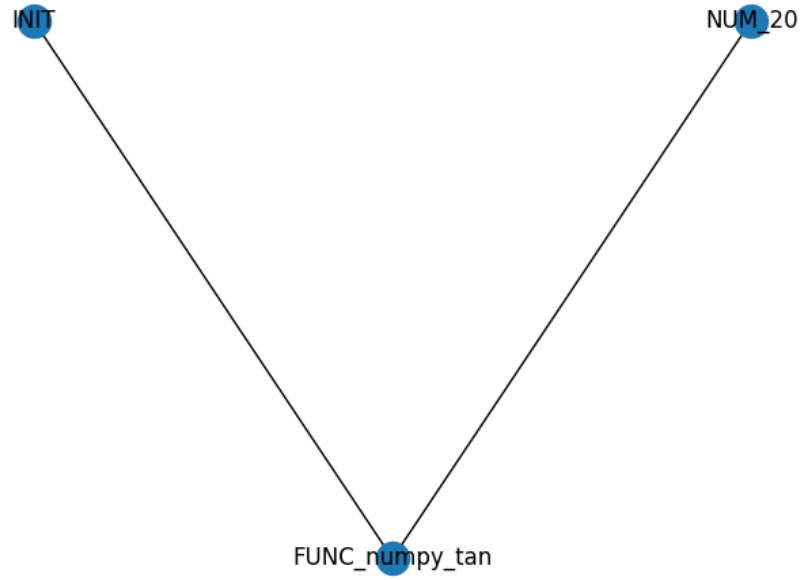



- Resultado

```
input>> numpy_cos(20)
From Node 1 []
Current Node:  NUMBER
From Node 2 [20]
Current Node:  FUNCTION_CALL
From Node 0 [0.408082061813392]
Current Node:  INITIAL
```

- numpy_tan(20): La tangente de 20 es 2,2371, a continuación adjuntamos pruebas del funcionamiento.

- Árbol

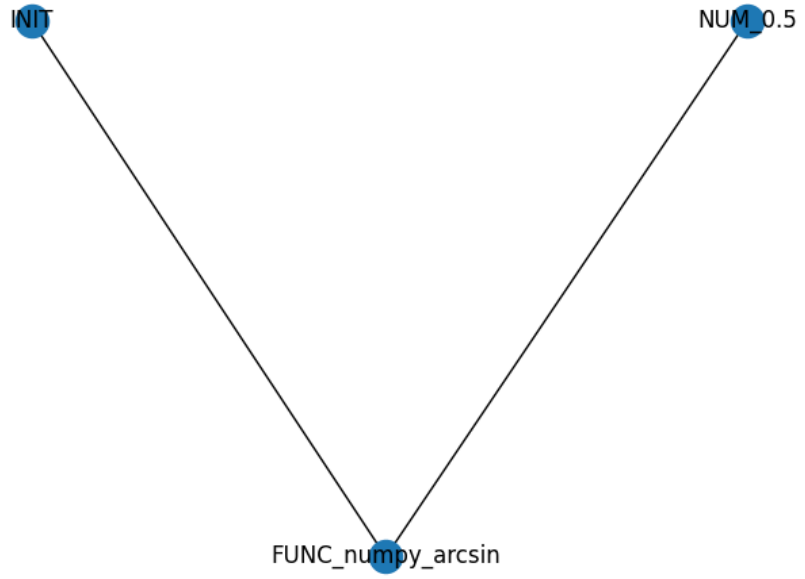



- Resultado

```
input>> numpy_tan(20)
From Node 1 []
Current Node:  NUMBER
From Node 2 [20]
Current Node:  FUNCTION_CALL
From Node 0 [2.2371609442247427]
Current Node:  INITIAL
```

- numpy_arcsin(0.5): El arcoseno de 0.5 es 0.5235, a continuación adjuntamos pruebas del funcionamiento.

