

Práctica 1

Compiladores (CS3402) 2021-0

Integrantes:

- Mayra Díaz Tramontana 201910147
- Joaquín Elías Ramírez Gutiérrez 201910277

Ejercicio 1:

¿Qué cadenas genera la gramática con la regla $A \rightarrow (A)A|\epsilon$? Formule la derivación de la cadena $((()()))()$

Genera expresiones de paréntesis balanceados.

((()())) m.i.

[illegible]

Ejercicio 2:

¿Qué cadenas genera la gramática ?

sec-sent \rightarrow sent; sec-sent | sent

sent \rightarrow s

Genere el árbol de análisis gramatical y abstracto para la cadena s;s,s,

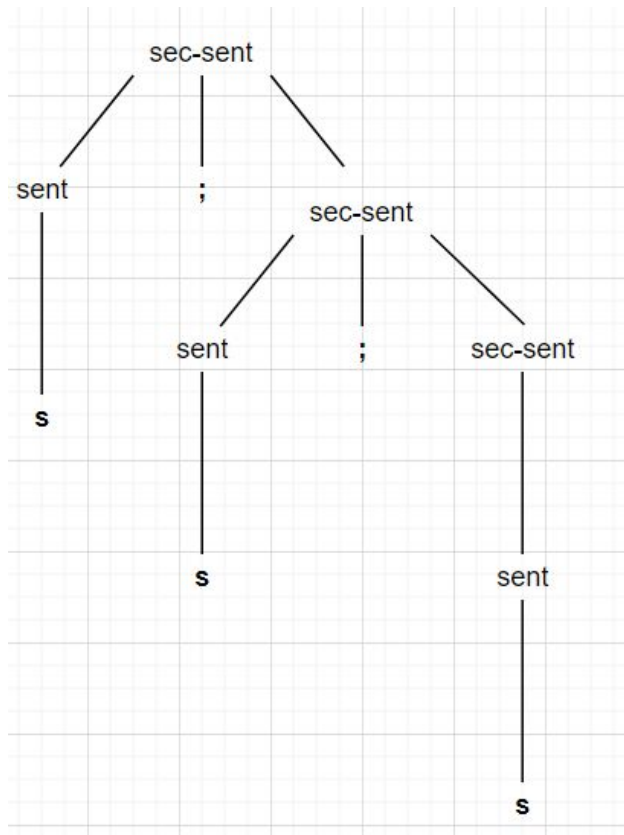
¿Cómo lograría que hayan secuencias de sentencias vacías?

Genera una secuencia de 's' separadas por ';' (la última no lleva ';' al final). No puede haber secuencias de sentencias vacías.

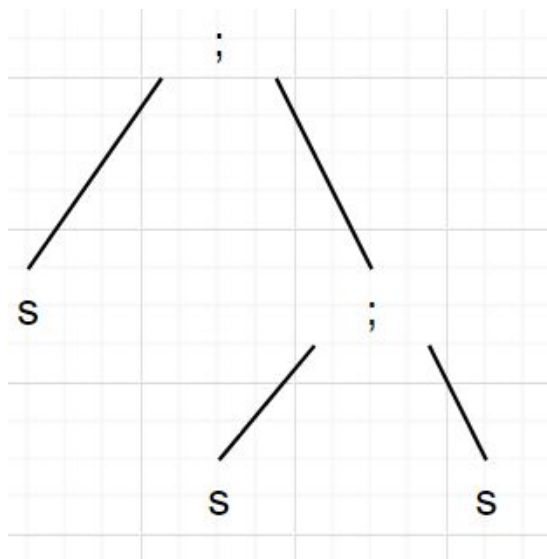
s;s;s m.i.

sec-sent \rightarrow sent; sec-sent \rightarrow s; sec-sent \rightarrow s; sent; sec-sent \rightarrow s;s; sec-sent \rightarrow s;s;s
 \rightarrow s;s;s

Gramatical:



Abstracto:



Ejercicio 3:

