Casos de uso sintácticos

1. Una variable al momento de ser declarada debe seguir el orden estricto de la gramática, que es “<tipodato> <NOMBREVARIABLE>;”, por lo que si se trata de asignar un valor al momento de declarar o no se sigue la estructura para declarar el compilador marca el error 140 que es un aviso de variable mal declarada.
Ejemplo.

```
ent x;  <--- correcto.
ent x = 5; <--- incorrecto.
```

2. La asignación de números negativos a variables numéricas debe ser por medio de paréntesis, si se abre el paréntesis pero no se cierra se marcará como error 155 indicando una mala asignación de número negativo, si no se abre pero se cierra solo se marcara error 150 de asignación incorrecta de valor. Se puede asignar valores positivos dentro de los paréntesis.

```
ent x;
ent y;
x = (-1);
y = (1);
```

3. La función *imprime* debe seguir la estructura declarada en la gramática del lenguaje de manera que cualquier falla en esta estructura, como puede ser la ausencia de paréntesis o de argumento, se marca con el error 160, el cual denota un error en la función imprime. La estructura debe ser: “<IMPRIME> (<CADENA> | <NOMBREVARIABLE>);”.

```
imprime (abc;
imprime abc);
imprime ();
```

4. Una condición debe estar conformada de dos valores y un signo comparativo siguiendo la estructura que la gramática indica, de tal manera que sólo se pueden utilizar valores numéricos y/o identificadores. Si la estructura es incorrecta se muestra al usuario el error 170 que denota la estructura incorrecta en la condición y la línea aproximada en la que ésta se encuentra.
Ejemplo.

```
mientras ( x < 55) <--- correcto.
si ( 55 < x )      <--- correcto.
mientras (x >= )   <--- incorrecto.
si ( == x)         <--- incorrecto.
```

5. La función *mientras* para ser aceptada como válida por el analizador sintáctico debe seguir la estructura “ <MIENTRAS>(<condicion>){<bloquecodigo>} ” por lo que cualquier

paréntesis o llave faltante será marcado como por el error 180 sobre la estructura errónea de la función.
Ejemplo.

```
mientras ( x <= 10) {
//bloque de código//
}
```

6. Para toda llave que abre “{“ debe de existir una llave de cierre “}” por lo que debe existir el mismo número de llaves de cierre que de abrir, si esto no se cumple se marca con el error 190 con el mensaje de que existe una llave que no se ha cerrado. Ejemplo.

```
mientras (condición) { }
si (condición) { }
sino { }
```

7. La función *si* se ve obligada a seguir la estructura “<SI>(<condicion>) { <bloquecodigo> }” señalada en la definición de la gramática, de no seguir esta estructura el compilador lo da a conocer al usuario por medio del error 200 mencionando también la línea donde se da el error. Ejemplo.

```
si ( x <= 10) {
//bloque de código//
}
```

8. La función *sino* se rige por la definición de la gramática “<SINO> {<bloquecodigo>}” por lo que si falta un llave en su estructura o por ejemplo se reemplaza una llave por un paréntesis se marca con el error 220 sobre una mal estructura de la función.
Ejemplo.

```
sino { //bloque de código// }  <--- correcto
sino ( //bloque de código// )  <--- incorrecto
sino //bloque de código//      <--- incorrecto
```

9. La función *sino* no puede ser usada sin anteriormente haber declarado una función *si*, por lo que si se trata de realizar esta acción, se le notifica al usuario por medio del error 225 mencionando la línea donde se encuentra esta estructura *sino.*
Ejemplo.

```
imprime (“hola”);
sino{
  //bloque de código//
}
```

10. La declaración de variables debe hacerse siguiendo la estructura que la gramática indica, “<tipodato> <NOMBREVARIABLE>;”, de no seguir este orden se toma como si se deseara asignar un valor a una variable y será marcado con el error 150.
Ejemplo.

```
ent x;      <--- correcto
cad hola;   <--- correcto
x ent;      <--- incorrecto
hola cad;   <--- incorrecto
```

11. En la asignación de valores a variables se pueden realizar operaciones aritméticas pero no más de una, por lo tanto como máximo deben ser dos operandos y un operador aritmético.
Ejemplo.

```
ent x;
ent y;
x = 5 * 2;
y = 2 * x;
```

12. Antes de comenzar a codificar se debe declarar la etiqueta o palabra clave *inicio* y posteriormente realizar el bloque de código que se desee, por lo que la palabra inicio será el primer token en la tabla de símbolos, de no ser así se marca el error 230 al usuario.
Ejemplo.

```
inicio
{
   ent x;
   x=5;
}
```

13. Para la etiqueta *inicio* se debe cumplir el hecho de abrir y cerrar llaves, según lo marca la producción gramatical “inicio { <bloquecodigo>}”, de no ser así el usuario recibe como retroalimentación el mensaje del error 240 donde se declara que la esctructura del método está equivoca.
Ejemplo.

```
inicio{
  //bloque de código//
}
```

14. En la función imprime se pueden imprimir variables con la condición de que antes de éstas debe haber una cadena y de preferencia no vacía y concatenar por medio del símbolo “.”, por el contrario se marca el error 160.
Ejemplo.

```
ent x;
x = 5;
cad abc;
abc = “hola”;
imprime (“hola ”.x);
imprime (“valor ”.abc);
```

14. Es posible concatenar una variable entero a una cadena por lo que el analizador permite esta acción por medio de la estructura <VARIABLE> = <CADENA> . <ENTERO>
Ejemplo.

```
cad msj ;
ent res ;
res = 5;
msj = "Resultado" . res;
```

15. Se puede asignar el valor positivo a una variable utilizando paréntesis pero forzosamente debe ser sin el signo “+”, de lo contrario se marca como una mala asignación de valor.
Ejemplo.

```
ent x;
x = (+1);  <--- incorrecto
x = (1);   <--- correcto
```