Manual Técnico
CompiScript+ - USAC

Autoría - Marco Fernando Cruz Mendoza

Carnet - 202001076

CompiScript+.

Manual Técnico

Guatemala ,24/04/2024

Autoría - Marco Fernando Cruz Mendoza

Tabla de contenido

Introducción

El sistema para carga y análisis de archivos de información y generador de grafica CompiScript+ está compuesto por una interfaz gráfica que permite el ingreso y manejo de datos según las especificaciones solicitadas y uso de cualquier usuario ,para poder cumplir los estándares y requerimiento se hizo uso del lenguaje de programación TypeScript y otras librerías en torno a este lenguaje adecuado de acuerdo a las especificaciones y paradigmas con las que este cuenta ,a través de esta tecnología la administración y manejo de la información contenida por CompiScript+ será de carácter cerrado dando la ventaja de poder manejar distintos conjuntos de información de manera eficiente.

Información destacada

El siguiente manual describe de forma detallada los aspectos técnicos internos y externos a nivel informático del programa CompiScript+ desarrollado para el manejo de información y creación de gráfica, haciendo énfasis en personal capacitado para administración entrada, salida y manejo de datos específicos relacionados con el entorno de un sistema de manejo y exportación de información de relevancia.

Que metodología se utilizó:

Para el proyecto CompiScript+ se utilizó una metodología de clases y objetos debido a la carga de información que solicitara la interfaz gráfica se requirió del uso y manejo de clases para mantener un orden, para la realización de la gráfica se utilizó otra librería ajena al análisis de los archivos, sin embargo, dicha generación de graficas enlaza por medio del contenido generado con el análisis realizado en los analizadores.

Objetivos

Instruir de forma detallada el uso y mantenimiento del sistema CompiScript+ a cualquier individuo o personal técnico especializado encargado de las actividades de mantenimiento, revisión, instalación, mejoramiento u optimización del mismo.

1. Requerimientos

- Sistema Operativo: Sin especificación:

El IDE utilizada para este sistema de almacenamiento y manejo de datos es compatible con varios sistemas operativos como Windows, Linux, IOS, entre otros.

- IDE Visual Studio Code Ultima version (recomendado):

Se recomienda la última versión del IDE Visual Studio Code y algunos componentes como librerías para reconocimiento del lenguaje usado debido a sus ventajas y versatilidad y debido al manejo de ciertos métodos y librerías en el programa se recomienda la misma versión para evitar incompatibilidad en el almacenamiento y manejo de datos.

- RAM: 1 GB:

Requisito mínimo para evitar mal funcionamiento o funcionamiento ineficiente del programa.

- Disco duro: 781 Mb de espacio disponible en disco mínimo:

Requisito mínimo para evitar mal funcionamiento debido al uso de almacenamiento de información.

- Tarjeta gráfica: Sin requerimientos:

Para la ejecución del sistema CompiScript+ no se requieren especificaciones graficas fuera de las integradas en el equipo ya que este se ejecuta en la web.

2.Uso

Para el uso de la interfaz para el sistema CompiScript+ , no se debe tener un conocimiento extenso sobre el código o el IDE que se utiliza, esto debido a que no se utiliza un conocimiento a profundidad sobre el entorno de desarrollo.

Primero se debe correr en terminal los comandos necesarios para ejecutar nuestra aplicación y así asegurar el despliegue correcto en nuestro navegador preferido el cual contiene todos los componentes del programa anexados a este una serie de archivos para la carga de datos como lo es el (.sc) para carga de información en el espacio correspondiente para su posterior análisis todo esto sin comprometer la información solicitada y al momento del almacenamiento de datos estos no se ven comprometidos debido a que para ellos se utilizan diversas clases que mantienen los atributos resguardados para posteriormente hacer uso de los datos.

Los pasos son los siguientes:

1. Descomprimir el archivo Zip sin un lugar específico puede descomprimirse en cualquier parte, para obtener los componentes necesarios del programa.

2. El siguiente paso es abrir el archivo mediante su IDE preferido con ayuda de la consola del equipo, para abrir consola en el sistema operativo Windows debe presionar la combinación de teclas Ctrl + R lo que abrirá la terminal o abrimos una terminal en el sistema **Linux** usando Ctrl + Alt + T, posteriormente pegamos la dirección del archivo.