Genere el árbol de análisis gramatical y abstracto para la cadena
if (0) otro else otro

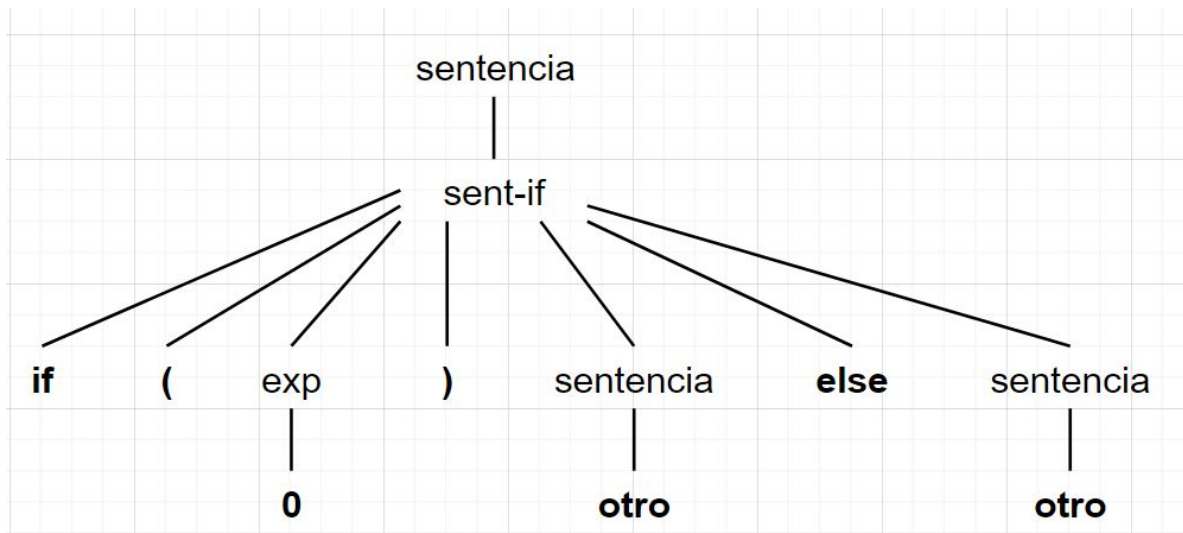
según la gramática:

sentencia \rightarrow sent-if | otro

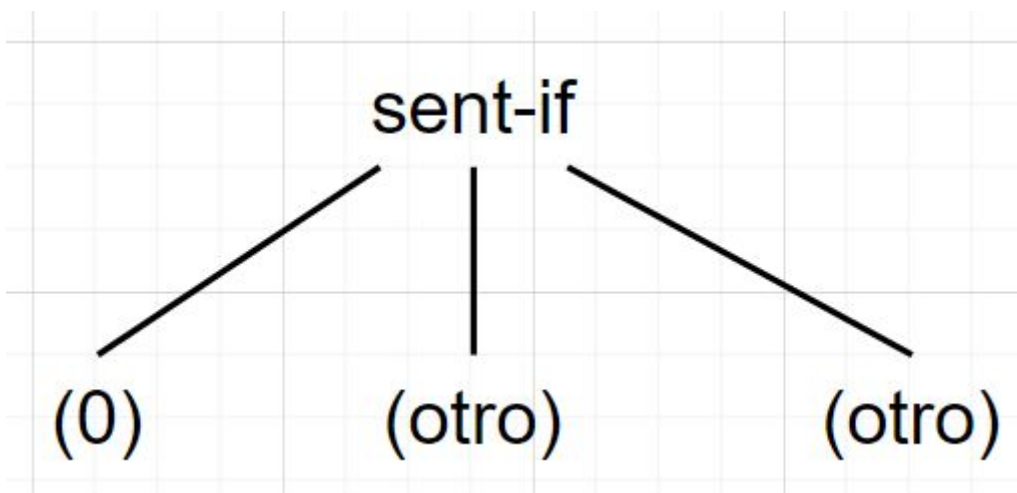
sent-if \rightarrow if (exp) sentencia | if (exp) sentencia else sentencia

exp \rightarrow 0|1

Gramatical:



