Universidad Nacional Autónoma de México.

Facultad de Estudios Superiores
"Aragón"

Ingeniería en Computación

Compiladores

"Proyecto Final, Interprete de mini C"

Jimenez Jimenez Roberto Salvador
Lara Martínez Christian Gael
Miguel Herrera Nestor Guillermo

Profesor:
MARCELO PEREZ MEDEL

2608

22 de Mayo del 2025

Proyecto final – Interprete mini C

Introducción

Este proyecto consiste en el desarrollo de un intérprete de Mini C, un lenguaje de programación simplificado basado en la sintaxis del lenguaje C. Su principal propósito es analizar, interpretar y ejecutar instrucciones básicas escritas en Mini C, permitiendo la evaluación de expresiones, el manejo de variables, la lectura y escritura de datos, así como el uso de estructuras de control como ciclos for.

Entre sus principales funcionalidades destacan:

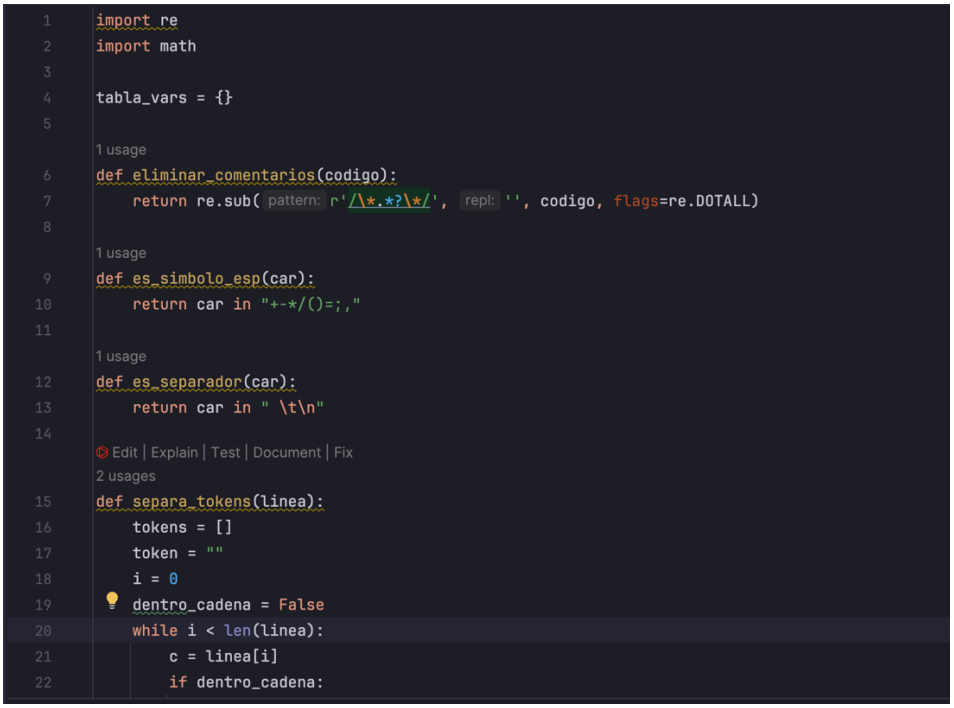
Declaración y asignación de variables.

Validación de errores comunes, como la redeclaración de variables o la omisión de punto y coma.

Ejecución de expresiones matemáticas, incluyendo funciones trigonométricas (sin(), cos(), tan()).

Entrada de datos mediante read() y salida con print() y println().

Implementación de ciclos for con evaluación de bloques internos.



```
import re
import math

tabla_vars = {}

1 usage
def eliminar_comentarios(codigo):
    return re.sub( pattern: r'/\*.*?\*/', repl: '', codigo, flags=re.DOTALL)

1 usage
def es_simbolo_esp(car):
    return car in "+-*/()=;,"

1 usage
def es_separador(car):
    return car in " \t\n"

Edit | Explain | Test | Document | Fix
2 usages
def separa_tokens(linea):
    tokens = []
    token = ""
    i = 0
    dentro_cadena = False
    while i < len(linea):
        c = linea[i]
        if dentro_cadena:
```

En las primeras lineas se importan las librerias re y math para poder realizar operaciones matematicas y expresiones regulares que son necesarias en nuestro interprete.

Se genera un diccionario global donde se guardan todas las variables declaradas por el usuario.

Cada variable tiene un tipo (int, float, string, char) y un valor.

Remueve comentarios estilo /* ... */ del código usando expresiones regulares.

Analisis Lexico.

es_simbolo_esp: verifica si un carácter es símbolo (+-*/()=;,)

es_separador: detecta espacios, tabulaciones o saltos de línea.

Toma una línea de código y la convierte en una lista de tokens individuales.

Detecta cadenas de texto, operadores como ==, y también ignora espacios en blanco.

Ejemplo: var int x;

Se convierte en ['var', 'int', 'x', ';']

Maneja:

Cadenas ("texto")

Operadores (+, =, ==)

Separadores

Palabras clave (var, read, print, etc.)

Se convierte en: ['var', 'int', 'x', ';']

```
55  def evaluar_expresion(tokens):
56      expr = ' '.join(tokens)
57      for var in tabla_vars:
58          expr = expr.replace(var, str(tabla_vars[var]["valor"]))
59      try:
60          return eval(expr, {"__builtins__": None}, {"sin": math.sin, "cos": math.cos, "tan": math.tan})
61      except:
62          print(f"Error al evaluar: {' '.join(tokens)}")
63          return None
64
```

Evaluación de expresiones.

