UNIVERSIDAD AUTONOMA DE AGUASCALIENTES

CENTRO DE CIENCIAS BÁSICAS
CAMPUS CENTRAL

Departamento de:
Sistemas Electrónicos.

Carrera:
Ingeniería en Sistemas Computacionales.

Materia: Compiladores I.

Análisis Sintáctico

Integrantes:
Gustavo Adolfo Avendaño Guevara. ID: 250884.
Pedro Román García Delgado. ID: 229338.
Fabián Reyes Medina. ID: 261274.
Semestre: 8. ***Grupo: A.***

Docente: Dra. Blanca Guadalupe Estrada Rentería.

Fecha de entrega: 23 de junio de 2025.

Tabla de contenido

Gramática del lenguaje

La gramática implementada es la de un análisis sintáctico descendente recursivo y define las siguientes reglas de producción:

programa → main { lista_declaracion }

lista_declaracion → lista_declaracion declaracion | declaracion

declaracion → declaracion_variable | sentencia

declaracion_variable → tipo identificador ;

identificador → id | identificador , id

sentencia → seleccion | iteracion | repeticion | sent_in | sent_out | asignacion

seleccion → if expresion then lista_sentencias [else lista_sentencias] end

iteracion → while expresion lista_sentencias end

repeticion → do lista_sentencias until expresion

sent_in → cin >> id ;

sent_out → cout << salida ;

salida → cadena | expresion

asignacion → id = sent_expresion ;

sent_expresion → expresion ; | ;

expresion → expresion_logica

expresion_logica → expresion_relacional [operador_logico expresion_relacional]

expresion_relacional → expresion_simple [operador_relacional expresion_simple]

expresion_simple → termino [operador_suma termino]

termino → factor [operador_multiplicacion factor]

factor → componente [^ componente]

componente → (expresion) | numero | id | bool | operador_logico componente

Estructura del proyecto

PyGFrame/

|---------- pygframe.py → Interfaz gráfica principal (IDE)

|---------- lexico.py → Analizador léxico

|---------- sintactico.py → Analizador sintáctico

|---------- ast_visualizer.py → Visualizador del AST

|---------- integration.py → Integración del analizador con el IDE

|---------- images/ → Iconos e imágenes necesarias

Diagrama de componentes

Características

Análisis Sintáctico

- Analizador descendente recursivo con manejo de errores.
- Construcción de AST (Árbol Sintáctico Abstracto) completo.
- Detección y reporte de errores sintácticos con ubicación precisa.
- Validación de estructura según la gramática definida.

Visualización del AST

- Árbol interactivo con capacidad de colapsar/expandir nodos.
- Agrupación inteligente de variables bajo sus tipos de datos.
- Expansión automática de operadores a su forma completa.
- Filtrado de nodos para mostrar información relevante.

Integración con el IDE

- Análisis en tiempo real desde la interfaz gráfica.
- Pestañas organizadas para AST y errores sintácticos.
- Sincronización con el análisis léxico previo.

Instalación y configuración

Requisitos del Sistema

- Python 3.7 o superior
- Tkinter (incluido en Python estándar)
- PIL (Pillow) para manejo de imágenes
- Módulo json (incluido en Python estándar)

Instalación

Descargar los archivos del proyecto

Instalar dependencias

pip install Pillow

Ejecución

python pygframe.py

Módulos del Sistema

1. sintactico.py - Analizador Sintáctico

Clase Nodo

Representa un nodo del Árbol Sintáctico Abstracto (AST).

Atributos principales:

- tipo: Tipo del nodo (PROGRAMA, DECLARACION, etc.)
- valor: Valor asociado al nodo (opcional)
- linea: Línea en el código fuente
- columna: Columna en el código fuente
- hijos: Lista de nodos hijos
- padre: Referencia al nodo padre

Métodos clave:

```
def agregar_hijo(self, hijo):
  # Agrega un hijo al nodo y establece la relación padre-hijo
def to_dict(self):
  # Convierte el nodo a diccionario para serialización
```

Clase AnalizadorSintactico

Implementa el analizador sintáctico descendente recursivo.