Abstracto:



Ejercicio 4:

Genere derivaciones por la izquierda, árboles de análisis gramatical abstractos según la gramática:

$\text{exp} \rightarrow \text{exp opsuma term} \mid \text{term}$

$\text{opsuma} \rightarrow + \mid -$

$\text{term} \rightarrow \text{term opmult factor} \mid \text{factor}$

$\text{opmult} \rightarrow *$

$\text{factor} \rightarrow (\text{exp}) \mid \text{numero}$

Para

a) $3 + 4 * 5 - 6$

b) $3 * (4 - 5 + 6)$

c) $3 - (4 + 5 * 6)$

a) **$3 + 4 * 5 - 6$**

m.i.

$\text{exp} \rightarrow$

$\text{exp opsuma term} \rightarrow$

$\text{exp opsuma term opsuma term} \rightarrow$

$\text{term opsuma term opsuma term} \rightarrow$

$\text{factor opsuma term opsuma term} \rightarrow$

$3 \text{ opsuma term opsuma term} \rightarrow$

$3 + \text{term opsuma term} \rightarrow$

$3 + \text{term opmult factor opsuma term} \rightarrow$

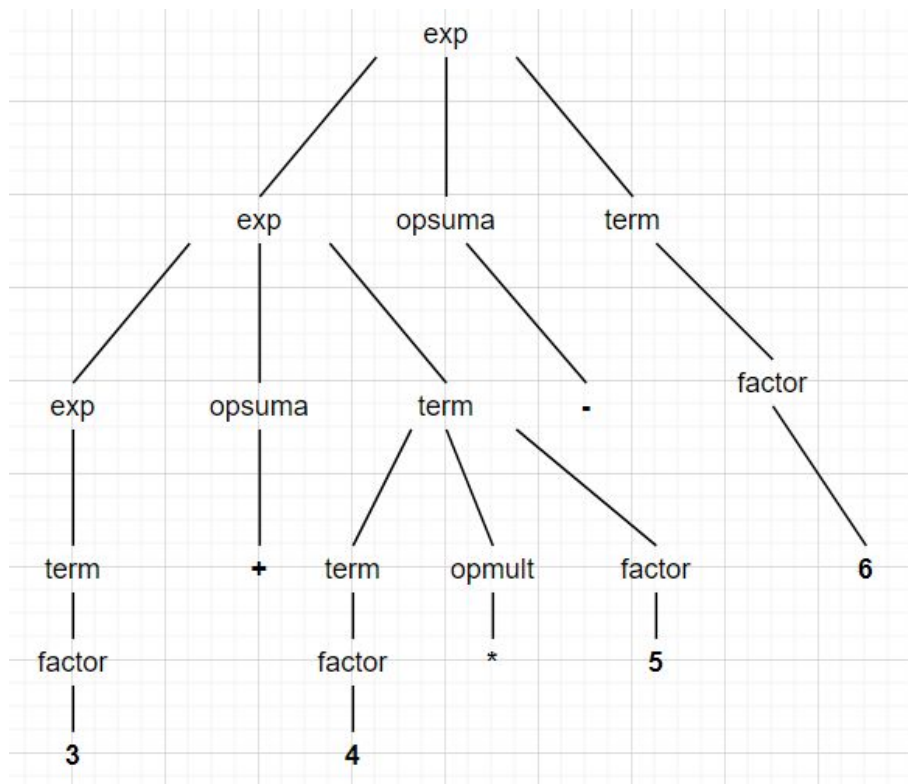
$3 + \text{factor opmult factor opsuma term} \rightarrow$

