

Práctica 09

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Compiladores

PRÁCTICA	TEMA	DURACIÓN
09	Recursive descent parsing	3 horas

1. Datos de los estudiantes

- Grupo: 1
- Git Hub: <https://github.com/CrazyDani17/Practica9-Compiladores>
- Integrantes:
 - Guillermo Alemán
 - Marvik Del Carpio
 - Daniel Mendiguri
 - Daniela Vilchez

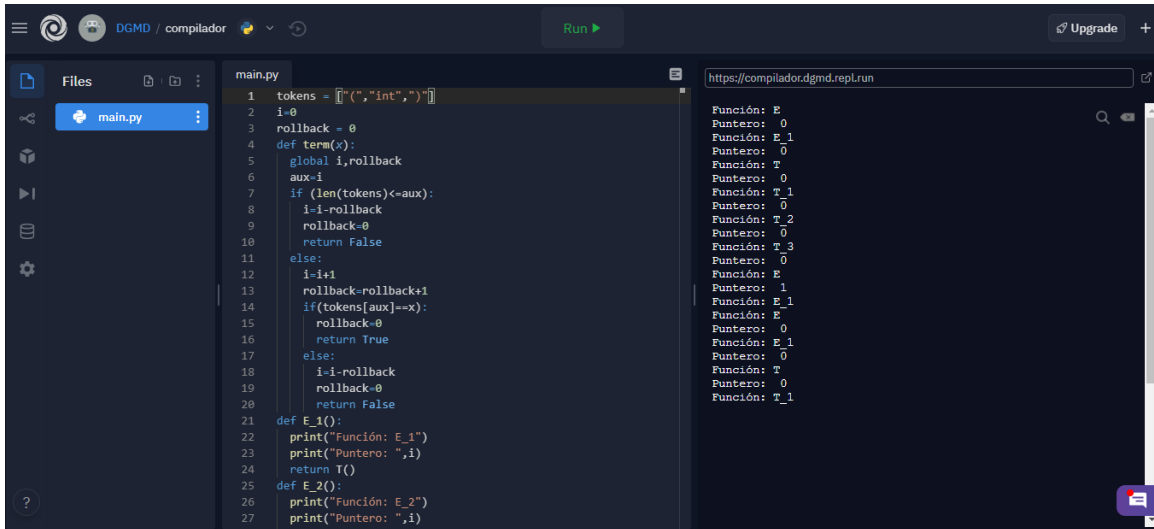
2. Ejercicios

1. Implemente el algoritmo recursive descent parsing para la siguiente gramática. Puede utilizar el lenguaje de su preferencia. La entrada al programa es una lista de tokens y la salida es un mensaje, indicando si la lista de tokens pertenece a la gramática

```
E -> T | T + E
T -> int | int * T | ( E )
```

Lenguaje utilizado: Python

```
tokens = ["(", "(", "(", "(", "int", ")", ")", ")", ")"]
i=0
rollback = 0
def term(x):
    global i, rollback
    aux=i
    if (len(tokens)<=aux):
        i=i-rollback
        rollback=0
        return False
    else:
        i=i+1
        rollback=rollback+1
        if(tokens[aux]==x):
            rollback=0
            return True
        else:
            i=i-rollback
            rollback=0
            return False
def E_1():
    print("Funcin: E_1")
    print("Puntero: ",i)
    return T()
def E_2():
    print("Funcin: E_2")
    print("Puntero: ",i)
    return T() and term ("+") and E()
def E():
    print("Funcin: E")
    print("Puntero: ",i)
    return E_1() or E_2()
def T_1():
    print("Funcin: T_1")
    print("Puntero: ",i)
    return term("int")
def T_2():
    print("Funcin: T_2")
    print("Puntero: ",i)
    return term("int") and term("*") and T()
def T_3():
    print("Funcin: T_3")
    print("Puntero: ",i)
    return term("(") and E() and term(")")
def T():
    print("Funcin: T")
    print("Puntero: ",i)
    return T_1() or T_2() or T_3()
print(E())
```