3. al hacer esto se desplegará la Interfaz del programa mostrando solo una ventana que es contenedora de todo el sistema de acceso del sistema CompiScript+ manteniendo un sistema cerrado de ingreso, al acceder a este se mostrará la interfaz la cual contiene, para instalar todas las dependencias que este necesita basta con ejecutar el siguiente comando en terminal “ npm i ”:

a) Ventana de Interfaz: Este sistema solo manipula la información cargada por el usuario, en ese sentido solo tiene una única ventana principal de despliegue con distintos campos que se deben cumplir y opciones para poder elegir, todo esto entorno al manejo de información y ventanas secundarias para mostrar el procesamiento de

los datos.

Para la creación de esta interfaz se utilizó la librería monaco-editor perteneciente a npm y graphviz-react esto para establecer la parte grafica del sistema CompiScript+ en general.
El sistema también cuenta con una graficación de datos en forma automática de la cual ya se hablara más adelante.

```
//<OpenNewWindowButton />
return (
  <div className="App">
    <div className="background-color: #F0F8FF p-2 text-white bg-opacity-75">
      <div className='text-center style="color: #F8F9FA;'>
        <h1>Proyecto 2 - OLC1</h1>
      </div>
      <br></br>
      <div className='text-center'>
        <div className="container">
          <div className="row">
            <input type="file" id="file" className="form-control form-control-lg" onChange={CargarArchivo} />
          </div>
          <br></br>
          <div className="row">
            <input type="button" value="Interpretar" id="btnCargar" className="btn btn-outline-secondary" onClick={interpretar}
            <input type="button" value="Error" id="btnCargar" className="btn btn-outline-secondary" onClick={mostrarHTMLErrores}
            <input type="button" value="AST" id="btnCargar" className="btn btn-outline-secondary" onClick={mostrarAST} />
          </div>
        </div>
      </div>
      <br></br>
      <div className='text-center style={{ height: "80%", width: "80%" }} '>
        <div className="container" >
          <div className="row">
            <div className="col">
              <p>Entrada</p>
              <Editor height="90vh" defaultLanguage="java" defaultValue="" theme="vs-dark" onMount={(editor) => handleEditorDidM
            </div>
```

Cargar Archivo:

Este botón le permite al usuario por medio de una carga automática ingresar los archivos que se necesiten dependiendo del analizador al sistema CompiScript+ que incluirán los datos necesarios para el llenado de la gráfica, de la siguiente forma.

```
const CargarArchivo = (event) => {
  var file = event.target.files[0];
  var reader = new FileReader();
  reader.onload = function (event) {
    var contents = event.target.result;
    editorRef.current.setValue(contents);
  };
  reader.readAsText(file);
}
```

Al momento de que el archivo es cargado mediante ente botón un campo en la parte inferior izquierda de la interfaz es llenado un elemento del tipo cuadro de texto de entrada el cual se llenara de los datos que contenga el archivo cargado permitiéndole al usuario visualizar y corroborar que ese sea el archivo que desea cargar.

Opción Error: Este botón permite al usuario por medio del procesamiento de datos ya realizado por medio de la opción de interpretación desplegar una nueva ventana la cual mostrara una tabla con los datos correspondientes a todos los errores encontrados por el sistema durante su respectivo análisis.

```
function reporteErrores() {
  fetch("http://localhost:4000/mostrarErrores", {
    method: "GET",
    headers: {
      "Content-Type": "application/json",
    },

  })
    .then(response => response.json())
    .then(data => {
      setArregloErrores(data.ArregloErrores);

      console.log(data.ArregloErrores)
      console.log("Arreglo de errores")
      console.log(arrErrores);

    })
    .catch((error) => {
      alert("Error al generar el reporte de errores.");
      console.error("Error:", error);
    });
}
```

Opción Interpretar: Este botón le permitirá al usuario realizar un análisis léxico y sintáctico del archivo que se está cargando o de lo que capture el cuadro de texto de entrada ubicado en la parte izquierda de la pantalla, este análisis permitirá almacenar datos referentes al archivo (.sc) para posteriormente almacenar trabajar sus datos por medio de distintos métodos y funciones.