$3 + 4 \text{ opmult factor opsuma term} \rightarrow$

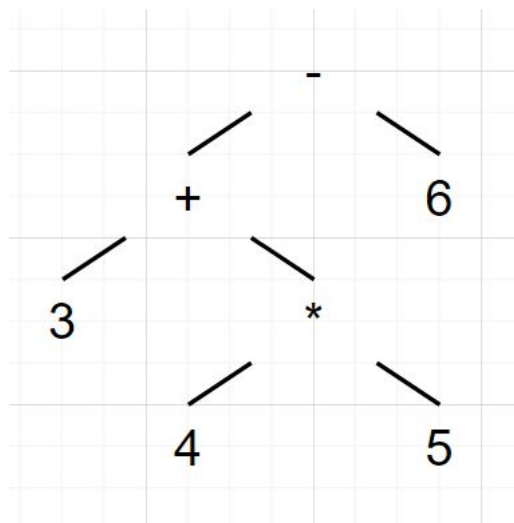
$3 + 4 * \text{factor opsuma term} \rightarrow$

$3 + 4 * 5 \text{ opsuma term} \rightarrow 3 + 4 * 5 - \text{term} \rightarrow 3 + 4 * 5 - \text{factor} \rightarrow 3 + 4 * 5 - 6$

Gramatical:



Abstracto:

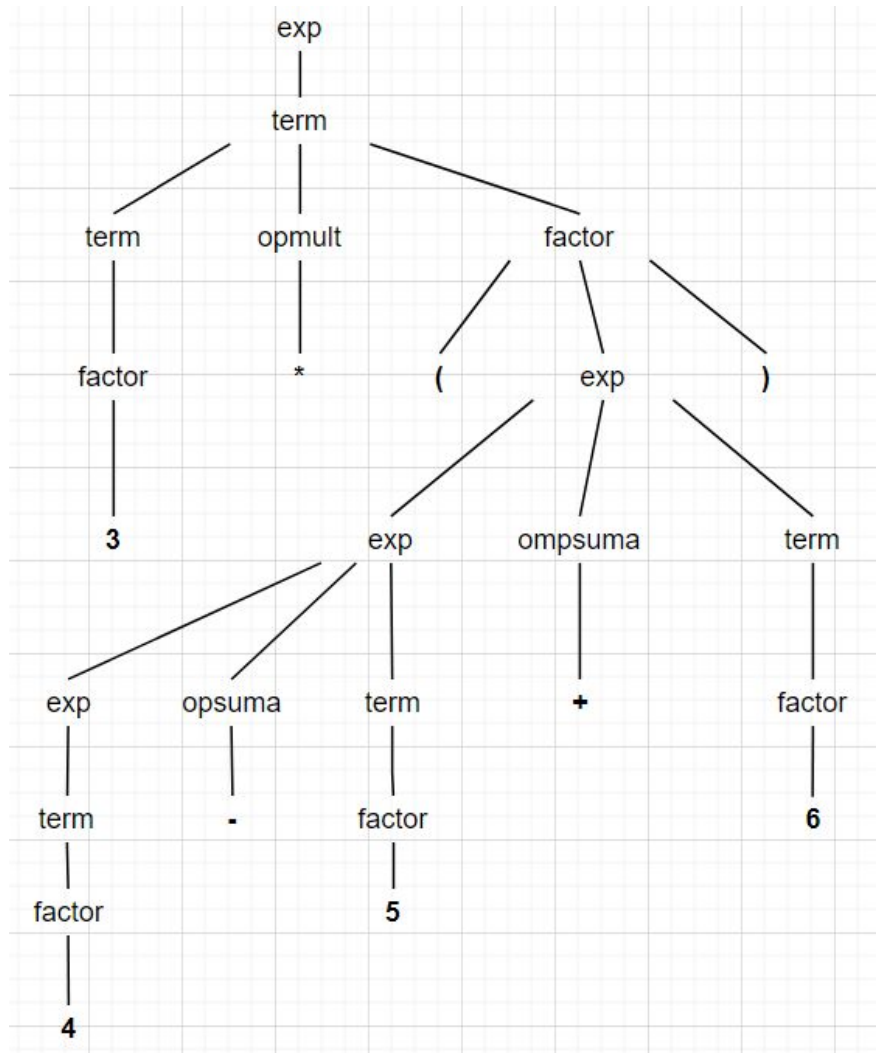


b) $3 * (4 - 5 + 6)$

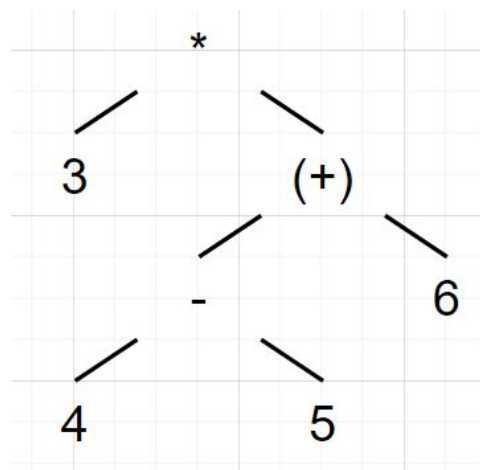
m.i.

$\text{exp} \rightarrow \text{term} \rightarrow \text{term opmult factor} \rightarrow \text{factor opmult factor} \rightarrow 3 \text{ opmult factor}$
 $\rightarrow 3 * \text{factor} \rightarrow 3 * (\text{exp}) \rightarrow 3 * (\text{exp opsuma term}) \rightarrow 3 * (\text{exp opsuma term opsuma term})$
 $\rightarrow 3 * (\text{term opsuma term opsuma term}) \rightarrow 3 * (\text{factor opsuma term opsuma term})$
 $\rightarrow 3 * (4 \text{ opsuma term opsuma term}) \rightarrow 3 * (4 - \text{term opsuma term})$
 $\rightarrow 3 * (4 - \text{factor opsuma term}) \rightarrow 3 * (4 - 5 \text{ opsuma term}) \rightarrow 3 * (4 - 5 + \text{term})$
 $\rightarrow 3 * (4 - 5 + \text{factor}) \rightarrow 3 * (4 - 5 + 6)$

Gramatical:



Abstracto:



c) $3 - (4 + 5 * 6)$

m.i.

exp \rightarrow

exp opsuma term \rightarrow term opsuma term \rightarrow factor opsuma term \rightarrow

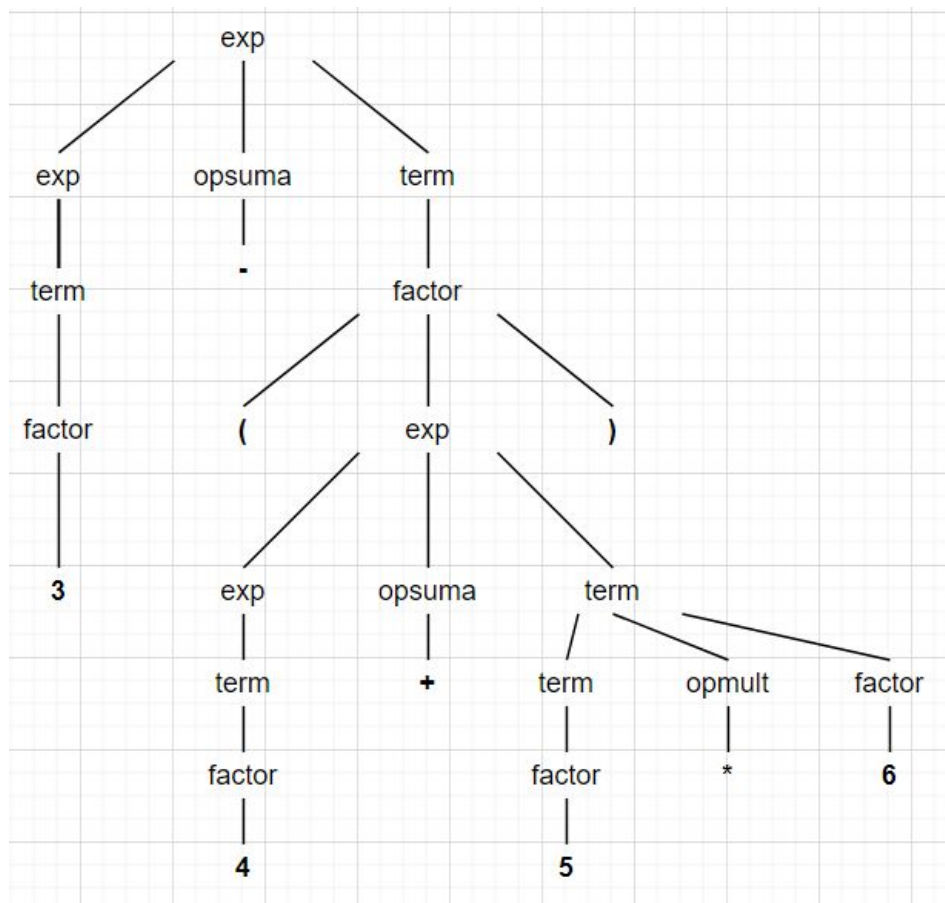
3 opsuma term \rightarrow 3 - term \rightarrow 3 - factor \rightarrow 3 - (exp) \rightarrow 3 - (exp opsuma term) \rightarrow