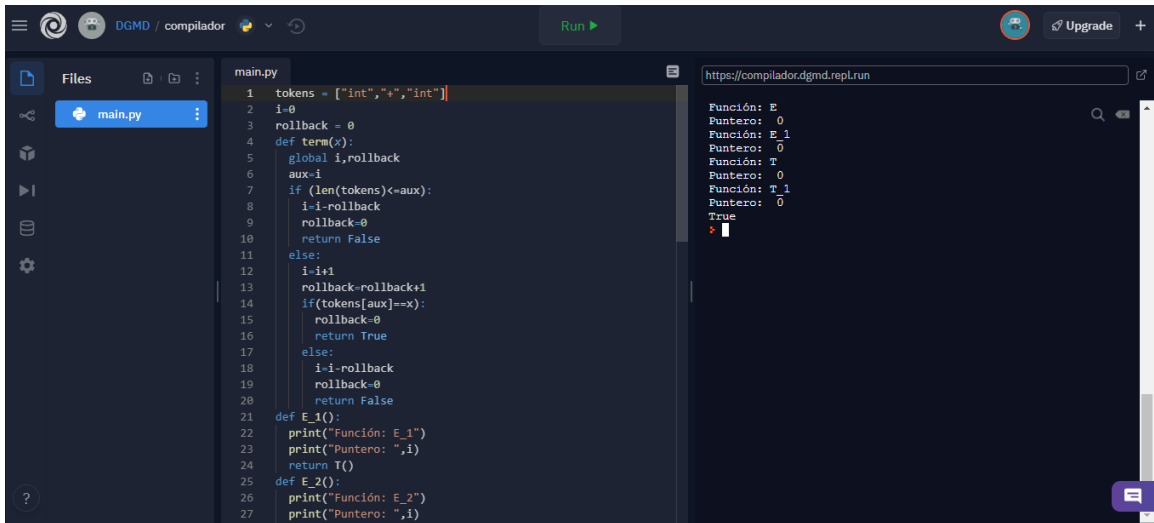
```

1 tokens = ['(', 'int', ')']
2 i=0
3 rollback = 0
4 def term(x):
5     global i,rollback
6     aux=i
7     if (len(tokens)<=aux):
8         i=i-rollback
9         rollback=0
10        return False
11    else:
12        i=i+1
13        rollback-rollback+1
14        if(tokens[aux]==x):
15            rollback=0
16            return True
17        else:
18            i=i-rollback
19            rollback=0
20            return False
21 def E_1():
22     print("Función: E_1")
23     print("Puntero: ",i)
24     return T()
25 def E_2():
26     print("Función: E_2")
27     print("Puntero: ",i)

```

Función: E
Puntero: 0
Función: E_1
Puntero: 0
Función: T
Puntero: 0
Función: T_1
Puntero: 0
Función: T_2
Puntero: 0
Función: E_3
Puntero: 0
Función: E
Puntero: 1
Función: E_1
Función: E
Puntero: 0
Función: E_1
Puntero: 0
Función: T
Puntero: 0
Función: T_1

Figura 1: Prueba 1 “(int)”



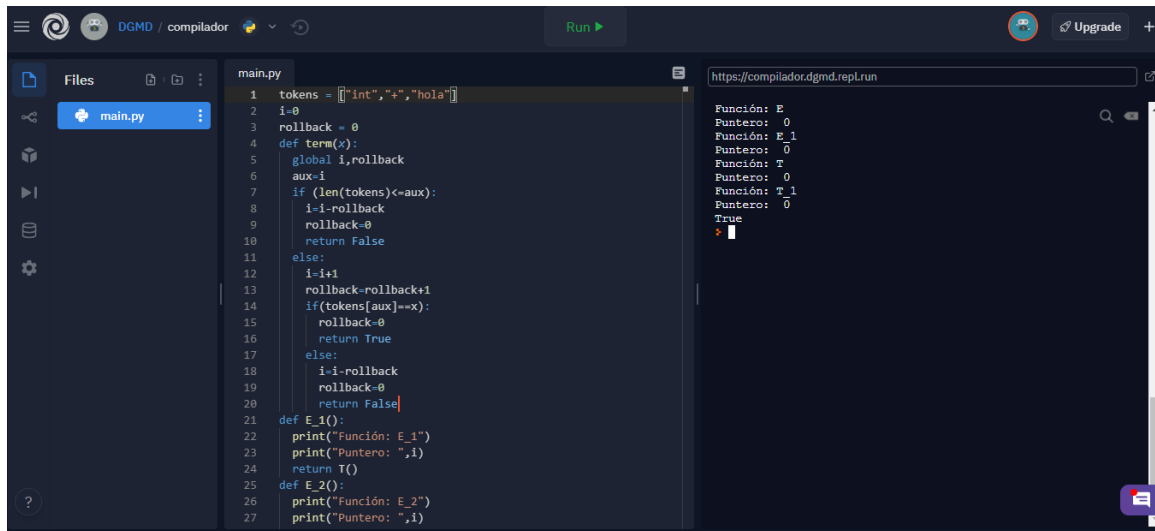
```

1 tokens = ['int', '+', 'int']
2 i=0
3 rollback = 0
4 def term(x):
5     global i,rollback
6     aux=i
7     if (len(tokens)<=aux):
8         i=i-rollback
9         rollback=0
10        return False
11    else:
12        i=i+1
13        rollback-rollback+1
14        if(tokens[aux]==x):
15            rollback=0
16            return True
17        else:
18            i=i-rollback
19            rollback=0
20            return False
21 def E_1():
22     print("Función: E_1")
23     print("Puntero: ",i)
24     return T()
25 def E_2():
26     print("Función: E_2")
27     print("Puntero: ",i)