- Árbol

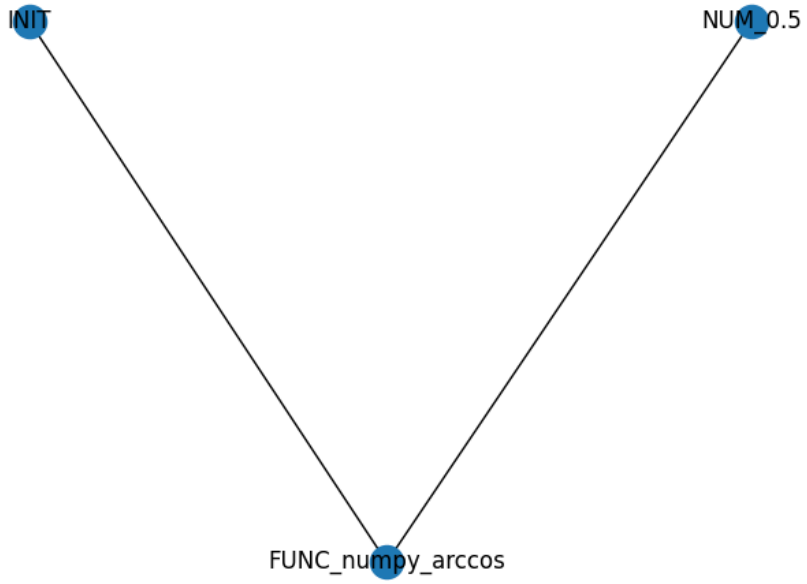



- Resultado

```
input>> numpy_arcsin(0.5)
From Node 1 []
Current Node:  NUMBER
From Node 2 [0.5]
Current Node:  FUNCTION_CALL
From Node 0 [0.5235987755982988]
Current Node:  INITIAL
```

- numpy_arccos(0.5): El arcocoseno de 0.5 es 1.0471, a continuación adjuntamos pruebas del funcionamiento.

- Árbol

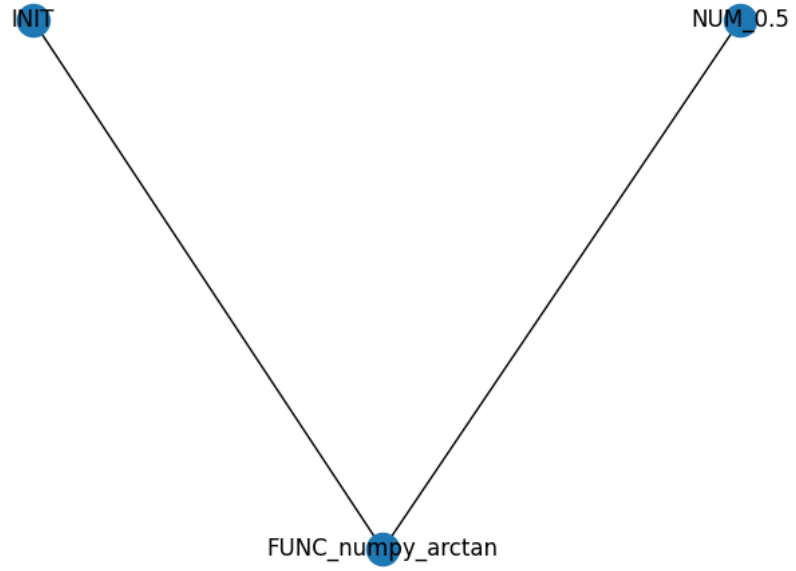



- Resultado

```
input>> numpy_arccos(0.5)
From Node 1 []
Current Node:  NUMBER
From Node 2 [0.5]
Current Node:  FUNCTION_CALL
From Node 0 [1.0471975511965976]
Current Node:  INITIAL
```

- numpy_arctan(0.5): El arcotangente de 0.5 es 0.4636, a continuación adjuntamos pruebas del funcionamiento.

