




# MANUAL DE USUARIO



Fernando Alvarado  
PROYECTO 1 Compiladores 1

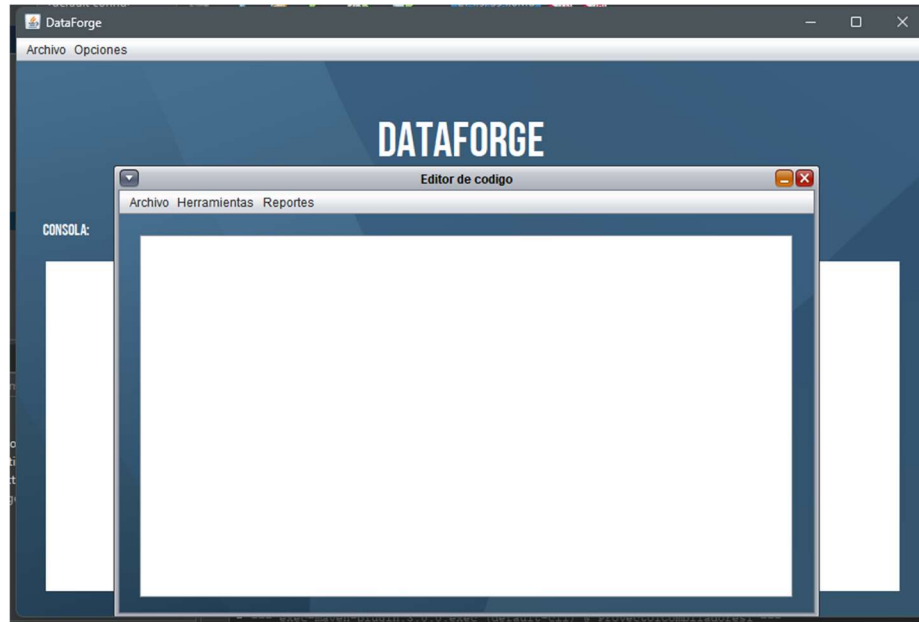
## **Introducción**

DataForge es un lenguaje de programación diseñado para realizar operaciones aritméticas, estadísticas y generar gráficos a partir de una colección de datos. El analizador léxico y sintáctico desarrollado le permitirá escribir y ejecutar código en DataForge de manera eficiente y precisa.

El objetivo de este manual es proporcionarle una comprensión completa de las funcionalidades, características y herramientas disponibles en el sistema analizador de DataForge. A lo largo de este manual, aprenderá a utilizar el editor integrado, ejecutar análisis léxicos y sintácticos, generar reportes y trabajar con las diferentes funcionalidades del lenguaje DataForge.

## Entorno de trabajo

El entorno de trabajo del Analizador de DataForge proporciona las herramientas necesarias para crear, editar, ejecutar y visualizar el código fuente de DataForge. Esta sección detalla las funcionalidades, características y herramientas disponibles en el entorno de trabajo.



### Entorno de Trabajo

El entorno de trabajo se compone de un editor, herramientas de ejecución, áreas de reportes y una consola de salida. A continuación, se describen cada uno de estos elementos.

#### Editor

El editor es una parte esencial del entorno de trabajo, que permite la creación, edición y visualización del código fuente de DataForge. Las funcionalidades del editor incluyen:

**Nuevo archivo:** Permite la creación de archivos en blanco para iniciar un nuevo proyecto.

**Abrir archivo:** Permite abrir archivos existentes con extensión .df para su edición.

**Guardar:** Permite guardar el código fuente en el archivo actual.

**Eliminar pestaña:** Permite cerrar pestañas de archivos abiertos, descartando los cambios si no se han guardado.

El editor también admite la apertura de múltiples pestañas, lo que facilita la gestión de varios archivos de código fuente.

## Funcionalidades

El entorno de trabajo ofrece las siguientes funcionalidades:

**Ejecutar:** Permite enviar el código fuente de la pestaña actual al intérprete para realizar el análisis léxico, sintáctico y la ejecución de instrucciones.

## Características

El editor presenta las siguientes características:

**Múltiples pestañas:** Permite crear y gestionar múltiples pestañas de archivos de código fuente para una mejor organización.

## Herramientas

El entorno de trabajo proporciona las siguientes herramientas:

**Reportes:** Genera reportes de tokens, errores y tabla de símbolos después de cada ejecución.

**Consola de salida:** Muestra los resultados, mensajes e información generada durante la ejecución del código.

El área de consola de salida no es editable por el usuario y únicamente muestra información relevante.

