Práctica 2
Analizador Sintáctico

Dado un programa en lenguaje PsiCoder, su tarea consiste en realizar el análisis sintáctico. Nos vamos a enfocar en los errores sintácticos generados. A continuación se muestra la manera correcta de generar las salidas correspondientes.

En caso de que el programa esté bien formado de acuerdo a las reglas de la gramática de PsiCoder, se debe mostrar el mensaje:

“El analisis sintactico ha finalizado exitosamente."

En caso contrario, es decir, si se encontró algún error sintáctico, se debe abortar el análisis y reportar únicamente el primer error sintáctico detectado.

Consideraciones gramaticales

A continuación se presentan algunas reglas de la gramática de PsiCoder a tener en cuenta:

- Todo programa en PsiCoder debe contener la función principal, la cual puede estar antecedida y/o precedida de declaraciones de otras funciones y estructuras.

  Ejemplo:

```
funciones y estructuras…

funcion_principal
   // codigo de funcion principal
fin_principal

funciones y estructuras…
```

  En caso de no contener la función principal se debe mostrar el siguiente error:

  Error sintactico: falta funcion_principal

- Cada comando del programa con las siguientes características debe terminar con el carácter ‘;’
  1. Después de la creación de una o más variables(separadas por ‘,’)

Ejemplo:

```
funcion_principal
  entero a;
  real x,y=0,z;
fin_principal
```

2. Después de la asignación de una variable
   Ejemplo:

```
funcion_principal
  entero a;
  a = 3 * (5) / b;
  a = a + 1          ;
  a = x;c = 1;
  b = “lenguajes de programacion”;
fin_principal
```

3. Al finalizar los comandos ‘imprimir’,‘leer’,’romper’,
   Ejemplo:

```
funcion_principal
  entero a;
  leer( a );
  imprimir( “ variable ”, a );
fin_principal
```

4. Al realizar el llamado de una función
   Ejemplo:

```
funcion entero sum(entero a, entero b) hacer
    retornar a + b;
fin_funcion
funcion_principal
    sum(1,2);
fin_principal
```

- El comando ‘imprimir’ debe ir seguido de paréntesis ‘(’, no puede ir vacío y debe terminar con paréntesis ‘)’.

Ejemplo:

```
funcion_principal
    imprimir( “letra ” , letter , “ contador ”, (cont * cont)  );
    imprimir( “total”, cont * 2 + 1 )
    ; // no es un error sintactico
    imprimir( “total”, cont * 2 + 1; //error sintáctico
```

```
    imprimir;( “total”, cont * 2 + 1);//error sintáctico
    imprimir( );//error sintáctico
fin_principal
```

- El comando ‘leer’ debe ir seguido de paréntesis ‘(’, no puede ir vacío, debe leer un identificador, y debe terminar con paréntesis ‘)’.
  Ejemplo:

```
funcion_principal
    leer( a );
    leer( a,b ); //error sintáctico
    leer( “a” ); //error sintáctico
    leer( a ; //error sintáctico
    leer( a );leer( b );
    leer( ); //error sintáctico
fin_principal
```

- El condicional si debe tener el siguiente formato;
  Ejemplo:

```
funcion_principal
    si ( num1 > num2 ) entonces
       res = num1;
    si_no
       res = num2;
    fin_si

 si ( num1 > num2 ) //error sintáctico, falta ‘entonces’
       res = num1;
    si_no
       res = num2;
    fin_si
fin_principal
```

  Consideraciones:
  - ❏ Después de realizar la condición debe continuar la palabra ‘entonces’.
  - ❏ El si_no es opcional.
  - ❏ Siempre que se crea un condicional si, este debe terminar con la palabra ‘fin_si’. En caso de agregar el si_no a la condición igualmente se debe terminar con un ‘fin_si’
  - ❏ En los condicionales y las estructuras de control cíclicas (mientras, para, hacer .. mientras) la condición siempre debe estar encerrada entre paréntesis ‘(’ ‘)’.

- El ciclo para debe tener el siguiente formato:

```
funcion_principal
    para ( entero i = 0 ; i < a ; 1 ) hacer
        imprimir(i);
    fin_para
    para ( j = 0; i < a || j  < 100  ; p ) hacer
        imprimir(j);
    fin_para
fin_principal
```

  En la parte inicial del ciclo sólo puede existir la inicialización de una única variable (por ejemplo: entero i = 0).

- El ciclo mientras debe tener el siguiente formato
  Ejemplo:

```
funcion_principal
    mientras ( a > 0 ) hacer
      a = a - 1;
    fin_mientras
fin_principal
```

- El ciclo mientras hacer debe tener el siguiente formato:
  Ejemplo

```
funcion_principal
    hacer
        a = a/2;
    mientras( a > 0 ); // debe terminar en punto y coma
fin_principal
```

- El comando selección múltiple tiene el siguiente formato:
  Ejemplo

```
funcion_principal
    seleccionar ( a ) entre
        caso 0 :
            imprimir(a);
        caso 1:
            imprimir(a*2);
            romper;
        defecto:
    fin_seleccionar
fin_principal
```

Consideraciones:

- ❏ Después de seleccionar puede existir 0 o más casos. Si el número de casos es 0 debe existir el caso por defecto así sea vacío.
- ❏ El caso por defecto es opcional a menos que no hayan más casos.
- ❏ El romper es opcional.