```

Función: E
Puntero: 0
Función: E_1
Puntero: 0
Función: T
Puntero: 0
Función: T_1
Puntero: 0
True
Puntero: 0

Figura 2: Prueba 2 “int + int”



The screenshot shows the DGMD compiler interface with the following code in `main.py`:

```

1 tokens = ["int", "+", "hola"]
2 i=0
3 rollback = 0
4 def term(x):
5     global i, rollback
6     aux=i
7     if (len(tokens)<=aux):
8         i=i-rollback
9         rollback=0
10        return False
11    else:
12        i=i+1
13        rollback=rollback+1
14        if(tokens[aux]==x):
15            rollback=0
16            return True
17        else:
18            i=i-rollback
19            rollback=0
20            return False
21 def E_1():
22     print("Función: E_1")
23     print("Puntero: ",i)
24     return T()
25 def E_2():
26     print("Función: E_2")
27     print("Puntero: ",i)

```

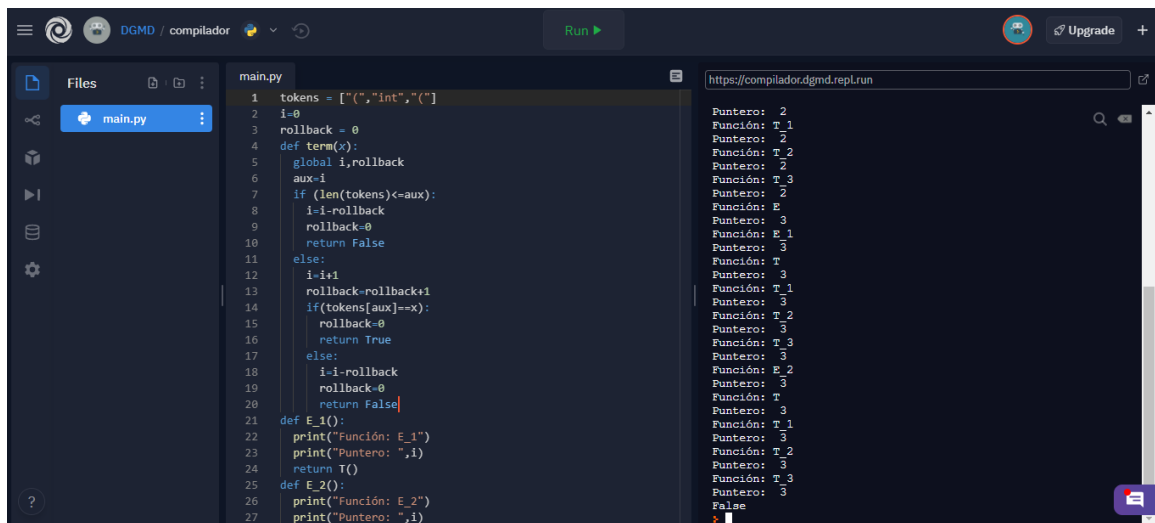
The output window on the right shows the execution results:

```

Función: E
Puntero: 0
Función: E_1
Puntero: 0
Función: T
Puntero: 0
Función: T_1
Puntero: 0
True

```

Figura 3: Prueba 3 “int + hola”