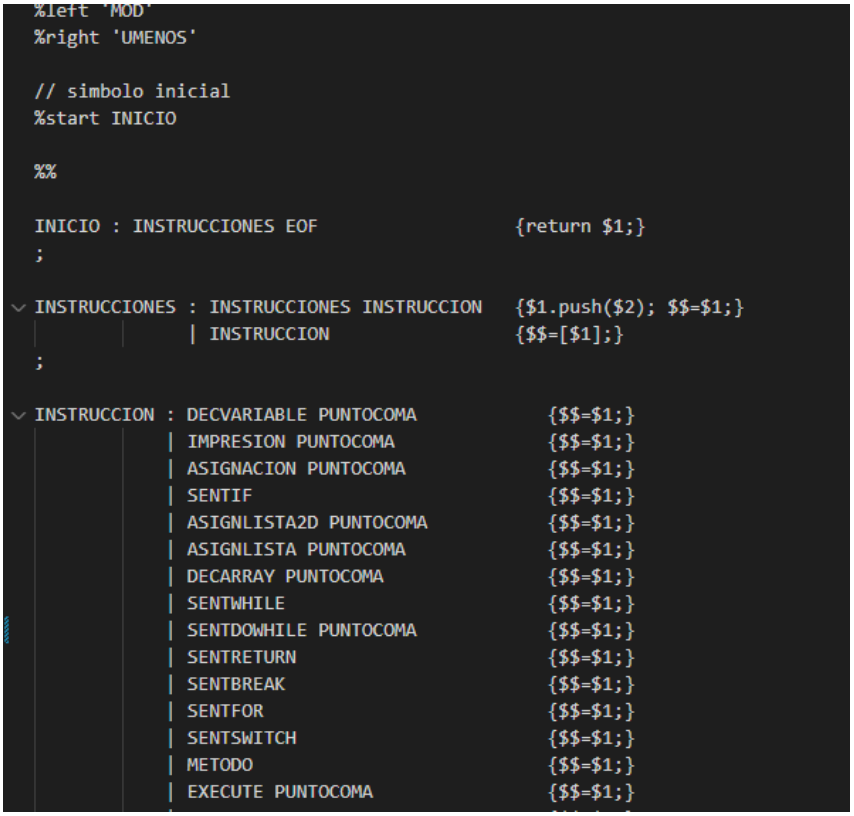
```
public interpretar(req: Request, res: Response) {
    ArregloErrores = new Array<Errores>
    try {
        AstDot = ""
        let parser = require('./analisis/analizador')
        let ast = new Arbol(parser.parse(req.body.entrada))
        let tabla = new tablaSimbolo()
        tabla.setNombre("Ejemplo 1")
```

```
function interpretar() {
  var entrada = editorRef.current.getValue();
  fetch('http://localhost:4000/interpretar', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify({ entrada: entrada }),
  })
    .then(response => response.json())
    .then(data => {
      consolaRef.current.setValue(data.Respuesta);
    })
    .catch((error) => {
      alert("Ocurrio un Error")
      console.error('Error:', error);
    });
}
```

El botón ejecutar CompiScript+ hace uso de una librería para su funcionamiento la cual es Jison pertenecientes a su debido lenguaje, la librería Jison se encarga de analizar de manera léxica el contenido del archivo cargado, de la siguiente manera:

Y de la misma forma esta misma librería se encarga de realizar el análisis sintáctico en tanto se haya realizado el análisis léxico mediante sus herramientas de la siguiente manera:

```
const Llamada = require('./instrucciones/Llamada')

%}

// analizador lexico

%lex
%options case-insensitive

COMMENTUL   "//"([^\r\n]*)?
COMMENTML   [/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]

%%
//comentarios ha ignorar
{COMMENTUL}             {}
{COMMENTML}             {}

//palabras reservadas
"int"                   return "TKINT";
"double"                return "TKDOUBLE";
"bool"                  return "TKBOOL";
"char"                  return "TKCHAR";
"std::string"           return "TKSTRING";
"true"                  return "TKTRUE";
"false"                 return "TKFALSE";
"pow"                   return "TKPOW";
"cout"                  return "TKCOUT"
"endl"                  return "TKENDL"
```



```
%left 'MOD'
%right 'UMENOS'

// simbolo inicial
%start INICIO

%%

INICIO : INSTRUCCIONES EOF                  {return $1;}
;

INSTRUCCIONES : INSTRUCCIONES INSTRUCCION   {$1.push($2); $$=$1;}
              | INSTRUCCION                 {$$=[$1];}
;

INSTRUCCION : DECVARIABLE PUNTOCOMA           {$$=$1;}
            | IMPRESION PUNTOCOMA             {$$=$1;}
            | ASIGNACION PUNTOCOMA            {$$=$1;}
            | SENTIF                          {$$=$1;}
            | ASIGNLISTA2D PUNTOCOMA          {$$=$1;}
            | ASIGNLISTA PUNTOCOMA            {$$=$1;}
            | DECARRAY PUNTOCOMA              {$$=$1;}
            | SENTWHILE                       {$$=$1;}
            | SENTDOWHILE PUNTOCOMA           {$$=$1;}
            | SENTRETURN                      {$$=$1;}
            | SENTBREAK                       {$$=$1;}
            | SENTFOR                         {$$=$1;}
            | SENTSWITCH                      {$$=$1;}
            | METODO                          {$$=$1;}
            | EXECUTE PUNTOCOMA               {$$=$1;}
```