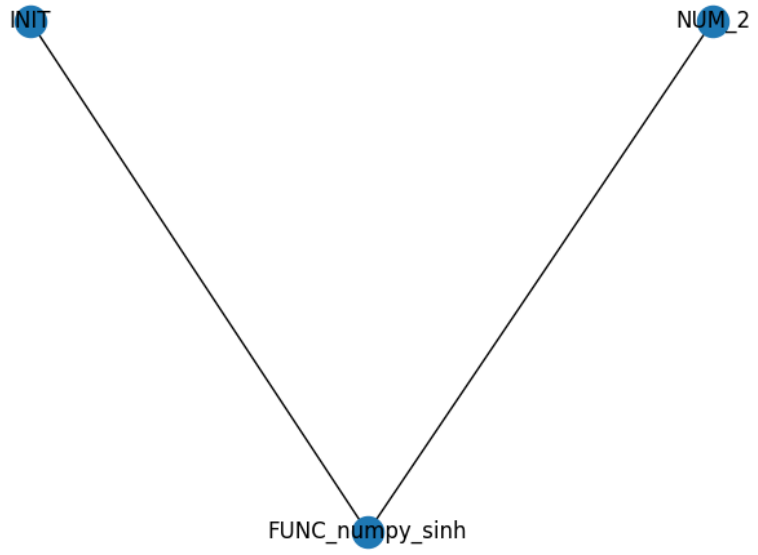
- Árbol




- Resultado

```
input>> numpy_arctan(0.5)
From Node 1 []
Current Node:  NUMBER
From Node 2 [0.5]
Current Node:  FUNCTION_CALL
From Node 0 [0.4636476090008061 5]
Current Node:  INITIAL
```

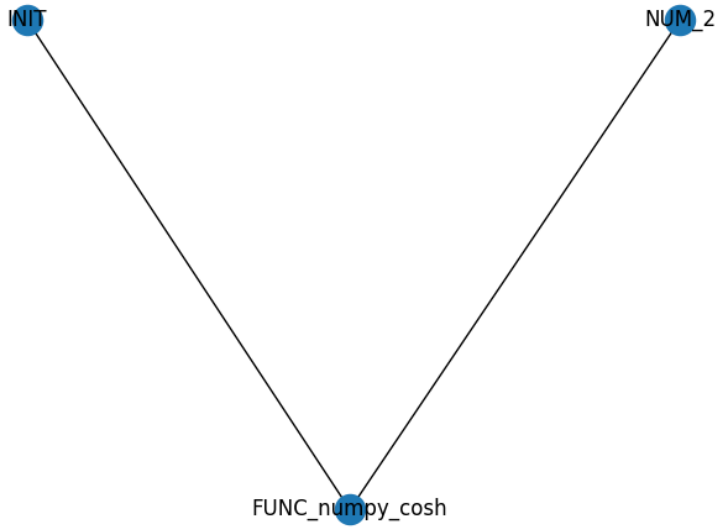
- numpy_sinh(2): El seno hiperbólico de 2 es 3.6268, a continuación adjuntamos pruebas del funcionamiento.
  - Árbol



- Resultado

```
input>> numpy_sinh(2)
From Node 1 []
Current Node:  NUMBER
From Node 2 [2]
Current Node:  FUNCTION_CALL
From Node 0 [3.626860407847018 6]
Current Node:  INITIAL
```

- numpy_cosh(2): El coseno hiperbólico de 2 es 3.7621, a continuación adjuntamos pruebas del funcionamiento.
  - Árbol

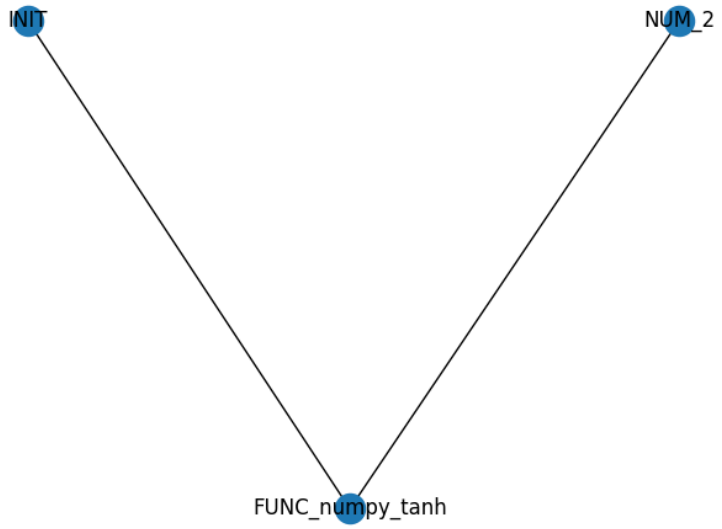



  - Resultado

```
input>> numpy_cosh(2)
From Node 1 []
Current Node:  NUMBER
From Node 2 [2]
Current Node:  FUNCTION_CALL
From Node 0 [3.762195691083631 4]
Current Node:  INITIAL
```

- numpy_tanh(2): La tangente hiperbólico de 2 es 0.9640, a continuación adjuntamos pruebas del funcionamiento.