Convierte los tokens en una cadena y evalúa la expresión matemática usando eval, reemplazando primero los nombres de variables por sus valores.

Solo permite funciones de trigonometría (sin, cos, tan) por seguridad.

```
def interpretar_linea(tokens):
    if not tokens:
        return

    if tokens[0] == "var" and tokens[1] in ["int", "float", "string", "char"]:
        nombre = tokens[2]
        if nombre in tabla_vars:
            print(f"Error: variable '{nombre}' ya declarada")
            return
        tabla_vars[nombre] = {"tipo": tokens[1], "valor": 0 if tokens[1] in ["int", "float"] else ""}
        return

    if tokens[0] == "read" and tokens[1] == "(" and tokens[3] == ")" and tokens[4] == ";":
        nombre = tokens[2]
        if nombre not in tabla_vars:
            print(f"Error: variable '{nombre}' no declarada")
            return
        tipo = tabla_vars[nombre]["tipo"]
        entrada = input(f"{nombre}? ")
        if tipo == "int":
            tabla_vars[nombre]["valor"] = int(entrada)
        elif tipo == "float":
            tabla_vars[nombre]["valor"] = float(entrada)
        elif tipo == "string":
            tabla_vars[nombre]["valor"] = str(entrada)
        else:
```

Se guarda en tabla_vars con tipo y valor inicial.

b. Lectura (read(nombre);)

Pide al usuario un valor y lo guarda en la variable.

c. Impresión

print() → imprime sin salto de línea.

println() → imprime con salto de línea.

```
    if tokens[0] in ["print", "println"] and tokens[1] == "(":
        contenido = []
        i = 2
        while tokens[i] != ")":
            if tokens[i] != ",":
                contenido.append(tokens[i])
            i += 1
        salida = ""
        for item in contenido:
            if item.startswith('"'):
                salida += item.strip('"')
            elif item in tabla_vars:
                salida += str(tabla_vars[item]["valor"])
            else:
                salida += item
        if tokens[0] == "println":
            print(salida)
        else:
            print(salida, end="")
        return

    if "=" in tokens:
        idx = tokens.index("=")
        nombre = tokens[idx - 1]
        if nombre not in tabla_vars:
            print(f"Error: variable '{nombre}' no declarada")
```

Imprime texto o variables. println agrega salto de línea; print, no.

Evalúa y guarda el resultado en tabla_vars[x]["valor"]

```
def interpretar_bloque(cuerpo_lineas):
    for linea in cuerpo_lineas:
        tokens = separa_tokens(linea.strip())
        interpretar_linea(tokens)

```

Interpreta una serie de líneas (por ejemplo, dentro de un for).

```
131  def main():
132      print("Intérprete Mini C - Ejecución directa")
133      print("Ingresa tu código (termina con 'end.'):")
134
135      codigo = ""
136      while True:
137          linea = input()
138          if linea.strip() == "end.":
139              break
140          codigo += linea + "\n"
141
142      codigo = eliminar_comentarios(codigo)
143      lineas = codigo.split("\n")
144      i = 0
145
146      while i < len(lineas):
147          linea = lineas[i].strip()
148          if not linea:
149              i += 1
150              continue
151
152          if not linea.endswith(";") and not linea.startswith("for"):
153              print(f"Error: falta ';' en línea {i+1}")
154              return
```

Es el nucleo de la ejecución del intérprete.

Esta es la función principal del programa:

- Pide al usuario que escriba su código hasta que escriba end..
- Elimina comentarios.
- Divide el código en líneas.
- Verifica si cada línea termina con ; (excepto ciclos).
- Si detecta un for, ejecuta su contenido varias veces.
- Llama a interpretar_linea() o interpretar_bloque() según sea necesario.

Quita los comentarios	10
Marca error cuando falta un “;”	10
Marca un error cuando una variable se redeclarada.	10
Implementa print() y println()	10
Implenta read()	10
Ejecuta expresiones	10
Implementa ciclos FOR	25
Implementa las funciones sin(), cos(), tan()	15

1. Quitar los comentarios.

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int x;
x = 5; /* Esto es un comentario que debe ser eliminado */
print (x);
end.
5
Process finished with exit code 0
```

2. Marcar error cuando falta un “;”

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int x
x = 5;
end.
Error: falta ';' en línea 1

Process finished with exit code 0
```

3. Marca error cuando ya es una variable se redeclara

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int x;
var int x;
end.
Error: variable 'x' ya declarada

Process finished with exit code 0
```

4. Implementacion de print y println

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int a;
a = 10;
print("Valor de a: ");
println(a);
end.
Valor de a: 10

Process finished with exit code 0
```

5. Implementacion de read ( )

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int edad;
read(edad);
println("Tu edad es:");
println(edad);
end.
edad? 20
Tu edad es:
20
```

6. Ejecuccion de operaciones

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int x;
x = 5 + 3 * 2;
println(x);
end.
11

Process finished with exit code 0
```

7. Implementa cicclos for

```
/usr/local/bin/python3.12 /Users/christianlara/Desktop/GitProye/compiladores/ProyectoFinal/main.py
Intérprete Mini C - Ejecución directa
Ingresa tu código (termina con 'end.'):
var int i;
for(i = 1; 3) {
    println(i);
}
end.
1
2
3

Process finished with exit code 0
```

8. Implementacion de funciones sin(), cos(), tan()