Métodos principales:

Método	Descripción
analizar()	Inicia el análisis sintáctico completo
programa()	Analiza la estructura principal del programa

declaracion_variable()	Procesa declaraciones de variables
sentencia()	Analiza diferentes tipos de sentencias
expresion()	Procesa expresiones con precedencia de operadores
error()	Registra errores sintácticos con ubicación

Tipos de nodos del AST:

Tipo de Nodo	Descripción	Hijos
PROGRAMA	Nodo raíz del AST	Lista de declaraciones
DECLARACION_VARIABLE	Declaración de variables	Tipo, identificadores
SELECCION	Estructura if-then-else	Expresión, sentencias if, sentencias else
ITERACION	Estructura while	Expresión, lista de sentencias
REPETICION	Estructura do-until	Lista de sentencias, expresión
ASIGNACION	Asignación de valores	Identificador, expresión
SENT_IN	Sentencia de entrada	Identificador
SENT_OUT	Sentencia de salida	Expresión o cadena

2. ast_visualizer.py - Visualizador del AST

Clase VisualizadorAST

Proporciona visualización interactiva del AST en el IDE.

Componentes principales:

- **Panel de controles**: Botones para expandir/colapsar árbol
- **TreeView**: Componente visual del árbol con columnas informativas

Esquema en el TreeView:

Elemento	Información mostrada
Estructura	Jerarquía del AST con indentación
Tipo	Tipo del nodo (PROGRAMA, EXPRESION, etc.)
Valor	Valor del token (operadores, identificadores, números)
Línea	Línea en el código fuente
Columna	Columna en el código fuente

Guía de Uso

Inicio Rápido

1. **Ejecutar la aplicación:**
2. python pygframe.py
3. **Escribir o cargar código:**
    - El editor soporta la sintaxis definida por la gramática
    - El resaltado léxico se aplica automáticamente
4. **Realizar análisis sintáctico:**
    - **Menú:** Análisis > Análisis Sintáctico
    - **Toolbar:** Hacer clic en el icono "S"
    - Los resultados aparecen en la pestaña "Sintáctico"
5. **Visualizar el AST:**
    - La pestaña "AST" muestra el árbol sintáctico
    - Usar botones "Expandir Todo" / "Colapsar Todo"
    - Clic en nodos para colapsar/expandir individualmente

Ejemplo de Código Soportado

```
main {
  int x, y, z;
```

```
float a, b, c;
suma = 45;
x = 32.32;
x = 23;
y = 2 + 3 - 1;
z = y + 7;
y = y + 1;
a=24.0+4-1/3*2+34-1;
x=(5-3)*(8/2);
y=5+3-2*4/7-9;
z = 8 / 2 + 15 * 4;
y = 14.54;
if 2>3 then
   y = a + 3;
else
   if 4>2 && true then
      b = 3.2;
   else
      b = 5.0;
   end
   y = y + 1;
end
a++;
c--;
x = 3 + 4;
do
```

```
    y = (y + 1) * 2 + 1;
    while x > 7
      x = 6 + 8 / 9 * 8 / 3;
      cin >> x;
      mas = 36 / 7;
    end
  until y == 5
  while y == 0
    cin >> mas;
    cout << x;
  end
}
```

Interpretación de Resultados

Pestaña Sintáctico

Muestra información del análisis:

- **Estado del análisis:** Exitoso o con errores
- **Información** de tipos de nodos encontrados