- Árbol



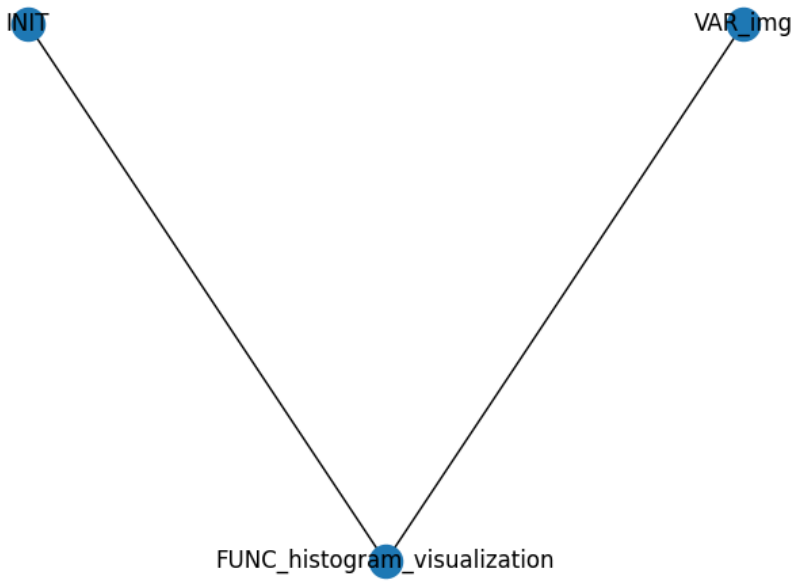

- Resultado

```
input>> numpy_tanh(2)
From Node 1 []
Current Node:  NUMBER
From Node 2 [2]
Current Node:  FUNCTION_CALL
From Node 0 [0.9640275800758169]
Current Node:  INITIAL
```

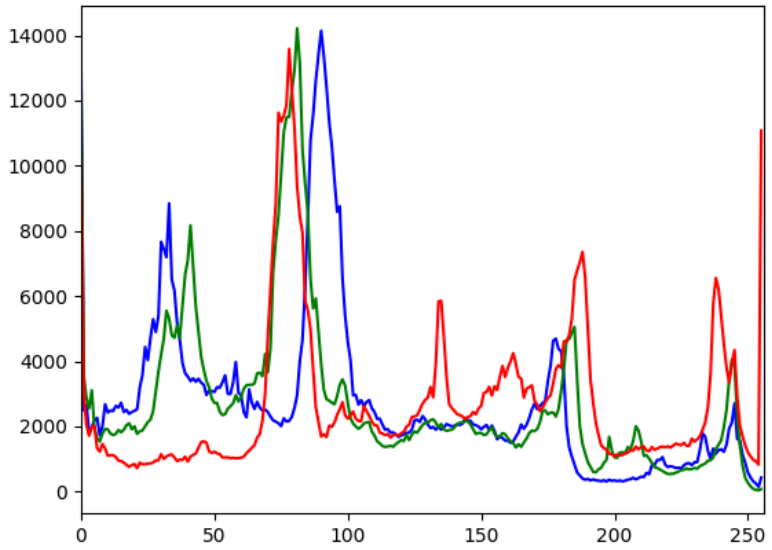
3. Implementación de visualización de histogramas con opencv.

Esta función nos permite pasar una imagen y enseñamos un histograma que muestra cómo varia cada uno de los tres canales (rgb) en la imagen. A continuación presentamos el árbol generado al realizar esta operación, como la imagen del histograma generado.