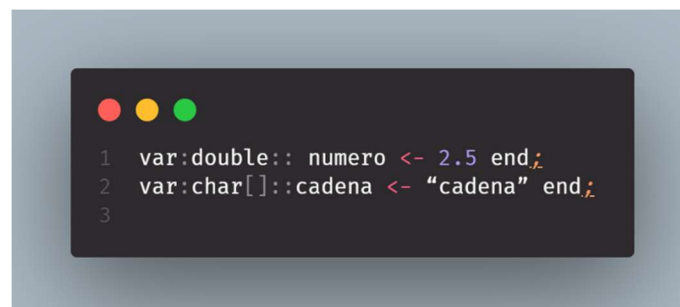
## Descripción del lenguaje

El lenguaje DataForge es un lenguaje de programación diseñado para realizar operaciones aritméticas, estadísticas y generar gráficos a partir de una colección de datos. En esta sección se detallan las características, tipos de datos, operaciones aritméticas, funciones estadísticas, impresión en consola y funciones de graficación disponibles en DataForge.

### Descripción del Lenguaje

- **Case Insensitive.** DataForge es un lenguaje case insensitive, lo que significa que no distingue entre mayúsculas y minúsculas.
- **Encapsulamiento.** Todas las sentencias del lenguaje deben estar encapsuladas entre las palabras reservadas PROGRAM y END PROGRAM.
- **Comentarios.** DataForge permite el uso de comentarios para dejar mensajes en el código fuente. Los comentarios pueden ser de una línea o multilínea:
- **Comentarios de una línea.** Comienzan con el símbolo ! y terminan con un salto de línea.
- **Comentarios multilínea.** Comienzan con <! y terminan con !>.
- **Tipos de Dato.** DataForge admite dos tipos de datos:
  - Cadena (Char[]): Un conjunto de caracteres delimitado por comillas dobles.
  - Decimal (Double): Valores numéricos de punto flotante.

Todas las variables deben ser declaradas antes de ser utilizadas, y tienen un tipo de dato y un nombre de identificador.

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is written in a light blue font and consists of two lines of variable declarations. The first line declares a double variable named 'numero' with the value 2.5. The second line declares a character array variable named 'cadena' with the value 'cadena'. Both lines end with a semicolon and are enclosed in a block structure with 'end;' at the end of each line.

```
1 var:double:: numero <- 2.5 end;  
2 var:char[]::cadena <- "cadena" end;  
3
```

- **Declaración de Arreglos.** Los arreglos en DataForge deben ser declarados antes de ser utilizados. Todos los arreglos tienen un tipo de dato y un nombre identificador que siempre inicia con el símbolo @.

```
1 arr:double::@darray <- [1, 2, 3, 4, 5] end; // Arreglo de tipo double
2 arr:char[]::@carray <- ["12", "2", "3"] end; // Arreglo de tipo string
3 arr:double::@carray <- [numero, copia, 7] end; // Puede usar variables
```

- **Operaciones Aritméticas.** DataForge admite varias operaciones aritméticas que pueden ser utilizadas en declaraciones de variables y arreglos. Estas operaciones pueden anidarse entre ellas.

Operaciones disponibles:

- Suma (SUM): Realiza la suma entre dos valores.
- Resta (RES): Realiza la resta entre dos valores.
- Multiplicación (MUL): Realiza la multiplicación entre dos valores.
- División (DIV): Realiza la división entre dos valores.
- Módulo (MOD): Obtiene el residuo de la división entre dos valores.

```
1 var:double:: suma <- SUM(5, 2) end;
2 var:double:: resta <- RES(3, 2) end;
3 var:double:: multi <- MUL(4, numero) end;
4 var:double:: division <- DIV(1, variable) end;
5 var:double:: modulo <- MOD(5, 4) end;
6
7 // Operaciones anidadas
8 var:double:: resultado <- MUL(SUM(7,3), RES(7, DIV(25,5))) end;
```

- **Funciones Estadísticas.** DataForge cuenta con funciones estadísticas que operan sobre arreglos de tipo double para calcular valores como la media, mediana, moda, varianza, máximo y mínimo.

Funciones disponibles:

- Media (Media): Calcula la media del conjunto de números en un arreglo.

- Mediana (Mediana): Calcula la mediana del conjunto de números en un arreglo.
- Moda (Moda): Calcula la moda del conjunto de números en un arreglo.
- Varianza (Varianza): Calcula la varianza del conjunto de números en un arreglo.
- Máximo (Max): Encuentra el valor más grande en un arreglo.
- Mínimo (Min): Encuentra el valor más pequeño en un arreglo.

```

1 var:double:: media1 <- Media([1, 2, SUM(3, b), 4, a]) end;
2 var:double:: media2 <- Mediana(@arreglo) end;
3
4 arr:double::@arreglo <- [Media(@data), Mediana(@data)] end;
5
6 var:double:: mitad <- DIV(SUM(Max(@data), Min(@data)), 2) end;

```

- **Impresión en Consola.** DataForge permite imprimir expresiones y arreglos en la consola de salida para visualizar resultados.

