# Diseño de Compiladores 2023: Individual Lenguaje Tavoscript

A continuación se describen las características generales del lenguaje que se deberá desarrollar. Es un lenguaje orientado a objetos basado en javascript pero con un tipado más estricto.
La estructura general de un programa escrito en Tavoscript es:

**Sheet** Nombre_sheet;
*<Declaración de variables globales>*
*<Declaración de Funciones>*
*<Declaración de Clases>*
**main()**
**{**
  <Estatutos>
**}**

*** Las secciones en itálicas son opcionales.**
*** Las palabras y símbolos en bold son Reservadas.**

**Para la Declaración de Variables globales y locales**
sintaxis:

**tipo** (PalabraS reservada )  lista_ids(var1,var2);

Tipos válidos de variables :

- **int**
- **float**
- **string**
- **char**
- **clase** *//* son dinámicas ya que el usuario puede crearlas

Ejemplo:
**int** var1,var2;
**float** double1;

**Para la Declaración de Funciones: (se pueden definir 0 ó más funciones)**
sintaxis:
<tipo_retorno> **func** nombre_funcion(<Parametros>)
{
  <Estatutos>
};

Los parámetros siguen la sintaxis de la declaración de variables simples y clases  y únicamente son de entrada. tipo-retorno puede ser de cualquier tipo soportado, clase, o bien void (si no regresa valor)

Ejemplo:

```
int func sum(int valueA,valueB)
{
  int valueC = valueA + valueB;
  return valueC;
}
```

**Para la Declaración de Clases: (se pueden definir 0 ó más clases)**

sintaxis:

```
class nombre_clase {
        <Declaración de atributos de clase>
        <Declaración de constructor>
        <Declaración de Funciones>
}
```

La declaración de atributos de clase es igual a la declaración de variables simples no otras clases. Todos los atributos son públicos y no pueden ser privados en esta versión de Tavoscript.

Ejemplo:

```
class Coordinate {
    int x,y;
    constructor  Coordinate(int initialX,initialY)
   {
        x = initialX;
        y = initialY;
   }
}
```

**Para la Declaración de Constructor**

```
constructor nombre_clase(<Parámetros>)
{
  <Estatutos>
}
```

La declaración de constructores tiene que ser dentro de una clase.

# Estatutos

**Asignación**

```
sum = Expresion
```

A un identificador (en este caso sum) se le asigna el valor de una expresión, pero no solo puede ser una expresión sino también puede ser una función, o una función concatenada con más expresiones.

```
sum = addValues(valueA,valueB)
average == addValues(valueA,valueB)/2
```

El valor resultante de las expresiones es asignado al identificador del lado izquierdo.

**Modulo Void**

El tipo void es utilizado en funciones que no regresan ningún valor y solo se utiliza una función sin asignación.

void **func** Hola(){};
Hola();

**Retorno de una Función**

Este estatuto **return** es utilizado en funciones para regresar un valor en caso de que la función sea void no se necesita.

**System Ops**

Son estatutos utilizados para hacer operaciones de lectura y escritura, para leer se utiliza: **read(**id,id2…**)** en el cual se puede leer uno o más identificadores separados por coma.
para escribir se utiliza:
**write(**"letrero" o expresión,"letrero o expresión" …**);** se escriben letreros de texto y/o expresiones separados por comas.

**Decisión**

if (expresión) // típica decisión doble
{ <Estatutos>; }
else // esta parte es opcional
{ <Estatutos>; }>

**Repetición**

**do** (expresión) **while**
{ <Estatutos>; }

**for Id**<dimensiones>= exp **to** exp **do**
{ <Estatutos>; } // Repite desde N hasta M brincando de 1 en 1

**Expresiones**

Las expresiones en Tavoscript son las tradicionales (como en C y en Java). Existen los operadores aritméticos,
lógicos y relacionales: +, - , *, / , &(and) , | (or), < , >, ==, etc.
Se manejan las prioridades tradicionales y se pueden emplear paréntesis para alterarla.