

Manual de usuario

CompiScript++

Bienvenido al Manual de Usuario de CompiScript+, un lenguaje de programación diseñado para facilitar el aprendizaje de los fundamentos de la programación y proporcionar una comprensión completa de los conceptos básicos de los lenguajes de programación.

Este manual está destinado a proporcionar a los usuarios una guía detallada sobre cómo utilizar CompiScript+ de manera efectiva. A lo largo de este documento, exploraremos las características clave del lenguaje, su sintaxis, estructuras de control, tipos de datos disponibles y cómo escribir programas utilizando CompiScript+.

¿Qué es CompiScript+?

CompiScript+ es un lenguaje de programación diseñado como parte del proyecto del curso de Organización de Lenguajes y Compiladores 1. Este lenguaje ha sido creado específicamente para los estudiantes del curso de Introducción a la Programación y Computación 1, con el objetivo de ayudarles a adquirir habilidades fundamentales de programación y comprender los conceptos esenciales de un lenguaje de programación.

Con CompiScript+, los usuarios pueden escribir programas que incluyan bucles, sentencias de control, definición de funciones y manipulación de datos. El lenguaje está diseñado para ser intuitivo y fácil de entender, lo que lo convierte en una herramienta ideal para aquellos que están dando sus primeros pasos en el mundo de la programación.

¿Para quién es este manual?

Este manual está dirigido a cualquier persona que esté interesada en aprender a programar utilizando CompiScript+. Ya sea que seas un estudiante que está comenzando su viaje en la programación o un entusiasta experimentado que quiere explorar un nuevo lenguaje, este manual te proporcionará la información necesaria para empezar a escribir programas en CompiScript+.

Instalación de DataForge

Para instalar DataForge en su sistema, siga estos pasos:

1. Descargar el Paquete de Instalación:

Obtenga el paquete de instalación de DataForge desde el repositorio oficial del proyecto en GitHub.

Utilice el comando git clone seguido de la URL del repositorio para clonar el código fuente en su máquina local.

2. Configuración del Entorno de Desarrollo:

Asegúrese de tener instalado el Kit de Desarrollo de Java (JDK) en su sistema. Puede descargarlo desde el sitio web oficial de Oracle.

Instale Apache Maven si aún no lo ha hecho. Maven se utiliza para compilar y gestionar las dependencias del proyecto.

3. Compilación del Proyecto:

Abra una terminal o línea de comandos y navegue hasta el directorio raíz del proyecto DataForge.

Ejecute el siguiente comando para compilar el proyecto y generar los archivos ejecutables:

```
mvn clean compile
```

4. Generación del Analizador Léxico y Sintáctico:

DataForge utiliza JFlex para el análisis léxico y CUP para el análisis sintáctico. Asegúrese de que estos generadores estén configurados correctamente en su entorno.

Ejecute los comandos correspondientes para generar el analizador léxico y el parser. Por ejemplo:

```
jflex Lexer.flex
```

```
java -jar java-cup-11b.jar -parser Parser -symbols Symbol <  
Parser.cup
```

5. Ejecución del Programa:

Una vez completados los pasos anteriores sin errores, estará listo para ejecutar DataForge.

Utilice el comando de ejecución adecuado para iniciar el programa desde la línea de comandos o terminal.

6. Verificación de la Instalación:

Asegúrese de que DataForge se haya ejecutado correctamente y que pueda realizar las tareas de análisis y visualización de datos según lo esperado.

Siguiendo estos pasos, habrá instalado y configurado DataForge en su sistema y estará listo para comenzar a utilizarlo para sus análisis y visualizaciones de datos.

Requerimientos mínimos del sistema

Para utilizar la aplicación de manera adecuada, se requieren los siguientes componentes y programas:

- Memoria RAM: 2GB o superior.
- Espacio disponible en disco: 500 MB.
- React Js
- Typescript: versión 4.2.4 o superior.
- NodeJS: versión 6.14.7.
- Graphviz: versión 2.38.0 (20140413.2041).
- Sistema Operativo: Windows 10 x64.

Es importante asegurarse de que tu sistema cumpla con estos requisitos mínimos para garantizar un funcionamiento óptimo de la aplicación.