Impresión de expresiones

```

1 console::print = "hola", numero, 15, "adios" end; // Salida: hola, 15, adios
2 console::print = 1, 2, SUM(3,5), Media(@arreglo) end; // Salida: 1, 2, 8, 15.7

```

Impresión de arreglos

```

1 arr:double::@darray <- [1, 2, 3, 4, 5] end;
2 console::column = "Enteros" => @darray end;

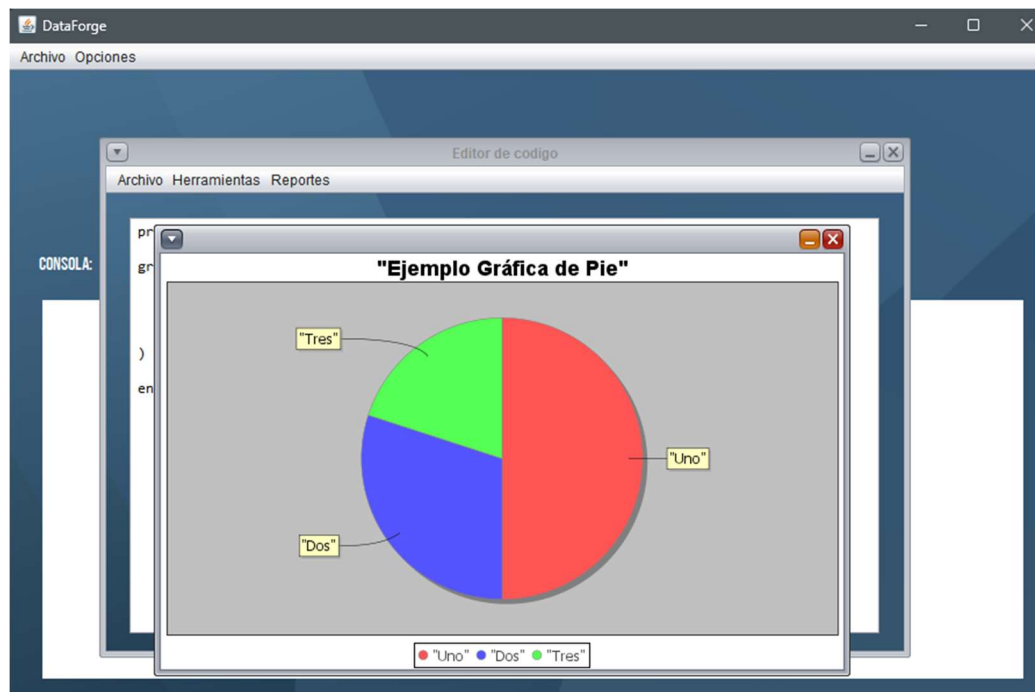
```

- Funciones de Graficación. DataForge cuenta con funciones para generar gráficas personalizadas a partir de conjuntos de datos.

Tipos de gráficas disponibles:

- Pie (grapPie): Gráfica de pastel.
- Barra (grapBar): Gráfica de barras.
- Línea (grapLine): Gráfica de líneas.
- Histograma (Histogram): Gráfica de histograma.

```
1 graphPie(  
2   label::char[] = ["Uno", "Dos", "Tres"] end;  
3   values::double = [50, 30, 20] end;  
4   titulo::char[] = "Ejemplo Gráfica de Pie" end;  
5   EXEC grapPie end;  
6 ) end.
```





## Reportes

Los reportes son una parte fundamental del sistema analizador de DataForge, ya que proporcionan información detallada sobre el análisis léxico, sintáctico y la ejecución del código. En esta sección se describen los diferentes reportes generados por el analizador y cómo interpretarlos.

### Tabla de Tokens

El reporte de tokens muestra información detallada sobre los tokens reconocidos durante el análisis léxico. Cada fila de la tabla representa un token y contiene los siguientes campos:

**Token:** El nombre del token reconocido.

**Tipo:** El tipo de token, como identificador, palabra reservada, operador, etc.

**Valor:** El valor asociado al token, si corresponde.

### Tabla de Errores

El reporte de errores muestra información sobre los errores léxicos y sintácticos encontrados durante el análisis del código fuente. Cada fila de la tabla representa un error y contiene los siguientes campos:

**Tipo de Error:** El tipo de error encontrado, como error léxico o error sintáctico.

**Descripción:** Una descripción del error encontrado.

**Ubicación:** La ubicación aproximada del error en el código fuente.

### Tabla de Símbolos

La tabla de símbolos muestra información sobre las variables y arreglos declarados en el código fuente, así como su tipo y valor. Cada fila de la tabla contiene los siguientes campos:

**Nombre:** El nombre del identificador de la variable o arreglo.

**Tipo:** El tipo de dato de la variable o arreglo.

**Valor:** El valor asignado a la variable o arreglo, si corresponde.

## **Generación de Reportes**

Los reportes son generados automáticamente después de cada ejecución del código fuente. Se presentan en formato HTML para una fácil lectura y comprensión. Es importante revisar los reportes después de cada ejecución para detectar y corregir posibles errores en el código.