DISEÑO DE COMPILADORES
ANALIZADOR SINTÁCTICO

CAMPUS QUERÉTARO
octubre, 2019

Para la segunda evaluación, debes entregar el analizador sintáctico del lenguaje que elegiste. Cada uno de ustedes tiene diferentes idiomas, con diferentes características y propósitos. Sin embargo, hay algunos requisitos mínimos que espero ver en su gramática definida: ver:

1. **Comentarios:** Tu lenguaje debe permitir comentarios. Por ejemplo, en C usas `/* */` o `//`.
2. **Estructuras anidadas:** Debe haber una manera de crear estructuras anidadas como estructuras condicionales, ciclos o jerarquías (como en XML). Por ejemplo, en C, puedes anidar tantos `if`'s como quieras.
3. **Variables y constantes:** Tu lenguaje debe permitir el uso de variables y constantes, tengan o no un tipo.
4. **Cadenas de caracteres (strings):** Tu lenguaje debe permitir el uso de cadenas como un token.
5. **Tipos de datos:** Si está usando un lenguaje de programación tipado, debe usar al menos 3 tipos (entero, booleano, cadena).
6. **Condicionales y ciclos:** Tu lenguaje de programación debe tener al menos una instrucción de ciclo y una instrucción condicional. Ambos deben admitir el anidamiento.

Todos los casos de prueba previo todavía son válidos. Ademas, espero ver el AST (Abstract Syntax Tree) de estos casos:

1. Un programa sencillo con la definición de una variable.
2. Un programa sencillo con la definición de una constante.
3. Un programa sencillo con un ciclo y una condicional. Un programa con todas las instrucciones definidas.

Los casos de prueba fallidos todavía deben fallar. Ademas, espero ver el AST de los siguientes errores:

1. Un programa sencillo con la definición de una variable en el lugar incorrecto y en el orden incorrecto.
2. Un programa sencillo que utiliza una cadena, variable y constante en un lugar que no está permitido.
3. Un programa sencillo con un ciclo definido pero usando una gramática incorrecta.

Los siguientes casos nuevos deben de pasar (aún cuando contengas entradas incorrectas).

- Asignar el valor a una variable que no corresponde con su tipo. Por ejemplo:

```
int a;
a = "hello";
```

- Llamar un método usando una clase que no contenga tal método. Por ejemplo:

```
class Rectangle {
private:
int width, height;
public:
```

```
void set_values(int, int);
int area();
};

int main() {
Rectangle rect;
rect.init(); // THIS METHOD WAS NOT DEFINED
...
return 0;
}
```

- Usar una variable fuera de su alcance. Por ejemplo:

```
int max(int num1, int num2) {
int result;
if (num1 > num2) {
result = num1;
} else {
result = num2;
}
return result;
}
...
printf( "Max value is : %d\n", result ); //result is not global variable
```

Errores
Todas las entradas que no sigan la estructura definida por tu gramática deberían crear un error. Debes detectar el primer error en la entrada. Si logras detectar todos los errores en todas las entradas dadas, eso te dará puntos extra.
Si alguno de estos requisitos no se aplica a su lenguaje de programación, deberás explicar por qué.