- Árbol




- Resultado

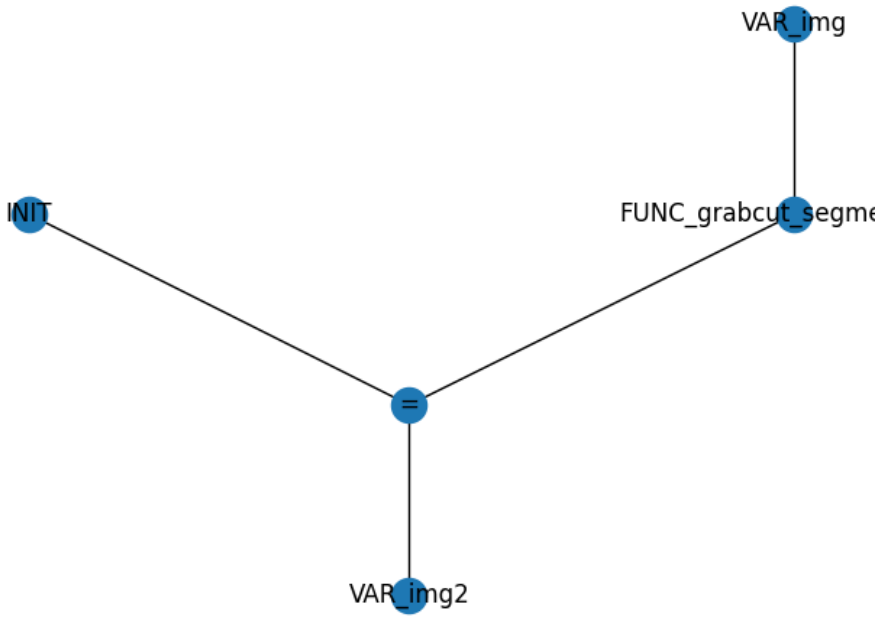



4. Implementación de un algoritmo complejo como herramienta en el lenguaje: WaterShed, Grabcut, TemplateMatching, CannyEdgeDetection.

En esta función se implementa el algoritmo grabcut. Al cual le pasamos una imagen que posee dos propiedades. Una es la imagen completa y la otra es una imagen que tiene líneas que delimitan lo que queremos cortar. Al pasar esta imagen por este algoritmo de opencv somos capaces de hacer la correcta segmentación de la imagen. Usamos unas transformaciones y lo pasamos por la función grabCut() de

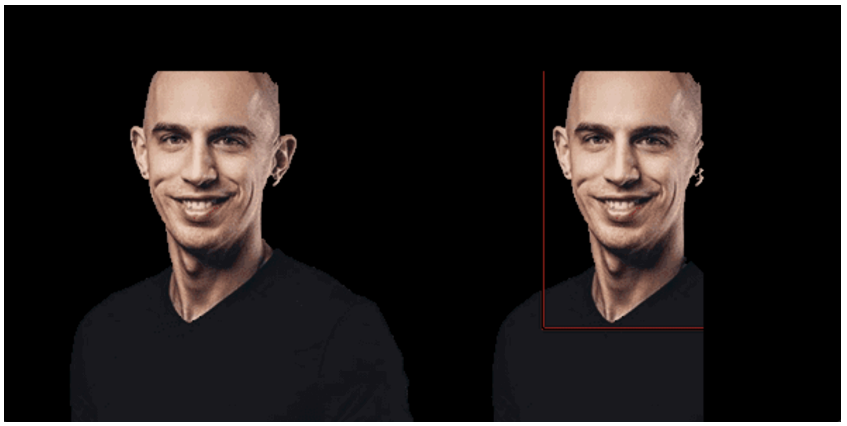
opencv, al final devolvemos una imagen que hace la segmentación correcta. A continuación presentamos el árbol generado, la imagen inicial y la final.