- Las estructuras deben tener el siguiente formato:

```
estructura C
    entero a;
fin_estructura
estructura Point
    entero x;
    entero y;
    real a,b,c;
    C c;
fin_estructura

funcion_principal
    Point p;
    p.x = 5;
    p.y = 10;
    p.c.a = 5;
fin_principal
```

- Las funciones deben tener el siguiente formato:

```
funcion entero sum(entero a, entero b) hacer
    retornar a + b;
fin_funcion
```

Errores sintácticos

En el caso de un error sintáctico, como en el programa que se muestra a continuación (donde falta un ";"):

```
funcion entero sum(entero a, entero b) hacer
    retornar a + b;
fin_funcion
funcion_principal
    sum(1,2)
fin_principal
```

Se debe imprimir el error en el siguiente formato:

<linea:col> Error sintactico: se encontro: lexema del token encontrado; se esperaba: lista de símbolos/tokens esperados separados por comas.

Donde:

- **linea** y **col** son los números de línea y columna donde se detectó el error.
- **lexema del token encontrado:** corresponde al lexema encontrado que no se esperaba encerrado entre comillas dobles.
- **lista de símbolos/tokens esperados separados por comas:** corresponde a lista de tokens esperados separados por comas y encerrados entre comillas dobles. Por ejemplo: “;”, “)”, “,”.

Para el caso del ejemplo anterior, la cadena de error será:

<6:1> Error sintactico: se encontro: “fin_principal”; se esperaba: “;”.

Nótese que el “;” podría estar en alguna de las siguientes líneas (después de sum(1,2)), siempre y cuando sea el siguiente token leído.

Para el siguiente ejemplo:

```
funcion_principal
   imprimir( “letra ” , letter , “ contador ”, (cont * cont)  );
   imprimir( “total”, cont * 2 + 1 );
   imprimir( “total”; //error sintáctico
   imprimir;( “total”, cont * 2 + 1);//error sintáctico
   imprimir( );//error sintáctico
fin_principal
```

El único error reportado debe ser el primero:

<4:22> Error sintactico: se encontro: “;”; se esperaba: “,”, “)”.

En casos como el anterior, cuando se debe imprimir una lista de símbolos/tokens esperados, éstos deben estar entre comillas dobles, separados por un espacio en blanco, y el orden en que debe aparecer la lista está determinado por la siguiente tabla.

símbolos/token	Token
+	tk_mas
-	tk_menos
*	tk_mult

/	tk_div
%	tk_mod
=	tk_asig
<	tk_menor
>	tk_mayor
<=	tk_menor_igual
>=	tk_mayor_igual
==	tk_igual
&&	tk_y
||	tk_o
!=	tk_dif
!	tk_neg
:	tk_dosp
‘	tk_comilla_sen
“	tk_comilla_dob
;	tk_pyc
,	tk_coma
.	tk_punto
(	tk_par_izq
)	tk_par_der
identificador	id
valor_entero	tk_entero
valor_real	tk_real
valor_caracter	tk_caracter
valor_cadena	tk_cadena
funcion_principal	funcion_principal
fin_principal	fin_principal
leer	leer
imprimir	imprimir

booleano	booleano
caracter	caracter
entero	entero
real	real
cadena	cadena
si	si
entonces	entonces
fin_si	fin_si
si_no	si_no
mientras	mientras
hacer	hacer
fin_mientras	fin_mientras
para	para
fin_para	fin_para
seleccionar	seleccionar
entre	entre
caso	caso
romper	romper
defecto	defecto
fin_seleccionar	fin_seleccionar
estructura	estructura
fin_estructura	fin_estructura
funcion	funcion
fin_funcion	fin_funcion
retornar	retornar
falso	falso
verdadero	verdadero
EOF	EOF

Por ejemplo el siguiente error sintáctico:

```
entero a = ;
```

Debe mostrarse de la siguiente manera:

<1,12> Error sintactico: se encontro: “;”; se esperaba: “-”, “(”, “identificador**”, “**entero**”.**

Nótese que el orden en que aparecen los tokens esperados es teniendo en cuenta la tabla de arriba hacia abajo.

Entrada

Para probar el analizador sintáctico se evaluarán distintos casos de prueba. Cada caso de prueba será pasado al programa por la entrada estándar. Cada entrada consiste en un programa escrito en el lenguaje PsiCoder.

Salida

Por cada archivo de entrada se debe mostrar la salida según lo especificado anteriormente.

Ejemplos

Entrada	**Salida**
funcion_principal imprimir (3+5); fin_principal	El analisis sintactico ha finalizado exitosamente.

Entrada	**Salida**

funcion_principal
  entero a1 = 5 * 3;
  imprimir(a1 / 10);
// comentario al final
	<5,1> Error sintactico: se encontro: “EOF”; se esperaba: “identificador”, “fin_principal”, “leer”, “imprimir”, “entero”, “real”, “caracter”, “cadena”, “booleano”, “si”, “mientras”, “para”, “seleccionar”, “romper”.

Entrada	**Salida**

/* esto no debe
importar, pero cuenta las lineas
*/
funcion_principal
sum(15, 52);
fin_principal
 	
funcion entero sum(entero a, entero b) hacer
retornar a + b;
// fin
fin_funcion
	El analisis sintactico ha finalizado exitosamente.

Entrada	**Salida**

funcion_principal
si (num1 > num2) entonces
res = num1;
si_no
res = num2;
fin_si
si (num1 > num2)
res = num1;
si_no
res = num2;
fin_si
fin_principal
	<8,8> Error sintactico: se encontro: “res”; se esperaba: “entonces”.

Entrada	**Salida**
funcion_principal para (entero i = 0 ; i < a ; 1) hacer imprimir(i); para (j = 0 ; i < a || j < 100 ; p) fin_principal	<5,1> Error sintactico: se encontro: “fin_principal”; se esperaba: “hacer”.

Entrada	**Salida**
estructura Point entero x; entero y; real a,b,c; C c; fin_estructura	Error sintactico: falta funcion_principal