Primeros pasos

Archivo Reportes

```
1 EXECUTE main();
2
3 int var1 = 0;
4
5
6 void main(){
7     cout << "Archivo de prueba\n";
8     cout << "Si sale compii" << endl;
9
10    int var1 = 10;
11
12    if(var1 == 0){
13        cout << "Manejo de ambitos erroneo :('" << endl;
14    }else{
15        cout << "Manejo de ambitos correcto" << endl;
16    }
17
18    // tabla de multiplicar
19    tablaMultiplicar(5);
```

Interpretar

Archivo de prueba\nSi sale compii
Manejo de ambitos correcto
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
Final de la tabla de multiplicar
6Funcion recursiva correcta
Fin de la prueba

Archivo Reportes

Crear archivo
Abrir archivo
Guardar archivo

```
1 EXECUTE main();
2
3 int var1 = 0;
4
5
6 void main(){
7     cout << "Archivo de prueba\n";
8     cout << "Si sale compii" << endl;
9
10    int var1 = 10;
11
12    if(var1 == 0){
13        cout << "Manejo de ambitos erroneo :('" << endl;
14    }else{
15        cout << "Manejo de ambitos correcto" << endl;
16    }
17
18    // tabla de multiplicar
19    tablaMultiplicar(5);
```

Interpretar

Archivo de prueba\nSi sale compii
Manejo de ambitos correcto
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
Final de la tabla de multiplicar
6Funcion recursiva correcta
Fin de la prueba

Archivo Reportes

Reporte de errores
Generar árbol AST
Reporte de tabla de símbolos

```
1 EXECUTE main();
2
3 int var1 = 0;
4
5
6 void main(){
7     cout << "Archivo de prueba\n";
8     cout << "Si sale compii" << endl;
9
10    int var1 = 10;
11
12    if(var1 == 0){
13        cout << "Manejo de ambitos erroneo :('" << endl;
14    }else{
15        cout << "Manejo de ambitos correcto" << endl;
16    }
17
18    // tabla de multiplicar
19    tablaMultiplicar(5);
```

Interpretar

Archivo de prueba\nSi sale compii
Manejo de ambitos correcto
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
Final de la tabla de multiplicar
6Funcion recursiva correcta
Fin de la prueba

El paso inicial antes de comenzar a utilizar CompiScript es aprender a utilizar la interfaz gráfica del programa y entender que función lleva a cabo cada botón. La interfaz gráfica del programa es la siguiente:

- Abrir archivo: Mostrará en pantalla un explorador de archivos, en el cual el usuario podrá elegir entre sus documentos archivos con la extensión .sc para comenzar a editarlos.
- Nuevo archivo: Esta función abrirá una nueva pestaña en la que el usuario podrá escribir código y posteriormente guardar ese código en un archivo .sc
- Consola: Mostrará información de la ejecución de los programas escritos por el usuario, la consola no es editable por el usuario.
- El reporte de errores mostrará todos los errores encontrados durante la ejecución del programa
- El reporte de la tabla de símbolos mostrara todos los símbolos reconocidos durante la ejecución del programa
- El botón generar árbol AST mostrará en pantalla el árbol AST construido durante la ejecución del programa.