Adicionalmente al realizar el análisis CompiScript+ en la parte derecha de la pestaña principal se cuenta con un componente del tipo recuadro de texto el cual mostrara las salidas que demande la ejecución del programa.

Opción AST:

Este botón le permite al usuario visualizar en la parte inferior de nuestra interfaz una gráfica la cual es generada gracias a la librería perteneciente al lenguaje React llamada graphviz-react la cual nos permite generar la imagen del AST que mostrara todos los datos reconocidos por nuestro sistema, mediante el ingreso de una cadena la cual generamos en el desarrollo de nuestra interpretación del lenguaje esta librería es capaz de generar dicha imagen.

```
const mostrarAST = () => {
  fetch('http://localhost:4000/getAST', {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
    },
  })
    .then(response => response.json())
    .then(jsonData => {
      //console.log(jsonData["AST"]);
      //globaljsonData = jsonData;
      //return jsonData;
      setValorCadenaAst(jsonData.AST);
      console.log(valorAst)
      // Realizar acciones basadas en los datos
    })
    .catch(error => console.error('Error al obtener JSON:', error));
```

A pesar de que la imagen que se genera es de una buena calidad en la parte inferior de nuestra interfaz es necesario pasar el cursor sobre la misma esto para que podamos hacer zoom sobre la imagen generada esto por medio de propiedades de la ventana en nuestro archivo .CSS.

```
.containerAst:hover {
    transform: scale(2.8);
    transition: transform 0.3s ease-in-out;
}
```

3.Configuracion

Para las configuraciones se debe tener en cuenta que el pilar de todo el programa y la interfaz están ligadas a las clases principales de ***OLCProyecto2*** llamadas ***cliente*** y ***servidor*** de la cual se desprenden y unen las otras clases.

Las consideraciones a optimizar pueden ser la optimización en el uso de librerías como Jison que para el sistema CompiScript+ se utilizó la librería con versiones recientes o la migración a otro tipo de lectura y el acompañamiento por defecto del lenguaje que se está usando para el correcto funcionamiento del sistema y su debida implementación.

Tomando en cuenta sus debidas importaciones para su adecuado funcionamiento en la implementación de dichas librerías y herramientas.

Y el resto de librerías utilizadas para la creación de gráficas que para este caso se utilizó la herramienta de graficación graphviz-react perteneciente a npm y dedicada a React de ser necesario también es posible migrar a otro tipo de librería de graficación.

4.Espesificaciònes

Se debe tener presente antes de interactuar con el programa o introducirnos directamente en el código del mismo la forma en la que este se desglosa la forma en la que las subclases se desprenden de la clase ***OLCProyecto2*** el orden de estas es el orden en el que se mostrara la interfaz al momento de interactuar con el programa lo primero con lo que interactúa el usuario es con la ventana que contiene todo lo relacionado al sistema CompiScript+ que se refiere a la interfaz con la que se accederá a la información y se visualizara la gráfica, al ingresar los archivos necesarios correctos este deberá habilitar la posibilidad de interactuar con las opciones de graficación de lo contrario se generara un error.

Toda la interfaz se maneja por interacción por el usuario y el programa, todo esto al momento que el usuario puede visualizar la interfaz CompiScript+ cabe resaltar que al momento de que el usuario genera la gráfica por medio de la interacción con el botón AST, para este apartado de graficas es necesario tomar en cuenta que se usó la librería graphviz-react con versión más reciente esto debido a que en versiones anteriores del mismo las importaciones a ciertas funcionalidades no son las mismas o no se invocan de la misma manera.