The screenshot shows the DGMD compiler interface with the following code in `main.py`:

```

1 tokens = ["(", "int", "("]
2 i=0
3 rollback = 0
4 def term(x):
5     global i, rollback
6     aux=i
7     if (len(tokens)<=aux):
8         i=i-rollback
9         rollback=0
10        return False
11    else:
12        i=i+1
13        rollback=rollback+1
14        if(tokens[aux]==x):
15            rollback=0
16            return True
17        else:
18            i=i-rollback
19            rollback=0
20            return False
21 def E_1():
22     print("Función: E_1")
23     print("Puntero: ",i)
24     return T()
25 def E_2():
26     print("Función: E_2")
27     print("Puntero: ",i)

```

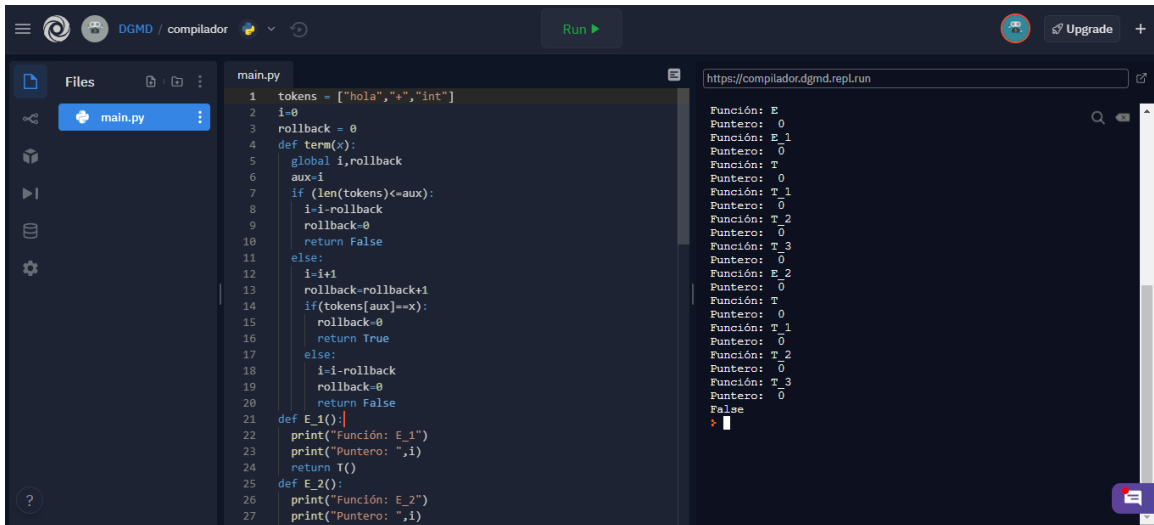
The output window on the right shows the execution results:

```

Puntero: 2
Función: T_1
Puntero: 2
Función: T_2
Puntero: 2
Función: T_3
Puntero: 2
Función: E
Puntero: 3
Función: E_1
Puntero: 3
Función: T
Puntero: 3
Función: T_1
Puntero: 3
Función: T_2
Puntero: 3
Función: T_3
Puntero: 3
Función: E_2
Puntero: 3
Función: T
Puntero: 3
Función: T_1
Puntero: 3
Función: T_2
Puntero: 3
Función: T_3
Puntero: 3
False

```

Figura 4: Prueba 4 “(int(”



The screenshot shows a web-based IDE interface. The top bar includes a menu icon, a logo, the text "DGMD / compilador", a "Run" button, and an "Upgrade" button. The left sidebar shows a "Files" panel with "main.py" selected. The main editor displays the following Python code:

```
1 tokens = ["hola", "+", "int"]
2 i=0
3 rollback = 0
4 def term(x):
5     global i,rollback
6     aux=i
7     if (len(tokens)<=aux):
8         i=i-rollback
9         rollback=0
10        return False
11    else:
12        i=i+1
13        rollback=rollback+1
14        if(tokens[aux]==x):
15            rollback=0
16            return True
17        else:
18            i=i-rollback
19            rollback=0
20            return False
21 def E_1():
22     print("Función: E_1")
23     print("Puntero: ",i)
24     return T()
25 def E_2():
26     print("Función: E_2")
27     print("Puntero: ",i)
```

The right panel shows the execution output:

```
Función: E
Puntero: 0
Función: E_1
Puntero: 0
Función: T
Puntero: 0
Función: T_1
Puntero: 0
Función: T_2
Puntero: 0
Función: T_3
Puntero: 0
Función: E_2
Puntero: 0
Función: T
Puntero: 0
Función: T_1
Puntero: 0
Función: T_2
Puntero: 0
Función: T_3
Puntero: 0
False
>
```

Figura 5: Prueba 5 “hola + int”