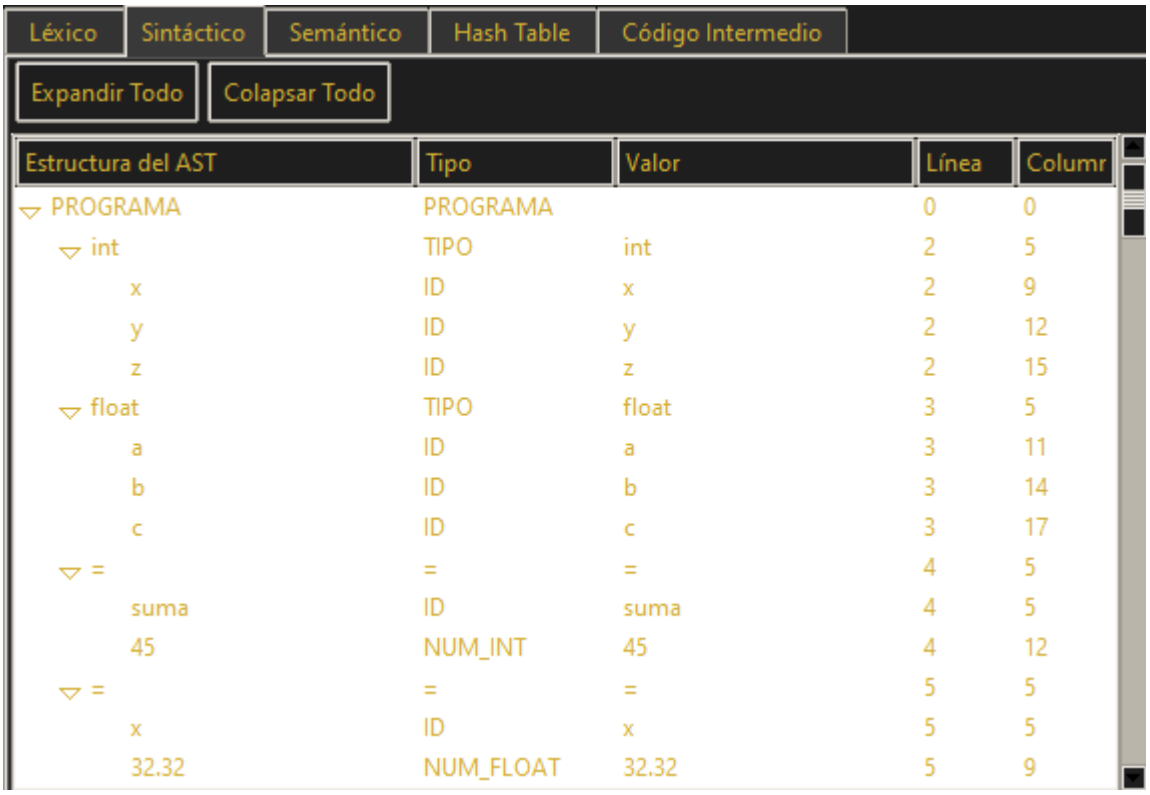
Pestaña Errores Sintácticos

Lista errores con detalles completos:

TIPO	DESCRIPCIÓN	LÍNEA	COLUMNA
Error Sintáctico	Se esperaba ';' después de declaración	3	15
Error Sintáctico	Se esperaba 'end' al final del if	8	5

AST

Visualización interactiva del árbol:



Especificaciones Técnicas

Estructura del AST

El AST se construye como una estructura de nodos con las siguientes características:

```
Nodo = {
  'tipo': str,      # Tipo del nodo
  'valor': str|None,  # Valor asociado (opcional)
  'linea': int,     # Línea en código fuente
  'columna': int,    # Columna en código fuente
  'hijos': [Nodo]    # Lista de nodos hijos
}
```

Algoritmo de Análisis

1. **Inicialización**: Se carga la lista de tokens del análisis léxico
2. **Análisis descendente**: Se inicia desde la regla programa

3. **Construcción de nodos**: Cada regla gramatical crea nodos en el AST
4. **Manejo de errores**: Se registran errores sin detener el análisis
5. **Validación**: Se verifica la estructura completa del programa
6. **Postprocesamiento**: Se optimiza y reorganiza el AST para visualización

Manejo de Errores

El analizador implementa **recuperación de errores** mediante:

- **Modo pánico**: Continúa el análisis después de encontrar un error
- **Sincronización**: Busca tokens de sincronización (;, end, etc.)
- **Contexto**: Proporciona información detallada sobre el error esperado vs encontrado
- **Ubicación precisa**: Línea y columna exacta del error

Integración con el IDE

El analizador sintáctico se integra completamente con el IDE mediante:

- **Análisis**: Funciones estandarizadas para análisis y visualización.
- **Sincronización**: Coordinación con el análisis léxico previo.