- Árbol



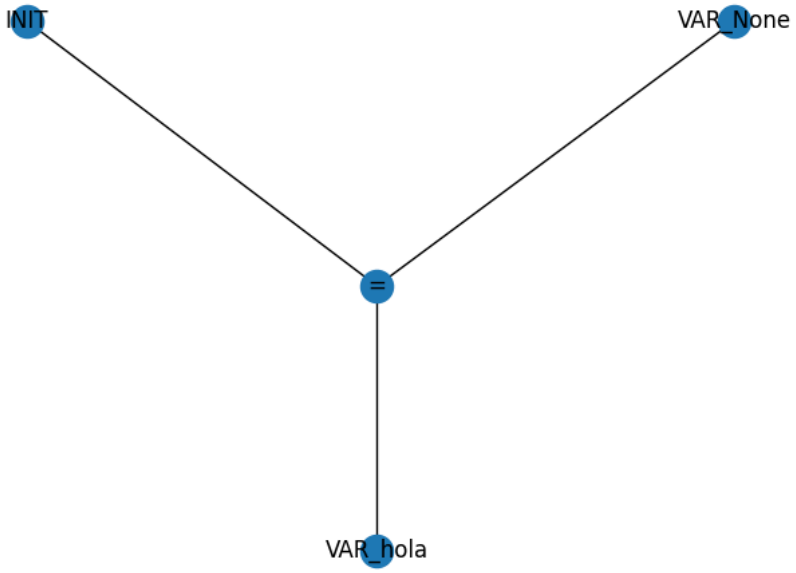

- Resultado
  - Imagen Inicial

- Imagen Final



5. Aceptar None como valor de la gramática para inicialización de variables. A continuación presentamos pruebas del funcionamiento.

- Árbol

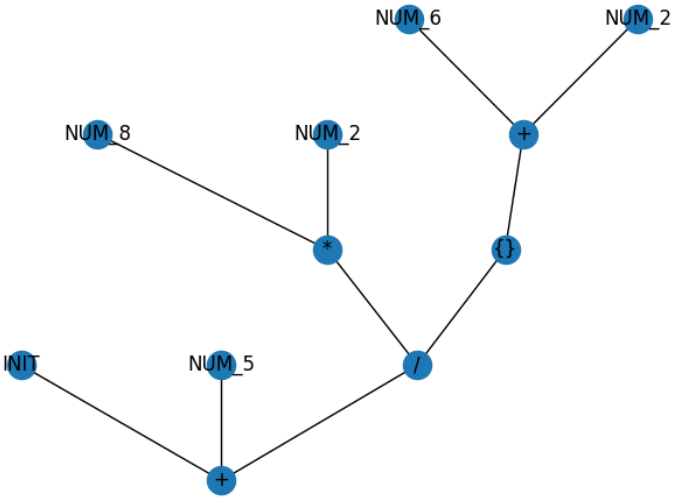



- Resultado

```
input>> hola = None
From Node 3 []
Current Node:  VARIABLE_ASSIGN
From Node 1 []
Current Node:  VARIABLE
From Node 2 ['hola', None]
Current Node:  ASSIGN
From Node 0 [None]
Current Node:  INITIAL
```

6. Exportación del árbol de sintaxis abstracto como texto incluyendo las matrices generadas. A continuación presentamos el árbol generado por la operación 5+8*2/(6+2), y enseñaremos como resultado la traducción que hace nuestro programa para exportar el árbol a un archivo.

- Árbol




- Resultado

```
Input: 5+8*2/(6+2)
Node 0: Type=INITIAL, Label=INIT
Node 1: Type=NUMBER, Label=NUM_5, Value=5
Node 2: Type=NUMBER, Label=NUM_8, Value=8
Node 3: Type=NUMBER, Label=NUM_2, Value=2
Node 4: Type=TIMES, Label=*, Value=
Node 5: Type=NUMBER, Label=NUM_6, Value=6
Node 6: Type=NUMBER, Label=NUM_2, Value=2
Node 7: Type=PLUS, Label=+, Value=
Node 8: Type=GROUP, Label={}, Value=
Node 9: Type=DIVIDE, Label=/, Value=
Node 10: Type=PLUS, Label=+, Value=
Edge from Node 0 to Node 10
Edge from Node 1 to Node 10
Edge from Node 2 to Node 4
Edge from Node 3 to Node 4
Edge from Node 4 to Node 9
Edge from Node 5 to Node 7
Edge from Node 6 to Node 7
Edge from Node 7 to Node 8
Edge from Node 8 to Node 9
Edge from Node 9 to Node 10
```

**Videos:**

- ¿Cómo ayuda este proyecto a completar, destruir o reinterpretar la idea personal sobre el uso de la computación para la resolución de problemas?
- ¿Cuál fue el reto más importante del proyecto de compiladores? ¿Qué aportaste para la solución?
- ¿Cómo consideras que este proyecto puede ayudar a resolver el reto? Si no lo hace, ¿ qué tipo de proyecto propones con el uso de traductores? Si si lo hace, ¿Qué tipo de aplicaciones puedes proyectar a 2 o 3 años con la solución del reto y este traductor?