\rightarrow 3 - (term opsuma term) \rightarrow 3 - (factor opsuma term) \rightarrow 3 - (4 opsuma term) \rightarrow

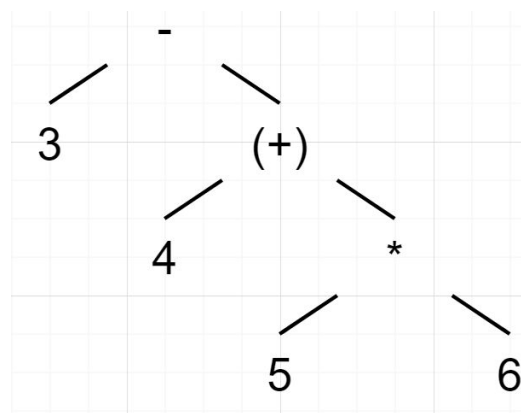
\rightarrow 3 - (4 + term) \rightarrow 3 - (4 + term opmult factor) \rightarrow 3 - (4 + factor opmult factor) \rightarrow

\rightarrow 3 - (4 + 5 opmult factor) \rightarrow 3 - (4 + 5 * factor) \rightarrow 3 - (4 + 5 * 6)

Gramatical:



Abstracto:



Ejercicio 5:

Elabore un programa en C++ que permita ingresar dos numeros enteros y compararlos con la funcion `assert(x < y)`. Si $x < y$ se imprime un mensaje con este resultado. En caso contrario termina el programa. Permita activar/desactivar `assert()` utilizando la directiva de preprocesamiento `NDEBUG`. Si esta definida, `assert` se desactiva. En caso contrario, se activa. Decida si definir (o no) `NDEBUG` al momento de compilacion.

Opcional: utilice un Makefile para elegir entre la opcion de compilacion con `NDEBUG` o sin `NDEBUG`.

Link de los archivos:

<https://drive.google.com/drive/folders/1tEsYSh8eK1XuAz6CouIVJ3zQK1wjrQaM?usp=sharing>

ejercicio5.cpp:

```
#include <iostream>
#define assert(x,y) (x<y)

int main(int argc, char *argv[]) {
    int x, y;
    std::cin >> x >> y;
    #ifndef NDEBUG
        if (assert(x, y))
            std::cout << x << " < " << y << '\n';
    #endif
}
```

Makefile:

```
#make ndebug=1 ej5          // ndebug ON
#make ej5                  // ndebug OFF
ej5:
ifdef ndebug
    g++ -DNDEBUG ejercicio5.cpp
else
    g++ ejercicio5.cpp
endif
clean:
    rm -r a.out
```