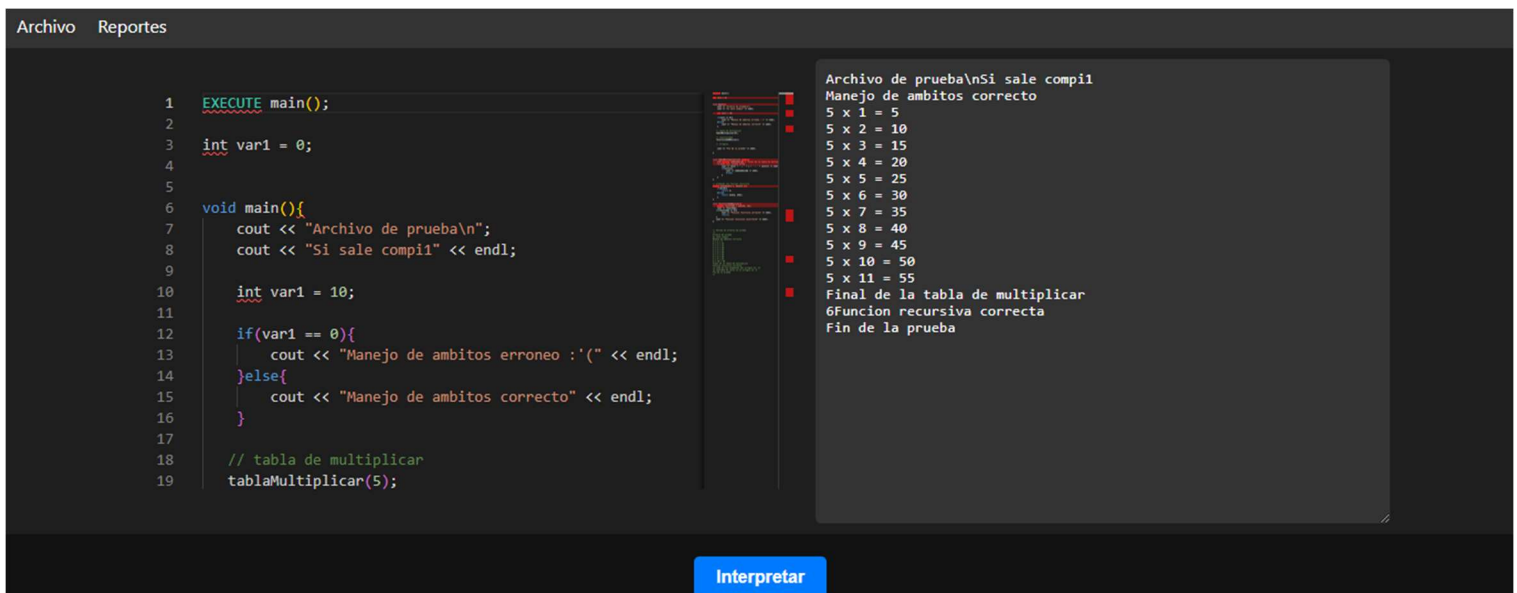
Conceptos básicos

Explicación del Lenguaje

1. Tipos: la aplicación es capaz de manejar 5 tipos primitivos de datos entre los cuales tenemos char, int, double, boolean, string.
2. Declaración de variables: para declarar una variable "TIPO ID = EXPRESION" o "TIPO ID" con la primera declaración podemos asignarle a la variable un valor inicial, mientras que en la segunda forma obtendrá el valor por defecto de cada tipo.
3. Asignación de variables: para poder asignar una variable se utiliza la siguiente estructura "ID = EXPRESION" con dicha instrucción podemos asignar un nuevo valor a la variable.
4. Declaración de vector: Para poder declarar un vector se utiliza la siguiente estructura "TIPO ID[] = NEW TIPO[EXP]" y la de una lista será existen diferentes variaciones de declaración, pudiendo realizarse la siguiente para el vector "TIPO ID[] = [EXP, EXP, ... EXP]"
5. Ciclos
 - a. FOR: la estructura de un for es la siguiente "FOR(DECLARACION/ASIGNACION; CONDICION; ACTUALIZACION){ INSTRUCCIONES}"
 - b. WHILE: la estructura de un while es la siguiente "WHILE(CONDICION){INSTRUCCIONES}"
 - c. DO WHILE: la estructura de un while es la siguiente "DO { INSTRUCCIONES} WHILE(CONDICION)"
6. Nativas: existen diferentes funciones nativas con la estructura "FUNCION(EXP)" en las cuales podemos mencionar, toString, length, round, toUpper, toLower
7. Funciones y métodos
 - a. funciones: los métodos tienen la siguiente estructura "TIPO ID (PARAMETROS) { INSTRUCCIONES }"
 - b. métodos: los métodos tienen la siguiente estructura "VOID ID (PARAMETROS) { INSTRUCCIONES }"

8. Execute: execute es una forma de llamar una función y a la vez indicara cual es nuestra función principal a ejecutar, solo puede existir uno por terminal "EXEC ID (PARAMETROS) "
9. Llamadas: para llamar una función o un método se utiliza la siguiente estructura " ID (PARAMETROS)"
10. Comentarios: el usuario podrá comentar su código de dos formas. "// comentario" para comentarios de una linea y "/* */" para comentarios de bloque

Ejemplo de funcionamiento:



The screenshot shows a C++ IDE with a dark theme. The left pane contains the source code, the middle pane shows the execution process, and the right pane shows the output.

```
1 EXECUTE main();
2
3 int var1 = 0;
4
5
6 void main(){
7     cout << "Archivo de prueba\n";
8     cout << "Si sale compii" << endl;
9
10    int var1 = 10;
11
12    if(var1 == 0){
13        cout << "Manejo de ambitos erroneo :'" << endl;
14    }else{
15        cout << "Manejo de ambitos correcto" << endl;
16    }
17
18    // tabla de multiplicar
19    tablaMultiplicar(5);
```

The output pane displays the following text:

```
Archivo de prueba\nSi sale compii
Manejo de ambitos correcto
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
5 x 11 = 55
Final de la tabla de multiplicar
6Funcion recursiva correcta
Fin de la prueba
```

At the bottom of the IDE, there is a blue button labeled "Interpretar".